A type-theory for higher-order amortized cost analysis

Vineet Rajani
vrajani@mpi-sws.org

Marco Gaboardi
gaboardi@bu.edu

Deepak Garg
dg@mpi-sws.org

Jan Hoffmann
jhoffmann@cmu.edu

Technical report MPI-SWS-2020-004
May 2020

Contents

1 Introduction 3

2 \(\lambda\)-amor†
   2.1 Syntax and Semantics 5
   2.2 Type system 6
   2.3 Model of types 7

3 Examples
   3.1 Church encoding 9
   3.2 Eager functional queue 10
   3.3 Okasaki’s implicit queue 11

4 Embedding Univariate RAML 12
   4.1 A brief primer on Univariate RAML 12
   4.2 Type-directed translation of Univariate RAML into \(\lambda\)-amor 13
   4.3 Semantic properties of the translation 14

5 \(\lambda\)-amor full (with sub-exponentials) 14
   5.1 Changes to the type system: syntax and type rules 15
   5.2 Model of types 16

6 Embedding \(\tilde{d}\)PCF 17
   6.1 A brief primer on \(\tilde{d}\)PCF 17
   6.2 Type-directed translation of \(\tilde{d}\)PCF into \(\lambda\)-amor 18
   6.3 Semantic properties of the translation 19

7 Related work 20

8 Conclusion 21

A Development for \(\lambda\)-amor
   A.1 Syntax 21
   A.2 Typesystem 22
   A.3 Semantics 26
   A.4 Model 27
   A.5 Embedding Univariate RAML
      A.5.1 Type preservation 55
      A.5.2 Cross-language model: RAMLU to \(\lambda\)-amor 77
      A.5.3 Re-deriving Univariate RAML’s soundness 90
## B Development of $\lambda$-amor (full)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Syntax</td>
<td>96</td>
</tr>
<tr>
<td>B.2 Typesystem</td>
<td>98</td>
</tr>
<tr>
<td>B.3 Semantics</td>
<td>102</td>
</tr>
<tr>
<td>B.4 Model</td>
<td>103</td>
</tr>
<tr>
<td>B.5 Embedding dlPCF</td>
<td>134</td>
</tr>
<tr>
<td>B.5.1 Type preservation</td>
<td>134</td>
</tr>
<tr>
<td>B.5.2 Cross-language model: dlPCF to $\lambda$-amor</td>
<td>144</td>
</tr>
<tr>
<td>B.5.3 Re-deriving dlPCF’s soundness</td>
<td>149</td>
</tr>
<tr>
<td>B.5.4 Cross-language model: Krivine to dlPCF</td>
<td>175</td>
</tr>
</tbody>
</table>

## C Examples

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1 Church numerals</td>
<td>177</td>
</tr>
<tr>
<td>C.2 Map</td>
<td>188</td>
</tr>
<tr>
<td>C.3 Append</td>
<td>189</td>
</tr>
<tr>
<td>C.4 Eager functional queue</td>
<td>190</td>
</tr>
<tr>
<td>C.5 Okasaki’s implicit queue</td>
<td>193</td>
</tr>
</tbody>
</table>
Abstract

This paper presents λ-amor, a new type-theoretic framework for amortized cost analysis of higher-order functional programs. λ-amor introduces a new modality for representing potentials—costs that have been accounted for, but not yet incurred, which are central to amortized analysis. Additionally, λ-amor relies on standard type-theoretic concepts like affineness (to prevent duplication of potentials, which could lead to unsoundness), refinement types (for expressiveness) and an indexed cost monad (for actual costs). Although λ-amor is proved sound using a rather simple logical relation, it is very expressive. In fact, we use it as a meta-framework to embed other existing type theories for cost analysis, ranging from call-by-value to call-by-name evaluation, and from effect to co-effect tracking of costs. Using one of our embeddings, we also show that λ-amor is relatively complete for all terminating PCF programs.

Keywords: amortized cost analysis, type theory, relative completeness

1 Introduction

Cost analysis refers to the static verification of an upper bound on the evaluation cost of a program measured in some (application-dependent) abstract unit such as the number of reduction steps, the number of function calls or the number of case analyses during execution of the program. Typically, cost analysis requires a sound static or type-theoretic approximation of the worst-case cost. However, in many cases, specifically, for operations on data structures with internal state, it is more useful to talk of the amortized or average cost of a sequence of n invocations of an operation, since some invocations may pay a huge cost to change the internal state in a way that reduces the cost of subsequent invocations [32]. This kind of an average cost analysis is called an amortized cost analysis. Elementary examples that rely on amortized analysis are a FIFO queue implemented using a pair of functional (LIFO) lists, Fibonacci heaps and the union-find data structure with path compression.

There are type systems for cost analysis, both amortized [16, 20, 19, 17, 18, 11, 26, 23] and non-amortized [6, 8, 12, 3, 24, 9, 10]. Our broad goal in this paper is to develop a type-theoretic framework to unify as many of these lines as possible. In doing so, we also extend the expressiveness of existing frameworks for amortized cost analysis. We call our framework λ-amor and examine its properties, including its expressiveness, in detail. In the remainder of this introduction, we give a high-level description of λ-amor and its properties, followed by a comparison to existing work.

Basics of amortized cost analysis. To motivate λ-amor’s design, we first describe the typical structure of amortized cost analysis. Suppose we want to prove that the average cost of running an operation on a data structure is $c$, when the actual cost of the operation depends on the internal state of the data structure. To do this, we find a function $\phi$ that maps the state $s$ of the data structure to a non-negative number, called a potential, and show (using a type theory like λ-amor) that an invocation of the operation that changes the data structure from state $s_i$ to $s_{i+1}$ has a cost upper-bounded by $\phi(s_i) - \phi(s_{i+1}) + c$. It immediately follows that a sequence of $n$ operations starting in state $s_0$ with $\phi(s_0) = 0$ has a total cost upper-bounded by $(\phi(s_0) - \phi(s_1) + c) + \ldots + (\phi(s_{n-1}) - \phi(s_n) + c)$. This is a telescopic series that equals $\phi(s_0) - \phi(s_n) + n \cdot c$, which in turn is upper-bounded by $n \cdot c$ since $\phi(s_0) = 0$ and $\phi(s_n)$ is non-negative. Hence, the average cost of each operation is no more than $c$, as required. The value $\phi(s)$ is called the potential associated with the state $s$. This potential is needed for verification only, i.e., it is ghost state and it does not exist at run time. The type theory is used to prove only that the cost of an individual operation is upper-bounded by $\phi(s_i) - \phi(s_{i+1}) + c$ (the rest is trivial).

Based on this intuition, we describe the requirements for a type theory to support amortized cost analysis, and how λ-amor satisfies these requirements.

1) The type theory must include some construct to associate the ghost potential to the type of a data structure. This is different from the standard notion of refinement (as $\phi(s)$) in a data structure, which must reflect its state to sufficient precision, to allow relating $p$ and $\tau$ meaningfully. λ-amor uses the type $\tau$ with associated potential $p$.

2) The potential $p$ is related to the state $s$ ($\tau$ equals $\phi(s)$), the type $\tau$ of the data structure must reflect its state to sufficient precision, to allow relating $p$ and $\tau$ meaningfully. λ-amor uses standard refinement types [33] for this. For instance, $L^n \tau$ and $\phi(s_i)$ is the type of lists of length $n$, and $[2n] (L^n \tau)$ is the type of lists of length $n$ with associated potential $2n$. Note how the refinement $n$ relates the aspect of the state (as a list) to the potential associated with it.

3) The type theory must represent execution costs since we need to establish upper-bounds on them. Costs can be represented as either an effect (monad) or a coeffect (comonad). Prior literature has considered both options [11, 16, 9, 10]. Somewhat arbitrarily, λ-amor uses effects: We include an indexed monad $\mathcal{M}_n \phi \tau$ to represent a computation that has a cost $\kappa$, which is also a non-negative number. However, we show that coeffect based cost analysis can be embedded or simulated in λ-amor using potentials.

4) The type theory must prevent duplication of any type that has a potential associated with it, else analysis may not be sound. For example, if a typing derivation duplicates the potential $\phi(s_i)$ even once, then

---

Footnote: Cost analysis can also be used to establish lower bounds but, here, as in most prior work, the focus is on upper bounds.
the operation’s real cost may be up to $2 \cdot \phi(s_i) - \phi(s_{i+1}) + c$, and the amortized analysis described earlier breaks completely. Hence, all potential-carrying types must be treated \textit{affinely} in the type theory. Accordingly, \lambda-amor is an affine type theory with the usual operators of affine logic like $\otimes$, $\&$, and $\oplus$. Duplicate resources are explicitly represented using the standard exponential $!$. To improve expressiveness, we allow the exponential $!$ to be indexed, following the dependent sub-exponential from Bounded Linear Logic \cite{DBLP:conf/lics/Gillies16}.

\textbf{Summary of \lambda-amor and our results.} Overall, \lambda-amor is a \lambda-calculus equipped with a type system that has the four features mentioned above – the construct $[p] \tau$ to associate potential to a type, type refinements, the indexed cost monad $\mathcal{M}_k \kappa \tau$, and affineness with an indexed exponential $![\cdot]$\footnote{The name “\lambda-amor” refers to both the calculus and its type system. The intended use can be disambiguated from the context.}.\footnote{The adjective “relative” means relative to having a refinement domain that is sufficiently expressive.} We give \lambda-amor a call-by-name semantics with eager evaluation for all pairs and sums. Although a call-by-name semantics might seem odd for cost analysis (since most languages are call-by-value or call-by-need), note that, here, costs are confined to a monad, so the semantics of pure evaluation are insignificant. We choose call-by-name since it simplifies our technical development. In fact, we show via an embedding how call-by-value amortized analysis can be simulated in \lambda-amor.

Despite the large number of features, \lambda-amor is conceptually very simple. We prove it sound using an elementary logical relation that extends Pym’s semantics of BI \cite{DBLP:journals/iandc/Pym10}.  The key novelty in building this relation is the treatment of potentials, and their interaction with the cost monad (available potential can offset the cost in the monad).

Finally, we show that two existing, state-of-the-art frameworks for (amortized) cost analysis can be embedded faithfully in \lambda-amor. Our first embedding is that of a core calculus for Resource-aware ML or RAML \cite{DBLP:conf/tpdp/GilliesLP15}, an implemented, widely-used framework for amortized cost analysis of ML programs. RAML is call-by-value, so this embedding shows how call-by-value (amortized) analysis can be simulated in \lambda-amor. Other effect-based type systems like the unary fragment of \cite{DBLP:journals/tcs/BrownH11} are conceptually simpler than RAML as they do not support amortized analysis, so embedding them in \lambda-amor is even easier.

Our second embedding is that of d\textit{L}PCF \cite{DBLP:conf/icsc/DanielssonO11}, a coeffect based type system for PCF, that is relatively complete.\footnote{The adjective “relative” means relative to having a refinement domain that is sufficiently expressive.} This embedding is difficult and shows two things: (a) coeffect-based cost analysis can be simulated in an effect-based system using potentials, and (b) \lambda-amor is also relatively complete for typing PCF in the sense that all terminating PCF programs can be typed with precise cost in \lambda-amor, establishing that \lambda-amor is extremely expressive.

Together, these embeddings show that \lambda-amor can represent cost analyses from very different settings ranging from call-by-value to call-by-name, and effects vs coeffects. In this sense, we may view it as a unifying framework for (amortized) cost analyses.

\textbf{Prior work on amortized cost analysis.} Besides being a unifying framework for cost analysis as just described, \lambda-amor also improves the expressiveness of prior work on amortized cost analysis in the call-by-value setting. The state of the art in this setting is the aforementioned RAML. However, RAML has difficulty dealing with the interaction between higher-order values and potential because it is not based on an affine calculus. To understand the issue, consider a Curried function of two arguments, of which the first carries potential that is used to offset the cost of executing the function. Suppose that this function is \textit{applied partially}. The resulting closure must not be duplicable because it captures the potential from the first argument. For this, one needs an affine type system. Since RAML (and its many extensions) are fundamentally not affine, they cannot handle this example correctly, and the best that exists so far is to limit potential to the last argument of a Curried function. In contrast, \lambda-amor, being fundamentally affine, can handle this example trivially.

There is also work on amortized cost analysis of call-by-need (lazy) programs, e.g., by \cite{DBLP:conf/icalp/DanielssonO08}, who formalizes the seminal thesis of Okasaki on the so-called method of debits \cite{DBLP:conf/icalp/Okasaki93}. Although we cannot embed this line of work \textit{faithfully} in \lambda-amor due to the fundamental difficulty of simulating call-by-need in call-by-name (this difficulty is not specific to our work), we show how one specific example from Okasaki’s/Danielsson’s work can also be analyzed in \lambda-amor’s type system. The cost and the potential functions do not change. We are fairly confident that this “porting” generalizes to most examples of Okasaki.

Finally, there is work on using \textit{program logics} for amortized cost analysis \cite{DBLP:conf/icsc/DanielssonO11, DBLP:conf/isaac/DanielssonO11}. While we expect program logics to be very expressive, this line of work \textit{does not actually} show any embeddings of existing frameworks. Hence, unlike our work, this line of work does not take concrete steps towards a common framework for (amortized) cost analysis. Note that, for our purposes, the choice between the use of a type system and program logic is one of personal taste; we could also have chosen to build \lambda-amor on a program logic instead of a type system and shown similar embeddings.

\textbf{Organization.} To simplify the presentation, we describe \lambda-amor in two stages. First, we describe \lambda-amor without indexing on exponentials (Section 2). This suffices for most examples (Section 3) and the embedding of RAML (Section 4), but not the embedding of d\textit{L}PCF. Then, we introduce the indexed exponentials (Section 5) and show how to embed d\textit{L}PCF (Section 6), thus establishing relative completeness for PCF programs. Appendix A, Appendix B and Appendix C describe the full technical details of \lambda-amor\textsuperscript{−}, \lambda-amor (full) and
2 \( \lambda \text{-amor} \)

To simplify the presentation, we first describe \( \lambda \text{-amor} \), a subset of \( \lambda \text{-amor} \) that only considers the standard exponential! from affine logic, without any indexing (that \( \lambda \text{-amor} \) supports).

2.1 Syntax and Semantics

The syntax of \( \lambda \text{-amor} \) is shown in Fig. 1. We describe the various syntactic categories below.

1. **Indices, sorts, kinds and constraints.** \( \lambda \text{-amor} \) is a refinement type system. (Static) indices, à la DML \([33]\), are used to track information like list lengths, computation costs and potentials. List lengths are represented using natural numbers (sort \( N \)). Potentials and costs are both represented using non-negative real numbers (sort \( \mathbb{R}^+ \)). Besides this, we also have index-level function and their application (required for some examples like that in Section 3.1). \( \lambda \text{-amor} \) also features kinds, denoted by \( K \). \( Type \) is the kind of standard affine types and \( S \rightarrow K \) represents a kind family indexed by the sort \( S \). Finally, constraints (denoted by \( C \)) are predicates (\( =, <, \land \)) over indices.

2. **Types.** \( \lambda \text{-amor} \) is an affine type system. The most important type is the modal type \( [p] \tau \), which ascribes values of type \( \tau \) that have potential \( p \) associated with them (as a ghost). We have the multiplicative unit type (denoted \( 1 \)) and an abstract base type (denoted \( b \)) to represent types like integers or booleans. Then, there are standard affine types – affine function spaces \( (\rightarrow) \), sums \( (\oplus) \), pairs (both the multiplicative \( \otimes \) and the additive \&) and the exponential \(! \), which ascribes expressions that can be duplicated. We include only one representative data type – the length-refined list type \( L^n \tau \), where the length \( n \) is drawn from the language of indices (described earlier). Other data types can be added if needed.

3. **Expressions.** \( \lambda \text{-amor} \) also has universal quantification over types and indices denoted by \( \forall \alpha : K.\tau \) and \( \forall i : S.\tau \), respectively, and existential quantification over indices denoted \( \exists i : S.\tau \). The constraint type \( C \Rightarrow \tau \) means that if constraint \( C \) holds then the underlying term has the type \( \tau \). The dual type \( C \land \tau \) means that the constraint \( C \) holds and the type of the underlying term is \( \tau \). For instance, the type of non-empty lists can be written as \( \exists n. (n > 0) \& (L^n \tau) \). We also have sort-indexed type families, which are type-level functions from sorts to kinds.

Finally, \( \lambda \text{-amor} \) has the monadic type \( M^k \tau \), which represents computations of cost at most \( k \). Technically, \( M^k \tau \) is a graded or indexed monad \([14]\). A non-zero cost can be incurred only by an expression of the monadic type. Following standard convention we call such expressions impure, while expressions of all other types are called pure.

4. **Expressions and values.** There are term-level constructors for all types (in the kind \( Type \)) except for the modal type \( [p] \tau \). The inhabitants of type \( [p] \tau \) are exactly those of type \( \tau \) since the potential is ghost state without runtime manifestation.

We describe the expression and value forms for some of the types. The term-level constructors for the constraint type \( (C \Rightarrow \tau) \), type and index-level quantification \( (\forall \alpha : K.\tau, \forall i : S.\tau) \) are all denoted \( \Lambda e \). (Note that indices, types and constraints do not occur in terms.) We also have a fixed point operator \( (fix) \) which is used to encode recursion.

The monadic type \( M^k \tau \) has several term constructors, including the standard monadic unit \( (\text{ret} e) \) and bind \( (\text{bind} x = e_1 \text{ in } e_2) \). The construct \( \text{store} e \) stores potential with a term and is the introduction form of the type.
Forcing reduction, \( e \downarrow^w v \)

\[
\begin{align*}
E\text{-return:} & \quad e \downarrow v_1 \quad v_1 \Uparrow^w v_1' \quad e_2 \left[v_1'/x\right] \downarrow v_2 \quad v_2 \Uparrow^w v_2' & \quad \text{E-bind:} & \quad \uparrow^w \psi^w e \quad \text{E-tick:}
\end{align*}
\]

\[
\begin{align*}
\text{bind} \ x = e_1 \ \text{in} \ e_2 \ Uparrow^{w_1+w_2} v_2' & \quad \text{E-release:} & \quad e \downarrow v \quad \text{E-store:}
\end{align*}
\]

Figure 2: Selected evaluation rules

Typing judgment: \( \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \)

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : (\tau_1 \otimes \tau_2) & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 : \langle \tau_1, \tau_2 \rangle \quad \text{T-tensorI} & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e : \tau \quad \text{T-ExpI}
\end{align*}
\]

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : !\tau & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 : \tau \quad \text{T-tensorE} & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e : !\tau \quad \text{T-ExpE}
\end{align*}
\]

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau' \quad \text{T-weaken} & \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau' \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau' \quad \text{T-ret}
\end{align*}
\]

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : M \kappa_1 \tau & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 : M \kappa_2 \tau_2 & \quad \Theta \vdash \kappa_1 : \mathbb{R}^+ & \quad \Theta + \kappa_2 : \mathbb{R}^+ & \quad \text{T-bind}
\end{align*}
\]

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau & \quad \Theta + p : \mathbb{R}^+ & \quad \text{T-store}
\end{align*}
\]

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \left[p_1\right] \tau_1 & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 : M(p_1 + p_2) \tau_2 & \quad \Theta + p_1 : \mathbb{R}^+ & \quad \Theta + p_2 : \mathbb{R}^+ & \quad \text{T-release}
\end{align*}
\]

Figure 3: Selected typing rules for \( \lambda\text{-amor}^+ \)

\([p] \tau\). Dually, \( \text{release } x = e_1 \ \text{in} \ \text{e}_2 \) releases potential stored with \( e_1 \) and makes it available to offset the cost of \( e_2 \).

Note that, \( \text{store } e \) and \( \text{release } x = e_1 \ \text{in} \ \text{e}_2 \) are useful only for the type system: they indicate ghost operations, i.e., where potentials should be stored and released, respectively. Operationally, they are uninteresting: \( \text{store } e \) evaluates exactly like \( \text{ret } e \), while \( \text{release } x = e_1 \ \text{in} \ \text{e}_2 \) evaluates exactly like \( \text{bind } x = e_1 \ \text{in} \ \text{e}_2 \). Finally, we have a construct for incurring non-zero cost – the “tick” construct denoted \( \uparrow^w \). This construct indicates that cost \( \kappa \) is incurred where it is placed. Programmers place the construct at appropriate points in a program to model costs incurred during execution, as in prior work \( \llbracket \rrbracket \).

Operational semantics. \( \lambda\text{-amor}^+ \) is a call-by-name calculus with eager evaluation.\(^4\) We use two evaluation judgments – pure and forcing. The pure evaluation judgment \( (e \Downarrow^w v) \) relates an expression \( e \) to the value \( v \) it evaluates to. All monadic expressions are treated as values in the pure evaluation. The rules for pure evaluation are standard so we defer them to the Appendix \( \llbracket \rrbracket \). The forcing evaluation judgment \( e \Downarrow^w v \) is a relation between terms of type \( \text{M} \kappa \tau \) and values of type \( \tau \). \( \kappa \) is the cost incurred in executing (forcing) \( e \). The rules of this judgment are shown in Fig. 2 E-return states that if \( e \) reduces to \( v \) in the pure reduction, then \( \text{ret } e \) forces to \( v \) with 0 cost. E-store is exactly like E-return, emphasizing the ghost nature of potential annotations in types. E-bind is the standard monadic composition of \( e_1 \) with \( e_2 \). The effect (cost) of bind is exactly the sum of the costs of forcing \( e_1 \) and \( e_2 \). E-release is similar. \( \uparrow^w \) is the only cost-consuming construct in the language. E-tick says that \( \uparrow^w \) forces to (\( \uparrow \)) and it incurs cost \( \kappa \).

2.2 Type system

The typing judgment of \( \lambda\text{-amor}^+ \) is written \( \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \). Here, \( \Psi \) is a context mapping type-level variables to their kinds, \( \Theta \) is a context mapping index-level variables to their sorts, \( \Delta \) is a context of constraints

\(^4\)Perhaps somewhat surprisingly, even additive (\( \& \)) pairs are evaluated eagerly. However, since all effects are confined to a monad, this choice does not matter. \( \uparrow \) is lazy as in a standard affine \( \lambda \)-calculus.
Appendix A.

κ type τ a function type over the argument and the return value. sub-potZero allows silently casting an expression of 

Theorem 1 (Soundness). ∀e, v, κ, κ′, τ ∈ Type. ⊢ e : Mκτ ∧ e ↓κ′ v → κ′ ≤ κ

2.3 Model of types

To prove the soundness of λ-amor, we develop a logical-relation model of its types. The model is an extension of Pym’s semantics of BI [31] with potentials, the cost monad, and type refinements. We also step-index the model to break a circularity in its definition, arising from impredicative quantification over types, as in the work of [29]. Because we use step-indices, we also have augmented operational semantics that count the number of rules (denoted T) used during evaluation. The revised judgments are written e ↓T v (pure) and e ↓T v (forcing). The expected details are in the Appendix A.

Figure 4: Selected subtyping rules
expression relation and substitution relations for the affine and non-affine context. The first two are mutually recursive on the lexicographic order (step index $T$), type $(\tau)$, value $<\text{expression}>$.

Value relation. The value relation (denoted by $\llbracket \cdot \rrbracket$) gives an interpretation to $\lambda$-amor$^-$ types (of kind $\tau$) as sets of triples of the form $(p, T, v)$. Importantly, the potential $p$ is an upper-bound on the ambient potential required to construct the value $v$. It must include potential associated with the (types of) subexpressions of $v$.

We describe interesting cases of this relation. The interpretation for the list type is defined by a further induction on list size. For a list of size 0 the value relation contains a nil value with any potential (since nil captures no potential). For a list of size $s + 1$, the value relation is defined inductively on $s$, similar to the tensor pair, which we describe next. For a tensor $(\otimes)$ pair, both components can be used. Therefore, the potential required to construct a tensor pair is at least the sum of the potentials needed to construct the two components. On the other hand, for a with $(\&)$ pair, either but not both of the components can be used by the context. So the potential needed for a $\&$ pair should be sufficient for each component separately. The type $\tau I$ contains $x$ when $e$ is in $\tau$. The important aspect here is that the potential associated with $e$ must be 0, otherwise we would have immediate unsoundness due to replication of potential, as described in Section 1.

Next, we explain the interpretation of the arrow type $\tau_1 \rightarrow \tau_2$: $\lambda x.e$ is in this type with potential $p$ if for any substitution $e'\ (\text{of type } \tau_1)$ that comes with potential $p'$, the total potential $p + p'$ is sufficient for the body $e[e'/x]$ (of type $\tau_2$).

The step indices $T$ play an important role only in the interpretation of the polymorphic type $\forall \alpha.\tau$. Since the type-level parameter $\alpha$ may be substituted by any type, potentially one even larger than $\tau$, the relation would not be well-founded by induction on types alone. Here, we rely on step-indices, noting that substituting $\alpha$ with a type consumes at least one step in our operational semantics, so the relation for $\tau$ (with the substitution) needs to be defined only at a smaller step index. This follows prior work [29].

Next, we come to the new, interesting types for potential and the cost monad. The potential type $[n] \tau$ contains $v$ with required potential $p$ if $p$ is sufficient to account for $n$ and the potential required for $v$. (Note that the same value $v$ is in the interpretation of both $\tau$ and $[n] \tau$.) The graded monadic type $[\kappa] \tau$ contains the (impure) value $v$ with required potential $p$ if $p$ and $\kappa$ together suffice to cover the cost $\kappa'$ of actually forcing $v$ and the potential $p'$ required for the resulting value, i.e., if $p + \kappa \geq \kappa' + p'$. The ambient potential $p$ and the cost $\kappa$ on the monad appear together in a sum, which explains why the typing rule T-release can offset cost on the monad using potential.

The interpretation of a type family $\lambda i.\tau$ is a type-level function, as expected. The interpretation of type-level application is an application of such a function. The remaining cases of the value relation of Fig. 5 should be self-explanatory.

Expression relation. The expression relation, denoted $\llbracket \cdot \rrbracket_\varepsilon$, maps a type to a set of triples of the form $(p, T, e)$. Its definition is fairly simple and standard: we simply check if the value $v$ obtained by pure evaluation of $e$ is in the value relation of the same type. The potential does not change during pure evaluation, but we need to
adjust the step index correctly.

Substitution relations. Finally, we describe the substitution relations for both the affine context ($\Gamma$) and the non-affine context ($\Omega$). Each relation maps the context to a set of valid substitutions for the context, paired with a step index and a potential. The two key points about the interpretation of $\Gamma$ are: 1) The substitution $\gamma$ should map each variable to a value of the correct type (semantically), and 2) The required potential $p$ for the context should be more than the sum of the required potentials for the substitutions of each of the variables. The interpretation of the non-affine context $\Omega$ is much simpler. It only demands that the substituted value is in the interpretation of the correct type with 0 potential.

Soundness. As is standard for logical-relations models, the main meta-theoretic property is the “fundamental theorem” (Theorem 2). The theorem basically says that if $\varepsilon$ is well-typed in some contexts at type $\tau$, then the application of any substitutions in the semantic interpretation of the contexts map $\varepsilon$ into the semantic interpretation of $\tau$. The important, interesting aspect of the theorem in $\lambda$-amor$^-$ is that the potential needed for $\varepsilon$ (after substitution) equals the potential coming from the context, $p_l$. This the crux of the soundness of (amortized) cost analysis in $\lambda$-amor$^-$.

**Theorem 2** (Fundamental theorem for $\lambda$-amor$^-$). \forall \Theta, \Omega, \Gamma, e, \tau, T, p_l, \gamma, \delta, \sigma, \iota.

$\Psi; \Theta; \Delta; \Gamma \vdash e : \tau \land (p_l, T, \gamma) \in [\Gamma \sigma_1]\varepsilon \land (0, T, \delta) \in [\Omega \sigma_1]\varepsilon \implies$

$(p_l, T, e \gamma \delta) \in [\tau \sigma_1]\varepsilon.$

Here, $\iota, \sigma, \delta$ and $\gamma$ denote substitutions for the index context $\Theta$, the type context $\Psi$, the non-affine context $\Omega$ and the affine context $\Gamma$, respectively. This theorem is proved by induction on the given typing judgment with a subinduction on the step-index for the case of fix. The supplementary material has the entire proof.

Theorem 2 is a direct corollary of this fundamental theorem. We can derive several additional corollaries about execution cost directly from this fundamental theorem. For instance, for open terms which only partially use the input potential and save the rest with the result, we can derive Corollary 3. Here, $e$ is a thunk that expects as input a unit argument, but with some associated potential $q$. When applied, $e$ returns a computation (of 0 cost) that forces to a value with a residual potential $q'$. The corollary says that if the context $\Gamma$ provides a potential $p_l$, then forcing $e$ (with the substitution $\gamma$) incurs a cost $J$ and produces a value $v$ that requires potential $p_v$ such that $J \leq (q + p_l) - (q' + p_v)$. This expression may look complex, but it is simply a difference of the incoming potentials of $e$ ($q$ and $p_l$) and the outgoing potentials of $e$ ($q'$ and $p_v$). In Section 4 we show an interesting use of this corollary for deriving an alternate proof of soundness of univariate RAML via its embedding in $\lambda$-amor$^-$.  

**Corollary 3.** \forall e, q, q', \tau, p_l, \gamma, J, v_l, v.

\[ \vdots \vdots \vdots \; \Gamma \vdash e : [g] 1 \rightarrow [M] 0 (([q'] \tau) \land (p_l, \gamma, \gamma)) \in [\Gamma]\varepsilon \land e () \gamma \gamma \Downarrow v_l \Downarrow J v \implies \]

\[ \exists p_v. (p_v, - v) \in [\tau] \land J \leq (q + p_l) - (q' + p_v) \]

3 Examples

Next, we show three nontrivial examples of cost analysis in $\lambda$-amor$^-$. Complete type derivations for the examples can be found in the supplementary material. That material also has two additional, simple examples (the standard list map and append functions) that we omit here.

3.1 Church encoding

Our first example types Church numerals and operations on them. Typing these constructions require nontrivial use of type and index families. The type we give to Church numerals is both general and expressive enough to encode and give precise cost to operations like addition, multiplication and exponentiation.

For exposition, we first consider the types of Church numerals without any cost. To recap, Church numerals encode natural numbers as function applications. For example, a Church zero is defined as $\lambda f.\lambda x. x$ (with zero applications), a Church one as $\lambda f.\lambda x. f x$ (with one application), a Church two as $\lambda f.\lambda x. f (f x)$ (with two applications) and so on. To type a Church numeral, we must specify a type for $f$. We assume that we have an $N$-indexed family of types $\alpha$, and that $f$ maps $\alpha i$ to $\alpha (i + 1)$ for every $i$. Then, the $n$th Church numeral, given such a function $f$, maps $\alpha 0$ to $\alpha n$.

Next, we consider costs. Just for illustration, we are interested in counting a unit cost for every function application. We want to encode the precise costs of operations like addition, multiplication and exponentiation in types. Classically, these operations are defined compositionally. For example, addition of $m$ and $n$ is defined by applying $m$ to the successor function for $f$ and $n$. This iterates the successor function $m$ times over $n$. To type this, the type of $f$ in the Church nats must be general enough. For this, we use a cost family $C$ from $N$ to $R$. The cost of applying $f$ depends on the index of the argument (called $j_n$ below). Then, given such a $f$, the $n$th Church numeral maps $\alpha 0$ to $\alpha n$ with cost $C(0) + \ldots + C(n - 1) + n$, where each $C i$ is the cost of using $f$
the \(i\)th time, and the last \(n\) is the cost of the \(n\) applications in the definition of the \(n\)th Church numeral. Our type for Church numerals, called \(\text{Nat}\) below, captures exactly this intuition.

\[
\text{Nat} = \lambda n. \forall \alpha : \mathbb{N} \rightarrow T\text{ype}. \forall C : \mathbb{N} \rightarrow \mathbb{R}^+.
\]

Below, we describe a term for the Church one, denoted \(\tau\), that has type \(\text{Nat}\ 1\). For notational simplification, we define \(e_1 \uparrow^1 e_2 \triangleq (\text{bind} - = \uparrow^1 \text{in ret}(e_1 e_2))\), which applies \(e_1\) to \(e_2\) and additionally incurs a cost of 1 unit.

\[
\begin{align*}
\tau & : \text{Nat}\ 1 \\
\tau & \triangleq \Lambda \alpha . \Lambda f . \text{ret} (\lambda x. \text{let} \ f_a = f \ \text{in let} \ \langle\langle y_1, y_2 \rangle\rangle = x \ \text{in release} \ - = y_2 \ \text{in bind} a = \text{store}() \ \text{in} \ f_a \ [\uparrow^1 \langle y_1, a \rangle])
\end{align*}
\]

The term \(\tau\) takes the input pair \(x\) of type \(\alpha 0 \otimes ((C 0 + 1) 1)\) and binds its two components to \(y_1\) and \(y_2\). It then releases the potential \(((C 0 + 1) y_2)\), stores \(C\) of the released potential in \(a\), and applies the input function \(f_a\) to \(\langle y_1, a \rangle\), incurring a cost of 1 unit. This incurred cost models the cost of the application (which we want to count). It is offset by the remaining 1 potential that was released from \(y_2\).

Next, we show the encoding for Church addition. Church addition is defined using a successor function (\(\text{succ}\)), which is also defined and type-checked in \(\lambda\)-amor\(^-\), but whose details we elide here. It is just enough to know that the cost of \(\text{succ}\) under our cost model is two units.

\[
\text{succ} : \forall n. \ [2] 1 \rightarrow M 0 (\text{Nat}[n] \rightarrow M 0 \text{Nat}[n + 1])
\]

An encoding of Church addition (\(\text{add}\)) in \(\lambda\)-amor\(^-\) is shown below. The type of \(\text{add}\) takes the required potential (which is \(4 * n_1 + 2\) here) along with two Church naturals (\(\text{Nat} n_1\) and \(\text{Nat} n_2\)) as arguments and computes their sum. The potential of \((4 * n_1 + 2)\) units corresponds to the precise cost of performing the Church addition under our cost model. The whole type is parameterized on \(n_1\) and \(n_2\). Ignoring the decorations, \(\text{add}\) simply applies \(\text{Nat} n_1\) to \(\text{succ}\) and \(\text{Nat} n_2\), as expected.

\[
\begin{align*}
\text{add} & : \forall n_1, n_2. (4 * n_1 + 2) 1 \rightarrow M 0 (\text{Nat} n_1 \rightarrow M 0 (\text{Nat} n_2 \rightarrow M 0 (\text{Nat} n_1 + n_2))) \\
\text{add} & \triangleq \Lambda \alpha . \Lambda a p . \text{ret}(\lambda x, y . \text{ret} (\lambda x . \text{rel} a = \text{p in} \ \text{bind} b = (\text{store}() \ \text{in} \ (\text{succ} [] b)) \ \text{in} \ b_1 \ [\uparrow^1 y_1])
\end{align*}
\]

Listing 1: Encoding of the Church addition in \(\lambda\)-amor\(^-\)

We have similarly encoded Church multiplication and exponentiation in \(\lambda\)-amor\(^-\). We are unaware of such a general encoding of Church numeral in a pure monadic cost framework without potentials.

### 3.2 Eager functional queue

Eager functional FIFO queues are often implemented using two LIFO stacks represented as standard functional lists, say \(l_1\) and \(l_2\). Enqueue is implemented as a push (\(\text{cons}\)) on \(l_1\). Dequeue is implemented as a pop (\(\text{head}\)) from \(l_2\) if it is non-empty. However, if \(l_2\) is empty, then the contents of \(l_1\) are transferred, one at a time, to \(l_2\) and the new \(l_2\) is popped. The transfer from \(l_1\) to \(l_2\) reverses \(l_1\), thus changing the stack's LIFO semantics to a queue's FIFO semantics. We describe the analysis of this eager queue in \(\lambda\)-amor\(^-\) queue. Here, we incur a unit cost for every list cons operation.

Note that the worst-case cost of dequeue is linear in the size of \(l_1\). However, the amortized cost of dequeue is actually constant. This analysis works by accounting the cost of transferring an element from \(l_1\) to \(l_2\) right when the time is enqueued in \(l_1\). This works because an enqueued element can be transferred at most once. Concretely, the enqueue operation has a cost (it requires a potential) of 3 units, of which 1 is used by the enqueue operation itself and the remaining 2 are stored with the element in the list \(l_1\) to be used later in the dequeue operation if required. This is reflected in the type of enqueue. The term for enqueue is straightforward, so we skip it here.

\[
\begin{align*}
enq & : \forall n. [m] 1 \rightarrow \tau \rightarrow L^n ([2] \tau) \rightarrow L^m \tau \rightarrow M 0 (L^{n+1} ([2] \tau) \otimes L^m \tau)
\end{align*}
\]

Observe how each element of the first list \(l_1\) in both the input and the output has a potential 2 associated with it. The dequeue operation (denoted by \(dq\) below) is a bit more involved. The constraints in the type of dequeue reflect a) dequeue can only be performed on a non-empty queue, i.e., if \(m + n > 0\) and b) the sum of the lengths of the resulting list is only 1 less than the length of the input lists, i.e., \(3m' + n'((m' + n' + 1) = (m + n))\). The full type and the term for the dequeue operation are described in Listing 2. Dequeue uses a function \(move\), which moves elements from the first list to the second. We skip the description of \(move\). Type-checked terms for \(enq\), \(dq\) and \(move\) are in the the Appendix.
dq : ∀ m, n. (m + n > 0) ⇒ L^m([2] τ) → L^n τ → M 0 (∃ m', n'. (m' + n' + 1) = (m + n)) & (L^m([2] τ ⊗ L^n τ))

dq ≡ Λ.Λ.Λ.λ l₁ l₂. match l₂ with

| nil → bind l₁ = move [] | l₁ nil in
| match l₁ with
|   | nil → fxr. x
|   | hₚ :: lₚ' → return Λ.⟨⟨ nil, lₚ' ⟩⟩
|   | l₂ :: l₂' → return Λ.⟨⟨ l₁, l₂' ⟩⟩

Listing 2: Dequeue operation for eager functional queue in λ-amor⁻

3.3 Okasaki’s implicit queue

Next we describe an encoding of a lazy data structure, namely, Okasaki’s implicit queue [30]. This is a FIFO functional queue, which supports log-time random access. Okasaki shows an analysis of this queue for memoizing (lazy) evaluation using his method of debits. [11] shows how to formalize this analysis in Agda. Here, we replicate Danielsson’s analysis in λ-amor⁻. Although the cost bounds we obtain are the same as Danielsson’s, we note that our bound does not apply to Okasaki’s/Danielsson’s lazy evaluation scheme. Rather, it applies to our monadic sequential execution, where we specifically encode operations to incur a unit cost at every case analysis on a queue. It turns out that Okasaki’s potential invariants work as-is for this cost model as well.

An implicit queue is either a shallow (trivial) queue of zero or one elements, or a deep (nontrivial) queue consisting of three parts called front, middle and rear. The front has one or two elements, the middle is a recursive, suspended implicit queue of pairs of elements, and the rear has zero or one elements. Because the recursive structure (the middle) is a queue of pairs, random lookup on the whole queue is logarithmic in time.

Overall, the queue can be described as a datatype of 6 constructors, corresponding to the two shallow cases and the four deep cases. Hereon, we assume a polymorphic queue type Queue(τ) with these 6 constructors, called C₀–C₅ has been added as a primitive to the language. The types of these constructors are shown below.

Note that each type encodes the potential needed to apply the constructor (e.g., a potential of 2 units is needed to apply C₄). These potentials match those of Okasaki.

\[
\begin{align*}
C₀ : & \text{Queue } \tau \\
C₂ : & [1] 1 \rightarrow M 0 (τ ⊗\text{Queue}(τ ⊗ \tau)) \rightarrow \text{Queue } \tau \\
C₄ : & [2] 1 \rightarrow M 0 ((τ ⊗ τ) ⊗ \text{Queue}(τ ⊗ \tau)) \rightarrow \text{Queue } \tau \\
C₁ : & \tau \rightarrow \text{Queue } \tau \\
C₃ : & [0] 1 \rightarrow M 0 (τ ⊗\text{Queue}(τ ⊗ \tau) ⊗ \tau) \rightarrow \text{Queue } \tau \\
C₅ : & [1] 1 \rightarrow M 0 ((τ ⊗ τ) ⊗ \text{Queue}(τ ⊗ \tau) ⊗ \tau) \rightarrow \text{Queue } \tau
\end{align*}
\]

Figure 6: Constructors of Okasaki’s implicit queue

We have implemented and analyzed the expected snoc, head and tail functions for this queue in λ-amor⁻. While we defer the details of these functions to the supplementary material, in Listing 3 we show the helper function headTail, which extracts both the head and the tail of a queue and is the crux of our implementation.

This function has an amortized cost of 3 units as indicated by its type. At the top-level, the function releases the 3 units of potential and immediately uses 1 unit to case analyze the form of the queue. The remaining 2 units are used – partially or fully – in the various cases. For example, when the input queue is C₁ x, we return a pair of x and the empty queue (C₀). The 2 units of potential are simply discarded in this case.

The cases C₂–C₅ are interesting. We describe here only the case C₃. From Fig. 6 we know that the suspension in C₃ needs 0 units of potential to be forced. So we store 0 units of potential in p′ and the remaining 2 units of potential in p₀, which is used later. We then force the suspension (named x) to obtain the front (f), middle (m) and rear (r). The front f is just returned as the head while the tail is constructed using the constructor C₅. Inside the suspension of C₅ we have 1 unit of additional potential available to us via p′. We use this 1 unit of potential from p′ along with the 2 units of potential from p₀, which we constructed earlier, to obtain a total of 3 units of potential, which allows us to make a recursive call to headTail on the middle. This gives us the head and tail of the middle. The rest of this case is straightforward.

Case C₂ is similar, while cases C₄ and C₅ do not involve recursive calls.

\[
\begin{align*}
\text{headTail} : & [3] 1 \rightarrow ∀ α.\text{Queue } α \rightarrow M 0 (α ⊗ \text{Queue } α) \\
\text{headTail} ≡ & \text{fix } H T.λ x.λ q. \\
& = \text{release } p \text{ in } = \uparrow^1 \text{ in } \text{ret} \\
\text{case } q \text{ of} \\
& \mid C₀ \mapsto \text{fix x. x} \\
& \mid C₁ x \mapsto \text{ret} ⟨⟨ x, C₀ ⟩⟩ \\
& \mid C₂ x \mapsto \\
& \quad \text{bind } p'' = \text{store} () \text{ in } \text{bind } p₀ = \text{store} () \text{ in} \\
& \quad \text{bind } x' = x \ p' \text{ in } \text{let} ⟨⟨ f, m ⟩⟩ = x' \text{ in} \\
& \quad \text{ret} ⟨⟨ f, C₄ (λ p'', = \text{release } p₀ \text{ in } = \text{release } p'' \text{ in } \text{bind } p₀ = \text{store} () \text{ in } H T \ p_r [] m) ⟩⟩⟩ \\
& \mid C₅ x \mapsto
\end{align*}
\]

\(^{1}\text{Okasaki does not need a corresponding function since he works in a non-affine setting. In some of his implementation he uses the same queue twice, once to extract its head and once to extract its tail. In our setting, a queue is affine so it cannot be used twice. Hence, we define this combined function.}
\[
\begin{align*}
\text{bind } p' &= \text{store()} \quad \text{bind } p_0 = \text{store()} \quad \text{in} \\
\text{bind } x' &= x \quad p' \quad \text{let } (f, m) = x' \quad \text{in} \\
\ret(f, C5) (\lambda p'. -) &= \text{release } p_0 \quad \text{in} \\
\text{bind } p'' &= \text{store()} \quad \text{in} \\
| C4 x \mapsto & \\
\text{bind } p' &= \text{store()} \quad \text{in} \\
\text{bind } x' &= x \quad p' \quad \text{let } (f, m) = x' \quad \text{in} \\
\ret(f_1, C2) (\lambda p'', \text{ret}(f_2, m'')) &= f \quad \text{in} \\
| C5 x \mapsto & \\
\text{bind } p' &= \text{store()} \quad \text{in} \\
\text{bind } x' &= x \quad p' \quad \text{let } (f, m) = x' \quad \text{in} \\
\ret(f_1, C3) (\lambda p'', \text{ret}(f_2, m'', r))) &= f \quad \text{in}
\end{align*}
\]

Listing 3: Function to obtain head and tail

The Appendix C contains full typing derivations for the headTail, head, tail and snoc operations.

4 Embedding Univariate RAML

In this section we describe an embedding of Resource Aware ML (RAML) \([19, 16]\) into \(\lambda\text{-amor}^-\). RAML is an effect-based type system for amortized analysis of OCaml programs using the method of potentials \([32, 7]\). The main motivation for this embedding is to show that: 1) \(\lambda\text{-amor}^-\) can also perform effect-based cost analysis like RAML and thus can be used to analyze all examples that have been tried on RAML and 2) \(\lambda\text{-amor}^-\), despite being a call-by-name framework, can embed RAML which is a call-by-value framework.

We describe an embedding of Univariate RAML \([19, 16]\) (which subsumes Linear RAML \([20]\)) into \(\lambda\text{-amor}^-\). We leave embedding multivariate RAML \([17]\) to future work but anticipate no fundamental difficulties in doing so.

4.1 A brief primer on Univariate RAML

We give a brief primer of Univariate RAML \([19, 16]\) here. The key feature of Univariate RAML is an ability to encode univariate polynomials in the size of the input data as potential functions. Such functions are expressed as non-negative linear combinations of binomial coefficients \(\binom{k}{n}\), where \(n\) is the size of the input data structure and \(k\) is some natural number. Vector annotations on the list type \(L^n\tau\), for instance, are used as a representation of such univariate polynomials. The underlying potential on a list of size \(n\) and type \(L^n\tau\) can then be described as \(\phi(q, n) \triangleq \sum_{1 \leq k \leq n} \binom{n}{k} q_k\) where \(q = \{q_1 \ldots q_k\}\). The authors of RAML show using the properties of binomial coefficients, that such a representation is amenable to an inductive characterization of polynomials which plays a crucial role in setting up the typing rules of their system. If \(q = \{q_1 \ldots q_k\}\) is the potential vector associated with a list then \(\langle\langle \bar{q} \rangle\rangle = \{q_1 + q_2, q_2 + q_3, \ldots q_{k-1} + q_k, q_k\}\) is the potential vector associated with the tail of that list. Trees follow a treatment similar to lists. Base types (unit, bools, ints) have zero potential and the potential of a pair is just the sum of the potentials of the components. A snippet of the definition of the potential function \(\Phi(a : A)\) (from \([19]\)) is described below.

\[
\begin{align*}
\Phi(a : A) &= 0 \quad \text{where } A = \{\text{unit, int, bool}\} \\
\Phi((a_1, a_2) : (A_1, A_2)) &= \Phi(a_1 : A_1) + \Phi(a_2 : A_2) \\
\Phi([\ell : L^n\tau A]) &= 0 \\
\Phi((a :: \ell) : L^n\tau A) &= q_1 + \Phi(a : A) + \Phi(\ell : L^n\tau A) \\
\quad \text{where } q = \{q_1 \ldots q_k\}
\end{align*}
\]

A type system is built around this basic idea with a typing judgment of the form \(\Sigma, \Gamma :: q, e_r : \tau\) where \(\Gamma\) is a typing context mapping free variables to their types, \(\Sigma\) is a context for function signatures mapping a function name to a type (this is separate from the typing context because RAML only has first-order functions that are declared at the top-level), \(q\) and \(q'\) denote the statically approximated available and remaining potential before and after the execution of \(e_r\), respectively, and \(\tau\) is the zero-order type of \(e_r\). Vector annotations are specified on list and tree types (as mentioned above). Types of first-order functions follow an intuition similar to the typing judgment above. \(\tau_1 \leadsto q_1 \times q_2\) denotes the type of a first-order RAML function which takes an argument of type \(\tau_1\) and returns a value of type \(\tau_2\), \(q\) units of potential are needed before this function can be applied and \(q'\) units of potential are left after this function has been applied. Intuitively, the cost of the function is upper-bounded by \((q + \text{potential of the input}) \times (q' + \text{potential of the result})\). Fig. 3 describe typing rules for function application and list cons. The \text{app} rule type-checks the function application with an input and remaining potential of \((q + K_{1\text{app}})\) and \((q' - K_{2\text{app}})\) units, respectively. RAML divides the cost of application between \(K_{1\text{app}}\) and \(K_{2\text{app}}\) units. Of the available \(q + K_{1\text{app}}\) units, \(q\) units are required by the function itself and \(K_{1\text{app}}\) units are consumed before the application is performed. Likewise, of the remaining \(q' - K_{2\text{app}}\) units, \(q'\) units are made available from the function and \(K_{2\text{app}}\) units are consumed after the application is performed. The cons rule requires an input potential of \(q + p_1 + K_{1\text{cons}}\) units of which \(p_1\) units are added to the potential of the resulting list and \(K_{1\text{cons}}\) units are consumed as the cost of performing this operation.

\(^\text{Every time a subtraction like } (I - J) \text{ appears, RAML implicitly assumes that there is a side condition } (I - J) \geq 0.\)
Soundness of the type system is defined by Theorem 3. Soundness is defined for top-level RAML programs (formalized later in Definition 5), which basically consist of first-order function definitions (denoted by $F$) and the "main" expression $e$, which uses those functions. Stack (denoted by $V$) and heap (denoted by $H$) are used to provide bindings for free variables and locations in $e$.

Theorem 4 (Univariate RAML’s soundness). \( \forall H, H', V, \Sigma, \tau, \Sigma, e, t \vdash H, H' \vdash e \vdash V \vdash \Gamma \vdash e \vdash \Sigma \vdash \tau \) \( \vdash H, V, (\Gamma + q) - (q' + \Phi_H(s : \tau)) \)

4.2 Type-directed translation of Univariate RAML into $\lambda$-amor$^-$

As mentioned above, types in Univariate RAML include types for unit, booleans, integers, lists, trees, pairs and the redundancy of treating similar types again and again. Ignoring trees. These simplifications only make the development more concise as we do not have to deal with.

The translation from Univariate RAML to $\lambda$-amor$^-$ is type-directed. We describe the type translation function (denoted by $\langle \_ \rangle$) from RAML types to $\lambda$-amor$^-$ types in Fig. 8.

Since RAML allows for full replication of unit and base types, we translate RAML’s base type, $b$, into $!b$ of $\lambda$-amor$^-$.

We translate a RAML pair type into a tensor (\( \otimes \)) pair. This is in line with how pairs are treated in RAML (both elements of the pair are available on elimination). Finally, a function type \( \tau_1 \rightarrow b \rightarrow \tau_2 \) in RAML is translated into the function type \( [q] : \tau_1 \rightarrow [q'] : \tau_2 \) \( \Rightarrow [q] : \tau_1 \\ \rightarrow [q'] : \tau_2 \)

We use this type translation function to produce a translation for Univariate RAML expressions by induction on RAML’s typing judgment. The translation judgment is $\Sigma: x: \tau, \exists \vdash e : \tau$ \( \Rightarrow e : \Sigma \). It basically means that a well-typed RAML expression $e$, is translated into a $\lambda$-amor$^-$ expression $e$.

The translation is expressive of the type \( [q] : \tau_1 \rightarrow [q']: \tau_2 \). We only describe the app rule here (Fig. 9). Since we know that the desired term must have the type \( [q + K_{app}'] : \tau_1 \rightarrow [q' + K_{app}'] : \tau_2 \). The translated term is a function which takes an argument, $u$, of the desired modal type and releases the potential to make it available for consumption. The continuation then consumes $K_{app}$ potential that leaves $q - K_{app}$ potential remaining for $\text{bind } P = \text{store}()$ in $E_1$.

We then store $q$ units of potential with the unit and use it to perform a function application. We get a result of type $M 0 (\langle q' \rangle \{\tau_2\})$. We release these $q'$ units of potential and consume $K_{app}$ units from it. This leaves us with a remaining potential of $q' - K_{app}$ units. We store this remaining potential with $f_2$ and box it up in a monad to get the desired type. Translations of other RAML terms (which we do not describe here) follow a similar approach. The entire translation is intuitive and relies extensively on the ghost operations store and release at appropriate places.
The fundamental theorem of this relation basically allows us to establish that the source expression and its $\lambda$-term is preserved in $\lambda$-amor$^\to$. We show that the translation is type-preserving by proving that the obtained $\lambda$-amor$^\to$ terms are well-typed (Theorem 5). The proof of this theorem works by induction on RAML’s type derivation.

**Theorem 5 (Type preservation: Univariate RAML to $\lambda$-amor$^\to$).** If $\Sigma; \Gamma \vdash_{q_i} e : \tau$ in Univariate RAML then there exists $e'$ such that $\Sigma; \Gamma \vdash_{q_i} e : \tau \Rightarrow e'$ and there is a derivation of $; ; ; \Rightarrow \left[ \Sigma; \Gamma \right] \vdash_{q_i} e' : [q_i] 1 \Rightarrow_0 M_0 \left[ (q_i)[q_i] \right]$ in $\lambda$-amor$^\to$.

As mentioned earlier, RAML only has first-order functions which are defined at the top-level. So, we need to lift this translation to the top-level. Definition 6 defines the top-level RAML program along with the translation.

**Definition 6 (Top level RAML program translation).** Given a top-level RAML program

$$P \triangleq F; e_{\text{main}}$$

where $F \triangleq f_1(x) = e_{f_1}, \ldots, f_n(x) = e_{f_n}$ s.t.

$$\Sigma, x : \tau_f \vdash_{q_i} e_{f_1} : \tau_f, \ldots, \Sigma, x : \tau_f \vdash_{q_i} e_{f_n} : \tau_f$$

and $\Sigma, \Gamma \vdash_{q_i} e_{\text{main}} : \tau$

where $\Sigma = f_1 : \tau_f \vdash_{q_i} e_{f_1}, \ldots, f_n : \tau_f \vdash_{q_i} e_{f_n}$.

The translation of $P$, denoted by $\overline{P}$, is defined as $(\overline{F}, \overline{e}_t)$ where

$$\overline{F} = \text{fix}_f, \lambda u. \lambda x. e_{t_1}, \ldots, \text{fix}_{f_n}, \lambda u. \lambda x. e_{t_n}$$

and

$$\Sigma, x : \tau_f \vdash_{q_i} e_{f_1} : \tau_f, \ldots, \Sigma, x : \tau_f \vdash_{q_i} e_{f_n} : \tau_f \Rightarrow e_{t_n}$$

4.3 Semantic properties of the translation

Besides type-preservation, we additionally:

1) prove that our translation preserves semantics and cost of the source RAML term and

2) re-derive RAML’s soundness result using $\lambda$-amor$^\to$-amor$^\to$’s fundamental theorem (Theorem 2) and properties of the translation. This is a sanity check to ensure that our type translation preserves cost meaningfully (otherwise, we would not be able to recover RAML’s soundness theorem in this way).

Semantics and cost preservation is formally stated in Theorem 7, which can be read as follows: if $e_s$ is a closed source (RAML) term which translates to a target (\(\lambda\)-amor$^\to$) term $e_t$ and if the source expression evaluates to a value (and a heap $H$, because RAML uses imperative boxed data structures) then the target term after applying to a unit (because the translation is always a function) can be evaluated to a value $t_{e_f}$ via pure $\left( (\overline{\circ}) \right)$ and forcing $\left( (\overline{\circ}) ^ {\circ} \right)$ relations s.t. the source and the target values are the same and the cost of evaluation in the target is at least as much as the cost of evaluation in the source.

**Theorem 7 (Semantics and cost preservation).** $\forall H, e, ^*v, p, p', q, q'$.

$$; ; ; ; ; \Rightarrow \left[ \Sigma; \Gamma \right] \vdash_{q_i} e_{t_1} : \tau_{t_1} \land \ldots \land \Sigma, x : \tau_f \vdash_{q_i} e_{t_n} : \tau_{t_n} \Rightarrow e_{t_n}$$

$\exists t_{e_f}, J, e_{t_1}(\left( \overline{\circ} \right) \downarrow) \Rightarrow t_{e_f} ^ {\circ} \Rightarrow ^* v = t_{e_f} \land p - p' \leq J$

The proof of Theorem 7 is via a cross-language relation between RAML and $\lambda$-amor$^\to$ terms. The relation (described in the Appendix) is complex because it has to relate RAML’s imperative data structures (like list which is represented as a chain of pointers in the heap) with $\lambda$-amor$^\to$’s purely functional datastructures. The fundamental theorem of this relation basically allows us to establish that the source expression and its translation are related, which basically internalizes semantics and cost preservation as required by Theorem 4.

Finally, we re-derive RAML’s soundness (Theorem 4) in $\lambda$-amor$^\to$ using $\lambda$-amor$^\to$’s fundamental theorem and the properties of the translation. To prove this theorem, we obtain a translated term corresponding to the term $e$ (of Theorem 4) via our translation. Then, using Theorem 5, we show that the cost of forcing the unit application of the target is lower-bounded by $p - p'$. After that, we use Corollary 3 to obtain the upper-bound on $p - p'$ as required in the statement of Theorem 4.

5 $\lambda$-amor full (with sub-exponentials)

Recall the Church encoding from Section 3.1. A Church numeral always applies the function argument a finite number of times. However, the type that we assigned to Church numeral specified an unbounded number of copies for the function argument. Similarly, the index $j_i$ can only take $n$ unique values in the range 0 to $(n-1)$, but it was left unrestricted in the type that we saw earlier. Both these limitations are due to $\lambda$-amor$^\to$’s lack of ability to specify these constraints at the level of types. These limitations, however, can be avoided by refining the exponential type (!$\tau$). In particular, we show that by using the dependent sub-exponential (!$_{i<n} \tau$) from
Bounded Linear Logic \cite{15} we can not only specify a bound on the number of copies of the underlying term, but can also specify the constraints on the index-level substitutions that are needed in the Church encoding. Morally, \(!i<\tau\) represent \(n\) copies of \(\tau\) in which \(i\) is uniquely substituted with all values from 0 to \(n - 1\).

Such an indexed sub-exponential has been used in the prior work. \(d\text{PCF} \ [9]\), for instance, uses it to obtain relative completeness of typing for PCF programs, which means every PCF program can be type checked in \(d\text{PCF}\), where the cost of the PCF program gets internalized in \(d\text{PCF}'s\) typing derivation. This is a very powerful result. However, cost analysis in \(d\text{PCF}\) works only for whole programs. This is because \(d\text{PCF}\) does not internalize cost into the types but rather tracks it only on the typing judgment. As a result, in order to verify the cost of \(e_2\) in the let expression, say let \(x = e_1\) in \(e_2\), we would need the whole typing derivation of \(e_1\) as cost is encoded on the judgment in \(d\text{PCF}\).

Contrast this with \(\lambda\text{-amor}\), where cost requirements are described in the types (\(M_k\kappa\tau\) for instance). In this case, the cost of \(e_2\) can be verified just by knowing the type of \(e_1\) (the whole typing derivation of \(e_1\) is not required to type check \(e_2\)). Therefore \(e_1\) can be verified separately.

We show that by adding such an indexed sub-exponential to \(\lambda\text{-amor}\), we can not only obtain the same relative completeness\footnote{Use of indexed sub-exponential is just one way of obtaining relative completeness. There could be other approaches, which we do not get into here.} result that \(d\text{PCF}\) obtains, but also provide a compositional alternative to the \(d\text{PCF}\) style of cost analysis. We describe the addition of \(!i<\tau\) to \(\lambda\text{-amor}\) in this section. We call the resulting system \(\lambda\text{-amor}\).

5.1 Changes to the type system: syntax and type rules

We take the same language as described earlier in Section 2 but replace the exponential type with an indexed sub-exponential type. There are no changes to the term syntax or semantics of the language. We just extend the index language with two specific counting functions described below.

\[ \text{Index} \]
\[
\begin{align*}
I, J, K &::= \ldots | \sum a<i I | \odot^J a K I | \ldots \\
\text{Types} &::= \ldots | \odot a\tau | \ldots \\
\text{Non-affine context} &::= \ldots | \Omega, x :: a<i \tau \\
\text{for term variables} &
\end{align*}
\]

\[ \Omega_1 + \Omega_2 \triangleq \left\{ \begin{array}{ll}
\Omega_2 & \Omega_1 = \ldots \\
(\Omega_1 + \Omega_2/x), x :: a<i+j \tau & \Omega_1 = \Omega_1', x :: a<i \tau[a/c] \land (x :: a<j \tau[f+b/c]) \in \Omega_2 \\
(\Omega_1 + \Omega_2), x :: a<i \tau & \Omega_1 = \Omega_1', x :: a<i \tau \land (x :: a<j \tau) \notin \Omega_2
\end{array} \right. \\
\sum \Omega \triangleq \left\{ \begin{array}{ll}
\Omega & \Omega = \Omega', x :: a<j \sigma[(\sum d<i J[d/a] + b)/c] \\
(\sum a<i \Omega), x :: a<i+j \sigma & \Omega = \Omega', x :: a<j \sigma
\end{array} \right.
\]

Figure 10: Changes to the type system syntax

We describe the changes introduced to the index and term language in Fig. 10. Since the sub-exponential type helps in specifying the number of copies of a term, we find inclusion of two specific counting functions to the index language very useful, both of which have been inspired from prior work \cite{9}. The first one is a function \(J\) ranging from 0 to \(J - 1\) inclusive, i.e., \([0]/a] + \ldots + [J-1]/a]\). The other function is used for computing the number of nodes in a graph structure like a forest of recursion trees. This is called the forest cardinality and denoted \(\odot^J a K I\). The forest cardinality \(\odot^J a K I\) counts the number of nodes in the forest (described by \(I\)) consisting of \(K\) trees starting from the \(Jth\) node. Nodes are assumed to be numbered in a pre-order fashion. It can be formally defined as in Fig. 11 and is used to count and identify children in the recursion tree of a fix construct.

\[ \odot^J a,0 K = 0 \]
\[ \odot^J a, J+1 K = \odot^J a, J K + (\odot^J a+1, a, J K, J K, J K, J K) K \]

Figure 11: Formal definition of forest cardinality from \cite{9}

The typing judgment is still the same: \(\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau\). However, the definition of \(\Omega\) is now different. The non-affine context \(\Omega\) now carries the constraints on the index variable described on the “\(\sim\)” as in \(x :: a<i \tau\) (Fig. 10). It specifies that there are \(I\) copies of \(x\) with type \(\tau\) in which the free \(a\) is substituted with unique values in the range from 0 to \(I - 1\). The non-affine context also differs in the definition of splitting. The definition of \(+\) (splitting, also referred to as the binary sum) for \(\Omega\) allows for the same variable to be present in the two contexts but by allowing splitting over the index ranges. Binary sum of \(\Omega_1\) and \(\Omega_2\) in \(\lambda\text{-amor}\) was just a disjoint union of the two contexts. However, here in \(\lambda\text{-amor}\), it permits \(\Omega_1\) and \(\Omega_2\) to have common variables but their multiplicities should add up. We also introduce a notion of bounded sum for the non-affine context denoted by \(\sum a<i \Omega\). Both binary and bounded sum over non-affine contexts are described in Fig. 10.

We only describe the type rules for the sub-exponential and the fixpoint in Fig. 12 as these are the only rules that change. T-subExpI is the rule for the introduction form of the sub-exponential. It says that if an
expression $e$ has type $\tau$ under a non-affine context $\Omega$ and $a < I$ s.t. $e$ does not use any affine resources (indicated by an empty $\Gamma$) then $e!$ has type $!_{a< I} \tau$ under context $\sum_{a < I} \Omega$. As before, we can always use the weakening rule to add affine resources to the conclusion. T-subExpE is similar to T-expE defined earlier but additionally it also carries the index constraint coming from the type of $e_j$ in the context for $e_2$.

The fixpoint expression ($fix.x.e$) encodes recursion by allowing $e$ to refer to $fix.x.e$ via $x$. T-fix defines the typing for such a fixpoint construct. It is a refinement of the corresponding rule in Fig. 4. The refinements serve two purposes: 1) they make the total number of recursive calls explicit (this is represented by $\lambda$) and 2) they identify each instance of the recursive call in a pre-order traversal of the recursive tree. This is represented by the index $(b + 1 + \sum_{a < I} a) I$ (representing the $a$th child of the $b$th node in the pre-order traversal). Using these two refinements, the T-fix rule in Fig. 12 can be read as follows: if for all $I$ copies of $x$ in the context we can type check $e$ with $\tau$, then we can also type check the top-most instance of $fix.x.e$ with type $\tau[0/b]$ ($0$ denotes the starting node in the pre-order traversal of the entire recursion tree). Contrast the rules described in Fig. 12 with the corresponding rules for $\lambda$-amor$^*$ described earlier in Fig. 5.

$$
\begin{align*}
\Psi; \Theta; \Delta; a < I; \Omega; \vdash e : \tau \\
\Psi; \Theta; \Delta; \sum_{a < I} \Omega; \vdash !e : !_{a < I} \tau \\
\Psi; \Theta; \Delta; \sum_{a < I} \Omega; \vdash \text{T-subExpI}
\end{align*}
$$

$$
\begin{align*}
\Psi; \Theta; \Delta; \sum_{a < I} \Omega; \vdash e : (l_{a < I} \tau) \\
\Psi; \Theta; \Delta; \sum_{a < I} \Omega; \vdash !e : !_{a < I} \tau \\
\Psi; \Theta; \Delta; \sum_{a < I} \Omega; \vdash \text{T-subExpE}
\end{align*}
$$

$$
\begin{align*}
\Psi; \Theta; \Delta; \sum_{a < I} \Omega; \vdash \text{T-fix}
\end{align*}
$$

We also introduce a new subtyping rule, sub-bSum. sub-bSum helps move the potential from the outside to the inside of a sub-exponential. This is sound because 1) potentials are really ghosts at the term level. Therefore terms of type $[\sum_{a < I} K] !_{a < I} \tau$ and $!_{a < I} [K] \tau$ are both just exponentials and 2) there is only a change in the position but no change of potential in going from $[\sum_{a < I} K] !_{a < I} \tau$ to $!_{a < I} [K] \tau$. We have proved that this new subtyping rule is sound wrt the model of $\lambda$-amor types by proving that if $\tau$ is a subtype of $\tau'$ according to the syntactic subtyping rules then the interpretation of $\tau$ is a subset of the interpretation of $\tau'$. This is formalized in Lemma 8. $\sigma$ and $\iota$ represent the substitutions for the type and index variables respectively.

$$
\begin{align*}
\Psi; \Theta; \Delta; \sum_{a < I} K; !_{a < I} \tau ; !_{a < I} [K] \tau \vdash \text{sub-bSum}
\end{align*}
$$

It is noteworthy that sub-bSum is the only rule in $\lambda$-amor which specifies how the two modalities, namely, the sub-exponential ($!_{a < I} \tau$) and the modal type ($[p] \tau$) interact with each other. People familiar with monads and comonads might wonder, why such an interaction between the sub-exponential and the monad is not required? We believe this is because we can always internalize the cost on the type using the store construct, so just relating exponential and potential modal type suffices. However, studying such interactions could be an interesting direction for future work.

**Lemma 8** (Value subtyping lemma). $\forall \Psi; \Theta; \Delta, \tau \in Type, \tau', \sigma, \iota.$

$$
\begin{align*}
\Psi; \Theta; \Delta; \vdash \tau < \tau' \land, \models \Delta \implies [\tau \sigma] \subseteq [\tau' \sigma \iota]
\end{align*}
$$

### 5.2 Model of types

We only describe the value relation for the sub-exponential here as the remaining cases of the value relation are exactly the same as before. $(p, T, !e)$ is in the value interpretation at type $!_{a < I} \tau$ iff the potential $p$ suffices for all $I$ copies of $e$ at the instantiated types $\tau[i/a]$ for $0 \leq i < I$. The other change to the model is in the interpretation of $\Omega$. This time we have $(p, \delta)$ instead of $(0, \delta)$ in the interpretation of $\Omega$ s.t. $p$ is sufficient for all copies of all variables in the context. The changes to the model are described in Fig. 13.

$$
\begin{align*}
[l_{a < I} \tau] \triangleq \{(p, e) \mid \exists p_0, \ldots, p_{i-1}, p_0 + \ldots + p_{i-1} \leq p \land \forall 0 \leq i < I, (p_i, e) \in [\tau[i/a]|e]\}
\end{align*}
$$

$$
\begin{align*}
[l_{a < I} \tau] \triangleq \{(p, \delta) \mid \exists f : \text{Vars} \rightarrow \text{Indices} \rightarrow \text{Rots.} \land (\forall x. a < I \tau) \in \Omega, 0 \leq i < I, (f x i, \delta(x)) \in [\tau[i/a]|e] \land (\sum_{x < I < I} \sum_{a < I} f x i) \leq p\}
\end{align*}
$$

Figure 13: Changes to the model

We prove the soundness of the model by proving a slightly different fundamental theorem (Theorem 9). There is an additional potential $(p_n)$ coming from the interpretation of $\Omega$ (which was 0 earlier).
Theorem 9 (Fundamental theorem). \( \forall \Psi, \Theta, \Delta, \Omega, \Gamma, e, \tau \in Type, p_i, p_m, \gamma, \delta, \sigma, \iota. \)

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \land (p_i, \gamma) \in \Gamma \sigma \iota \in \Gamma \land (p_m, \delta) \in \Omega \sigma \iota \in \Omega \land \vdash \Delta \iota \implies (p_i + p_m, e \gamma \delta) \in \tau \sigma \iota \]

The proof of the theorem proceeds in a manner similar to that of Theorem 2, i.e., by induction on the typing derivation. Now, in the fix case, we additionally induct on the recursion tree (this also involves generalizing the induction hypothesis to account for the potential of the children of a node in the recursion tree). The Appendix has the entire proof.

6 Embedding \( d\ell PCF \)

In this section we describe an embedding of \( d\ell PCF \) into \( \lambda\text{-amor} \). \( d\ell PCF \) is a \textit{coeffect-based} type system (contrast this with RAML which is an \textit{effect-based} type system) which has been shown to be relatively complete for cost analysis of PCF programs. The objective of this embedding is to show that \( \lambda\text{-amor} \) perform \textit{coeffect-based} cost analysis like \( d\ell PCF \) and hence is relatively complete for PCF too. Additionally, \( \lambda\text{-amor} \) (due to its ability to internalize cost in types) can be seen as a compositional extension of \( d\ell PCF \) (which can analyze whole programs only), as pointed out earlier.

6.1 A brief primer on \( d\ell PCF \)

\( d\ell PCF \) is a call-by-name calculus with an affine type system for doing cost analysis of PCF programs. Terms and types of \( d\ell PCF \) are described in Fig. 14. \( d\ell PCF \) works with the standard PCF terms but refines the standard types of PCF a bit to perform cost analysis. The type of natural numbers is refined with two indices \( Nat[I, J] \) to capture types for natural numbers in the range \([I, J]\) specified by the indices. Function types are refined with index constraints in the negative position. For instance, \([a < I] \tau_1 \rightarrow \tau_2 \) is the type of a function which when given \( I \) copies of an expression (since \( d\ell PCF \) is call-by-name) of type \( \tau_1 \) will produce a value of type \( \tau_2 \). The \([a < I] \) acts both as a constraint on what values \( a \) can take and also as a binder for free occurrence of \( a \) in \( \tau_1 \) (but not in \( \tau_2 \)). \([a < I] \tau_1 \rightarrow \tau_2 \) is morally equivalent to \((\tau_1[i/a] \otimes \ldots \otimes \tau_1[I + I/a]) \rightarrow \tau_2 \).

\[
\begin{align*}
dPCF \text{ terms} & : t ::= n \ | \ s(t) \ | \ p(t) \ | \ ifz t then u else v \ | \ \lambda x.t \ | \ tu \ | \ fix x.t \\
dPCF \text{ types} & : \sigma ::= Nat[I,J] \ | \ A \rightarrow \sigma \\
A & ::= [a < I] \sigma
\end{align*}
\]

Figure 14: \( d\ell PCF \)’s syntax of terms and types from [9].

The typing judgment of \( d\ell PCF \) is given by \( \Theta; \Delta; \Gamma \vdash_{C} : \tau \). \( \Theta \) denotes a context of index variables, \( \Delta \) denotes a context for index constraints, \( \Gamma \) denotes a context of term variables and \( C \) denotes the cost of evaluation of \( e_d \). This cost \( C \) is the number of variable lookups in a full execution of \( e_d \). \( \xi \) on the turnstile denotes an equational program used for interpreting the function symbols of the index language. Like in the negative position of the function type, multiplicities also show up with the types of the variables in the typing context. The typing rules are designed to track these multiplicities (which is a coeffect in the system). For illustration, we only show the typing rule for function application in Fig. 15. Notice how the cost in the conclusion is lower bounded by the sum of: a) the number of times the argument of \( e_1 \) can be used by the body, i.e., \( I \), b) the cost of \( e_1 \), i.e., \( J \) and c) the cost of \( I \) copies of \( e_2 \), i.e., \( \sum_{a < I} K \). The authors of [9] show that this kind of coeffect tracking in the type system actually suffices to give an upper-bound on the cost of execution on a \( K_{PCF} \) machine, a Krivine-style machine [25] for PCF.

\[
\begin{align*}
\Theta; \Delta; \Gamma \vdash_{C} e_1 : ([a < I], \tau_1) \rightarrow \tau_2 \\
\Theta; a; \Delta; a < I; \Delta + K e_2 : \tau_1 \\
\Gamma' \sqsupseteq \Gamma + \sum_{a < I} \Delta \\
H \geq I + J + \sum_{a < I} K
\end{align*}
\]

\( \Theta; \Delta; \Gamma' \vdash_{H} e_1 : \tau_2 \)

Figure 15: Typing rule for function application from [9].

States of the \( K_{PCF} \) machine consist of triples of the form \((t, \rho, \theta)\) where \( t \) is a \( d\ell PCF \) term, \( \rho \) is an environment for variable binding and \( \theta \) is stack of closures. A closure (denoted by \( C \)) is simply a pair consisting of a term and an environment. The left side of Fig. 16 describes some evaluation rules of the \( K_{PCF} \) machine from [9]. For instance, the application triple \((e_1 e_2, \rho, \theta)\) reduces in one step to \( e_1 \), and \( e_2 \) along with the current closure is pushed on top of the stack for later evaluation. This is how one would expect an evaluation to happen in a call-by-name scheme. One final ingredient that we need to describe for the soundness of \( d\ell PCF \) is a notion of the size of a term, denoted by \([t]\). The size of a \( d\ell PCF \) term is defined in [9] (we describe some of the clauses on the right side of Fig. 16).

Finally \( d\ell PCF \) soundness (Theorem 10) states that the execution cost (denoted by \( n \)) is upper-bounded by the product of the size of the initial term, \( t \) and \((I + 1)\). \( d\ell PCF \) states the soundness result for base (bounded naturals) types only and soundness for functions is derived as a corollary. \( \downarrow^n \) is a shorthand for \( n \rightarrow \) (\( n \)-step closure under the \( K_{PCF} \) reduction relation).
\((e_1, e_2, \rho, \theta) \rightarrow (e_1, \rho, (e_2, \rho, \theta))\) \hspace{1cm} |x| = 1
\((\lambda x. e, \rho, C, \theta) \rightarrow (e, \rho, C, \theta)\) \hspace{1cm} |x| = 1
\((x, (t_0, \rho_0), \ldots, (t_n, \rho_n), \theta) \rightarrow (t_0, \rho_0, \ldots, t_n, \rho_n, \theta)\) \hspace{1cm} |x| = 1 + |e| + 1
\((\text{fix} x. e, \rho, \theta) \rightarrow (e, (\text{fix} x. e, \rho, \theta))\) \hspace{1cm} |x| = 1 + 1

\(\top_i t : \text{Nat}[J,K] \land t \downarrow^m \implies n \leq |t| \times (I + 1)\)

### 6.2 Type-directed translation of \(dlfPCF\) into \(\lambda\)-amor

Without loss of generality, as in RAML’s embedding, we abstract the type of naturals and treat them as a general abstract base type \(b\). Like RAML, \(dlfPCF\)’s embedding is also type directed. The type translation function is described in Fig. 17, \(dlfPCF\)’s base type is translated into the base type of \(\lambda\)-amor. The function type \((a \,<\, I)\) translates to a function which takes \(I\) copies of the monadic translation of \(\tau\) (following Moggi [28]) and \(I\) units of potential (to account for \(I\) substitutions during application) as a modal unit type, and returns a monadic type of translation of \(\tau_2\). The monad on the return type is essential as a function cannot consume \(I\) units of potential and still return a pure value. The translation of the typing context is defined pointwise for every monadic type in the context. Since all variables in the \(dlfPCF\)’s typing context have comonadic types (carrying multiplicities), \(dlfPCF\)’s typing context is translated into the non-affine typing context of \(\lambda\)-amor.

The translation judgment is of the form \(\Theta; \Delta; \Gamma \vdash J \bot \sigma[a/\theta] : \tau \rightsquigarrow e_{\lambda}b\), where \(e_{\lambda}b\) denotes the translated \(\lambda\)-amor term. \(\xi\) never changes in any of \(dlfPCF\)’s typing rules, so for simplification we assume it to be present globally and thus we omit it from the translation judgment. The expression translation of \(dlfPCF\) terms is defined by induction on typing judgments (Fig. 18). Notice that in the variable rule (var) we place a deliberate tick construct which consumes one unit of potential. This is done to match the cost model of \(dlfPCF\). Without this accounting our semantics and cost preservation theorem would not hold. The translation of function application and the fixpoint construct make use of a coercion function (\(coerce\), which is written in \(\lambda\)-amor itself). It helps convert an application of exponentials into an exponential of an application. The coercion function is described in the box along with the expression translation rules in Fig. 18.

\[\begin{align*}
\Theta; \Delta \vdash J \geq 0 & \quad \Theta; \Delta \vdash I \geq 1 & \quad \Theta; \Delta \vdash \sigma[a/\theta] : \tau & \quad \Theta; \Delta \vdash |a < I|\sigma[\theta] = p & \quad \Theta; \Delta \vdash |a < I|\gamma = 0 & \\
\Theta; \Delta; \Gamma, x : |a < I|\tau \vdash \lambda \rho.release = e in bind = \uparrow^m in x & \quad \text{lam} & \\
\Theta; \Delta; \Gamma, x : |a < I|\tau \vdash \lambda x.e : (|a < I|\tau_1) \rightsquigarrow \tau_2 \rightsquigarrow \text{let } x = y in \tau_1 in \tau_2 in \text{let } x = y in \text{release } = \text{p}_1 in \text{release } = \text{p}_2 in \text{bind } a = \text{store}(e) in e_1 a & \\
\Theta; \Delta; \Gamma, x : |a < I|\tau \vdash \text{app} & \\
\Theta; \Delta; \Gamma, x : |a < I|\tau \vdash \text{T-fix} & \\
\Theta; \Delta; \Gamma, x : |a < I|\tau \vdash \text{fix}\ x.e in \text{mu} on \text{mu} = \text{E}_0
\end{align*}\]

\[\begin{align*}
E_0 = \text{fixY} & \quad \text{coerce F X = } \text{let } f = \text{f in let } x = \text{X in f(x)} & \\
E_1 = \text{bind A} & \quad \text{bind C} in \text{let } x = (\text{E}_{21}, \text{E}_{22}) in \text{bind } C = \text{store}(e) in e_1 C & \\
E_{21} = \text{coerce } Y & \quad E_{22} = (\lambda u.()\ A) & \\
\end{align*}\]

Figure 18: Expression translation: \(dlfPCF\) to \(\lambda\)-amor

We want to highlight another point about this translation. This is the second instance (the first one was embedding of Church numerals, Section 3.1) where embedding using just a cost monad (without potentials) does not seem to work. To understand this, let us try to translate \(dlfPCF\)’s function type \((a < I)\) using only the cost monad and without the potentials. One possible translation of \(|a < I|\tau_1 \rightsquigarrow \tau_2\) is \((|a < I|\theta) \rightsquigarrow \text{M}(\theta)\).
The \( I \) in the monadic type is used to account for the cost of substitution of the \( I \) copies of the argument in \( d\text{PCF} \). Now, in the rule for function abstraction we have to generate a translated term of the type \( \mathsf{M}(\llbracket J + \text{count}(\Gamma) \rrbracket (a < I) \tau_1 \Rightarrow \mathsf{M} I \tau_2) \). From the induction hypothesis, we have a term of type \( \mathsf{M}(I + J + \text{count}(\Gamma)) \tau_2 \). A possible term translation can be \( \text{ret} x \), let \( ! x = y \) in \( e_I \). This would require us to type \( e_I \) with \( \mathsf{M} I \tau_2 \) under the given context with a free \( x \). However, \( e_I \) can only be typed with \( \mathsf{M}(I + J + \text{count}(\Gamma)) \tau_2 \) (which cannot be coerced to the desired type). Hence, the translation with just cost monads does not work. We believe that such a translation can be made to work by adding appropriate coercion axioms for the cost monads.

However, there is an alternate way to make this translation work, using the modal type and that is what we use. The idea is to capture the \( I \) units as a potential using the modal type of \( \lambda\text{-amor} \) (in the negative position) instead of capturing it (in the positive position) as a cost on the monad. Concretely, this means that, instead of translating \( [a < I] \tau_1 \Rightarrow \mathsf{M} I \tau_2 \), we translate it to \( (a < I) \mathsf{M} 0 \| \tau_1 \Rightarrow [I] \mathsf{M} 0 \| \tau_2 \) (as described in Fig. 17 earlier). Likewise, the typing judgment is also translated using the same potential approach (as described in Theorem 11). Following this approach, we obtain a term of type \( (J + I + \text{count}(\Gamma)) \mathsf{M} 0 \| \tau_2 \) from the induction hypothesis and we are required to produce a term of type \( [J + \text{count}(\Gamma)] \mathsf{M} 0 \| \tau_2 \) from \( \text{count}(\tau_1) \) potential as input (in the argument position of the translated type): \( I \) accounts for the substitutions coming from function applications in the \( d\text{PCF} \) expression and \( \text{count}(\Gamma) \) accounts for the total number of possible substitutions of context variables. All translated expressions release the input potential coming from the argument. This is later consumed using a tick as in the variable rule or stored with a unit value to be used up by the induction hypothesis.

We prove that this translation is type preserving (Theorem 11). In particular, we show that the translated term can be typed at the function type \( [I + \text{count}(\Gamma)] \mathsf{M} 0 \| \tau_2 \), where count is defined as \( \text{count}(\Gamma, x : \llbracket a < I \rrbracket) = \text{count}(\Gamma) + I \) (with \( \text{count}(\llbracket \rrbracket) = 0 \) as the base case). Since \( d\text{PCF} \) counts cost for each variable lookup in a terminating \( K_{PCF} \) reduction, the translated term must have enough potential to make sure that all copies of free variables in the context can be used. This is ensured by having \( (I + \text{count}(\Gamma)) \) potential as input (in the argument position of the translated type): \( I \) accounts for the substitutions coming from function applications in the \( d\text{PCF} \) expression and \( \text{count}(\Gamma) \) accounts for the total number of possible substitutions of context variables.

3.6 Semantic properties of the translation

Besides type preservation we also prove semantics and cost preservation for the translation. To achieve that, we define a cross-language relation between \( d\text{PCF} \) terms and \( \lambda\text{-amor} \) using the properties of the translation only. But \( d\text{PCF} \)’s soundness is defined wrt reduction on a \( K_{PCF} \) machine [23], as described earlier. So, we would like to re-derive a proof of Theorem 12.

Theorem 11 (Type preservation: \( d\text{PCF} \) to \( \lambda\text{-amor} \)). If \( \Theta; \Delta; \Gamma \vdash_I e : \tau \text{ in } d\text{PCF} \) then there exists \( e' \) such that \( \Theta; \Delta; \Theta; \Delta; \Gamma \vdash e' : \tau' \text{ such that there is a derivation of } \vdash : \Theta; \Delta; \Theta; \Delta; \Gamma : \vdash e' : (I + \text{count}(\Gamma)) \mathsf{M} 0 \| \tau_2 \) in \( \lambda\text{-amor} \).

6.3 Semantic properties of the translation

To prove Theorem 12 we need a way of relating \( K_{PCF} \) triples to \( \lambda\text{-amor} \) terms. So, we come up with an approach for decomposing \( K_{PCF} \) triples into \( d\text{PCF} \) terms (which we can then transitively relate to \( \lambda\text{-amor} \) terms via our translation). The decomposition is defined as a function (denoted by \( \llbracket \rrbracket \)) from Krivine triples to \( d\text{PCF} \) terms. We first define decomposition for closures (\( \llbracket \rrbracket \) is overloaded), by induction on the environment. For an empty environment the decomposition is simply an identity on the given term. And for an environment of the form \( c_1, \ldots, c_n \), the decomposition is given by closing off the open parts of the given term. Direct substitution of closures in \( e \) won’t remain cost-preserving. So instead, we decompose it using lambda abstraction and application as described on the left side of Fig. 19. Using this closure decomposition, we define decomposition for the full Krivine triples. When stack is empty, it is just the decomposition of the underlying closure. And when stack is non-empty, the closures on the stack are applied one after the other on the closed term obtained via the transition of the closure. This is described on the right side of Fig. 19. We prove that the decomposition preserves type, cost and semantics of the Krivine triple.

\[ \text{This is a generalized version of } d\text{PCF} \text{’s soundness (Theorem 10), where we prove the cost bound for terms of arbitrary types.} \]
\[
\begin{align*}
\langle e, [] \rangle & \triangleq e \\
\langle e, C_1, \ldots, C_n \rangle & \triangleq (\lambda x_1 \ldots x_n . e) \langle C_1 \rangle \ldots \langle C_n \rangle
\end{align*}
\]

Figure 19: Decompilation of closure (left) and Krivine triple (right)

Finally, we compose the decompilation of Krivine triples to \(d\text{-PCF}\) terms with the translation of \(d\text{-PCF}\) to \(\lambda\text{-amor}\) terms to obtain a composite translation from Krivine triples to \(\lambda\text{-amor}\). We then prove that this composite translation preserves the meaning of cost annotations wrt to the intentional soundness criteria stated in Theorem 12. The proof is quite involved, but due to lack of space we cannot get into the technicalities of that proof.

7 Related work

Literature on cost analysis is very vast; we summarize and compare to only a representative fraction, covering several prominent styles of cost analysis.

Type and effect systems. Several type and effect system have been proposed for amortized analysis using the method of potentials. Early approaches \[23\] \[22\] allow the potential associated with a value to only be a linear function of the value’s size. Univariate RAML \[19\] generalizes this to polynomial potentials. Multivariate RAML \[17\] is a substantial generalization where a single potential, that is a polynomial of the sizes of multiple input variables, can be associated to all of them together. These approaches are inherently first-order in their treatment of potentials—closures that capture potentials are disallowed by the type systems. We already showed how to embed Univariate RAML in \(\lambda\text{-amor}\) in Section 4 and we believe that the embedding can be extended to Multivariate RAML.

AARA \[21\] \[18\] extends RAML with limited support for closures and higher-order functions. \[18\] cannot handle Curry-style functions at all, while \[21\] can handle Curried functions only when the potential is associated with the last argument. As explained in Section 1 these limitations arise from incomplete support for affineness. In contrast, \(\lambda\text{-amor}\), being affine, does not have such limitations.

Some prior work such as the unary fragment of \[6\] uses effect-based type systems for non-amortized cost analysis. A significant line of work tracing lineage back to at least \[8\] uses sized types and cost represented in a writer monad for cost analysis. More recently, \[12\] \[3\] \[24\], show how to extend this idea to extract sound recurrences from programs. These recurrences can be solved to establish cost bounds. However, this line of work does not support potentials or amortized analysis. Conceptually, it is simpler than the above-mentioned work on amortized cost analysis (it corresponds to RAML functions where the input and output potentials are both 0) and, hence, can be easily simulated in \(\lambda\text{-amor}\).

Cost analysis using program logics. As an alternative to type systems, a growing line of work uses variants of Hoare logic for amortized cost analysis \[1\] \[5\] \[27\]. The common idea is to represent the potential before and after the execution of a code segment as ghost state in the pre- and post-condition of the segment, respectively. Conceptually, this idea is not very different from how we encode potentials using our \([p] \tau\) construct in the inputs and outputs of functions (e.g., in embedding RAML in Section 3). However, unlike \(\lambda\text{-amor}\), prior work shows neither embeddings of existing frameworks, nor any (relative) completeness result. \[24\] introduce a new concept called time receipts, which can be used for lower-bound analysis, something that \(\lambda\text{-amor}\) does not support yet.

We note that we could have developed \(\lambda\text{-amor}\) and showed our embeddings and the relative completeness result using a program logic in place of a type theory. For our purposes, the difference between the two is only a matter of personal preference.

Cost analysis of lazy programs. Some prior work \[11\] \[26\] \[23\] develops methods for cost analysis of lazy programs. While the semantics of laziness, as in call-by-need, cannot be directly embedded in \(\lambda\text{-amor}\), we can replicate the analysis of some lazy programs, with nearly identical potentials, in the call-by-name setting of \(\lambda\text{-amor}\). We already showed an example of this by verifying Okasaki’s implicit queue in Section 3.3, replicating an analysis by \[11\]. We believe that other examples of Okasaki \[30\] can be replicated in \(\lambda\text{-amor}\) (in a call-by-name setting).

An interesting aspect is that amortized analysis of lazy programs does not necessarily require affineness. \[11\] circumvents the issue related to duplication of potentials by representing potential using the same indexed monad that represents cost. To represent offsetting of cost by potential, he introduces a primitive coercion “pay” of type \(M(\kappa_1 + \kappa_2) \tau \rightarrow M\kappa_1 (M\kappa_2 \tau)\), which, in a way, encodes paying \(\kappa_1\) part of the cost \(\kappa_1 + \kappa_2\) using potential from outside. An interesting question is whether we could use the same idea and do away with our construct for potentials. It turns out that our construct for potentials satisfies additional properties that are needed for embedding \(d\text{-PCF}\). In particular, making \(d\text{-PCF}\)’s embedding work in Danielsson’s style requires a different coercion of type \((\tau' \rightarrow M(\kappa_1 + \kappa_2) \tau) \rightarrow M\kappa_1 (\tau' \rightarrow M\kappa_2 \tau)\).

Coeffect-based cost analysis. \(d\text{-PCF}\) \[9\] and \(d\text{-PCFv}\) \[10\] are coeffect-based type systems for non-amortized cost analysis of PCF programs in the call-by-name and call-by-value settings, respectively. Both systems count the number of variable lookups during execution on an abstract machine (the Krivine machine for call-by-name and the CEK machine for call-by-value \[25\] \[13\]). This is easily done by tracking (as a coeffect) the number of
uses of each variable in an affine type system with dependent exponentials (λ-amor borrows this exponentials, but does not use coeffects for tracking cost). A common limitation is that these type systems cannot internalize the cost of a program into its type; instead the cost is a function of the typing derivation. We showed in Section 5 that λ-amor can embed dPCF and internalize its costs into types. Hence, λ-amor advances beyond dPCF. We expect that λ-amor can also embed dPCF_V, but have not tried this embedding yet.

A.1 Syntax

Expressions

\[ c ::= v | e_1 e_2 | \langle e_1, e_2 \rangle | \text{let} (x, y) = e_1 \text{ in } e_2 | (c, e) | \text{fst}(e) | \text{snd}(e) | \text{inl}(e) | \text{inr}(e) | \text{case } e, x.e, y.e | \text{let } !x = e_1 \text{ in } e_2 | e :: e | e [] | e; x.e \]

Values

\[ v ::= x | () | c | \lambda x.e | \langle v_1, v_2 \rangle | (v, v) | \text{inl}(e) | \text{inr}(e) | !e | \text{nil} | \Lambda.e | \text{ret } e | \text{bind } x = e_1 \text{ in } e_2 | \uparrow | \text{release } x = e_1 \text{ in } e_2 | \text{store } e \]

(No value forms for \( I \tau \))

Index

\[ I ::= i | N | R | I + I | I - I | \lambda i : S.I | I I \]

Sort

\[ S ::= N | R^+ | S \rightarrow S \]

Kind

\[ K ::= T_{\text{type}} | S \rightarrow K \]

Types

\[ \tau ::= \bot | b | \tau_1 \rightarrow \tau_2 | \tau_1 \times \tau_2 | \tau_1 \& \tau_2 | \tau_1 \lor \tau_2 | \tau I | ! \tau | [I] \tau | \mathbb{M} I \tau | L^I \tau \]

\[ \alpha | \forall \alpha : K.\tau | \forall i : S.\tau | \lambda i : S.\tau | \tau I | \exists i : S.\tau | c \Rightarrow \tau | c & \tau \]

Constraints

\[ c ::= I = I | I < I | c \wedge c \]

Lin. context

\[ \Gamma ::= . | \Gamma, x : \tau \]

for term variables

Unres. context

\[ \Omega ::= . | \Omega, x : \tau \]

for term variables

Unres. context

\[ \Theta ::= . | \Theta, i : S \]

for index variables

Unres. context

\[ \Psi ::= . | \Psi, \alpha : K \]

for type variables

Definition 13 (Binary sum of multiplicity context).

\[ \Omega_1 \oplus \Omega_2 \triangleq \begin{cases} \Omega_1 & \Omega_1 = . \\ \Omega_2 & \Omega_2 = . \\ \Omega_1 = \Omega_1'_{\bot} & X \in \Omega_2 \\ \Omega_1 = \Omega_1'_{\bot} & X \in \Omega_2 \\ \text{undefined} & \text{otherwise} \end{cases} \]

Definition 14 (Binary sum of affine context).

\[ \Gamma_1 \oplus \Gamma_2 \triangleq \begin{cases} \Gamma_2 & \Gamma_1 = . \\ \Gamma_1 & \Gamma_2 = . \\ \Gamma_1 = \Gamma_1'_{\bot} & X \in \Gamma_2 \\ \Gamma_1 = \Gamma_1'_{\bot} & X \in \Gamma_2 \\ \text{undefined} & \text{otherwise} \end{cases} \]
A.2 Typesystem

<table>
<thead>
<tr>
<th>Typing $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash x : \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{match } e \text{ with }</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \exists s. \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau, e_2 : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x.e : (\tau_1 \to \tau_2)$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{let } \langle x, y \rangle = e \text{ in } e' : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash c : b$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{nil} : L\tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{match } e \text{ with }</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \exists s. \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau, e_2 : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x.e : (\tau_1 \to \tau_2)$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{let } \langle x, y \rangle = e \text{ in } e' : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash c : b$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{nil} : L\tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{match } e \text{ with }</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \exists s. \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau, e_2 : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x.e : (\tau_1 \to \tau_2)$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{let } \langle x, y \rangle = e \text{ in } e' : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash c : b$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{nil} : L\tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{match } e \text{ with }</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \exists s. \tau$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau, e_2 : \tau'$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x.e : (\tau_1 \to \tau_2)$</td>
</tr>
<tr>
<td>$\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{let } \langle x, y \rangle = e \text{ in } e' : \tau'$</td>
</tr>
</tbody>
</table>
\[
\begin{align*}
\text{T-fst} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (\tau_1 \& \tau_2) \\
\text{T-snd} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{snd}(e) : \tau_2 \\
\text{T-inl} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau_1 \\
\text{T-inr} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{inr}(e) : \tau_1 \oplus \tau_2 \\
\text{T-case} & : \Psi; \Theta; \Delta; \Omega; \Gamma_1 \oplus \Gamma_2 \vdash \text{case } e, x, e_1, y, e_2 : \tau \\
\text{T-Exp} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash !\tau \\
\text{T-tabs} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda e : (\forall \alpha : K. \tau) \\
\text{T-fix} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{fix } e : \tau \\
\text{T-ret} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{ret } e : M \theta \\
\text{T-bind} & : \Theta ; \Delta ; I : \tau^+ ; \Theta ; \Delta ; I_1 : \tau^+ ; \Theta ; \Delta ; I_2 : \tau^+ \\
\text{T-tick} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash I^! : M I^! \\
\text{T-release} & : \Psi; \Theta; \Delta; \Omega; \Gamma_1 \oplus \Gamma_2 \vdash \text{release } x = e_1 \text{ in } e_2 : M(I_1 + I_2) \tau_2 \\
\text{T-store} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{store } e : M I ([I] \tau) \\
\text{T-CE} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \\
\text{T-AndI} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (c \& \tau) \\
\text{T-AndE} & : \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (c \& \tau) \rightarrow \text{let } x = e \text{ in } e' : \tau' \\
\end{align*}
\]
\[
\begin{align*}
\text{sub-refl} & : \Psi; \Theta; \Delta \vdash \tau \prec \tau' \\
\text{sub-arrow} & : \Psi; \Theta; \Delta \vdash \tau_1 \prec \tau_2 \\
\text{sub-tensor} & : \Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 \prec \tau'_1 \otimes \tau'_2 \\
\text{sub-sum} & : \Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 \prec \tau'_1 \oplus \tau'_2 \\
\text{sub-monad} & : \Psi; \Theta; \Delta \vdash \tau \prec \tau' \\
\text{sub-potential} & : \Psi; \Theta; \Delta \vdash \tau \prec \tau' \\
\text{sub-list} & : \Psi; \Theta; \Delta \vdash L^n \tau \prec L^n \tau' \\
\text{sub-typePoly} & : \Psi; \Theta; \Delta \vdash \forall \alpha. \tau_1 \prec \forall \alpha. \tau_2 \\
\text{sub-constraint} & : \Psi; \Theta; \Delta \vdash \tau_1 \prec \tau_2 \\
\text{sub-mBase} & : \Psi; \Theta; \Delta \vdash \Omega \subseteq \cdot \\
\text{sub-mInd} & : x : \tau' \in \Omega_1, \Psi; \Theta; \Delta \vdash \tau' \prec \tau, \Psi; \Theta; \Delta \vdash \Omega_1/x \subseteq \Omega_2, x : \tau \\
\text{sub-lBase} & : \Psi; \Theta; \Delta \vdash \Gamma \subseteq \cdot \\
\text{sub-lBase} & : x : \tau' \in \Gamma_1, \Psi; \Theta; \Delta \vdash \tau' \prec \tau, \Psi; \Theta; \Delta \vdash \Gamma_1/x \subseteq \Gamma_2, x : \tau \\
\text{S-var} & : \Theta, i : S \vdash i : S \\
\text{S-nat} & : \Theta, \Delta \vdash N : N \\
\text{S-real} & : \Theta, \Delta \vdash R : R^+ \\
\text{S-real1} & : \Theta, \Delta \vdash i : N \\
\text{S-add-Nat} & : \Theta, \Delta \vdash I_1 : \mathbb{N}, \Theta, \Delta \vdash I_2 : \mathbb{N}, \Theta, \Delta \vdash I_1 + I_2 : \mathbb{N} \\
\text{S-add-Real} & : \Theta, \Delta \vdash I_1 : R^+, \Theta, \Delta \vdash I_2 : R^+, \Theta, \Delta \vdash I_1 + I_2 : R^+ \\
\text{S-minus-Real} & : \Theta, \Delta \vdash I_1 : R^+, \Theta, \Delta \vdash I_2 : R^+, \Theta, \Delta \vdash I_1 - I_2 : R^+ \\
\text{S-family} & : \Theta, i \vdash S, \Theta, \Delta \vdash i : S' \\
\text{S-family} & : \Theta, \Delta \vdash \lambda_i i : S \rightarrow S \\
\end{align*}
\]
\[ \frac{\Psi; \Theta; \Delta \vdash \mathbf{1} : Type}{\Psi; \Theta; \Delta \vdash \mathbf{1} : K} \quad \text{K-unit} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \mathbf{b} : Type}{\Psi; \Theta; \Delta \vdash \mathbf{b} : K} \quad \text{K-base} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau : K}{\Psi; \Theta; \Delta \vdash I : S} \quad \text{K-List} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau_1 : K \quad \Psi; \Theta; \Delta \vdash \tau_2 : K}{\Psi; \Theta; \Delta \vdash \tau_1 \rightarrow \tau_2 : K} \quad \text{K-arrow} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau_1 : K \quad \Psi; \Theta; \Delta \vdash \tau_2 : K}{\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 : K} \quad \text{K-with} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau : K}{\Psi; \Theta; \Delta \vdash !\tau : K} \quad \text{K-Exp} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau : K \quad \Theta; \Delta \vdash I : \mathbb{R}^+}{\Psi; \Theta; \Delta \vdash M\, I\, \tau : K} \quad \text{K-monad} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau : K \quad \Theta; \Delta \vdash I : \mathbb{R}^+}{\Psi; \Theta; \Delta \vdash M\, I\, \tau : K} \quad \text{K-family} \]

\[ \frac{\Psi; \Theta; \Delta \vdash \tau : K}{\Psi; \Theta; \Delta \vdash c \vdash \tau : K} \quad \text{K-constraint} \]
\[ \frac{\Psi; \Theta; \Delta \vdash c \vdash \tau : K}{\Psi; \Theta; \Delta \vdash c \& \tau : K} \quad \text{K-consAnd} \]
\[ \frac{\Psi; \Theta; \Delta \vdash \tau : S \rightarrow K \quad \Theta; \Delta \vdash I : S}{\Psi; \Theta; \Delta \vdash \lambda_i.\tau : S \rightarrow K} \quad \text{K-iapp} \]

Figure 25: Kind rules for types
A.3 Semantics

<table>
<thead>
<tr>
<th>Pure reduction, $e \Downarrow_t v$</th>
<th>Forcing reduction, $e \Downarrow_t^\nu v$</th>
</tr>
</thead>
</table>

- $e_1 \Downarrow_t l \quad e_2 \Downarrow_t l$ E-cons
- $e_1 :: e_2 \Downarrow_t l \quad v :: l$ E-cons
- $e_1 \Downarrow_t v_1 \quad e_2 \Downarrow_t v_2$ E-match
- $e_1 \Downarrow_t \text{nil} \quad e_2 \Downarrow_t v$ E-matchNil
- $e_1 \Downarrow_t v_1 \quad e_2[v/x] \Downarrow_t v'$ E-exist

- $e_1 \Downarrow_t l \quad e_2 \Downarrow_t [v/h][t/l] \Downarrow_t v$ E-matchCons
- $e_1 \Downarrow_t \text{nil} \quad e_2 \Downarrow_t [h :: t \mapsto e_3] \Downarrow_t v$ E-matchCons

- $e \Downarrow_t \lambda x.e' \quad e'[e_2/x] \Downarrow_t v'$ E-app
- $e \Downarrow_t (e_1,e_2) \Downarrow_t v_1 \Downarrow_t v_2$ E-TI

- $\text{let}(x,y) = e \text{ in } e' \Downarrow_t \Downarrow_t v_1$ E-TE
- $\text{let} x = e_1 \text{ in } e_2 \Downarrow_t v_2$ E-WI
- $\text{let} (v_1,v_2) = e \Downarrow_t \Downarrow_t v_1 \Downarrow_t v_2$ E-Fst

- $e \Downarrow_t \text{in}(v) \quad e'[v/x] \Downarrow_t v'$ E-case1
- $e \Downarrow_t \text{in}(v) \quad e''[v/y] \Downarrow_t v''$ E-case2

- $\text{let} ! x = e \text{ in } e'[v/x] \Downarrow_t v$ E-fix

- $v \in \{0,1\}, \text{bind } x = e_1 \text{ in } e_2, \text{ release } x = e_1 \text{ in } e_2, \text{ store } e \mapsto v$ E-val

- $e \Downarrow_t \lambda x.e' \quad e'[v/x] \Downarrow_t v'$ E-fix

- $e \Downarrow_t l \quad e'[e_2/x] \Downarrow_t v'$ E-CE

- $e \Downarrow_t l \quad e'[e_2/x] \Downarrow_t v'$ E-CE

- $\text{let} x = e_1 \text{ in } e_2 \Downarrow_t v_2 \Downarrow_t v'$ E-CandE

- $e \Downarrow_t l \quad e'[e_2/x] \Downarrow_t v'$ E-CE

- $e \Downarrow_t l \quad e'[e_2/x] \Downarrow_t v'$ E-CE

- $e \Downarrow_t l \quad e'[e_2/x] \Downarrow_t v'$ E-CE

- $e \Downarrow_t l \quad e'[e_2/x] \Downarrow_t v'$ E-CE

Figure 26: Evaluation rules: pure and forcing
A.4 Model

Definition 16 (Interpretation of typing contexts).

\[ \begin{align*}
[\Gamma]_\varepsilon & \triangleq \{(p, T, \gamma) \mid \exists f : \text{Vars} \to \text{Pots}, \\
& \quad (\forall x \in \text{dom}(\Gamma). f(x), T, \gamma(x)) \in [\Gamma(x)]_\varepsilon \land \sum_{x \in \text{dom}(\Gamma)} f(x) \leq p\} \\
[\Omega]_\varepsilon & \triangleq \{(0, T, \delta) \mid (\forall x \in \text{dom}(\Omega). 0, T, \delta(x)) \in [\tau]_\varepsilon\}
\end{align*} \]

Definition 17 (Type and index substitutions). \( \sigma : \text{TypeVar} \to \text{Type}, \iota : \text{IndexVar} \to \text{Index} \)

Lemma 18 (Value monotonicity lemma). \( \forall p, p', v, \tau. (p, T, v) \in [\tau] \land p \leq p' \land T' \leq T \implies (p', T', v) \in [\tau] \)

Proof. Proof by induction on \( \tau \)

Lemma 19 (Expression monotonicity lemma). \( \forall p, p', v, \tau. (p, T, e) \in [\tau]_\varepsilon \land p \leq p' \land T' \leq T \implies (p', T', e) \in [\tau]_\varepsilon \)

Proof. From Definition [15] and Lemma [61]

Theorem 20 (Fundamental theorem). \( \forall \Theta, \Omega, \Gamma, e, \tau, p_1, \gamma, \delta, \sigma, \iota. \)

\[ \Psi ; \Theta ; \Delta ; \Omega ; \Gamma \vdash e : \tau \land (p_1, T, \gamma) \in [\Gamma \sigma_1]_\varepsilon \land (0, T, \delta) \in [\Omega \sigma_1]_\varepsilon \implies (p_1, T, e \gamma \delta) \in [\tau \sigma_1]_\varepsilon. \]

Proof. Proof by induction on the typing judgment

1. T-var1:

\[ \Psi ; \Theta ; \Delta ; \Omega ; \Gamma, x : \tau \vdash x : \tau \quad \text{T-var1} \]

Given: \( (p_1, T, \gamma) \in [\Gamma, x : \tau \sigma_1]_\varepsilon \) and \( (0, T, \delta) \in [\Omega \sigma_1]_\varepsilon \)

To prove: \( (p_1, T, x \delta \gamma) \in [\tau \sigma_1]_\varepsilon \)

Since we are given that \( (p_1, T, \gamma) \in [\Gamma, x : \tau \sigma_1]_\varepsilon \) therefore from Definition [16] we know that \( \exists f. (f(x), T, \gamma(x)) \in [\tau \sigma_1]_\varepsilon \) where \( f(x) \leq p_1 \)

Therefore from Lemma [62] we get \( (p_1, T, x \delta \gamma) \in [\tau \sigma_1]_\varepsilon \)

2. T-var2:

\[ \Psi ; \Theta ; \Delta ; \Omega, x : \tau \vdash \Gamma, x : \tau \quad \text{T-var2} \]

Given: \( (p_1, T, \gamma) \in [\Gamma, \sigma_1]_\varepsilon \) and \( (0, T, \delta) \in [(\Omega, x : \tau \sigma_1]_\varepsilon \)

To prove: \( (p_1, T, x \delta \gamma) \in [\tau \sigma_1]_\varepsilon \)
Since we are given that \((0, T, \delta) \in \llbracket (\Omega, x : \tau) \sigma_i \rrbracket_c\) therefore from Definition 16 we know that \((0, T, \delta(x)) \in \llbracket \tau \sigma_i \rrbracket_c\).

Therefore from Lemma 62 we get \((p_l, T, x \delta \gamma) \in \llbracket \tau \sigma_i \rrbracket_c\)

3. T-unit:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash () \triangleright T\text{-unit}
\]

Given: \((p_l, T, \gamma) \in \llbracket \Gamma \sigma_i \rrbracket_c\), \((0, T, \delta) \in \llbracket \Omega \sigma_i \rrbracket_c\)

To prove: \((p_l, T, c) \in \llbracket b \rrbracket_c\)

From Definition 15 it suffices to prove that

\[
\forall T' < T, v \cdot (\cdot) \downarrow_T v' \implies (p_l, T - T', v') \in [1]
\]

This means given some \(T' < T, v'\) s.t \(\downarrow_T v'\) it suffices to prove that \((p_l, T - T', v') \in [1]\)

From (E-val) we know that \(T' = 0\) and \(v' = ()\), therefore it suffices to prove that \((p_l, T, c) \in [1]\)

We get this directly from Definition 15

4. T-base:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash c : b \text{T-base}
\]

Given: \((p_l, T, \gamma) \in \llbracket \Gamma \sigma_i \rrbracket_c\), \((0, T, \delta) \in \llbracket \Omega \sigma_i \rrbracket_c\)

To prove: \((p_l, T, c) \in \llbracket b \rrbracket_c\)

From Definition 15 it suffices to prove that

\[
\forall T' < T, v', c \cdot \downarrow_T v' \implies (p_l, T - T', v') \in [1]
\]

This means given some \(T' < T, v'\) s.t \(c \downarrow_T v'\) it suffices to prove that \((p_l, T - T', v') \in [1]\)

From (E-val) we know that \(T' = 0\) and \(v' = c\), therefore it suffices to prove that \((p_l, T, c) \in [b]\)

We get this directly from Definition 15

5. T-nil:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{nil} : L^0 \tau \text{T-nil}
\]

Given: \((p_l, T, \gamma) \in \llbracket \Gamma \sigma_i \rrbracket_c\), \((0, T, \delta) \in \llbracket \Omega \sigma_i \rrbracket_c\)

To prove: \((p_l, T, \text{nil} \delta \gamma) \in \llbracket L^0 \tau \sigma_i \rrbracket_c\)

From Definition 15 it suffices to prove that

\[
\forall T' < T, v \cdot \text{nil} \downarrow_T v' \implies (p_l, T - T', v') \in [L^0 \tau \sigma_i]
\]

This means given some \(T' < T, v'\) s.t \(\text{nil} \downarrow_T v'\) it suffices to prove that \((p_l, T - T', v') \in [L^0 \tau \sigma_i]\)

From (E-val) we know that \(T' = 0\) and \(v' = \text{nil}\), therefore it suffices to prove that \((p_l, T, \text{nil}) \in [L^0 \tau \sigma_i]\)

We get this directly from Definition 15

6. T-cons:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \tau \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2 : L^n \tau \quad \Theta \vdash n : \mathbb{N} \text{T-cons}
\]

Given: \((p_l, T, \gamma) \in \llbracket (\Gamma_1 \oplus \Gamma_2) \sigma_i \rrbracket_c\), \((0, T, \delta) \in \llbracket (\Omega) \sigma_i \rrbracket_c\)

To prove: \((p_l, T, (e_1 : e_2) \delta \gamma) \in \llbracket L^{n+1} \tau \sigma_i \rrbracket_c\)

From Definition 15 it suffices to prove that

\[
\forall t < T, v \cdot (e_1 : e_2) \delta \gamma \downarrow_t v' \implies (p_l, T - t, v') \in [L^{n+1} \tau \sigma_i]
\]

This means given some \(t < T, v'\) s.t \((e_1 : e_2) \delta \gamma \downarrow_t v'\), it suffices to prove that \((p_l, T - t, v') \in [L^{n+1} \tau \sigma_i]\)

From (E-cons) we know that \(\exists v_f, l, v' = v_f :: l\)
7. T-match:

\[ \exists p_1, p_2, p_1 + p_2 \leq p_1 \wedge (p_1, T - t, v_f) \in \llbracket \tau \sigma_i \rrbracket \wedge (p_2, T - t, l) \in \llbracket L^n \tau \sigma_i \rrbracket \]  \quad \text{(F-C0)}

From Definition 14 and Definition 13 we know that \( \exists p_1, p_1 + p_2 = p_1 \) s.t. \((p_1, \gamma) \in [\Gamma_1]_{\sigma_l} \) and \((p_2, \gamma) \in [\Gamma_2]_{\sigma_l} \)

**IH1:**

\((p_1, T, e_1 \delta \gamma) \in [\tau \sigma_i]_{\xi} \)

Therefore from Definition 15 we have

\[ \forall t \in T. e_1 \delta \gamma \llrrarrow v_f \implies (p_1, T - t, v_f) \in \llbracket \tau \rrbracket \]

Since we are given that \( (e_1 :: e_2) \delta \gamma \llrrarrow v_f : l \) therefore from E-cons we also know that \( \exists t_1 < t. e_1 \delta \gamma \llrrarrow v_f \)

Since \( t_1 < t < T \), therefore we have \((p_1, T - t, v_f) \in \llbracket \tau \sigma_i \rrbracket \)  \quad \text{(F-C1)}

**IH2:**

\((p_2, T, e_2 \delta \gamma) \in [L^n \tau \sigma_i]_{\xi} \)

Therefore from Definition 15 we have

\[ \forall t \in T. e_2 \delta \gamma \llrrarrow l \implies (p_2, T - t, l) \in \llbracket L^n \tau \sigma_i \rrbracket \]

Since we are given that \( (e_1 :: e_2) \delta \gamma \llrrarrow v_f : l \) therefore from E-cons we also know that \( \exists t_2 < t - 1. e_2 \delta \gamma \llrrarrow l \)

Since \( t_2 < t - 1 < t < T \), therefore we have \((p_2, T - t, l) \in [L^n \tau \sigma_i] \)  \quad \text{(F-C2)}

In order to prove (F-C0) we choose \( p_1 \) as \( p_1 \) and \( p_2 \) as \( p_2 \) and it suffices to prove that

\((p_1, T - t, v) \in [\tau \sigma_i] \wedge (p_2, T - t, l) \in [L^n \tau \sigma_i] \)

Since \( t = t_1 + t_2 + 1 \) therefore from (F-C1) and Lemma 61 we get \((p_1, T - t, v) \in [\tau \sigma_i] \)

Similarly from (F-C2) and Lemma 61 we also get \((p_2, T - t, l) \in [L^n \tau \sigma_i] \)

7. T-match:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : L^n \tau \\
\Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_1 : \tau' \\
\Theta \vdash n : \mathbb{N} \\
\Psi; \Theta; \Delta \vdash \tau' : \mathbb{T}
\]

T-match

\[
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e \text{ with } |n| \mapsto e_1 \left[ h : t \mapsto e_2 \right] \delta \gamma \llrrarrow \psi_t v_f \implies (p_1, T - t, v_f) \in [\tau \sigma_i] \]

This means given some \( t < T, v_f \) s.t. \( \text{match } e \text{ with } |n| \mapsto e_1 \left[ h : t \mapsto e_2 \right] \delta \gamma \llrrarrow \psi_t v_f \) it suffices to prove that \((p_1, T - t, v_f) \in [\tau' \sigma_i] \)

\quad \text{(F-M0)}

From Definition 15 and Definition 13 we know that \( \exists p_1, p_2, p_1 + p_2 = p_1 \) s.t. \((p_1, \gamma) \in [\Gamma_1]_{\sigma_l} \) and \((p_2, \gamma) \in [\Gamma_2]_{\sigma_l} \)

**IH1:**

\((p_1, T, e_1 \delta \gamma) \in [L^n \tau \sigma_i]_{\xi} \)

This means from Definition 15 we have

\[ \forall t' \in T. e_1 \delta \gamma \llrrarrow v_1 \implies (p_1, T - t', v_1) \in \llbracket L^n \tau \sigma_i \rrbracket \]

Since we know that \( \text{match } e \text{ with } |n| \mapsto e_1 \left[ h : t \mapsto e_2 \right] \delta \gamma \llrrarrow \psi_t v_f \) therefore from E-match we know that \( \exists t' < t, v_1, e_1 \delta \gamma \llrrarrow v_1. \)

Since \( t' < t < T \), therefore we have \((p_1, T - t', v_1) \in [L^n \tau \sigma_i] \)

2 cases arise:

(a) \( v_1 = n\).

In this case we know that \( n = 0 \) therefore

**IH2:**

\((p_2, T, e_1 \delta \gamma) \in [\tau' \sigma_i]_{\xi} \)

This means from Definition 15 we have

\[ \forall t_1 \in T. e_1 \delta \gamma \llrrarrow l \implies (p_2, T - t_1, v_f) \in \llbracket \tau' \sigma_i \rrbracket \]

Since we know that \( \text{match } e \text{ with } |n| \mapsto e_1 \left[ h : t \mapsto e_2 \right] \delta \gamma \llrrarrow \psi_t v_f \) therefore from E-match we know that \( \exists t_1 < t. e_1 \delta \gamma \llrrarrow l, v_f. \)
Since \( t_1 < t < T \) therefore we have
\((p_{t_2}, T - t_1, v_f) \in [\tau' \sigma_l]_E\)

And from Lemma 61 we get
\((p_{t_2} + p_{t_1}, T - t, v_f) \in [\tau' \sigma_l']_E\)

And finally since \( p_l = p_{t_1} + p_{t_2} \) therefore we get
\((p_l, T - t, v_f) \in [\tau' \sigma_l']_E\)

And we are done

(b) \( v_1 = v :: l \):

In this case we know that \( n > 0 \)

\( \text{IH} \)
\((p_{t_2} + p_{t_1}, T, e_2 \delta \gamma') \in [\tau' \sigma_l']_E\)

where
\( \gamma' = \gamma \cup \{ h \mapsto v \} \cup \{ t \mapsto t' \} \)
\( t' = t \cup \{ i \mapsto n - 1 \} \)

This means from Definition 15 we have
\( \forall t_2 < T, e_2 \delta \gamma' \upsilon_{t_2} v_f \implies (p_{t_2} + p_{t_1}, T - t_2, v_f) \in [\tau' \sigma l']_E \)

Since we know that \( \text{match } e \text{ with } \| n1 \mapsto e_1 \| h :: t \mapsto e_2 \| \delta \gamma \| t \mapsto v_f \) therefore from E-match we know that \( 3t_2 < t, e_2 \delta \gamma' \upsilon_{t_2} v_f. \)

Since \( t_2 < t < T \) therefore we have
\((p_{t_2} + p_{t_1}, T - t_2, v_f) \in [\tau' \sigma l']_E\)

From Lemma 61 we get
\((p_{t_2} + p_{t_1}, T - t, v_f) \in [\tau' \sigma l']_E\)

And finally since \( p_l = p_{t_1} + p_{t_2} \) therefore we get
\((p_l, T - t, v_f) \in [\tau' \sigma l']_E\)

And finally since we have \( \Psi; \Theta; \Delta \vdash \tau' : K \) therefore we also have
\((p_l, T - t, v_f) \in [\tau' \sigma l]_E\)

And we are done

8. T-existI:

\[
\begin{array}{r}
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau[n/s] \\
\Theta \vdash n : S
\end{array}
\]

\( T \)-existI

Given: \( (p_l, T, \gamma) \in [\Gamma \sigma_l]_E, (0, T, \delta) \in [(\Omega \sigma l)]_E \)

To prove: \( (p_l, T, e \delta \gamma) \in [\exists s. T]_E \)

From Definition 15 it suffices to prove that
\( \forall t < T, v_f, e \gamma \ll v_f \implies (p_l, T - t, v_f \delta \gamma) \in [\exists s. T \sigma_l] \)

This means given some \( t < T, v_f \) s.t \( e \gamma \ll v_f \) it suffices to prove that
\( (p_l, T - t, v_f) \in [\exists s. T \sigma_l] \)

From Definition 15 it suffices to prove that
\( \exists s'. (p_l, T - t, v_f) \in [\tau[s'/s] \sigma_l] \) (F-E0)

\( \text{IH} : (p_l, T, e \delta \gamma) \in [\tau[n/s] \sigma_l]_E \)

This means from Definition 15 we have
\( \forall t' < T.e \delta \gamma \ll v_f \implies (p_l, T - t', v_f) \in [\tau[n/s] \sigma_l] \)

Since we are given that \( e \delta \gamma \ll v_f \) therefore we get
\( (p_l, T - t', v_f) \in [\tau[n/s] \sigma_l] \) (F-E1)

To prove (F-E0) we choose \( s' \) as \( n \) and we get the desired from (F-E1)

9. T-existsE:

\[
\begin{array}{r}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \exists s. T \\
\Psi; \Theta; s; \Delta; \Omega; \Gamma_2, x : \tau \vdash e' : \tau' \\
\Theta \vdash \tau'
\end{array}
\]

\( T \)-existsE

Given: \( (p_l, T, \gamma) \in [(\Gamma_1 + \Gamma_2) \sigma_l]_E, (0, T, \delta) \in [(\Omega) \sigma_l]_E \)

To prove: \( (p_l, T, (e; x.e')) \delta \gamma \in [\tau' \sigma l]_E \)

From Definition 15 it suffices to prove that
\( \forall t < T, v_f, (e; x.e') \delta \gamma \ll v_f \implies (p_l, T - t, v_f) \in [\tau' \sigma l] \)
This means given some \( t < T, v_f \) s.t. \((e; x.e') \delta \gamma \Downarrow t v_f\) it suffices to prove that 
\((p_1, T - t, v_f) \in [\tau' \sigma_1]_E\) \hspace{3cm} (F-EE0)

From Definition\([15]\) and Definition\([14]\) we know that \(\exists \gamma p_1, p_2: p_1 + p_2 = p_t\) s.t. 
\((p_1, \gamma) \in [(\Gamma_1)_{\sigma_1}]_E\) and \((p_2, \gamma) \in [(\Gamma_2)_{\sigma_1}]_E\)

**IH1**

\((p_1, T, e \delta \gamma) \in [\exists s.t \sigma_1]_E\)

This means from Definition\([15]\) we have

\(\forall t < T.e \delta \gamma \Downarrow t v_1 \implies (p_1, T - t_1, v_1) \in [\exists s.t \sigma_1]_E\)

Since we know that \((e; x.e') \delta \gamma \Downarrow t v_f\) therefore from E-existE we know that \(\exists t_1 < t, v_1. e \delta \gamma \Downarrow t_1 v_1\). Therefore we have

\((p_1, T - t_1, v_1) \in [\exists s.t \sigma_1]_E\)

Therefore from Definition\([15]\) we have

\(\exists s'. (p_1, T - t_1, v_1) \in [\tau'[s'/s] \sigma_1]_E\) \hspace{3cm} (F-EE1)

**IH2**

\((p_1 + p_2, T, e' \delta' \gamma) \in [\tau' \sigma_1']_E\)

where \(\delta' = \delta \cup \{x \mapsto e_1\}\) and \(\iota' = \iota \cup \{s \mapsto s'\}\)

This means from Definition\([15]\) we have

\(\forall t < T.e' \delta' \gamma \Downarrow t_2 v_f \implies (p_1 + p_2, T - t_2, v_f) \in [\tau' \sigma_1']_E\)

Since we know that \((e; x.e') \delta \gamma \Downarrow t v_f\) therefore from E-existE we know that \(\exists t_2 < t. e' \delta' \gamma \Downarrow t_2 v_f\). Since \(t_2 < t < T\) therefore we have

\((p_1 + p_2, T - t_2, v_f) \in [\tau' \sigma_1']_E\)

Since \(p_t = p_1 + p_2\) therefore we get

\((p_t, T - t_2, v_f) \in [\tau' \sigma_1']_E\)

From Lemma\([61]\) we get

\((p_t, T - t_2, v_f) \in [\tau' \sigma_1']_E\)

And finally since we have \(\Psi; \Theta \vdash \tau'\) therefore we also have

\((p_t, T - t, v_f) \in [\tau' \sigma_1]_E\)

And we are done

10. T-lam:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma, x : \tau_1 \vdash e : \tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x.e : (\tau_1 \rightarrow \tau_2)} \quad \text{T-lam}
\]

Given: \((p_t, T, \gamma) \in [\Gamma, \sigma_1]_E\), \((0, T, \delta) \in [\Omega, \sigma_1]_E\)

To prove: \((p_t, T, \lambda x.e \delta \gamma) \in [(\tau_1 \rightarrow \tau_2) \sigma_1]_E\)

From Definition\([15]\) it suffices to prove that

\(\forall t < T, v_f. (\lambda x.e) \delta \gamma \Downarrow t v_f \implies (p_t, T - t, v_f) \in [(\tau_1 \rightarrow \tau_2) \sigma_1]_E\)

This means given some \( t < T, v_f\) s.t. \((\lambda x.e) \delta \gamma \Downarrow t v_f\). From E-val we know that \(t = 0\) and \(v_f = (\lambda x.e) \delta \gamma\)

Therefore it suffices to prove that

\((p_t, T, (\lambda x.e) \delta \gamma) \in [(\tau_1 \rightarrow \tau_2) \sigma_1]_E\)

From Definition\([15]\) it suffices to prove that

\(\forall p', e', T' < T. (p' + p', T', e') \in [\tau_1 \sigma_1]_E \implies (p_t + p', T', e'[\varepsilon/x]) \in [\tau_2 \sigma_1]_E\)

This means given some \( p', e', T' < T\) s.t. \((p', T', e') \in [\tau_1 \sigma_1]_E\) it suffices to prove that

\((p_t + p', T', e'[\varepsilon/x]) \in [\tau_2 \sigma_1]_E\) \hspace{3cm} (F-L1)

From IH we know that

\((p_t + p', T, e \delta \gamma') \in [\tau_2 \sigma_1]_E\)

where

\(\gamma' = \gamma \cup \{x \mapsto e'\}\)

Therefore from Lemma\([62]\) we get the desired
11. T-app:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : (\tau_1 \to \tau_2) \quad \Psi; \Theta; \Delta; \Omega_2; \Gamma_2 \vdash e_2 : \tau_1}{\Psi; \Theta; \Delta; \Omega_1 \oplus \Omega_2; \Gamma_1 \oplus \Gamma_2 \vdash e_1 e_2 : \tau_2}
\]

\text{T-app}

Given: \((p_1, T, \gamma) \in \llbracket (\Gamma_1 \oplus \Gamma_2) \sigma_1 \rrbracket \in \llbracket (\Omega \sigma_1) \rrbracket \)

To prove: \((p_1, T, e_1 e_2 \delta \gamma) \in \llbracket \tau_2 \sigma_1 \rrbracket \)

From Definition \[15\] it suffices to prove that

\(\forall t < T, v_f, (e_1 e_2) \delta \gamma \Downarrow_t v_f \implies (p_1, T-t, v_f) \in \llbracket \tau_2 \sigma_1 \rrbracket \)

This means given some \(t < T, v_f\) s.t. \((e_1 e_2) \delta \gamma \Downarrow_t v_f\) it suffices to prove that \((p_1, T-t, v_f) \in \llbracket \tau_2 \sigma_1 \rrbracket \) (F-A0)

From Definition \[16\] and Definition \[14\] we know that \(\exists p_{11}, p_{12}, p_{11} + p_{12} = p_1\) s.t. \((p_{11}, \gamma) \in \llbracket (\Gamma_1) \sigma_1 \rrbracket \) and \((p_{12}, \gamma) \in \llbracket (\Gamma_2) \sigma_1 \rrbracket \)

\text{IH1}

\((p_{11}, T, e_1 \delta \gamma) \in \llbracket (\tau_1 \to \tau_2) \sigma_1 \rrbracket \)

This means from Definition \[15\] we have

\(\forall t_1 < T. e_1 \Downarrow_{t_1} \lambda x. e \implies (p_{11}, T-t_1, \lambda x. e) \in \llbracket (\tau_1 \to \tau_2) \sigma_1 \rrbracket \)

Since we know that \((e_1 e_2) \delta \gamma \Downarrow_t v_f\) therefore from E-app we know that \(\exists t_1 < t. e_1 \Downarrow_{t_1} \lambda x. e\), therefore we have

\((p_{11}, T-t_1, \lambda x. e) \in \llbracket (\tau_1 \to \tau_2) \sigma_1 \rrbracket \)

Therefore from Definition \[16\] we have

\(\forall p', e_1, T_1 < T-t_1. (p', T_1, e_1') \in \llbracket \tau_1 \sigma_1 \rrbracket \implies (p_{11} + p', T_1, e[e'/x]) \in \llbracket \tau_2 \sigma_1 \rrbracket \) (F-A1)

\text{IH2}

\((p_{12}, T-t_1 - 1, e_2 \delta \gamma) \in \llbracket \tau_1 \sigma_1 \rrbracket \) (F-A2)

Instantiating (F-A1) with \(p_{12}, e_2 \delta \gamma\) and \(T-t_1 - 1\) we get

\((p_{11} + p_{12}, T-t_1 - 1, e[e_2 \delta \gamma/x]) \in \llbracket \tau_2 \sigma_1 \rrbracket \)

This means from Definition \[15\] we have

\(\forall t_2 < T-t_1 - 1. e[e_2 \delta \gamma/x] \Downarrow_{t_2} v_f \implies (p_{11} + p_{12}, T-t_1 - 1 - t_2, v_f) \in \llbracket \tau_2 \sigma_1 \rrbracket \)

Since we know that \((e_1 e_2) \delta \gamma \Downarrow_t v_f\) therefore from E-app we know that \(\exists t_2. e[e_2 \delta \gamma/x] \Downarrow_{t_2} v_f\) where \(t_2 = t - t_1 - 1\), therefore we have

\((p_{11} + p_{12}, T-t_1 - 1 - t_2, v_f) \in \llbracket \tau_2 \sigma_1 \rrbracket\) where \(p_{11} + p_{12} = p_1\)

Since from E-app we know that \(t = t_1 + t_2 + 1\), therefore we have proved (F-A0)

12. T-sub:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \quad \Theta \vdash \tau <; \tau'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau'}
\]

\text{T-sub}

Given: \((p_1, T, \gamma) \in \llbracket (\Gamma) \sigma_1 \rrbracket \in \llbracket (\Omega \sigma_1) \rrbracket \)

To prove: \((p_1, T, e \delta \gamma) \in \llbracket \tau' \sigma_1 \rrbracket \)

\text{IH} \((p_1, T, e \delta \gamma) \in \llbracket \tau \sigma_1 \rrbracket \)

We get the desired directly from IH and Lemma \[22\]

13. T-weaken:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \quad \Psi; \Theta; \Delta \vdash \Gamma' <; \Gamma \quad \Psi; \Theta; \Delta \vdash \Omega' <; \Omega}{\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e : \tau}
\]

\text{T-weaken}

Given: \((p_1, T, \gamma) \in \llbracket (\Gamma') \sigma_1 \rrbracket \in \llbracket (\Omega') \sigma_1 \rrbracket \)

To prove: \((p_1, T, e \delta \gamma) \in \llbracket \tau \sigma_1 \rrbracket \)

Since we are given that \((p_1, T, \gamma) \in \llbracket (\Gamma') \sigma_1 \rrbracket \) therefore from Lemma \[23\] we also have \((p_1, T, \gamma) \in \llbracket (\Gamma) \sigma_1 \rrbracket \)

Similarly since we are given that \((0, T, \delta) \in \llbracket (\Omega') \sigma_1 \rrbracket \) therefore from Lemma \[25\] we also have \((0, T, \delta) \in \llbracket \tau \sigma_1 \rrbracket \)

\text{IH} \((p_1, T, e \delta \gamma) \in \llbracket \tau \sigma_1 \rrbracket \)

We get the desired directly from IH
14. T-tensorE:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \tau_1 \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2 : \tau_1}{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \otimes \Gamma_2 \vdash \langle e_1, e_2 \rangle : \langle \tau_1 \otimes \tau_2 \rangle} \quad \text{T-tensorE}
\]

Given: \((p_1, T, \gamma) \in [(\Gamma_1 \otimes \Gamma_2) \sigma_T]_\mathcal{E}, (0, T, \delta) \in [(\Omega) \sigma_T]_\mathcal{E}\)

To prove: \((p_1, T, \langle e_1, e_2 \rangle) \vdash \delta \gamma \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\)

From Definition 15 it suffices to prove that

\[\forall t < T. (e_1, e_2) \vdash \delta \gamma \quad \iff \quad (p_1, T-t, \langle v_{f_1}, v_{f_2} \rangle) \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\]

This means given some \(t < T\) s.t. \(\langle e_1, e_2 \rangle \vdash \delta \gamma \quad \iff \quad \langle v_{f_1}, v_{f_2} \rangle \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\)

From Definition 14 and Definition 13 we know that \(\exists p_{12}, p_{12} + p_{12} = p_1\) s.t.

\(\exists (p_1, \gamma) \in [(\Gamma_1) \sigma_T]_\mathcal{E}\) and \((p_2, \gamma) \in [(\Gamma_2) \sigma_T]_\mathcal{E}\)

**H1:**

\((p_1, T, e_1 \delta \gamma) \in [\tau_1 \sigma_T]_\mathcal{E}\)

Therefore from Definition 15 we have

\[\forall t < T. e_1 \delta \gamma \quad \iff \quad (p_1, T-t, v_{f_1}) \in [\tau_1 \sigma_T]_\mathcal{E}\]

Since we are given that \(\langle e_1, e_2 \rangle \delta \gamma \quad \iff \quad \langle v_{f_1}, v_{f_2} \rangle \in [\tau_1 \otimes \tau_2]_\mathcal{E}\)

Hence we have \((p_1, T-t_1, v_{f_1}) \in [\tau_1 \sigma_T]_\mathcal{E}\) (F-T11)

**H2:**

\((p_2, T, e_2 \delta \gamma) \in [\tau_2 \sigma_T]_\mathcal{E}\)

Therefore from Definition 15 we have

\[\forall t < T. e_2 \delta \gamma \quad \iff \quad (p_2, T-t_2, v_{f_2}) \in [\tau_2 \sigma_T]_\mathcal{E}\]

Since we are given that \(\langle e_1, e_2 \rangle \delta \gamma \quad \iff \quad \langle v_{f_1}, v_{f_2} \rangle \in [\tau_1 \otimes \tau_2]_\mathcal{E}\)

Since \(t_2 < t < T\) therefore we have

\((p_2, T-t_2, v_{f_2}) \in [\tau_2 \sigma_T]_\mathcal{E}\) (F-T12)

Applying Lemma 61 on (F-T11) and (F-T12) and by using Definition 15 we get the desired.

15. T-tensorE:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : (\tau_1 \otimes \tau_2) \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1, y : \tau_2 \vdash e' : \tau}{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \pi \Gamma_2 \vdash \text{let}(\langle x, y \rangle) = e \text{ in } e' : \tau} \quad \text{T-tensorE}
\]

Given: \((p_1, T, \gamma) \in [(\Gamma_1 \otimes \Gamma_2) \sigma_T]_\mathcal{E}, (0, T, \delta) \in [(\Omega) \sigma_T]_\mathcal{E}\)

To prove: \((p_1, T, \text{let}(\langle x, y \rangle) = e \text{ in } e') \delta \gamma \in [\tau \sigma_T]_\mathcal{E}\)

From Definition 15 it suffices to prove that

\[\forall t < T, v_f. (\text{let}(\langle x, y \rangle) = e \text{ in } e') \delta \gamma \quad \iff \quad (p_1, T-t, v_f) \in [\tau \sigma_T]_\mathcal{E}\]

This means given some \(t < T, v_f\) s.t. \(\text{let}(\langle x, y \rangle) = e \text{ in } e') \delta \gamma \quad \iff \quad (p_1, T-t, v_f) \in [\tau \sigma_T]_\mathcal{E}\)

From Definition 15 and Definition 13 we know that \(\exists p_{11}, p_{12}, p_{12} + p_{12} = p_1\) s.t.

\(\exists (p_1, \gamma) \in [(\Gamma_1) \sigma_T]_\mathcal{E}\) and \((p_2, \gamma) \in [(\Gamma_2) \sigma_T]_\mathcal{E}\)

**H1:**

\((p_1, T, e \delta \gamma) \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\)

This means from Definition 15 we have

\[\forall t < T. e \delta \gamma \quad \iff \quad (p_1, T-t_1, \langle v_{f_1}, v_{f_2} \rangle) \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\]

Since we know that \(\text{let}(\langle x, y \rangle) = e \text{ in } e') \delta \gamma \quad \iff \quad \langle v_{f_1}, v_{f_2} \rangle \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\)

Therefore we have

\((p_1, T-t_1, \langle v_{f_1}, v_{f_2} \rangle) \in [(\tau_1 \otimes \tau_2) \sigma_T]_\mathcal{E}\)

From Definition 15 we know that

\(\exists p_{12}, p_{12} + p_{12} = p_1 \land (p_1, T, v_1) \in [\tau_1 \sigma_T] \land (p_2, T, v_2) \in [\tau_2 \sigma_T]_\mathcal{E}\) (F-TE1)

**H2:**

\((p_2 + p_1 + p_2, T, e' \delta \gamma') \in [\tau \sigma_T]_\mathcal{E}\)
where
\[ \gamma' = \gamma \cup \{x \mapsto v_1\} \cup \{y \mapsto v_2\} \]

This means from Definition\[15\] we have
\[ \forall t < T . e' . \delta \gamma' \downarrow t . v_f \implies (p_{t2} + p_1 + p_2, T - t_2, v_f) \in [\tau \sigma_i] \]

Since we know that (let(⟨x, y⟩) = e in e') \( \delta \gamma \downarrow t . v_f \) therefore from E-TE we know that \( \exists t_2 < t.e' \delta \gamma' \downarrow t_2 . v_f. \)

Therefore we have
\( (p_{t2} + p_1 + p_2, T - t_2, v_f) \in [\tau \sigma_i] \)

From Lemma\[61\] we get
\( (p_1, T - t, v_f) \in [\tau \sigma_i] \)

And we are done

16. T-withI:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau_1 \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_2 : \tau_1}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \langle e_1, e_2 \rangle : (\tau_1 \& \tau_2)} \quad \text{T-withI}
\]

Given: \( (p_1, T, \gamma) \in [\Gamma \sigma_i]_{\xi}, (0, T, \delta) \in [\Omega \sigma_i]_{\xi} \)

To prove: \( (p_1, T, (e_1, e_2) \delta \gamma) \in [(\tau_1 \& \tau_2) \sigma_i]_{\xi} \)

From Definition\[15\] it suffices to prove that
\[ \forall t < T . (e_1, e_2) \delta \gamma \downarrow t . (\langle v_f_1, v_f_2 \rangle) \implies (p_1, T - t, \langle v_f_1, v_f_2 \rangle) \in [(\tau_1 \& \tau_2) \sigma_i] \]

This means given \( (e_1, e_2) \delta \gamma \downarrow t . (\langle v_f_1, v_f_2 \rangle) \) it suffices to prove that
\( (p_1, T - t, \langle v_f_1, v_f_2 \rangle) \in [(\tau_1 \& \tau_2) \sigma_i] \) (F-WI0)

**IH1:**
\( (p_1, T, e_1 \delta \gamma) \in [\tau_1 \sigma_i]_{\xi} \)

Therefore from Definition\[15\] we have
\[ \forall t_1 < T . e_1 \delta \gamma \downarrow t_1 . v_f \implies (p_1, T - t_1, v_f) \in [\tau_1 \sigma_i] \]

Since we are given that \( (e_1, e_2) \delta \gamma \downarrow t . (\langle v_f_1, v_f_2 \rangle) \) therefore from E-WI we know that \( \exists t_1 < t.e_1 \delta \gamma \downarrow t_1 . v_f. \)

Since \( t_1 < t < T \), therefore we have
\( (p_1, T - t_1, v_f) \in [\tau_1 \sigma_i] \) (F-WI1)

**IH2:**
\( (p_1, T, e_2 \delta \gamma) \in [\tau_2 \sigma_i]_{\xi} \)

Therefore from Definition\[15\] we have
\[ \forall t_2 < T . e_2 \delta \gamma \downarrow t_2 . v_f \implies (p_1, T - t_2, v_f) \in [\tau_2 \sigma_i] \]

Since we are given that \( (e_1, e_2) \delta \gamma \downarrow t . (\langle v_f_1, v_f_2 \rangle) \) therefore from E-WI we also know that \( \exists t_2 < t.e_2 \delta \gamma \downarrow t_2 . v_f. \)

Since \( t_2 < t < T \), therefore we have
\( (p_1, T - t_2, v_f) \in [\tau_2 \sigma_i] \) (F-WI2)

Applying Lemma\[61\] on (F-W1) and (F-W2) we get the desired.

17. T-fst:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (\tau_1 \& \tau_2) \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{fst}(e) : \tau_1}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash t : \tau_1 \quad \text{T-fst}}
\]

Given: \( (p_1, T, \gamma) \in [(\Gamma \sigma_i)]_{\xi}, (0, T, \delta) \in [\Omega \sigma_i]_{\xi} \)

To prove: \( (p_1, T, (\text{fst}(e)) \delta \gamma) \in [(\tau_1 \sigma_i)]_{\xi} \)

From Definition\[15\] it suffices to prove that
\[ \forall t < T . v_f . (\text{fst}(e)) \delta \gamma \downarrow t . v_f \implies (p_1, T - t, v_f) \in [\tau_1 \sigma_i] \]

This means given some \( t < T . v_f \) s.t. \( (\text{fst}(e)) \delta \gamma \downarrow t . v_f \) it suffices to prove that
\( (p_1, T - t, v_f) \in [\tau_1 \sigma_i] \) (F-F0)

**IH**

\( (p_1, T, e \delta \gamma) \in [(\tau_1 \& \tau_2) \sigma_i]_{\xi} \)

This means from Definition\[15\] we have
\[ \forall t_1 < T . e \delta \gamma \downarrow t_1 . (v_1, v_2) \delta \gamma \implies (p_1, T - t_1, (v_1, v_2)) \in [(\tau_1 \& \tau_2) \sigma_i] \]

Since we know that \( (\text{fst}(e)) \delta \gamma \downarrow t . v_f \) therefore from E-fst we know that \( \exists t_1 < t.v_1, v_2.e \delta \gamma \downarrow t_1 . (v_1, v_2). \)

Since \( t_1 < t < T \), therefore we have

34
From Definition 15 we know that $(p_t, T - t, v_1) \in [\tau_1 \sigma_1]$.

Finally using Lemma 61 we also have $(p_t, T - t, v_1) \in [\tau_1 \sigma_1]$.

Since from E-fst we know that $v_f = v_1$, therefore we are done.

18. T-snd:

Similar reasoning as in T-fst case above.

19. T-inr:

Given: $(p_t, T, \gamma) \in [\Gamma \sigma_1]E$, $(0, T, \delta) \in [\Omega \sigma_1]E$

To prove: $(p_t, T, \inl(e) \delta \gamma) \in [\tau_1 \sigma_1]E$

From Definition 15 it suffices to prove that

$\forall t < T. \inl(e) \delta \gamma \in \Gamma \sigma_1$.

This means from Definition 15 we have

$\forall t < T, v_1 \in \Gamma \sigma_1$.

Hence we have $(p_t, T - t, v_1) \in [\tau_1 \sigma_1]$. From Lemma 61 we get $(p_t, T - t, v_1) \in [\tau_1 \sigma_1]$.

And finally from Definition 15 we get (F-IL0).

20. T-inr:

Similar reasoning as in T-inr case above.

21. T-case:

Given: $(p_t, T, \gamma) \in [\Gamma_1 \oplus \Gamma_2]E$, $(0, T, \delta) \in [\Omega \sigma_1]E$

To prove: $(p_t, T, \text{case } e, x, e_1, y, e_2) \delta \gamma \in [\tau \sigma_1]E$

From Definition 15 it suffices to prove that

$\forall t < T, v_f, (\text{case } e, x, e_1, y, e_2) \delta \gamma \in \Gamma \sigma_1$.

This means from Definition 15 we know that $\exists p_{11}, p_{12}, p_{11} + p_{12} = p_t$ s.t.

$(p_{11}, \gamma) \in [\Gamma_1 \sigma_1]E$ and $(p_{12}, \gamma) \in [\Gamma_2 \sigma_1]E$.

IH1

$(p_{11}, T, e \delta \gamma) \in [\tau_1 \sigma_1]E$

This means from Definition 15 we have

$\forall t' < T. e \delta \gamma \in \Gamma_1 \sigma_1$.

Since we know that $(\text{case } e, x, e_1, y, e_2) \delta \gamma \in [\tau_1 \sigma_1]$.

Since $t' < T$, therefore we have

$(p_{11}, T - t', v_1) \in [\tau_1 \sigma_1]$.

2 cases arise:

(a) $v_1 = \inl(v)$:

IH2

$(p_{12} + p_{11}, T - t', e_1 \delta \gamma') \in [\tau \sigma_1]E$

where
\[ \gamma' = \gamma \cup \{ x \mapsto v \} \]

This means from Definition 15 we have
\[ \forall t_1 < T - t, e_1 \delta' \downarrow t_1, v_f \quad \Rightarrow \quad (p_1, T - t, e_1, v_f, v_f) \in [\tau \sigma] \]

Since we know that (case \( e \cdot x.e_1, y.e_2 \)) \( \delta_1 \downarrow e_1 \) \( v_f \) therefore from E-case we know that \( \exists t_1, e_1 \delta' \downarrow v_f \)

where \( t_1 = t - t' - 1 \).

Since \( t_1 = t - t' - 1 < T - t' \) therefore we have
\[ (p_2, T - t' - t_1, v_f) \in [\tau \sigma] \]

From Lemma 61 we get
\[ (p_2 + p_1, T - t, v_f) \in [\tau \sigma] \]

And finally since \( p_1 = p_1 + p_2 \) therefore we get
\[ (p_1, T - t, v_f) \in [\tau \sigma] \]

And we are done

(b) \( v_1 = \text{inr} (v) \):

Similar reasoning as in the inl case above.

22. T-ExpI:

\[ \Psi; \Theta; \Delta; \Omega; \vdash e : \tau \quad \Psi; \Theta; \Delta; \Omega; \vdash ! e : \tau \quad \text{T-ExpI} \]

Given: \( (p_1, T, \gamma) \in [I \sigma] \), \( (0, T, \delta) \in [\Omega \sigma] \)

To prove: \( (p_1, T, ! e \delta) \in [I \sigma] \)

From Definition 15 it suffices to prove that
\[ \forall t < T . (! e \delta \downarrow e (t) \delta) \quad \Rightarrow \quad (p_1, T - t, (! e \delta)) \in [I \sigma] \]

This means given some \( t < T \) s.t \( (! e \delta \downarrow e (t) \delta) \) it suffices to prove that
\[ (p_1, T - t, (! e \delta)) \in [I \sigma] \]

From Definition 15 it suffices to prove that
\[ (0, T - t, ! e \delta) \in [I \sigma] \]

\[ \text{IH:} \ (0, T - t, ! e \delta) \in [I \sigma] \]

We get the desired directly from IH

23. T-ExpE:

\[ \Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \tau \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e : \tau' \quad \text{T-ExpE} \]

Given: \( (p_1, T, \gamma) \in [(\Gamma_1 \oplus \Gamma_2) \sigma] \), \( (0, T, \delta) \in [(\Omega) \sigma] \)

To prove: \( (p_1, T, ! e \delta) \in [(\Gamma_2) \sigma] \)

From Definition 15 it suffices to prove that
\[ \forall t < T, v_f . (! e \delta \downarrow e (t) \delta) \quad \Rightarrow \quad (p_1, T - t, v_f) \in [(\Gamma_2) \sigma] \]

This means given some \( t < T, v_f \) s.t \( (! e \delta \downarrow e (t) \delta) \) it suffices to prove that
\[ (p_1, T - t, v_f) \in [(\Gamma_2) \sigma] \]

From Definition 15 and Definition 15 we know that \( \exists p_1, p_2, p_1 + p_2 = p_1 \) s.t
\[ (p_1, \gamma) \in [(\Gamma_1) \sigma] \]

\[ \text{IH1:} \ (p_1, T, ! e \delta) \in [(\Gamma_1) \sigma] \]

This means from Definition 15 we have
\[ \forall t_1 < T . e \delta \downarrow t_1, ! e_1 \delta \quad \Rightarrow \quad (p_1, T - t_1, ! e_1 \delta) \in [(\Gamma_1) \sigma] \]

Since we know that \( (\text{let} ! e \downarrow e (t) \delta) \downarrow e \downarrow e_1 \delta \) therefore from (E-ExpE) we know that \( \exists t_1 < t, e_1 \downarrow e_1 \delta \downarrow e_1 \delta \).

Since \( t_1 < t < T \), therefore we have
\[ (p_1, T - t_1, ! e_1 \delta) \in [(\Gamma_1) \sigma] \]

This means from Definition 15 we have
\[ (0, T - t_1, e_1 \delta) \in [\tau] \quad \text{(E-E1)} \]

\[ \text{IH2:} \]

36
(p_2, T - t_1, e', \delta' \gamma) \in [\tau', \sigma\iota]_\varepsilon

where
\delta' = \delta \cup \{x \mapsto e_1\}

This means from Definition 15 we have
\forall t_2 < T - t_1, e', \delta' \gamma \triangleright \psi_{t_2} v_f \implies (p_2, T - t_1 - t_2, v_f) \in [\tau', \sigma\iota]

Since we know that (let ! x = e in e') \delta' \gamma \triangleright \psi_f therefore from (E-ExpE) we know that \exists t_2, e' \delta' \gamma \triangleright v_f where t_2 = t - t_1 - 1.

Since t_2 = t - t_1 - 1 < T - t_1, therefore we have
(p_2, T - t_1 - t_2, v_f) \in [\tau', \sigma\iota]

From Lemma 62 we get
(p_2 + p_1, T - t, v_f) \in [\tau', \sigma\iota]

And finally since p_1 = p_1 + p_2 therefore we get
(p_1, T - t, v_f) \in [\tau', \sigma\iota]

And we are done

24. T-tabs:

\[
\begin{array}{c}
\Psi, \alpha : K; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda. e : (\forall \alpha : K, \tau) \\
\hline
\text{T-tabs}
\end{array}
\]

Given: \((p_1, T, \gamma) \in [\Gamma, \sigma\iota]_\varepsilon, (0, T, \delta) \in [\Omega, \sigma\iota]_\varepsilon\)

To prove: \((p_1, T, (\Lambda e) \delta' \gamma) \in [(\forall \alpha. \tau) \sigma\iota]_\varepsilon\)

From Definition 15 it suffices to prove that
\forall t < T, v_f. (\Lambda e) \delta' \gamma \triangleright v_f \implies (p_1, T - t, v_f) \in [(\forall \alpha. \tau) \sigma\iota]

This means given some \(t < T, v_f\) s.t. \((\Lambda e) \delta' \gamma \triangleright v_f\). From E-val we know that \(t = 0\) and \(v_f = (\Lambda e) \delta' \gamma\) Therefore it suffices to prove that
\((p_1, T, (\Lambda e) \delta' \gamma) \in [(\forall \alpha. \tau) \sigma\iota]\)

From Definition 15 it suffices to prove that
\forall \tau', T' < T. (p_1, T', e) \in [\tau[\tau'/\alpha] \sigma\iota]_\varepsilon

This means given some \(\tau', T' < T\) it suffices to prove that
\((p_1, T', e) \in [\tau[\tau'/\alpha] \sigma\iota]_\varepsilon\) \quad \text{(F-TAB0)}

From IH we know that
\((p_1, T, e \delta' \gamma) \in [\tau \sigma'\iota]_\varepsilon\)

where
\sigma' = \gamma \cup \{\alpha \mapsto \tau'\}

Therefore from Lemma 62 we get the desired

25. T-tapp:

\[
\begin{array}{c}
\Psi, \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \\
\Psi; \Theta \vdash \tau' \\
\hline
\text{T-tapp}
\end{array}
\]

Given: \((p_1, T, \gamma) \in [\Gamma \sigma\iota]_\varepsilon, (0, T, \delta) \in [\Omega \sigma\iota]_\varepsilon\)

To prove: \((p_1, T, e [\tau') \sigma\iota]) : (\forall \alpha. \tau) \sigma\iota]_\varepsilon\)

From Definition 15 it suffices to prove that
\forall t < T, v_f. (e [\tau') \sigma\iota) \delta' \gamma \triangleright v_f \implies (p_1, T - t, v_f) \in [\tau[\tau'/\alpha] \sigma\iota]

This means given some \(t < T, v_f\) s.t \((e [\tau') \sigma\iota) \delta' \gamma \triangleright v_f\) it suffices to prove that
\((p_1, T - t, v_f) \in [\tau[\tau'/\alpha] \sigma\iota]\) \quad \text{(F-A0)}

IH
\((p_1, T, e \delta' \gamma) \in [(\forall \alpha. \tau) \sigma\iota]_\varepsilon\)

This means from Definition 15 we have
\forall t_1 < T. e \delta' \gamma \triangleright \Lambda e \implies (p_1, T - t_1, \Lambda e) \in [(\forall \alpha. \tau) \sigma\iota]

Since we know that \((e [\tau') \sigma\iota) \delta' \gamma \triangleright v_f\) therefore from E-tapp we know that \(\exists t_1 < t. e \delta' \gamma \triangleright v_f\), therefore we have
\((p_1, T - t_1, \Lambda e) \in [(\forall \alpha. \tau) \sigma\iota]\)

Therefore from Definition 15 we have
∀τ″, T₁ < T − t₁. (pᵢ, T₁, e) ∈ [τ″/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ (F-A1)

Instantiating (F-A1) with the given τ″ and T − t₁ − 1 we get

(pᵢ, T − t₁ − 1, e) ∈ [τ″/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ

From Definition 13 we have
∀τ₂ < T − t₂ − 1. e  ψ₂ v₂j ⇒ (pᵢ, T − t₂ − 1, v₂j) ∈ [τ″/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ

Since we know that (e []) δγ ψ₂ v₂j therefore from E-tapp we know that Ξτ₂.e  ψ₂ v₂j where t₂ = t₁ − 1

Since t₂ = t − t₁ − 1 < T − t₁ − 1, therefore we have
(pᵢ, T − t₂ − 1, v₂j) ∈ [τ″/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ and we are done.

26. T-ibs:

Ψ; Θ, i : S; Δ; Ω; Γ ⊢ e : τ

Given: (pᵢ, T, γ) ∈ [Γ, σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (∀i : S.τ)

To prove: (pᵢ, T, (Δ.e) δγ) ∈ [[(∀i.τ) σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ]

From Definition 13 it suffices to prove that
∀τ < T, v₂j.(Δ.e) δγ ψ₂ v₂j ⇒ (pᵢ, T − τ, v₂j) ∈ [[(∀i.τ) σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ]

This means given some t < T, v₂j s.t. (Δ.e) δγ ψ₂ v₂j. From E-val we know that τ = 0 and v₂j = (Δ.e) δγ

Therefore it suffices to prove that
(pᵢ, T, (Δ.e) δγ) ∈ [[(∀i.τ) σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : τ]

From Definition 13 it suffices to prove that
∀τ, T′ < T : (pᵢ, T′, e) ∈ [τ[I/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

(F-IAB0)

From IH we know that
(pᵢ, T, e δγ) ∈ [[τ σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

where

τ′ = γ ∪ {i → I}

Therefore from Lemma 62 we get the desired

27. T-iapp:

Ψ; Θ, i : S; Δ; Ω; Γ ⊢ e : (∀i : S.τ)

Ψ; Θ, i : S; Δ; Ω; Γ ⊢ e : (τ[I/i])

Given: (pᵢ, T, γ) ∈ [Γ, σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

To prove: (pᵢ, T, e δγ) ∈ [[τ σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

From Definition 13 it suffices to prove that
∀τ < T, v₂j.(e []) δγ ψ₂ v₂j ⇒ (pᵢ, T − τ, v₂j) ∈ [[τ[I/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

This means given some t < T, v₂j s.t. (e []) δγ ψ₂ v₂j it suffices to prove that
(pᵢ, T − τ, v₂j) ∈ [[τ[I/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

(F-A0)

IH

(pᵢ, T, e δγ) ∈ [[(∀i.τ) σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

This means from Definition 13 we have
∀τ₁ < T.e  ψ₁, Λ.e ⇒ (pᵢ, T − τ₁, Λ.e) ∈ [(∀i.τ) σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

Since we know that (e []) δγ ψ₁ v₁ therefore from E-tapp we know that Ξτ₁ < T.e  ψ₁, Λ.e, therefore we have
(pᵢ, T − τ₁, Λ.e) ∈ [(∀i.τ) σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

Therefore from Definition 13 we have
∀I, T₁ < T − t₁. (pᵢ, T₁, e) ∈ [[τ[I/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

(F-IAP1)

Instantiating (F-IAP1) with the given I and T − t₁ − 1 we get
(pᵢ, T − t₁ − 1, e) ∈ [[τ[I/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

From Definition 13 we have
∀τ₂ < T − t₂ − 1.e  ψ₂ v₂j ⇒ (pᵢ, T − t₂ − 1, v₂j) ∈ [[τ[I/i] σ| Ψ; Θ; Δ; Ω; Γ ⊢ e : (τ[I/i])

38
Since we know that \( (e \|) \) \( \delta \gamma \downarrow_t v_f \) therefore from E-iapp we know that \( \exists t_2, e \|_{t_2} v_f \) where \( t_2 = t - t_1 - 1 \)
Since \( t_2 = t - t_1 - 1 < T - t_1 - 1 \), therefore we have 
\((p_1, T - t_1 - t_2 - 1, v_f) \in [\tau \| t] \sigma_i \) and we are done.

28. T-CI:

\[
\Psi; \Theta; \Delta, e; \Gamma \vdash e : \tau \\
\Psi; \Theta; \Delta, \Omega; \Gamma \vdash \lambda \cdot e : (c \rightarrow \tau) \quad \text{T-CI}
\]

Given: \((p_1, T, \gamma) \in [\Gamma \sigma_i]_\xi, (0, T, \delta) \in [\Omega \sigma_i]_\xi \) and \( \vdash \Delta \ i \)
To prove: \((p_1, T, \lambda \cdot e \ \delta \gamma) \in [(c \rightarrow \tau) \ \sigma_i]_\xi \)

From Definition [15] it suffices to prove that
\[
\forall t, t < T \cdot \Delta, e \ \delta \gamma \downarrow_t v \implies (p_1, T - t, v) \in [(c \rightarrow \tau) \ \sigma_i]_\xi
\]

This means given some \( v, t < T \) s.t \( \Lambda, e \ \delta \gamma \downarrow_t v \) and from \( (E-val) \) we know that \( v = \Lambda, e \ \delta \gamma \) and \( t = 0 \) therefore it suffices to prove that
\[(p_1, T, \lambda \cdot e \ \delta \gamma) \in [(c \rightarrow \tau) \ \sigma_i]_\xi \)

From Definition [15] it suffices to prove that
\[. \vdash c \ i \implies (p_1, T, \delta \gamma) \in [\tau \sigma_i]_\xi \]

This means given that \( . \vdash c \ i \) it suffices to prove that
\[(p_1, T, \lambda \cdot e \ \delta \gamma) \in [\tau \ \sigma_i]_\xi \]

\( \text{IH} \) \( (p_1, T, \lambda \cdot e \ \delta \gamma) \in [\tau \ \sigma_i]_\xi \)

We get the desired directly from IH

29. T-CE:

\[
\Psi; \Theta; \Delta, e; \Gamma \vdash e : (c \rightarrow \tau) \\
\Theta; \Delta \vdash e : c \quad \text{T-CE}
\]

Given: \((p_1, T, \gamma) \in [\Gamma \sigma_i]_\xi, (0, T, \delta) \in [\Omega \sigma_i]_\xi \) and \( \vdash \Delta \ i \)
To prove: \((p_1, T, e \ | \) \( \delta \gamma) \in [\tau \sigma_i]_\xi \)

From Definition [15] it suffices to prove that
\[
\forall v_f, t < T \cdot (e \ |) \ \delta \gamma \downarrow_t v_f \implies (p_1, T - t, v_f) \in [(\tau) \ \sigma_i]_\xi
\]

This means given some \( v_f, t < T \) s.t \( (e \ |) \ \delta \gamma \downarrow_t v_f \) it suffices to prove that
\[(p_1, T - t, v_f) \in [(\tau) \ \sigma_i] \]

\( \text{IH} \)
\[(p_1, T, e \ \delta \gamma) \in [(c \rightarrow \tau) \ \sigma_i]_\xi \]

This means from Definition [15] we have
\[
\forall v', t' < T \cdot e \ \delta \gamma \downarrow_t v' \implies (p_1 + p_m, v') \in [(c \rightarrow \tau) \ \sigma_i]_\xi
\]

Since we know that \( (e \ |) \ \delta \gamma \downarrow_t v_f \) therefore from E-CE we know that \( \exists t', t.e \delta \gamma \downarrow_t \Lambda, e' \), an since \( t' < t < T \) therefore we have
\[(p_1, T - t', \Lambda, e') \in [(c \rightarrow \tau) \ \sigma_i]_\xi \)

Therefore from Definition [15] we have
\[. \vdash c \ i \implies (p_1, T - t', e' \ \delta \gamma) \in [\tau \ \sigma_i]_\xi \]

Since we are given \( \Theta; \Delta \vdash c \) and \( . \vdash \Delta \ i \) therefore we know that \( . \vdash c \ i \). Hence we get
\[(p_1, T - t', e' \ \delta \gamma) \in [\tau \ \sigma_i]_\xi \]

This means from Definition [15] we have
\[
\forall v_f, t'' < T - t'. (t'' \delta \gamma \downarrow_t v_f) \implies (p_1, T - t' - t'', v_f) \in [(\tau) \ \sigma_i]_\xi
\]

From Definition [15] we get \( (c \rightarrow \tau) \) s.t \( t = t' + t'' + 1 \) therefore instantiating (F-CE1) with the given \( v_f \) and \( t'' \) we get
\[(p_1, T - t' - t'', v_f) \in [(\tau) \ \sigma_i]_\xi \]

Since \( t = t' + t'' + 1 \) therefore from Lemma [31] we get the desired.

30. T-CanId:

\[
\Psi; \Theta; \Delta, e; \Gamma \vdash e : \tau \\
\Theta; \Delta \vdash e : c \quad \text{T-CanId}
\]

Given: \((p_1, T, \gamma) \in [\Gamma \ \sigma_i]_\xi, (0, T, \delta) \in [\Omega \ \sigma_i]_\xi \)
31. T-CAnE:

To prove: \((p_1, e \delta \gamma) \in [c \& \tau \sigma]_\text{E}\)

From Definition 15 it suffices to prove that
\(\forall v_f, t < T. \delta \gamma \triangledown_{t} v_f \Rightarrow (p_1, T - t, v_f) \in [c \& \tau \sigma]_\text{E}\)

This means given some \(v_f, t < T\) s.t \(e \delta \gamma \triangledown_{t} v_f\) it suffices to prove that
\((p_1, T - t, v_f) \in [c \& \tau \sigma]_\text{E}\)

From Definition 15 it suffices to prove that
\(\vdash e \wedge (p_1, T - t, v_f) \in [\tau \sigma]_\text{E}\)

Since we are given that \(\vdash \Delta \wedge \Theta; \Delta \vdash e\) therefore it suffices to prove that
\((p_1, T - t, v_f) \in [\tau \sigma]_\text{E}\)  (F-CAI0)

IH: \((p_1, T, e \delta \gamma) \in [\tau \sigma]_\text{E}\)

This means from Definition 15 we have
\(\forall t < T. e \delta \gamma \triangledown_{t} v_f \Rightarrow (p_1, T - t, v_f) \in [\tau \sigma]_\text{E}\)

Since we are given that \(e \delta \gamma \triangledown_{t} v_f\) therefore we get
\((p_1, T - t, v_f) \in [\tau \sigma]_\text{E}\)  (F-CAI1)

We get the desired from (F-CAI1)

\(\dashv\)

Given: \((p_1, T, \gamma) \in [(\Gamma_1 \oplus \Gamma_2) \sigma]_\text{E}, (0, T, \delta) \in [(\Omega) \sigma]_\text{E}\)

To prove: \((p_1, T, (\text{clet} x = e \in e' \delta \gamma)) \in [\tau' \sigma]_\text{E}\)

From Definition 15 it suffices to prove that
\(\forall v_f, t < T. (\text{clet} x = e \in e' \delta \gamma \triangledown_{t} v_f \Rightarrow (p_1, T - t, v_f) \in [\tau' \sigma]_\text{E}\)

This means given some \(v_f, t < T\) s.t \((\text{clet} x = e \in e' \delta \gamma \triangledown_{t} v_f\) it suffices to prove that
\((p_1, T - t, v_f) \in [\tau' \sigma]_\text{E}\)  (F-CAE0)

From Definition 15 and Definition 14 we know that \(\exists p_1, p_2, p_1 + p_2 = p_1\) s.t
\((p_1, T, \gamma) \in [(\Gamma_1) \sigma]_\text{E}\) and \((p_2, T, \gamma) \in [(\Gamma_2) \sigma]_\text{E}\)

\(\\)

IH1

\((p_1, T, e \delta \gamma) \in [c \& \tau \sigma]_\text{E}\)

This means from Definition 15 we have
\(\forall t < T. e \delta \gamma \triangledown_{t} v_1 \Rightarrow (p_1, T - t, v_1) \in [c \& \tau \sigma]_\text{E}\)

Since we know that \((\text{clet} x = e \in e' \delta \gamma \triangledown_{t} v_f)\) therefore from E-CAnE we know that \(\exists v_1, t_1 < t. e \delta \gamma \triangledown_{t_1} v_1\). Therefore we have
\((p_1, T - t_1, v_1) \in [c \& \tau \sigma]_\text{E}\)

Therefore from Definition 15 we have
\(\vdash \gamma \cup \{x \mapsto v_1\}\)  (F-CAE1)

\(\\)

IH2

\((p_2 + p_1, T, e' \delta \gamma') \in [\tau' \sigma]_\text{E}\)

where
\(\gamma' = \gamma \cup \{x \mapsto v_1\}\)

This means from Definition 15 we have
\(\forall t_2 < T. e' \delta \gamma' \triangledown_{t_2} v_f \Rightarrow (p_2 + p_1, T - t_2, v_f) \in [\tau' \sigma]_\text{E}\)

Since we know that \((\text{clet} x = e \in e' \delta \gamma \triangledown_{t_2} v_f)\) therefore from E-CAnE we know that \(\exists t_2 < t. e' \delta \gamma \triangledown_{t_2} v_f\). Therefore we have
\((p_2 + p_1, T - t_2, v_f) \in [\tau' \sigma]_\text{E}\)

Since \(p_1 = p_1 + p_2\) therefore we get
\((p_1, T - t_2, v_f) \in [\tau' \sigma]_\text{E}\)

And finally from From Lemma 61 we get
\((p_1, T - t, v_f) \in [\tau' \sigma]_\text{E}\)

And we are done.
32. **T-fix:**

\[ \Psi; \Theta; \Delta; \Omega; x : \tau ; \vdash e : \tau \]

\[ \Psi; \Theta; \Delta; \Omega; \vdash \text{fix}x.e : \tau \]  

**T-fix**

Given: \((0, T, \gamma) \in [\sigma]_\varepsilon\), \((0, T, \delta) \in [\Omega \sigma l]_\varepsilon\)

To prove: \((0, T, (\text{fix}x.e) \delta \gamma) \in [\tau l]_\varepsilon\)  

\((\text{F-FX0})\)

We induct on \(T\)

**Base case, \(T = 1\):**

It suffices to prove that \((0, 1, (\text{fix}x.e) \delta \gamma) \in [\tau \sigma l]_\varepsilon\)

This means from Definition 15 it suffices to prove

\[ \forall t < 1, (\text{fix}x.e) \delta \gamma \downarrow_v v \implies (0, 1 - t, v) \in [\tau \sigma l] \]

This further means that given \(t < 1\) s.t. \((\text{fix}x.e) \delta \gamma \downarrow v\) it suffices to prove that

\((0, 1 - t, v) \in [\tau \sigma l] \)

Since from E-fix we know that minimum value of \(t\) can be 1 therefore \(t < 1\) is not possible and the goal holds vacuously.

**Inductive case:**

**IH:** \((0, T - 1, (\text{fix}x.e) \delta \gamma) \in [\tau \sigma l]_\varepsilon\)

Therefore from Definition 16 we have

\((0, T - 1, \delta') \in [\Omega, x : \tau \sigma l]_\varepsilon\) where \(\delta' = \delta \cup \{x \mapsto \text{fix}x.e \delta\}\)

Applying Definition 15 on \((\text{F-FX0})\) it suffices to prove that

\[ \forall t < T, (\text{fix}x.e) \delta \gamma \downarrow_v v_f \implies (0, T - t, v_f) \in [\tau \sigma l] \]

This means given some \(t < T\) s.t. \((\text{fix}x.e) \delta \gamma \downarrow v\) it suffices to prove that

\((0, T - t, v_f) \in [\tau \sigma l] \)  

\((\text{F-FX0.0})\)

Now from IH of outer induction we have

\((0, T - 1, e \delta' \gamma) \in [\tau \sigma l]_\varepsilon\)

This means from Definition 15 we have

\[ \forall t' < T - 1, e \delta' \gamma \downarrow_v v_f \implies (0, T - 1 - t', v_f) \in [\tau \sigma l] \]

Since we know that \((\text{fix}x.e \delta \gamma \downarrow v)\) therefore from E-fix we know that \(\exists t' = t - 1\) s.t. \(e \delta' \gamma \downarrow v\)

Since \(t < T\) therefore \(t' = t - 1 < T - 1\) hence we have

\((0, T - t, v_f) \in [\tau \sigma l] \)

Therefore we are done

33. **T-ret:**

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \]

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{ret} e : M \cdot 0 \tau \]  

**T-ret**

Given: \((p_l, T, \gamma) \in [\Gamma \sigma l]_\varepsilon\), \((0, T, \delta) \in [\Omega \sigma l]_\varepsilon\)

To prove: \((p_l, T, \text{ret} e \delta) \in [M \cdot 0 \tau \sigma l]_\varepsilon\)

From Definition 15 it suffices to prove that

\[ \forall t < T, v_f.(\text{ret} e) \delta \gamma \downarrow_t v_f \implies (p_l, T - t, v_f) \in [M \cdot 0 \tau \sigma l] \]

It means we are given some \(t < T, v_f\) s.t. \((\text{ret} e) \delta \gamma \downarrow_t v_f\). From E-val we know that \(t = 0\) and \(v_f = (\text{ret} e) \delta \gamma\).

Therefore it suffices to prove that

\((p_l, T, (\text{ret} e) \delta) \in [M \cdot 0 \tau \sigma l] \)

From Definition 15 it further suffices to prove that

\[ \forall t < T, (\text{ret} e) \delta \gamma \downarrow_t v_f \implies \exists p'.n' + p' \leq p_l \land (p', T - t', v_f) \in [\tau \sigma l] \]

This means given some \(t' < T\) s.t. \((\text{ret} e) \delta \gamma \downarrow v_f\) it suffices to prove that

\[ \exists p'.n' + p' \leq p_l \land (p', T - t', v_f) \in [\tau \sigma l] \]

From \((\text{E-ret})\) we know that \(n' = 0\) therefore we choose \(p'\) as \(p_l\) and it suffices to prove that

\((p_l, T - t', v_f) \in [\tau \sigma l] \)  

\((\text{F-R0})\)

**IH**

\((p_l, T, e \delta) \in [\tau \sigma l]_\varepsilon\)
This means from Definition 15 we have
\[ \forall t < T . (e) \delta \gamma \Downarrow_{t} v_f \implies (p_t, T - t_1, v_f) \in [\tau \sigma_t] \]

Since we know that \( (\text{ret } e) \delta \gamma \Downarrow^0_{t} v_f \) therefore from (E-ret) we know that \( \exists t_1.e \delta \gamma \Downarrow_{t_1} v_f \)

Since \( t_1 < t < T \) therefore we have
\[ (p_t, T - t_1, v_f) \in [\tau \sigma_t] \]

And finally from Lemma 61 we get
\[ (p_t, T - t, v_f) \in [\tau \sigma_t] \]

and we are done.

34. T-bind:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma_1 + \Gamma_2 + e_1 : M \mid n_1 \tau_1 \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 + x : \tau_1 \vdash e_2 : M \mid n_2 \tau_2 \quad \Theta \vdash n_1 : \mathbb{R}^+ \quad \Theta \vdash n_2 : \mathbb{R}^+ \quad \text{T-bind}
\]

Given: \( (p_t, T, \gamma) \in \llbracket (\Gamma_1 \cup \Gamma_2) \sigma_t \rrbracket_E \)

To prove: \( (p_t, T, \text{bind } x = e_1 \text{ in } e_2 \delta \gamma) \in \llbracket M(n_1 + n_2) \tau_2 \sigma_t \rrbracket_E \)

From Definition 15 it suffices to prove that
\[ \forall t < T, v.(\text{bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{\tau_2} v = (p_t, T - t, v) \in \llbracket M(n_1 + n_2) \tau_2 \sigma_t \rrbracket_E \]

This means given some \( t < T, v \) s.t \( \text{ bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{\tau_2} v \). From E-val we know that \( t = 0 \) and \( v = (\text{bind } x = e_1 \text{ in } e_2 \delta \gamma) \)

Therefore it suffices to prove that
\[ (p_t, T, (\text{bind } x = e_1 \text{ in } e_2 \delta \gamma)) \in \llbracket M(n_1 + n_2) \tau_2 \sigma_t \rrbracket_E \]

This means from Definition 15 it suffices to prove that
\[ \forall t' < T, v_f.(\text{bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{\tau_2} v_f = \exists p', \tau_2 + p' \leq p_t + n \land (p', T - t', v_f) \in \llbracket \tau_2 \sigma_t \rrbracket_E \]

This means given some \( t' < T, v_f \) s.t \( \text{ bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{\tau_2} v_f \) and we need to prove that
\[ \exists p', s' \leq p_t + n \land (p', T - t', v_f) \in \llbracket \tau_2 \sigma_t \rrbracket_E \] (F-B0)

From Definition 15 and Definition 14 we know that \( \exists p_{t_1}, p_{t_2}. p_{t_1} + p_{t_2} = p_t \) s.t
\[ (p_{t_1}, \gamma) \in \llbracket (\Gamma_1) \sigma_t \rrbracket_E \text{ and } (p_{t_2}, \gamma) \in \llbracket (\Gamma_2) \sigma_t \rrbracket_E \]

**H1**
\[ (p_{t_1}, T, e_1 \delta \gamma) \in \llbracket M(n_1) \tau_1 \sigma_t \rrbracket_E \]

From Definition 15 it means we have
\[ \forall t_1 < T . (e_1) \delta \gamma \Downarrow_{t_1} v_{m_1} = (p_{t_1}, T - t_1, v_{m_1}) \in \llbracket M(n_1) \tau_1 \sigma_t \rrbracket_E \]

Since we know that \( \text{ bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{t_1} v_{m_1} \)

Since \( t_1 < t' < T \), therefore we have
\[ (p_{t_1}, T - t_1, v_{m_1}) \in \llbracket M(n_1) \tau_1 \sigma_t \rrbracket_E \] (F-B1)

This means from Definition 15 we are given that
\[ \forall t_1 < T - t_1, v_{m_1} \Downarrow_{e_1} v_1 \implies \exists p_1, s_1 + p_1 \leq p_{t_1} + n_1 \land (p_1', T - t_1 - t_1', v_1) \in \llbracket \tau_1 \sigma_t \rrbracket_E \]

Since we know that \( \text{ bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{t_1} v_1 \)

Since \( t_1' < t - t_1 < T - t_1 \) therefore means we have
\[ \exists p_1', s_1 + p_1' \leq p_{t_1} + n_1 \land (p_1', T - t_1 - t_1', v_1) \in \llbracket \tau_1 \sigma_t \rrbracket_E \] (F-B1)

**H2**
\[ (p_{t_2} + p_{t_1}', T - t_1 - t_1', e_2 \delta \gamma \cup \{ x \mapsto v_1 \}) \in \llbracket M(n_2) \tau_2 \sigma_t \rrbracket_E \]

From Definition 15 it means we have
\[ \forall t_2 < T - t_1 - t_1'.(e_2) \delta \gamma \cup \{ x \mapsto v_1 \} \Downarrow_{t_2} v_{m_2} = (p_{t_2} + p_{t_1}', T - t_1 - t_1' - t_2, v_{m_2}) \in \llbracket M(n_2) \tau_2 \sigma_t \rrbracket_E \]

Since we know that \( \text{ bind } x = e_1 \text{ in } e_2 \delta \gamma \Downarrow_{t_1} v_{m_2} \)

Since \( t_2 < t' - t_1 - t_1' < T - t_1 - t_1' \) therefore we have
\[ (p_{t_2} + p_{t_1}', T - t_1 - t_1' - t_2, v_{m_2}) \in \llbracket M(n_2) \tau_2 \sigma_t \rrbracket \]

This means from Definition 15 we are given that
∀t′ < T − t_1 − t_1′ − t_2,v_{n2} \bullet_{t_2} v_2 \implies \exists p'_2,s_2 + p'_2 \leq p_2 + p_1 + n_2 \land (p'_2,T − t_1 − t_1′ − t_2,v_2) \in [\tau_2 \sigma t]$

Since we know that (bind \( x = e_1 \) in \( e_2 \)) \( \delta \gamma \parallel_t v \) therefore from E-bind we know that \( \exists t'_2 < t' - t_1 - t_1′ - t_2,v_2,v_{n2} \bullet_{t_2} v_2 \).

This means we have

\[ \exists p'_2,s_2 + p'_2 \leq p_2 + p_1 + n_2 \land (p'_2,T − t_1 − t_1′ − t_2 − t_2′,v_2) \in [\tau_2 \sigma t] \quad (F-B2) \]

In order to prove (F-B0) we choose \( p' \) as \( p'_2 \) and it suffices to prove

(a) \( s' + p'_2 \leq p_1 + n \):

Since from (F-B2) we know that

\( s_2 + p'_2 \leq p_2 + p_1 + n_2 \)

Adding \( s_1 \) on both sides we get

\( s_1 + s_2 + p'_2 \leq p_2 + s_1 + p_1 + n_2 \)

Since from (F-B1) we know that

\( s_1 + p_1' \leq p_1 + n_1 \)

therefore we also have

\( s_1 + s_2 + p'_2 \leq p_2 + p_{11} + n_1 + n_2 \)

And finally since we know that \( n = n_1 + n_2, s' = s_1 + s_2 \) and \( p_1 = p_{11} + p_{22} \) therefore we get the desired

(b) \( (p'_2,T − t_1 − t_1′ − t_2 − t_2′,v_f) \in [\tau_2 \sigma t] \):

From E-bind we know that \( v_f = v_2 \) therefore we get the desired from (F-B2)

35. T-tick:

\[ \Theta \vdash n : \mathbb{R}^+ \]

\[ \Psi, \Theta ; \Delta ; \Omega ; \Gamma \vdash \tau^n : \mathbb{M} n \ 1 \]

T-tick

Given: \( (p_1,T,\gamma) \in [\Gamma \sigma t]_E, (0,T,\delta) \in [\Omega \sigma t]_E \)

To prove: \( (p_1,T,\tau^n,\delta,\gamma) \in [\mathbb{M} n \ 1 \ \sigma t]_E \)

From Definition \([15]\) it suffices to prove that

\[ \forall t < T,v.(\tau^n) \delta \gamma \parallel_t v \implies (p_1,T − t,v) \in [\mathbb{M} n \ 1 \ \sigma t] \]

This means we are given some \( t < T,v \) s.t. \( (\tau^n) \delta \gamma \parallel_t v \). From E-val we know that \( t = 0 \) and \( v = (\tau^n) \delta \gamma \)

Therefore it suffices to prove that

\( (p_1,T,\tau^n,\delta,\gamma) \in [\mathbb{M} n \ 1 \ \sigma t]_E \)

From Definition \([15]\) it suffices to prove that

\[ \forall t' < T.(\tau^n) \delta \gamma \parallel_{t'} (v) \implies \exists p',n' + p' \leq p_1 + n \land (p',T − t',()) \in [1] \]

This means given some \( t' < T \) s.t. \( (\tau^n) \delta \gamma \parallel_{t'} (v) \) it suffices to prove that

\( \exists p',n' + p' \leq p_1 + n \land (p',T − t',()) \in [1] \)

From (E-tick) we know that \( n' = n \) therefore we choose \( p' \) as \( p_1 \) and it suffices to prove that

\( (p_1,T,\tau'_n,()) \in [1] \)

We get this directly from Definition \([15]\)

36. T-release:

\[ \Psi, \Theta ; \Delta ; \Omega ; \Gamma_1 \vdash e_1 : [n_1] \tau_1 \]

\[ \Psi, \Theta ; \Delta ; \Omega ; \Gamma_2 ; x : \tau_1 \vdash e_2 : \mathbb{M}(n_1 + n_2) \tau_2 \]

\[ \Theta \vdash \tau : \mathbb{R}^+ \]

\[ \Theta \vdash n_2 : \mathbb{R}^+ \]

\[ \Psi, \Theta ; \Delta ; \Omega ; \Gamma_1 \vdash \text{release } x = e_1 \text{ in } e_2 : \mathbb{M}(n_2) \tau_2 \]

T-release

Given: \( (p_1,T,\gamma) \in [(\Gamma_1 \oplus \Gamma_2) \sigma t]_E, (0,T,\delta) \in [\Omega \sigma t]_E \)

To prove: \( (p_1,T,\text{release } x = e_1 \text{ in } e_2 \delta,\gamma) \in [\mathbb{M}(n_2) \tau_2 \sigma t]_E \)

From Definition \([15]\) it suffices to prove that

\[ \forall t < T,v.(\text{release } x = e_1 \text{ in } e_2) \delta \gamma \parallel_t v \implies (p_1,T − t,v) \in [\mathbb{M}(n_2) \tau_2 \sigma t] \]

This means given some \( t < T,v \) s.t. \( (\text{release } x = e_1 \text{ in } e_2) \delta \gamma \parallel_t v \). From E-val we know that \( t = 0 \) and \( v = (\text{release } x = e_1 \text{ in } e_2) \delta \gamma \)

Therefore it suffices to prove that

\( (p_1,T,\text{release } x = e_1 \text{ in } e_2 \delta,\gamma) \in [\mathbb{M}(n_2) \tau_2 \sigma t] \)

This means from Definition \([15]\) it suffices to prove that

43
∀t′ < T, v.f.(release x = e₁ in e₂ δ) ⊨ v′ f f ⇒ ∃p′, s′ + p′ ≤ p₁ + n₂ ∧ (p′, T − t′, v.f) ∈ [t₂ σ₁]

This means given some t′ < T, v.f s.t (release x = e₁ in e₂ δ) ⊨ v′ f f and we need to prove that ∃p′, s′ + p′ ≤ p₁ + n₂ ∧ (p′, T − t′, v.f) ∈ [t₂ σ₁]

(F-R0)

From Definition 15 and Definition 14 we know that ∃p₁₁, p₁₂ p₁₁ + p₁₂ = p₁ s.t (p₁₁, γ) ∈ [((Γ₁)σ₁)₂] and (p₁₂, γ) ∈ [((Γ₂)σ₁)₂]

IH1
(p₁, T, e₁ δ) ∈ [[n₁] σ₁]₂

From Definition 15 it means we have
∀t₁ < T . (e₁) δ ⊨ v₁ ⇒ (p₁₁, T − t₁, v₁) ∈ [[n₁] σ₁]

Since we know that (release x = e₁ in e₂ δ) ⊨ v¹ f f therefore from E-rel we know that ∃t₁ < t′ (e₁) δ ⊨ v¹, v₁.

Since t₁ < t′ < T, therefore we have
(p₁, T − t₁, v₁) ∈ [[n₁] σ₁]

This means from Definition 15 we have
∃p₁, p₁ + p₁ ≤ p₁ ∧ (p₁, T − t₁, v₁) ∈ [t₁]

(F-R1)

IH2
(p₁ + p₁′, T − t₁, e₂ δ ⊨ v₁) ∈ [[M(n₁ + n₂) σ₁]₂]

From Definition 15 it means we have
∀t₂ < T − t₁ (e₂) δ ⊨ v₂, v₁m₂ v₁ = v₂ ⇒ (p₁ + p₁′, T − t₂, v₁m₂) ∈ [[M(n₁ + n₂) σ₁]

Since we know that (release x = e₁ in e₂ δ) ⊨ v¹ f f therefore from E-rel we know that ∃t₂ < t₁, (e₂) δ ⊨ v₁, v₂.

This means we have
(p₁ + p₁′, T − t₂, v₁m₂) ∈ [[M(n₁ + n₂) σ₁]

(F-R2)

In order to prove (F-R0) we choose p′ as p₂ and it suffices to prove
(a) s′ + p₂ ≤ p₁ + n₂:
Since from (F-R2) we know that
s₂ + p₂ ≤ p₁ + p₁′ + n₁ + n₂
Since from (F-R1) we know that
p₁′ + n₁ ≤ p₁

therefore we also have
s₂ + p₂ ≤ p₂ + p₁ + n₁ + n₂

And finally since we know that s₂ = s₂, p₁ = p₁ + p₂ and 0 = p₃ we therefore get the desired
(b) (p₂, T − t₂, t₂′, v.f) ∈ [t₂ σ₁]

From E-rel we know that v₂ = v₂ therefore we get the desired from (F-R2)

37. T-store:

Ψ; Θ; Δ; Ω; Γ ⊢ e : τ → Θ ⊢ n : R⁺
Ψ; Θ; Δ; Ω; Γ ⊢ store e : n ([n] τ)

T-store

Given: (p₁, T, γ) ∈ [Γ σ₁]₂, (0, T, δ) ∈ [Ω σ₁]₂

To prove: (p₁, T, store e δ) ∈ [[M n ([n] τ) σ₁]₂

From Definition 15 it suffices to prove that
∀t < T, v.(store e) δ ⊨ v ⇒ (p₁, T − t, v) ∈ [[M n ([n] τ) σ₁]

This means we are given some t < T, v s.t (store e) δ ⊨ v, v. From E-val we know that t = 0 and v = (store e) δ

Therefore it suffices to prove that
(p, T, (store e) δγ) ∈ [[M n (τ)] σi t]

From Definition 15 it suffices to prove that
∀v′ < T, v′, v′′. (store e) δγ v′′ v′ =⇒ p′ < p ∧ (p′, T − t′, v) ∈ [[n] τ σi t]

This means given some v′ < T, v, v′′ s.t. (store e) δγ v′′ v′ if it suffices to prove that
∀v′ < T, v′, v′′. (store e) δγ v′′ v′ =⇒ v′′ ≤ v′ ∧ (v′, T − t′, v) ∈ [[n] τ σi t]

From (E-store) we know that n′ = 0 therefore we choose p′ as p + n and it suffices to prove that
(p + n, T − t′, v) ∈ [[n] τ σi t]

This further means that from Definition 15 we have
∃v′ < T, v′, v′′ =⇒ p′ < p ∧ (p′, T − t′, v) ∈ [[n] τ σi t]

We choose p′ as p and it suffices to prove that
(p + n, T − t′, v) ∈ [[n] τ σi t] (F-S0)

IH
(p, T, e δγ) ∈ [[τ σi t] e

This means from Definition 15 we have
∀v′ < T, T, e δγ v′, v =⇒ (p, T − l1, v) ∈ [[τ σi t]

Since we know that (store e) δγ v′, v therefore from (E-store) we know that ∃v′ < T, e δγ v′, v

Since v′ < T therefore we have
(p, T − l1, v) ∈ [[τ σi t]

and finally from Lemma 11 we have
(p, T − t′, v) ∈ [[τ σi t]

Lemma 21 (Value subtyping lemma). ∀Ψ, Θ, τ ∈ Type, ν, ∀Ψ, Θ, Δ ⊢ τ <: τ′ ∧ ∨ Δ =⇒ [[τ σi t]] ⊆ [[τ′ σi t]]

Proof. Proof by induction on the Ψ; Θ; Δ ⊢ τ <: τ′ relation
1. sub-refl:

To prove: ∀(p, T, v) ∈ [[τ σi t]] =⇒ (p, T, v) ∈ [[τ σi t]]

Trivial

2. sub-arrow:

To prove: ∀(p, T, x, e) ∈ [[τ σi t]] =⇒ (p, T, λx.e) ∈ [[(τ σi t)]]

This means given some (p, T, λx.e) ∈ ((τ σi t)] we need to prove
(p, T, λx.e) ∈ (((τ σi t)]

From Definition 15 we are given that
∀p′ < T, p′, e′. (p, T, e) ∈ [[τ σi t] e =⇒ (p + p′, T′, e′) ∈ [[τ σi t] e (F-SL0)

Also from Definition 15 it suffices to prove that
∀p′′ < T, p′′, e′′. (p′′, T′′, e′′) ∈ [[τ σi t] e =⇒ (p + p′′, T′′, e′′) ∈ [[τ σi t] e (F-SL1)

IH1: [[τ σi t]] ⊆ [[τ σi t]]

Since we have (p′′, T′′, e′′) ∈ [[τ σi t]] therefore from IH1 we also have (p′′, T′′, e′′) ∈ [[τ σi t]]

Therefore instantiating (F-SL0) with p′, T′, e′ we get
(p + p′, T′, e′) ∈ [[τ σi t] e

And finally from Lemma 11 we get the desired
3. sub-tensor:

\[
\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau'_1 \quad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau'_2}{\Psi; \Theta; \Delta \vdash \tau_1 \otimes \tau_2 <: \tau'_1 \otimes \tau'_2} \quad \text{sub-tensor}
\]

To prove: \( \forall (p, T, \langle v_1, v_2 \rangle) \in [(\tau_1 \otimes \tau_2) \sigma] \implies (p, T, \langle v_1, v_2 \rangle) \in [(\tau'_1 \otimes \tau'_2) \sigma] \)

This means given \( (p, T, \langle v_1, v_2 \rangle) \in [(\tau_1 \otimes \tau_2) \sigma] \)

It suffices to prove that \( (p, T, \langle v_1, v_2 \rangle) \in [(\tau'_1 \otimes \tau'_2) \sigma] \)

This means from Definition [15] we are given that \( \exists p_1, p_2, p_1 + p_2 \leq p \land (p_1, T, v_1) \in [\tau_1 \sigma] \land (p_2, T, v_2) \in [\tau_2 \sigma] \)

Also from Definition [15] it suffices to prove that \( \exists p'_1, p'_2, p'_1 + p'_2 \leq p \land (p'_1, T, v_1) \in [\tau'_1 \sigma] \land (p'_2, T, v_2) \in [\tau'_2 \sigma] \)

IH1 \( [(\tau_1) \sigma] \subseteq [(\tau'_1) \sigma] \)
IH2 \( [(\tau_2) \sigma] \subseteq [(\tau'_2) \sigma] \)

Choosing \( p_1 \) for \( p'_1 \) and \( p_2 \) for \( p'_2 \) we get the desired from IH1 and IH2

4. sub-with:

\[
\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau'_1 \quad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau'_2}{\Psi; \Theta; \Delta \vdash \tau_1 \& \tau_2 <: \tau'_1 \& \tau'_2} \quad \text{sub-with}
\]

To prove: \( \forall (p, T, \langle v_1, v_2 \rangle) \in [(\tau_1 \& \tau_2) \sigma] \implies (p, T, \langle v_1, v_2 \rangle) \in [(\tau'_1 \& \tau'_2) \sigma] \)

This means given \( (p, T, \langle v_1, v_2 \rangle) \in [(\tau_1 \& \tau_2) \sigma] \)

It suffices to prove that \( (p, T, \langle v_1, v_2 \rangle) \in [(\tau'_1 \& \tau'_2) \sigma] \)

This means from Definition [15] we are given that \( (p, T, v_1) \in [\tau_1 \sigma] \land (p, T, v_2) \in [\tau_2 \sigma] \)

(F-SW0)

Also from Definition [15] it suffices to prove that \( (p, T, v_1) \in [\tau'_1 \sigma] \land (p, T, v_2) \in [\tau'_2 \sigma] \)

IH1 \( [(\tau_1) \sigma] \subseteq [(\tau'_1) \sigma] \)
IH2 \( [(\tau_2) \sigma] \subseteq [(\tau'_2) \sigma] \)

We get the desired from (F-SW0), IH1 and IH2

5. sub-sum:

\[
\frac{\Psi; \Theta; \Delta \vdash \tau_1 <: \tau'_1 \quad \Psi; \Theta; \Delta \vdash \tau_2 <: \tau'_2}{\Psi; \Theta; \Delta \vdash \tau_1 \oplus \tau_2 <: \tau'_1 \oplus \tau'_2} \quad \text{sub-sum}
\]

To prove: \( \forall (p, T, \langle v_1, v_2 \rangle) \in [(\tau_1 \oplus \tau_2) \sigma] \implies (p, T, \langle v_1, v_2 \rangle) \in [(\tau'_1 \oplus \tau'_2) \sigma] \)

This means given \( (p, T, v) \in [(\tau_1 \oplus \tau_2) \sigma] \)

It suffices to prove that \( (p, T, v) \in [(\tau'_1 \oplus \tau'_2) \sigma] \)

This means from Definition [15] two cases arise

(a) \( v = \text{inl}(v') \):

This means from Definition [15] we have \( (p, T, v') \in [\tau_1 \sigma] \) \quad (F-SW0)

Also from Definition [15] it suffices to prove that

\( (p, T, v') \in [\tau'_1 \sigma] \)

IH \( [(\tau_1) \sigma] \subseteq [(\tau'_1) \sigma] \)

We get the desired from (F-SW0), IH

(b) \( v = \text{inr}(v') \):

Symmetric reasoning as in the inl case
6. sub-list:

\[
\Psi; \Theta; \Delta \vdash \tau <: \tau'
\]

\[
\Psi; \Theta; \Delta \vdash L^n \tau <: L^n \tau'
\]

sub-list

To prove: \(\forall (p, T, v) \in [[L^n \tau \sigma i]], (p, T, v) \in [[L^n \tau' \sigma i]]\)

This means given \((p, T, v) \in [L^n \tau \sigma i]\) and we need to prove
\((p, T, v) \in [L^n \tau' \sigma i]\)

We induct on \((p, T, v) \in [L^n \tau \sigma i]\)

(a) \((p, T, nil) \in [L^0 \tau \sigma i]:\)

We need to prove \((p, T, nil) \in [L^0 \tau' \sigma i]\)

We get this directly from Definition 15

(b) \((p, T, v'::l') \in [L^{m+1} \tau \sigma i]:\)

In this case we are given \((p, T, v'::l') \in [L^{m+1} \tau \sigma i]\)

and we need to prove \((p, T, v'::l') \in [L^{m+1} \tau' \sigma i]\)

This means from Definition 15 are given
\(\exists p_1, p_2: p_1 + p_2 \leq p \land (p_1, T, v') \in [\tau \sigma i] \land (p_2, T, l') \in [L^m \tau \sigma i]\) (Sub-List0)

Similarly from Definition 15 we need to prove that
\(\exists p_1', p_2': p_1' + p_2' \leq p \land (p_1', T, v') \in [\tau' \sigma i] \land (p_2', T, l') \in [L^m \tau' \sigma i]\)

We choose \(p_1'\) as \(p_1\) and \(p_2'\) as \(p_2\) and we get the desired from (Sub-List0) IH of outer induction and IH of inner induction

7. sub-exist:

\[
\Psi; \Theta, s; \Delta \vdash \tau <: \tau'
\]

\[
\Psi; \Theta; \Delta \vdash \exists s. \tau <: \exists s. \tau'
\]

sub-exist

To prove: \(\forall (p, T, v) \in [[\exists s. \tau \sigma i]], (p, T, v) \in [[\exists s. \tau' \sigma i]]\)

This means given some \((p, T, v) \in [[\exists s. \tau \sigma i]]\) we need to prove
\((p, T, v) \in [[\exists s. \tau' \sigma i]]\)

From Definition 15 we are given that
\(\exists s'. (p, T, v) \in [[\tau \sigma i[s'/s]]\) (F-exist0)

IH: \([[\tau] \sigma i \cup \{s \mapsto s'\}] \subseteq [[\tau'] \sigma i \cup \{s \mapsto s'\}]\)

Also from Definition 15 it suffices to prove that
\(\exists s''. (p, T, v) \in [[\tau' \sigma i[s''/s]]\)

We choose \(s''\) as \(s'\) and we get the desired from IH

8. sub-potential:

\[
\Psi; \Theta, s; \Delta \vdash n <: n'
\]

\[
\Psi; \Theta; \Delta \vdash [n] \tau <: [n'] \tau'
\]

sub-potential

To prove: \(\forall (p, T, v) \in [[[n] \tau \sigma i]], (p, T, v) \in [[[n'] \tau' \sigma i]]\)

This means given \((p, T, v) \in [[[n] \tau \sigma i]]\) we need to prove
\((p, T, v) \in [[[n'] \tau' \sigma i]]\)

This means from Definition 15 we are given
\(\exists p', p'' + n \leq p \land (p', T, v) \in [\tau \sigma i]\) (F-SP0)

And we need to prove
\(\exists p'', p'' + n' \leq p \land (p'', T, v) \in [\tau' \sigma i]\) (F-SP1)

In order to prove (F-SP1) we choose \(p''\) as \(p'\)

Since from (F-SP0) we know that \(p' + n \leq p\) and we are given that \(n' \leq n\) therefore we also have \(p' + n' \leq p\)

IH: \([[\tau \sigma i]] \subseteq [[\tau' \sigma i]]\)

We get the desired directly from IH
9. sub-monad:

\[ \Psi; \Theta; \Delta \vdash \tau <: \tau' \]
\[ \Psi; \Theta; \Delta \vdash m n \tau <: m n' \tau' \]

sub-monad

To prove: \( \forall (p, T, v) \in [\text{MM } n \tau \sigma_l].(p, T, v) \in [\text{MM } n' \tau' \sigma_l] \)

This means given \( (p, T, v) \in [\text{MM } n \tau \sigma_l] \) and we need to prove
\( (p, T, v) \in [\text{MM } n' \tau' \sigma_l] \)

This means from Definition \([15]\) we are given
\( \forall \tau' < T, n_1, v. \psi_{\tau, n_1}^{\tau', v} \implies \exists p_1. n_1 + p_1 \leq n + (p', T - t', v') \in [\tau \sigma_l] \) (F-SM0)

Again from Definition \([15]\) we need to prove that
\( \forall \tau'' < T, n_2, v', v. \psi_{\tau, n_2}^{\tau'', v'} \implies \exists p_2. n_2 + p'' \leq p + n' \land (p'', T - t'', v'') \in [\tau' \sigma_l] \)

This means given some \( \tau'' < T, v', n_2 \) s.t. \( v. \psi_{\tau, n_2}^{\tau'', v'} \) it suffices to prove that
\( \exists p'. n_2 + p' \leq p + n' \land (p', T - t'', v'') \in [\tau' \sigma_l] \) (F-SM1)

Instantiating (F-SM0) with \( t'', n_2, v'' \) Since \( v. \psi_{\tau, n_2}^{\tau'', v''} \) therefore from (F-SM0) we know that
\( \exists p'. n_2 + p' \leq p + n \land (p', T - t'', v'') \in [\tau' \sigma_l] \) (F-SM2)

\( \text{IH} [\tau \sigma_l] \subseteq [\tau' \sigma_l] \)

In order to prove (F-SM1) we choose \( p'' \) as \( p' \) and we need to prove
(a) \( n_2 + p'' \leq p + n' \)
Since we are given that \( n \leq n' \) therefore we get the desired from (F-SM2)
(b) \( (p', v') \in [\tau' \sigma_l] \)
We get this directly from IH and (F-SM2)

10. sub-Exp:

\[ \Psi; \Theta; \Delta \vdash \tau <: \tau' \]

sub-Exp

To prove: \( \forall (p, T, v) \in [\text{!}\tau \sigma_l].(p, T, v) \in [\text{!}\tau' \sigma_l] \)

This means given \( (p, T, v) \in [\text{!}\tau \sigma_l] \) and we need to prove
\( (p, T, v) \in [\text{!}\tau' \sigma_l] \)

This means from Definition \([15]\) we are given
\( (0, T, e) \in [\tau \sigma_l] \) (F-SE0)

Again from Definition \([15]\) we need to prove that
\( (0, T, e) \in [\tau' \sigma_l] \) (F-SE1)

\( \text{IH} [\tau \sigma_l] \subseteq [\tau' \sigma_l] \)

Therefore from (F-SE0) and IH we get \( (0, T, e) \in [\tau' \sigma_l] \) and we are done.

11. sub-typePoly:

\[ \Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 \]

sub-typePoly

To prove: \( \forall (p, T, \Lambda, e) \in [\forall \tau \sigma_l].(p, T, \Lambda, e) \in [\forall \tau_2 \sigma_l] \)

This means given some \( (p, T, \Lambda, e) \in [\forall \tau \sigma_l] \) we need to prove
\( (p, T, \Lambda, e) \in [\forall \tau_2 \sigma_l] \)

From Definition \([15]\) we are given that
\( \forall \tau, T' < T. (p, T', e) \in [\tau_1 [\tau/\alpha]] \) (F-STEP0)

Also from Definition \([15]\) it suffices to prove that
\( \forall \tau', T'' < T. (p, T'', e) \in [\tau_2 [\tau'/\alpha]] \)

This means given some \( \tau', T'' < T \) and we need to prove
\( (p, T'', e) \in [\tau_2 [\tau'/\alpha]] \) (F-STEP1)

\( \text{IH} [\tau_1 [\tau/\alpha] \cup \{ \alpha \rightarrow \tau' \}] \subseteq [\tau_2 [\tau'/\alpha] \cup \{ \alpha \rightarrow \tau' \}] \)

Instantiating (F-STEP0) with \( \tau', T'' \) we get
\( (p, T'', e) \in [\tau_1 [\tau'/\alpha]] \)

and finally from IH we get the desired.
12. sub-indexPoly:

\[
\Psi; \Theta; i; \Delta \vdash \tau_1 <: \tau_2 \quad \text{sub-indexPoly}
\]

To prove: \( \forall (p, T, \Lambda, e) \in \llbracket \forall i. \tau_1 \rrbracket \sigma_i, (p, T, \Lambda, e) \in \llbracket \forall i. \tau_2 \rrbracket \sigma_i \)

This means given some \((p, T, \Lambda, e) \in \llbracket \forall i. \tau_1 \rrbracket \sigma_i\) we need to prove \((p, T, \Lambda, e) \in \llbracket \forall i. \tau_2 \rrbracket \sigma_i\)

From Definition \[15\] we are given that
\( \forall I, T' < T. (p, T', e) \in [\tau_1[I/i]]E \)  \quad (F-SIP0)

Also from Definition \[15\] it suffices to prove that
\( \forall I', T'' < T. (p, T'', e) \in [\tau_2[I'/i]]E \)

This means given some \(I', T'' < T\) and we need to prove
\( (p, T'', e) \in [\tau_2[I'/i]]E \)  \quad (F-SIP1)

\( \text{IH: } \llbracket (\tau_1 \sigma_i \cup \{i \mapsto I'\}) \rrbracket \subseteq \llbracket (\tau_2 \sigma_i \cup \{i \mapsto I'\}) \rrbracket \)

Instantiating (F-SIP0) with \(I', T''\) we get
\( (p, T'', e) \in [\tau_1[I'/i]]E \) and finally from IH we get the desired

13. sub-constraint:

\[
\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 \quad \Theta; \Delta \vdash c_2 \implies c_1 \quad \text{sub-constraint}
\]

To prove: \( \forall (p, T, \Lambda, e) \in \llbracket (c_1 \Rightarrow \tau_1) \sigma_i \rrbracket, (p, T, \Lambda, e) \in \llbracket (c_2 \Rightarrow \tau_2) \sigma_i \rrbracket \)

This means given some \((p, T, \Lambda, e) \in \llbracket (c_1 \Rightarrow \tau_1) \sigma_i \rrbracket\) we need to prove
\((p, T, \Lambda, e) \in \llbracket (c_2 \Rightarrow \tau_2) \sigma_i \rrbracket\)

From Definition \[15\] we are given that
\( . \models c_{1i} \iff (p, T, e) \in [\tau_1 \sigma_i]E \)  \quad (F-SC0)

Also from Definition \[15\] it suffices to prove that
\( . \models c_{2i} \iff (p, T, e) \in [\tau_2 \sigma_i]E \)

This means given some \( . \models c_{2i} \) and we need to prove
\( (p, T, e) \in [\tau_2 \sigma_i]E \)  \quad (F-SC1)

Since we are given that \( \Theta; \Delta \vdash c_2 \implies c_1 \) therefore we know that \( . \models c_{1i} \)

Hence from (F-SC0) we have
\( (p, T, e) \in [\tau_1 \sigma_i]E \)  \quad (F-SC2)

\( \text{IH: } \llbracket (\tau_1 \sigma_i) \rrbracket \subseteq \llbracket (\tau_2 \sigma_i) \rrbracket \)

Therefore we get the desired from IH and (F-SC2)

14. sub-CAnd:

\[
\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 \quad \Theta; \Delta \vdash c_1 => c_2 \quad \text{sub-CAnd}
\]

To prove: \( \forall (p, v) \in \llbracket (c_1 \& \tau_1) \sigma_i \rrbracket, (p, v) \in \llbracket (c_2 \& \tau_2) \sigma_i \rrbracket \)

This means given some \((p, v) \in \llbracket (c_1 \& \tau_1) \sigma_i \rrbracket\) we need to prove
\((p, v) \in \llbracket (c_2 \& \tau_2) \sigma_i \rrbracket\)

From Definition \[15\] we are given that
\( . \models c_{1i} \& (p, e) \in [\tau_1 \sigma_i]E \)  \quad (F-SCA0)

Also from Definition \[15\] it suffices to prove that
\( . \models c_{2i} \& (p, e) \in [\tau_2 \sigma_i]E \)

Since we are given that \( \Theta; \Delta \vdash c_2 \implies c_1 \) and \( . \models c_{1i} \) therefore we also know that \( . \models c_{2i} \)
Also from (F-SCA0) we have \((p,e) \in \llbracket \tau_1 \sigma_i \rrbracket_E\) (F-SCA1)

\[\text{IH: } \llbracket (\tau_1) \sigma_i \rrbracket \subseteq \llbracket (\tau_2) \sigma_i \rrbracket\]

Therefore we get the desired from IH and (F-SCA1)

15. sub-familyAbs:

\[
\Psi; \Theta, i : S \vdash \tau <: \tau' \quad \text{sub-familyAbs}\]

To prove:

\[
\forall f \in \llbracket \lambda_i : S. \tau \rrbracket. f \in \llbracket \lambda_i : S. \tau' \rrbracket
\]

This means given \(f \in \llbracket \lambda_i : S. \tau \rrbracket\) and we need to prove

\(f \in \llbracket \lambda_i : S. \tau' \rrbracket\)

This means from Definition 15 we are given

\[\forall I.f I \in \llbracket \tau[I/i] \rrbracket\] (F-SFAbs0)

This means from Definition 15 we need to prove

\[\forall I'. f I' \in \llbracket \tau'[I'/i] \rrbracket\]

This further means that given some \(I'\) we need to prove

\(f I' \in \llbracket \tau'[I'/i] \rrbracket\) (F-SFAbs1)

Instantiating (F-SFAbs0) with \(I'\) we get

\(f I' \in \llbracket \tau'[I'/i] \rrbracket\)

From IH we know that \(\llbracket \tau \sigma_i \cup \{i \mapsto I' \} \rrbracket \subseteq \llbracket \tau' \sigma_i \cup \{i \mapsto I' \} \rrbracket\)

And this completes the proof.

16. Sub-tfamilyApp1:

\[
\Psi; \Theta; \Delta \vdash \lambda_i : S. \tau I <: \tau[I/i] \quad \text{sub-familyApp1}\]

To prove:

\[
\forall (p,T,v) \in \llbracket \lambda_i : S. \tau I \rrbracket. (p,T,v) \in \llbracket \tau[I/i] \rrbracket
\]

This means given \((p,T,v) \in \llbracket \lambda_i : S. \tau I \rrbracket\) and we need to prove

\((p,T,v) \in \llbracket \tau[I/i] \rrbracket\)

This means from Definition 15 we are given

\((p,T,v) \in \llbracket \lambda_i : S. \tau I \rrbracket\)

This further means that we have

\((p,T,v) \in f I \sigma_i \) where \(f I \sigma_i = \llbracket \tau[I/i] \rrbracket\)

This means we have \((p,T,v) \in \llbracket \tau[I/i] \rrbracket\)

And this completes the proof.

17. Sub-tfamilyApp2:

\[
\Psi; \Theta; \Delta \vdash \lambda_i : S. \tau I <: \lambda_i : S. \tau I \quad \text{sub-familyApp2}\]

To prove: \(\forall (p,T,v) \in \llbracket \tau[I/i] \rrbracket. (p,T,v) \in \llbracket \lambda_i : S. \tau I \rrbracket\)

This means given \((p,T,v) \in \llbracket \tau[I/i] \rrbracket\) (Sub-tF0)

And we need to prove

\((p,T,v) \in \llbracket \lambda_i : S. \tau I \rrbracket\)

This means from Definition 15 it suffices to prove that

\((p,T,v) \in \llbracket \lambda_i : S. \tau I \rrbracket\)

It further suffices to prove that

\((p,T,v) \in f I \sigma_i \) where \(f I \sigma_i = \llbracket \tau[I/i] \rrbracket\)

which means we need to show that

\((p,T,v) \in \llbracket \tau[I/i] \rrbracket\)

We get this directly from (Sub-tF0)

\[\square\]

Lemma 22 (Expression subtyping lemma). \(\forall \Psi, \Theta, \tau, \tau'. \Psi; \Theta \vdash \tau <: \tau' \implies \llbracket \tau \sigma_i \rrbracket_E \subseteq \llbracket \tau' \sigma_i \rrbracket_E\)
Proof. To prove: \( \forall (p, T, e) \in [\tau \_\sigma t]_E \Rightarrow (p, T, e) \in [\tau' \_\sigma t]_E \)
This means given some \((p, T, e) \in [\tau \_\sigma t]_E \) it suffices to prove that \((p, T, e) \in [\tau' \_\sigma t]_E \)
This means from Definition [15] we are given
\( \forall t < T, v, e \Downarrow v \Rightarrow (p, T - t, v) \in [\tau \_\sigma t] \) (S-E0)
Similarly from Definition [15] it suffices to prove that
\( \forall t' < T, v', e \Downarrow v' \Rightarrow (p, T - t, v') \in [\tau' \_\sigma t] \)
This means some \( t' < T, v' \; s.t. \; e \Downarrow v' \) it suffices to prove that
\((p, T - t', v') \in [\tau' \_\sigma t] \)
Instantiating (S-E0) with \( t', v' \) we get \((p, T - t', v') \in [\tau' \_\sigma t] \)
And finally from Lemma [21] we get the desired.

Lemma 23 (\( \Gamma \) subtyping lemma). \( \forall \Psi, \Theta, \Gamma_1, \Gamma_2, \sigma, \iota. \)
\( \Psi; \Theta \vdash \Gamma_1 <: \Gamma_2 \Rightarrow [\Gamma_1 \_\sigma t] \subseteq [\Gamma_2 \_\sigma t] \)

Proof. Proof by induction on \( \Psi; \Theta \vdash \Gamma_1 <: \Gamma_2 \)
1. sub-lBase:
\[
\begin{array}{c}
\psi; \Theta \vdash \Gamma <: \tau' \\
\hline
\psi; \Theta \vdash \tau' <: \tau \\
\psi; \Theta \vdash \Gamma_1 <: \Gamma_2 \\
\end{array}
\]
To prove: \( \forall (p, T, \gamma) \in [\Gamma_1 \_\sigma t]_E \; (p, T, \gamma) \in [\Gamma_2 \_\sigma t]_E \)
This means given some \( (p, T, \gamma) \in [\Gamma_1 \_\sigma t]_E \) it suffices to prove that \( (p, T, \gamma) \in [\Gamma_2 \_\sigma t]_E \)
From Definition [16] it suffices to prove that
\( \exists f : \text{Vars} \rightarrow \text{Pos}, (\forall x \in \text{dom}(\cdot). (f(x), T, \gamma(x)) \in [\Gamma(x)]_E) \land (\sum_{x \in \text{dom}(\cdot)} f(x) \leq p) \)
We choose \( f \) as a constant function \( f' = 0 \) and we get the desired
2. sub-lInd:
\[
\begin{array}{c}
x : \tau' \in \Gamma_1 \\
\psi; \Theta \vdash \tau' <: \tau \\
\psi; \Theta \vdash \Gamma_1 / x <: \Gamma_2 \\
\hline
\psi; \Theta \vdash \Gamma_1 \subset: \Gamma_2, x : \tau \\
\end{array}
\]
To prove: \( \forall (p, T, \gamma) \in [\Gamma_1 \_\sigma t]_E \; (p, T, \gamma) \in [\Gamma_2, x : \tau]_E \)
This means given some \( (p, T, \gamma) \in [\Gamma_1 \_\sigma t]_E \) it suffices to prove that \( (p, T, \gamma) \in [\Gamma_2, x : \tau]_E \)
This means from Definition [16] we are given that
\( \exists f : \text{Vars} \rightarrow \text{Pos}, (\forall x \in \text{dom}(\Gamma_2), f(x), T, \gamma(x)) \in [\Gamma(x)]_E \) (L0)
\( (\sum_{x \in \text{dom}(\Gamma_1)} f(x) \leq p) \) (L1)
Similarly from Definition [16] it suffices to prove that
\( \exists f' : \text{Vars} \rightarrow \text{Pos}, (\forall y \in \text{dom}(\Gamma_2, x : \tau), f'(y), T, \gamma(y)) \in [\Gamma_2, x : \tau(y)]_E \) \land \( (\sum_{y \in \text{dom}(\Gamma_2, x : \tau)} f'(y) \leq p) \)
We choose \( f' \) as \( f \) and it suffices to prove that
(a) \( \forall y \in \text{dom}(\Gamma_2, x : \tau), f'(y), T, \gamma(y)) \in [\Gamma_2, x : \tau(y)]_E : \)
This means some \( y \in \text{dom}(\Gamma_2, x : \tau) \) it suffices to prove that
\( (f(y), T, \gamma(y)) \in [\Gamma_2, x : \tau(y)]_E \) where say \( (\Gamma_2, x : \tau(y)) = \tau_2 \)
From Lemma [23] we know that
\( y : \tau_2 \in \Gamma_1 \land \psi; \Theta \vdash \tau_2 <: \tau_2 \)
By instantiating (L0) with the given \( y \)
\( (f(y), T, \gamma(y)) \in [\tau_2]_E \)
Finally from Lemma [22] we also get \( (f(y), T, \gamma(y)) \in [\tau_2]_E \)
And we are done.
(b) \( (\sum_{y \in \text{dom}(\Gamma_2, x : \tau)} f(y) \leq p) : \)
From (L1) we know that \( (\sum_{x \in \text{dom}(\Gamma_1)} f(x) \leq p) \) and since from Lemma [24] we know that \( \text{dom}(\Gamma_2, x : \tau) \subseteq \text{dom}(\Gamma_1) \) therefore we also have
\( (\sum_{y \in \text{dom}(\Gamma_2, x : \tau)} f(y) \leq p) \)

**Lemma 24** (Γ Subtyping: domain containment). \( \forall \rho, \gamma, \Gamma_1, \Gamma_2. \)
\[ \Psi; \Theta \vdash \Gamma_1 <\Gamma_2 \implies \forall x : \tau \in \Gamma_2, x : \tau' \in \Gamma_1 \land \Psi; \Theta \vdash \tau' <\tau \]

**Proof.** Proof by induction on \( \Psi; \Theta \vdash \Gamma_1 <\Gamma_2 \)

1. sub-lBase:
\[ \frac{\Psi; \Theta \vdash \Gamma_1 <\Gamma_2}{\Psi; \Theta \vdash \psi <\psi} \]

To prove: \( \forall x : \tau' \in (\_), x : \tau \in \Gamma_1 \land \Psi; \Theta \vdash \tau' <\tau \)

Trivial

2. sub-lInd:
\[ \frac{x : \tau' \in \Gamma_1 \quad \Psi; \Theta \vdash \tau' <\tau \quad \Psi; \Theta \vdash \Gamma_1/x <\Gamma_2}{\Psi; \Theta \vdash \Gamma_1 <\Gamma_2, x : \tau_x} \]

To prove: \( \forall y : \tau_1 \in (\Gamma_2, x : \tau_x), y : \tau \in \Gamma_1 \land \Psi; \Theta \vdash \tau' <\tau \)

This means given some \( y : \tau \in (\Gamma_2, x : \tau_x) \) it suffices to prove that \( y : \tau \in \Gamma_1 \land \Psi; \Theta \vdash \tau' <\tau \)

The following cases arise:

- \( y = x \):
  - In this case we are given that \( x : \tau' \in \Gamma_1 \land \Psi; \Theta \vdash \tau' <\tau \)
  - Therefore we are done

- \( y \neq x \):
  - Since we are given that \( \Psi; \Theta \vdash \Gamma_1/x <\Gamma_2 \) therefore we get the desired from IH

\( \square \)

**Lemma 25** (Ω subtyping lemma). \( \forall \Psi, \Theta, \Omega_1, \Omega_2, \sigma, \iota. \)
\[ \Psi; \Theta \vdash \Omega_1 <\Omega_2 \implies [\Omega_1 \sigma \iota] \subseteq [\Omega_2 \sigma \iota] \]

**Proof.** Proof by induction on \( \Psi; \Theta \vdash \Omega_1 <\Omega_2 \)

1. sub-lBase:
\[ \frac{\Psi; \Theta \vdash \Omega_1 <\Omega_2}{\Psi; \Theta \vdash \Omega <\Omega} \]

To prove: \( \forall (0, T, \delta) \in [\Omega_1 \sigma \iota], (0, T, \delta) \in [\Omega]_\varepsilon \)

This means given some \( (0, T, \delta) \in [\Omega_1 \sigma \iota] \) it suffices to prove that \( (0, T, \delta) \in [\Omega]_\varepsilon \)

We get the desired directly from Definition [10]

2. sub-lInd:
\[ \frac{x : \tau' \in \Omega_1 \quad \Psi; \Theta \vdash \tau' <\tau \quad \Psi; \Theta \vdash \Omega_1/x <\Omega_2}{\Psi; \Theta \vdash \Omega_1 <\Omega_2, x : \tau} \]

To prove: \( \forall (0, T, \delta) \in [\Omega_1 \sigma \iota], (0, T, \delta) \in [\Omega_2, x : \tau]_\varepsilon \)

This means given some \( (0, T, \delta) \in [\Omega_1 \sigma \iota] \) it suffices to prove that \( (0, T, \delta) \in [\Omega_2, x : \tau]_\varepsilon \)

This means from Definition [10], we are given that
\[ (\forall x : \tau \in \Omega_1, (0, T, \delta(x)) \in [\tau]_\varepsilon) \quad (L0) \]

Similarly from Definition [10], it suffices to prove that
\[ (\forall y : \tau_y \in (\Omega_2, x : \tau), (0, T, \delta(y)) \in [\tau_y]_\varepsilon) \]

This means given some \( y : \tau_y \in (\Omega_2, x : \tau) \) it suffices to prove that
\[ (0, T, \delta(y)) \in [\tau_y]_\varepsilon \]

From Lemma [26] we know that \( \exists \tau', y : \tau' \in dom(\Omega_1) \land \Psi; \Theta \vdash \tau' <\tau_y \)

Instantiating (L0) with \( y : \tau' \) we get \( (0, T, \delta(y)) \in [\tau']_\varepsilon \)

And finally from Lemma [22] we get the desired

\( \square \)

**Lemma 26** (Ω Subtyping: domain containment). \( \forall \Psi, \Theta, \Omega_1, \Omega_2. \)
\[ \Psi; \Theta \vdash \Omega_1 <\Omega_2 \implies \forall x : \tau \in \Omega_2, x : \tau' \in \Omega_1 \land \Psi; \Theta \vdash \tau' <\tau \]

**Proof.** Proof by induction on \( \Psi; \Theta \vdash \Omega_1 <\Omega_2 \)

52
1. sub-lBase:

\[
\text{Proof.}
\]

To prove: \( \forall x : \tau \in (.) : \Psi: \Theta \vdash \tau' <: \tau \)

Trivial

2. sub-lInd:

\[
\begin{array}{c}
x : \tau' \in \Omega_1 \\
\vdash \Psi: \Theta \vdash \tau' <: \tau \\
\Psi: \Theta \vdash \Omega_1 <: \Omega_2 \\
\vdash \Psi: \Theta \vdash \Omega_1 / x <: \Omega_2
\end{array}
\]

To prove: \( \forall y : \tau \in (\Omega_2, x : \tau_x) : y : \tau' \in \Omega_1 \land \Psi: \Theta \vdash \tau' <: \tau \)

This means given some \( y : \tau \in (\Omega_2, x : \tau) \) it suffices to prove that \( y : \tau' \in \Omega_1 \land \Psi: \Theta \vdash \tau' <: \tau \)

The following cases arise:

- \( y = x \):
  In this case we are given that
  \( x : \tau' \in \Omega_1 \land \Psi: \Theta \vdash \tau' <: \tau \)
  Therefore we are done

- \( y \neq x \):
  Since we are given that \( \Psi: \Theta \vdash \Omega_1 / x <: \Omega_2 \) therefore we get the desired from IH

\[ \square \]

**Theorem 27** (Soundness 1). \( \forall e, n, n', \tau \in \text{Type}, t. \)

\( \vdash e : [M n \tau \land e \downarrow_n \nu] \Rightarrow n' \leq n \)

**Proof.** From Theorem 20 we know that \( (0, t + 1, e) \in [M n \tau]_\varepsilon \)

From Definition 15 this means we have

\( \forall t' < t + 1 \land e \downarrow_{t'} \nu' \Rightarrow (0, t + 1 - t' \nu') \in [M n \tau] \)

From the evaluation relation we know that \( e \downarrow_{0} e \) therefore we have

\( (0, t + 1, e) \in [M n \tau] \)

Again from Definition 15 it means we have

\( \forall t'' < t + 1 \land e \downarrow_{t'' \nu'} \nu \Rightarrow \exists p', n' + p' \leq 0 + n \land (p', t + 1 - t'' \nu', v) \in \tau \)

Since we are given that \( e \downarrow_{t'' \nu'} \nu \) therefore we have

\( \exists p', n' + p' \leq n \land (p', 1, v) \in \tau \)

Since \( p' \geq 0 \) therefore we get \( n' \leq n \)

\[ \square \]

**Theorem 28** (Soundness 2). \( \forall e, n, n', \tau \in \text{Type}. \)

\( \vdash e : [n] 1 \rightarrow M 0 \tau \land e \downarrow_{t_1} \rightarrow \downarrow_{t_2} \nu \Rightarrow n' \leq n \)

**Proof.** From Theorem 20 we know that \( (0, t_1 + t_2 + 2, e) \in [[n] 1 \rightarrow M 0 \tau]_{\varepsilon} \)

From Definition 15 we know that

\( \forall t' < t_1 + t_2 + 2, v.e \downarrow_{t'} \nu \Rightarrow (0, t_1 + t_2 + 2 - t' \nu', v) \in [[n] 1 \rightarrow M 0 \tau] \) \hspace{1cm} (S0)

Since we know that \( e (.) \downarrow_{t_1} \) therefore from E-app we know that \( \exists e'. e \downarrow_{t_1} \lambda x. e' \)

Instantiating (S0) with \( t_1, \lambda x. e' \) we get \( (0, t_2 + 2, \lambda x. e') \in [[n] 1 \rightarrow M 0 \tau] \)

This means from Definition 15 we have

\( \forall p', e', t'' < t_2 + 2, (p', t', e'' \nu) \in [[n] 1]_{\varepsilon} \Rightarrow (0 + p', t', e' \nu / x) \in [M 0 \tau]_{\varepsilon} \) \hspace{1cm} (S1)

Claim: \( \forall l. (I, l, ()) \in [[I] 1]_{\varepsilon} \)

**Proof:**

From Definition 15 it suffices to prove that

\( (.) \downarrow_{0} v \Rightarrow (I, l, v) \in [[I] 1] \)

Since we know that \( v = (.) \) therefore it suffices to prove that

\( (I, l, v) \in [[I] 1] \)

From Definition 15 it suffices to prove that

\( \exists p', p' \leq I \land (p', l, v) \in [1] \)
We choose $p'$ as 0 and we get the desired

Instantiating (S1) with $n, (t, t_2 + 1$ we get $(n, t_2 + 1, e'((/x)) ∈ [M0 τ]_ε$

This means again from Definition 15 we have

$∀v < t_2 + 1.e'((/x)) ⊢_v v' ⇒ (n, t_2 + 1 - τ', v') ∈ [M0 τ]$

From E-val we know that $v' = e'((/x)$ and $t' = 0$ therefore we have

$(n, t_2 + 1, e'((/x)) ∈ [M0 τ]$

Again from Definition 15 we have

$∀v < t_2 + 1.e'((/x)) ⊢_v v' ⇒ ∃p'.n' + p' ≤ n + 0 ∧ (p', t_2 + 1 - τ', v') ∈ [τ]$

Since we are given that $e ⊢_t -1 = v_2$ therefore we get

$∃p'.n' + p' ≤ n ∧ (p', 1, v''') ∈ [τ]$

Since $p' ≥ 0$ therefore we have $n'' ≤ n$

**Corollary 29** (Soundness). $∀Γ, e, q, q', τ, T, p, n.$

| . . . | $Γ ⊢ e : [q] 1 −→ M0 [q'] τ ∧$
| (p_T, T_γ) ∈ [Γ]_ε ∧ |
| $e () γ ⊢ t, v_t ⊢ t, v ∧$ |
| $t_1 + t_2 < T$ |
| $⇒$ |
| $∃p_c.(p_c, T - t_1 - t_2, v) ∈ [τ] ∧ J ≤ (q + p_t) - (q' + p_c)$ |

**Proof:** From Theorems 20 we know that $(p_T, e, [q] 1 −→ M0 [q'] τ]_ε$

Therefore from Definition 15 we know that

$∀T'<T, e.γ ⊢_T v, v ⇒ (p_t, T - T', v) ∈ [q] 1 −→ M0 [q'] τ$  (S0)

Since we know that $e () γ ⊢ t, v_t$ therefore from E-app we know that

$∃e'.e ⊢ x.e'. e'((/x) ∈ [q] 1 −→ M0 [q'] τ]_ε$

Instantiating (S0) with $t_0, λ.x.e'$ we get $(p_T, T - t_0, λ.x.e') ∈ [q] 1 −→ M0 [q'] τ$

This means from Definition 15 we have

$∀p', T'<(T - t_0), e'.(p', T', e'((/x)) ∈ [q] 1 −→ M0 [q'] τ]_ε$  (S1)

Claim: $∀T.(I, T, ()) ∈ [I]_ε$

**Proof:**

From Definition 15 it suffices to prove that

$∀T'<T, v, () ⊢_{T''} v ⇒ (I, T - T'', v) ∈ [I]_ε$

From (E-val) we know that $T'' = 0$ and $v = ()$ therefore it suffices to prove that

$(I, T, ()) ∈ [I]_ε$

From Definition 15 it further suffices to prove that

$∃p'.p' + I ≤ I ∧ (p', T, ()) ∈ [I]_ε$

We choose $p'$ as 0 and we get the desired

Using the claim we know that we have $(q, T - t_0 - 1, ()) ∈ [q] 1 −→ M0 [q'] τ]_ε$

Instantiating (S1) with $q, T - t_0 - 1, ()$ and using the claim proved above we get

$(p + q, T - t_0 - 1, e'((/x)) ∈ [M0 [q'] τ]_ε$

This means again from Definition 15 we have

$∀T_1 < T - t_0 - 1.e'((/x)) ⊢_v v' ⇒ (p + q, T - t_0 - 1 - T_1, v') ∈ [M0 [q'] τ]$

Instantiating with $t_0', v_t$ and since $t_1 < T$, therefore we also have $t_0' < T - t_0'$. Also since we are given that $e() γ ⊢ v_t$ therefore we know that $v' = v_t$. Thus, we have

$(p + q, T - t_0 - 1 - t_0', v_t) ∈ [M0 [q'] τ]$

Again from Definition 15 we have

$∀v', t_2 < T - t_0 - 1.v_t ⊢ t_2, v' ⇒ ∃p'.J + p' ≤ p + q ∧ (p', T - t_0 - 1 - t_2, v') ∈ [q] τ$  (S2)
Since we have \((p', T - t_1 - t_2, v) \in \llbracket \tau \rrbracket\) therefore from Definition \ref{def:typing} we have \(\exists p'. p' + q' \leq p' \land (p', T - t_1' - t_2' - 1 - t_2, v) \in \llbracket \tau \rrbracket\) (S3)

In order to prove \(\exists p_v. (p_v, T - t_1 - t_2, v) \in \llbracket \tau \rrbracket \land J \leq (q + p_v) - (q' + p_v)\) we choose \(p_v\) as \(p'_1\) and we need to prove:

1. \((p'_1, T - t_1 - t_2, v) \in \llbracket \tau \rrbracket\):
   
   Since from (S3) we have \((p'_1, T - t_1' - t_2' - 1 - t_2, v) \in \llbracket \tau \rrbracket\) and since \(t_1' + t_1'' + 1 = t_1\) therefore also have \((p'_1, T - t_1 - t_2, v) \in \llbracket \tau \rrbracket\)

2. \(J \leq (q + p_v) - (q' + p_v)\):
   
   From (S2) and (S3) we get \(J \leq (p_v + q) - (q' + p_v)\)

\[\Box\]

### A.5 Embedding Univariate RAML

Univariate RAML's type syntax

\[
\begin{align*}
\text{Types} & \quad \tau ::= \; \text{b} \mid L^\tau \tau \mid (\tau_1, \tau_2) \\
\text{A} & \quad ::= \; \tau \mathbin{\triangleright} q/\bar{q} \tau
\end{align*}
\]

Type translation

\[
\begin{align*}
\langle \text{unit} \rangle & \quad = \; 1 \\
\langle \text{b} \rangle & \quad = \; \top \text{b} \\
\langle L^\tau \tau \rangle & \quad = \; \exists s. (\langle \text{phi}(\bar{q}, s) \rangle 1 \otimes L^s(\tau)) \\
\langle \langle \tau_1, \tau_2 \rangle \rangle & \quad = \; (\langle \tau_1 \rangle \otimes \langle \tau_2 \rangle) \\
\langle \tau_1 \mathbin{\triangleright} q/\bar{q} \tau_2 \rangle & \quad = \; (\langle q \rangle \mapsto \langle \tau_1 \rangle \mapsto M(0, \langle \bar{q} \rangle \langle \tau_2 \rangle))
\end{align*}
\]

Type context translation

\[
\begin{align*}
\langle \cdot \rangle & \quad = \; \cdot \\
\langle \Gamma, x : \tau \rangle & \quad = \; \langle \Gamma \rangle, x : (\tau_1)
\end{align*}
\]

Function context translation

\[
\begin{align*}
\langle \cdot \rangle & \quad = \; \cdot \\
\langle \Sigma, x : \tau \rangle & \quad = \; \langle \Sigma \rangle, x : (\tau_1)
\end{align*}
\]

Judgment translation

\[
\Sigma; \Gamma \vdash_q e_x : \tau \quad \leadsto \quad \vdash_q \text{e}_x : [q] \mapsto M(0, \langle \bar{q} \rangle \langle \tau_1 \rangle)
\]

**Definition 30.** \(\phi(\bar{q}, u) \triangleq \sum_{1 \leq i \leq k} ^n q_i\) as defined in \[10, 19\]

**Expression translation**

\[
\begin{align*}
\Sigma; \vdash_q^{K^\text{unit}} (\text{unit}) : \text{unit} \leadsto \lambda u. \text{release} & = u \text{ in bind } \leadsto \uparrow_K^{\text{unit}} \text{ in bind } a = \text{store}() \text{ in ret}(a) \\
\Sigma; \vdash_q^{K^\text{base}} c : \text{b} \leadsto \lambda u. \text{release} & = u \text{ in bind } \leadsto \uparrow_K^{\text{base}} \text{ in bind } a = \text{store}(c) \text{ in ret}(a) \\
\Sigma; x : \tau \vdash_q^{K^\text{var}} x : \tau \leadsto \lambda u. \text{release} & = u \text{ in bind } \leadsto \uparrow_K^{\text{var}} \text{ in bind } a = \text{store} x \text{ in ret}(a) \\
\Sigma; x : \tau_1 \vdash_q^{K^\text{app}} f : x : \tau_2 \leadsto \lambda u. E_0 & \quad \uparrow^{K^\text{app}} \text{ in bind } a \in \Sigma(f) \quad \text{app}
\end{align*}
\]

where

\[
\begin{align*}
E_0 = \text{release} & = u \text{ in bind } \leadsto \uparrow_K^{\text{app}} \text{ in bind } P = \text{store}() \text{ in } E_1 \\
E_1 = \text{bind } f_1 & = (f, P, x) \text{ in release } f_2 = f_1 \text{ in bind } f_3 = \text{store } f_2 \text{ in ret } f_3
\end{align*}
\]

\[
\begin{align*}
\Sigma; \emptyset \vdash_q^{K^\text{unit}} \text{nil} : L^\text{nil} \leadsto \lambda u. \text{release} & = u \text{ in bind } \leadsto \uparrow_K^{\text{unit}} \text{ in bind } b = \text{store}(\langle a, \text{nil} \rangle) \text{ in ret}(b)
\end{align*}
\]
\[ \Sigma; x_h : \tau, x_t : L^q \vdash p \vdash q; p_1 + K_{cons} \quad cons(x_h, x_t) : L^p \tau \rightsquigarrow \lambda u. \text{release} \quad \text{u} \in \text{bind} \quad \tau \downarrow q; q_2 + K_{cons} \quad e_2 : \tau' \rightsquigarrow e_2 \]

where

\[ E_0 = x_1 : x, \text{let} \langle x_1, x_2 \rangle = x \in E_1 \]

\[ E_1 = \text{release} \quad \text{u} \in \text{bind} \quad a = \text{store()} \quad \text{in} \quad b = \text{store} \langle a, x_h :: x_2 \rangle \quad \text{in} \quad \text{ret}(b) \]

\[ \Sigma; \Gamma \vdash q; K_{maxN} \quad e_1 : \tau' \rightsquigarrow e_1 \quad \Sigma; \Gamma, h : \tau, t : L^q \vdash q; K_{maxC} \quad e_2 : \tau' \rightsquigarrow e_2 \]

\[ \Sigma; \Gamma, x : L^p \tau \vdash q; q \quad \text{match} \quad x \text{ with } \begin{cases} \text{nil} \mapsto e_1 \mid h \mapsto e_2 : \tau' \rightsquigarrow \lambda u. E_0 \end{cases} \]

\[ \Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash q; q \quad e : \tau' \rightsquigarrow e_a \quad \tau = \tau_1 \cap \tau_2 \quad \tau = \tau_1 = \tau_2 = 1 \]

Share-unit

\[ E_0 = \lambda u. E_1 \]

\[ E_1 = \text{bind} \quad a = \text{coerce}_{1,1,1} \quad z \quad \text{in} \quad \text{let} \langle x, y \rangle = a \in e_a \quad u \]

\[ \text{coerce}_{1,1,1} : (\{1\} \twoheadrightarrow M \{0 \} (\{1\} \otimes \{1\})) \]

\[ \text{coerce}_{1,1,1} \triangleq \lambda u. \text{ret}(!() \otimes ()()) \]

\[ \Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash q; q \quad e : \tau' \rightsquigarrow e_a \quad \tau = \tau_1 \cap \tau_2 \quad \tau = \tau_1 = \tau_2 = b \]

Share-base

\[ E_0 = \lambda u. E_1 \]

\[ E_1 = \text{bind} \quad a = \text{coerce}_{b,b,b} \quad z \quad \text{in} \quad \text{let} \langle x, y \rangle = a \in e_a \quad u \]

\[ \text{coerce}_{b,b,b} \triangleq \lambda u. \text{let} \quad u' = u \quad \text{in} \quad \text{ret}(!() \otimes u'()) \]

\[ \tau = L^p \tau'' \quad \tau_1 = L^{p_1 \tau_1} \quad \tau_2 = L^{p_2 \tau_2} \quad \tau'' = \tau_1'' \cap \tau_2'' \quad \tau'' = \tau_1'' \cap \tau_2'' \quad \tau'' = \tau_1'' \cap \tau_2'' \]

Share-list
\[ E_0 \triangleq \text{release } - = p \text{ in } E_1 \]
\[ E_1 \triangleq \text{match } \lambda l \text{ with } |\text{nil} \mapsto E_{2.1} | h : t \mapsto E_3 \]
\[ E_{2.1} \triangleq \text{bind } z_1 = \text{store}(\) \text{ in } E_{2.2} \]
\[ E_{2.2} \triangleq \text{bind } z_2 = \text{store}(\) \text{ in } E_{2.3} \]
\[ E_{2.3} \triangleq \text{ret} \langle \langle z_1, \text{nil} \rangle, \langle z_2, \text{nil} \rangle \rangle \]
\[ E_1 \triangleq \text{bind } H = g' \ h \text{ in } E_{3.1} \]
\[ E_{3.1} \triangleq \text{bind } a_0 = () \text{ in } E_{3.2} \]
\[ E_{3.2} \triangleq \text{bind } T = f' \ \langle a_0, t \rangle \text{ in } E_4 \]
\[ E_4 \triangleq \text{let } \langle H_1, H_2 \rangle = H \text{ in } E_5 \]
\[ E_5 \triangleq \text{let } \langle T_1, T_2 \rangle = T \text{ in } E_6 \]
\[ E_6 \triangleq T_1; t p_1, \text{let } \langle p'_1, k'_1 \rangle = t p_1 \text{ in } E_{7.1} \]
\[ E_{7.1} \triangleq T_2; t p_2, \text{let } \langle p'_2, k'_2 \rangle = t p_2 \text{ in } E_{7.2} \]
\[ E_{7.2} \triangleq \text{release } = - = p'_1 \text{ in } E_{7.3} \]
\[ E_{7.3} \triangleq \text{release } = - = p'_2 \text{ in } E_{7.4} \]
\[ E_{7.4} \triangleq \text{bind } a_1 = \text{store}(\) \text{ in } E_{7.5} \]
\[ E_{7.5} \triangleq \text{bind } a_2 = \text{store}(\) \text{ in } E_8 \]
\[ E_8 \triangleq \text{ret} \langle \langle a_1, H_1 :: T_1 \rangle, \langle a_2, H_2 :: T_2 \rangle \rangle \]

\[
\Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash \theta_y \ e : \tau' \vdash e_a \quad \tau = \tau_1 \ Y \tau_2 \quad \tau = (\tau_a, \tau_b) \quad \tau_1 = (\tau'_a, \tau'_b) \quad \tau_2 = (\tau''_a, \tau''_b) \quad \text{Share-pair}
\]

\[
\Sigma; \Gamma, z : \tau \vdash e[z/x, z/y] : \tau' \vdash e_0 \quad \text{Sub}
\]
\[ \Sigma; \Gamma \vdash e' \ e : \tau \vdash e_a \quad \tau < : \tau' \quad \Sigma; \Gamma \vdash e' \ e : \tau' \vdash e_a \]
\[ \Sigma; \Gamma \vdash \theta_y e : \tau \vdash e_a \quad \tau' < : \tau_1 \quad \Sigma; \Gamma \vdash \theta_y e : \tau' \vdash e_a \]
\[ \Sigma; \Gamma \vdash \theta_y e : \tau \vdash e_a \quad q \geq p \quad q' \geq q - p' \quad \Sigma; \Gamma \vdash \theta_y e : \tau \vdash \lambda o. E_0 \quad \text{Relax}
\]

where
\[ E_0 = \text{release } - = o \text{ in } E_1 \]
\[ E_1 = \text{bind } a = \text{store}() \text{ in } E_2 \]
\[ E_2 = \text{bind } b = e_a \text{ a in } E_3 \]
\[ E_3 = \text{release } c = b \text{ in } \text{store} c \]
\[ \Sigma; \Gamma_1 \vdash \theta_y \ e_1 : \tau_1 \vdash e_{a1} \quad \Sigma; \Gamma_2 \vdash \theta_y K_i \ e_2 : \tau_1 \vdash e_{a2} \quad \Sigma; \Gamma_1, \Gamma_2 \vdash \theta_y let \ x \mapsto e_1 \text{ in } e_2 : \tau \vdash E_t \quad \text{Let}
\]

where
\[ E_t = \lambda u. E_0 \]
\[ E_0 = \text{release } - = u \text{ in } E_1 \]
\[ E_1 = \text{bind } n - = \theta \text{ in } E_2 \]
\[ E_2 = \text{bind } a = \text{store}() \text{ in } E_3 \]
\[ E_3 = \text{bind } b = e_{a1} \text{ a in } E_4 \]
\[ E_4 = \text{release } x = b \text{ in } E_5 \]
\[ E_5 = \text{bind } \lambda u. E_0 \]
\[ E_6 = \text{bind } c = \text{store()} \text{ in } E_7 \]
\[ E_7 = \text{bind } d = e \text{ in } E_8 \]
\[ E_8 = \text{release } f = d \text{ in } E_9 \]
\[ E_9 = \text{bind } \lambda u. E_0 \]
\[ E_{10} = \text{bind } g = \text{store } f \text{ in } E_{10} \]
\[
\Sigma; x_1 : \tau_1, x_2 : \tau_2 \vdash^{q + K_{\text{map}}^{\text{ret}}} (x_1, x_2) : (\tau_1, \tau_2) \rightsquigarrow E_t \quad \text{pair}
\]
where
\[ E_0 = \lambda u. E_0 \]
\[ E_1 = \text{release } \lambda u. E_0 \]
\[ E_2 = \text{bind } \lambda u. E_0 \]
\[ E_3 = \text{bind } d = \text{store } (x_1, x_2) \text{ in } \text{ret } a \]
\[ E_4 = \text{bind } a = \text{store } (x_1, x_2) \text{ in } \text{ret } a \]
\[ \tau = (\tau_1, \tau_2) \quad \Sigma, \Gamma, x_1 : \tau_1, x_2 : \tau_2 \vdash^{q + K_{\text{map}}^{\text{ret}}} e : \tau' \rightsquigarrow e_t \]
\[ \Sigma; \Gamma, x : \tau \vdash^{q} \text{match } x \text{ with } (x_1, x_2) \rightarrow e : \tau' \rightsquigarrow E_t \quad \text{matP} \]

\[ E_t = \text{bind } d = \text{store } c \text{ in } \text{ret } d \]
\[ \Sigma; \Gamma \vdash^{q} e : \tau \rightsquigarrow e_a \quad \text{Augment} \]

\[ \Sigma; \Gamma \vdash^{q} e : \tau \rightsquigarrow e_a \]

A.5.1 Type preservation

**Theorem 31 (Type preservation: Univariate RAML to \(\lambda\)-amor).** If \(\Sigma; \Gamma \vdash^{q} e : \tau\) in Univariate RAML then there exists \(e'\) such that \(\Sigma; \Gamma \vdash^{q} e : \tau \rightsquigarrow e'\) such that there is a derivation of \(\vdots; \vdots; (\Sigma), (\Gamma) \vdash e' : [q] 1 \rightsquigarrow M_0 ([q'] [r])\) in \(\lambda\)-amor.

**Proof.** By induction on \(\Sigma; \Gamma \vdash^{q} e : \tau\):

1. unit:
   \[ \frac{}{\Sigma; \Gamma \vdash^{q + K_{\text{unit}}} () : \text{unit} \rightsquigarrow \lambda u. \text{release } = = \text{u in } \text{bind } = = \lambda u. \text{unit} \text{ in } \text{bind } a = \text{store } (\text{in } \text{ret } a) \]

   \[ E_0 = \lambda u. \text{release } = = \text{u in } \text{bind } = = \lambda u. \text{unit} \text{ in } \text{bind } a = \text{store } (\text{in } \text{ret } a) \]

   \[ E_1 = \text{release } = = \text{u in } \text{bind } = = \lambda u. \text{unit} \text{ in } \text{bind } a = \text{store } (\text{in } \text{ret } a) \]

   \[ T_0 = [q + K_{\text{unit}}] 1 \rightsquigarrow M_0 ([q'] [\text{unit}]) \]

   \[ T_1 = [q + K_{\text{unit}}] 1 \]

   \[ T_2 = M (q + K_{\text{unit}}) ([q] 1) \]

   \[ T_{2,1} = M (q) ([q] 1) \]

   \[ T_3 = M K_{\text{unit}} 1 \]

   \[ T_4 = M 0 ([q] 1) \]

   \[ T_5 = M q ([q] 1) \]

   D1:
   \[ \frac{}{\vdots; \vdots; (\Sigma) ; \vdash \text{store } : T_5} \]
   \[ \frac{}{\vdots; \vdots; (\Sigma) ; \vdash a : [q] 1 \vdash \text{ret } a : T_4} \]

   \[ \frac{}{\vdots; \vdots; (\Sigma) ; \vdash \text{bind } a = \text{store } (\text{in } \text{ret } a) : T_5} \]

   D0:
   \[ \frac{}{\vdots; \vdots; (\Sigma) ; \vdash K_{\text{unit}} : T_3} \]
D0.0: \[ D0 \quad D1 \]
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{bind } a = \uparrow K^{\text{base}} \text{ in bind } a = \text{store}(t1) \text{ in ret } : T_2 \quad \text{T-bind}
\]

Main derivation:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{u : } T_1 \vdash u : T_1 \quad \text{T-var} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_1 : T_4 \quad \text{T-release} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_0 : T_0 \quad \text{T-lam}
\]

2. base:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{bind } a = \text{store}(t1) \text{ in ret } : T_2 \quad \text{T-var} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_1 : T_4 \quad \text{T-release} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_0 : T_0 \quad \text{T-lam}
\]

\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{store}(t1) : T_5 \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash a : \tau \vdash \text{ret } : T_1 \quad \text{T-lam}
\]

D0:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \uparrow K^{\text{base}} : T_3
\]

D0.0:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{bind } a = \text{store}(t1) \text{ in ret } : T_2 \quad \text{T-bind}
\]

Main derivation:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{u : } T_1 \vdash u : T_1 \quad \text{T-var} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_1 : T_4 \quad \text{T-release} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_0 : T_0 \quad \text{T-lam}
\]

3. var:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{u : } T_1 \vdash u : T_1 \quad \text{T-var} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_1 : T_4 \quad \text{T-release} \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash E_0 : T_0 \quad \text{T-lam}
\]

\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{store}(t1) : T_5 \\
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash a : \tau \vdash \text{ret } : T_1 \quad \text{T-lam}
\]

D0:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \uparrow K^{\text{var}} : T_3
\]

D0.0:
\[
\vdash \cdot \cdot \cdot ; \langle \Sigma \rangle ; . \vdash \text{bind } a = \text{store}(t1) \text{ in ret } : T_2 \quad \text{T-bind}
\]

59
Main derivation:

\[ \vdash \vdash: \{ \Sigma \}; u : T_1 \vdash u : T_1 \quad \text{T-var} \quad D0.0 \]
\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau \}, u : T_1 \vdash E_1 : T_4 \quad \text{T-release} \]
\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau \} \vdash E_0 : T_0 \quad \text{T-lam} \]

4. app:

\[ \frac{\tau_1 \overset{q/q'}{\tau_2} \in \Sigma(f) \quad \text{app}}{\Sigma; x : \tau_1 \vdash q + K^{app}_2 \overset{f}{\tau_2} \sim \lambda u. E_0} \]

where

\[
E_0 = \text{release } t = u \text{ in bind } \Rightarrow = \uparrow K^{app}_1 \text{ in bind } P = \text{store}(t) \text{ in } E_1 \\
E_1 = \text{bind } f_1 = (f \ P \ x) \text{ in release } f_2 = f_1 \text{ in bind } \Rightarrow = \uparrow K^{app}_2 \text{ in bind } f_3 = \text{store } f_2 \text{ in ret } f_3 \\
E_{1.1} = \text{release } f_2 = f_1 \text{ in bind } \Rightarrow = \uparrow K^{app}_2 \text{ in bind } f_3 = \text{store } f_2 \text{ in ret } f_3 \\
E_{1.2} = \text{bind } \Rightarrow = \uparrow K^{app}_2 \text{ in bind } f_3 = \text{store } f_2 \text{ in ret } f_3 \\
E_{1.3} = \text{bind } f_3 = \text{store } f_2 \text{ in ret } f_3 \\
E_{1.4} = \text{store } f_2 \\
E_{1.5} = \text{ret } f_3 \\
E_{0.1} = \text{bind } \Rightarrow = \uparrow K^{app}_1 \text{ in bind } F = f \text{ in } E_4 \]

\[
T_0 = \{ q + K^{app}_1 \} 1 \rightarrow M \ 0 \ \{ [q' - K^{app}_2(\tau)] \} \\
T_{0.1} = \{ q + K^{app}_1 \} 1 \\
T_{0.2} = M \ 0 \ \{ [q' - K^{app}_2(\tau)] \} \\
T_1 = M \ (q + K^{app}_1) 1 \\
T_{1.2} = M \ 0 \ \{ [q' - K^{app}_2(\tau)] \} \\
T_2 = M \ (K^{app}_1) 1 \\
T_3 = M \ (q) \ \{ \tau_2 \} \\
T_4 = M \ (q) \ \{ \tau_1 \} \rightarrow M \ 0 \ \{ q' \} \ \{ \tau_2 \} \\
T_{1.1} = \{ \tau_1 \} \rightarrow M \ 0 \ \{ q' \} \ \{ \tau_2 \} \\
T_{1.2} = M \ 0 \ \{ q' \} \ \{ \tau_2 \} \\
T_{1.3} = \{ q' \} \ \{ \tau_2 \} \\
T_{1.4} = M \ (q' - K^{app}_2) \ \{ q' - K^{app}_2 \} \ \{ \tau_2 \} \\
T_{1.41} = M \ (q') \ \{ q' - K^{app}_2 \} \ \{ \tau_2 \} \\
T_{1.5} = \{ q' - K^{app}_2 \} \ \{ \tau_2 \} \\
T_{1.6} = M \ 0 \ \{ q' - K^{app}_2 \} \ \{ \tau_2 \} \\
\]

D2.3:

\[
\vdash \vdash: \{ \Sigma \}; f_2 : \{ \tau_2 \} \vdash E_{1.4} : T_{4.4} \quad \vdash \vdash: \{ \Sigma \}; f_3 : T_{4.5} \vdash E_{1.5} : T_{4.6} \\
\vdash \vdash: \{ \Sigma \}; f_2 : \{ \tau_2 \}, f_3 : T_{4.5} \vdash E_{1.3} : T_{4.4} \quad D2.3 \]

D2.2:

\[ \vdash \vdash: \{ \Sigma \}; \vdash \uparrow K^{app}_2 : M \ K^{app}_2 1 \quad \text{D2.3} \]
\[ \vdash \vdash: \{ \Sigma \}; f_2 : \{ \tau_2 \} \vdash E_{1.2} : T_{4.41} \quad D2.2 \]

D2.1:

\[ \vdash \vdash: \{ \Sigma \}; f_1 : T_{4.3} \vdash f_1 : T_{4.3} \quad \text{D2.2} \]
\[ \vdash \vdash: \{ \Sigma \}; f_1 : T_{4.3} \vdash E_{1.1} : T_{1.2} \quad \text{D2.1} \]

D2:

\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau_1 \}, P : [q] 1 \vdash f \ P \ x : T_{4.2} \quad \text{D2.1} \]
\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau_1 \}, P : [q] 1 \vdash E_1 : T_{1.2} \quad \text{D2} \]

D1:

\[ \vdash \vdash: \{ \Sigma \}; \vdash \text{store}() : M \ q \ [q] 1 \quad \text{D2} \]
\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau_1 \} \vdash \text{bind } P = \text{store}() \text{ in } E_1 : T_{1.2} \quad \text{D1} \]

D0:

\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau_1 \} \vdash \uparrow K^{app}_1 : T_1 \quad \text{D1} \]
\[ \vdash \vdash: \{ \Sigma \}; x : \{ \tau_1 \} \vdash E_{0.1} : T_{1.2} \quad \text{D0} \]
Main derivation:

\[
\vdots \vdots (\Sigma); u : T_{0,1} \vdash u : T_{0,1} \quad \text{T-var} \\
\vdots \vdots (\Sigma); x : \langle \tau \rangle, u : T_{0,1} \vdash E_0 : T_{0,2} \\
\vdots \vdots (\Sigma); x : \langle \tau \rangle \vdash \lambda u. E_0 : T_0
\]

5. nil:

\[
\Sigma; \emptyset \vdash 0 \vdash q + K^{nil} \quad \text{nil}
\]

\[
\begin{align*}
E_0 &= \lambda u. \text{release} \vdash u \quad \text{in \ bind} \vdash \uparrow K^{nil} \quad \text{in \ bind} \\
E_1 &= \text{release} \vdash u \quad \text{in \ bind} \vdash \uparrow K^{nil} \quad \text{in \ bind} \\
E_2 &= \text{bind} \vdash \uparrow K^{nil} \quad \text{in \ bind} \\
E_3 &= \text{bind} \vdash \uparrow K^{nil} \quad \text{in \ bind} \\
E_4 &= \text{bind} \vdash \uparrow K^{nil} \quad \text{in \ bind} \\
E_5 &= \text{bind} \vdash \uparrow K^{nil} \quad \text{in \ bind}
\end{align*}
\]

\[
\begin{align*}
T_0 &= \langle q + K^{nil} \rangle 1 \\
T_2 &= M(0) ([q] \exists n. \phi(q, n) \otimes \text{list}[n][\tau]) \\
T_3 &= M(q + K^{nil}) ([q] \exists n. \phi(q, n) \otimes \text{list}[n][\tau]) \\
T_4 &= M K^{nil} 1 \\
T_5 &= M(q) ([q] \exists n. \phi(q, n) \otimes \text{list}[n][\tau]) \\
T_{5,1} &= ((q) \exists n. \phi(q, n) \otimes \text{list}[n][\tau]) \\
T_6 &= M(0) ([q] \exists n. \phi(q, n) \otimes \text{list}[n][\tau])
\end{align*}
\]

D4:

\[
\phi(\bar{p}, 0) = 0
\]

\[
\vdots \vdots (\Sigma); a : [0] 1 \vdash a : [0] 1 \\
\vdots \vdots (\Sigma); a : [0] 1 \vdash \langle a, nil \rangle : T_0[0/n] \\
\vdots \vdots (\Sigma); a : [0] 1 \vdash \langle a, nil \rangle : T_0
\]

D3:

\[
\vdots \vdots (\Sigma); b : T_{5,1} \vdash E_5 : T_0
\]

D2:

\[
\vdots \vdots (\Sigma); a : [0] 1 \vdash \text{store}\langle a, nil \rangle : T_5
\]

D1:

\[
\vdots \vdots (\Sigma); \vdash \text{store}() : M 0 [0] 1 \\
\vdots \vdots (\Sigma); \vdash E_3 : T_3
\]

D0:

\[
\vdots \vdots (\Sigma); \vdash \uparrow K^{nil} : T_4 \\
\vdots \vdots (\Sigma); \vdash E_2 : T_3
\]

Main derivation:

\[
\vdots \vdots (\Sigma); u : T_1 \vdash u : T_1 \\
\vdots \vdots (\Sigma); u : T_1 \vdash E_1 : T_2 \\
\vdots \vdots (\Sigma); \vdash E_0 : T_0
\]

6. cons:

\[
\bar{p} = (p_1, \ldots, p_k)
\]

\[
\Sigma; x_h : \tau, x_t : L^* \bar{p} \vdash q + p_1 + K^{\text{cons}} \quad \text{cons}(x_h, x_t) : L^p \tau \quad \lambda u. \text{release} \vdash u \quad \text{in \ bind} \vdash \uparrow K^{\text{cons}} \quad \text{in \ E_0}
\]

where
$E_0 = x_1; x. \text{let} \langle x_1, x_2 \rangle = x \text{ in } E_1$

$E_1 = \text{release } \ast = x_1 \text{ in } \text{bind } a = \text{store()} \text{ in } \text{store}\langle a, x_h :: x_2 \rangle$

$T_0 = [q + p_1 + K^{\text{cons}}] \ 1 \rightarrow M 0 ([q] \exists n'. [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_1 = [q + p_1 + K^{\text{cons}}] 1$

$T_2 = M 0 ([q] \exists n'. [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_{2.1} = M(q + p_1) ([q] \exists n'. [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_{2.2} = M(q + p_1 + \phi(\bar{q}, s)) ([q] \exists n'. [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_{2.3} = M(q) ([q] \exists n'. [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_{2.4} = \exists n'. [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_{2.5} = [\phi(p, n')] 1 \otimes L^n(\tau)]$

$T_3 = ([p_1 + \phi(\bar{q}, s)] 1$

$T_4 = \exists s, ([\phi(\bar{q}, s)] 1 \otimes L^s(\tau)]$

$T_{11} = ([\phi(\bar{q}, s)] 1 \otimes L^s(\tau)]$

$T_{12} = [\phi(\bar{q}, s)] 1$

$T_{13} = L^s(\tau)]$

### D1.4:

| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_h : \langle \tau \rangle, x_2 : T_{13}, a : T_3 + \langle a, x_h :: x_2 \rangle : T_{2.5} ; (s + 1) / n' |$ | $s : \mathbb{N} + s + 1 : \mathbb{N}$ |
| --- | --- |
| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_h : \langle \tau \rangle, x_2 : T_3, a : T_3 + \langle a, x_h :: x_2 \rangle : T_{2.4}$ | $\text{D1.4}$ |
| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_h : \langle \tau \rangle, x_2 : T_{13}, a : T_3 + \text{store}\langle a, x_h :: x_2 \rangle : T_{2.3}$ | $\text{D1.3}$ |

### D1.2:

| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_h : \langle \tau \rangle, x_2 : T_{13}, y : T_3 + \langle a, x_h :: x_2 \rangle : T_{2.3}$ | $\text{D1.2}$ |

### D1.1:

| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x : T_{11} + x : T_1$ | $\text{D1.1}$ |
| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_h : \langle \tau \rangle, x_1 : T_1, x_2 : T_{12}, x_2 : T_{13} + E_1 : T_{2.1}$ | $\text{D1.1}$ |

### D1:

| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_1 : T_{12} + x_1 : T_1$ | $\text{D1}$ |
| $s : \mathbb{N}; 1 \langle \Sigma \rangle; x_h : \langle \tau \rangle, x_1 : T_1 + \text{let} \langle x_1, x_2 \rangle = x \text{ in } E_1 : T_{2.1}$ | $\text{D1}$ |

### D0:

| $s : \mathbb{N}; 1 \langle \Sigma \rangle; u : T_1$ | $\text{D0}$ |

Main derivation:

$\vdash s : \mathbb{N}; 1 \langle \Sigma \rangle; u : T_1 + u : T_1$

7. match:

$\Sigma; \Gamma, x : L^p \tau + q^m \text{ match } x \text{ with } \text{in} \rightarrow e_1 | h : t \rightarrow e_2 : \tau' \rightarrow \lambda u. E_0$ match

where

$E_0 = \text{release } \ast = u \text{ in } E_{0.1}$

$E_{0.1} = x_1; \text{a. let} \langle x_1, x_2 \rangle = a \text{ in } E_1$

$E_1 = \text{match } x_2 \text{ with } \text{in} \rightarrow E_2 | h : l \rightarrow E_3$

$E_2 = \text{bind } \ast = T^{K^{\text{matN}}}_{\text{in}} \text{ in } E_{2.1}$
\[E_{2.1} = \text{bind } b = \text{store}() \text{ in } E'_{2.1}\]
\[E_2 = \text{bind } c = (e_{a1} b) \text{ in } E'_{2.1}\]
\[E_{2.1} = \text{release } d = c \text{ in } E'_{2.2}\]
\[E_{2.2} = \text{bind } d = \uparrow K^{matC}_2 \text{ in } E'_{2.3}\]
\[E_{2.3} = \text{release } = x_1 \text{ in } \text{store } d\]
\[E_3 = \text{bind } = \uparrow K^{matC}_1 \text{ in } E_{3.1}\]
\[E_{3.1} = \text{release } = x_1 \text{ in } E_{3.2}\]
\[E_{3.2} = \text{bind } b = \text{store}() \text{ in } E_{3.3}\]
\[E_{3.3} = \text{bind } t = \text{ret}(b, l) \text{ in } E_{3.4}\]
\[E_{3.4} = \text{bind } d = \text{store}() \text{ in } E_{3.5}\]
\[E_{3.5} = \text{bind } f = e_{a2} d \text{ in } E_{3.6}\]
\[E_{3.6} = \text{release } g = f \text{ in } E_{3.7}\]
\[E_{3.7} = \text{bind } = \uparrow K^{matC}_2 \text{ in } \text{store } g\]

\[T_0 = [q] \mathcal{1} \rightarrow \mathcal{0} ([q]' \uparrow' \mathcal{1})\]
\[T_1 = [q] \mathcal{1}\]
\[T_2 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_2.0 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_2.1 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_2.10 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_2.11 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_2.12 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_2.13 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_3 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_3.0 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_3.1 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_3.2 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_4.0 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_4.10 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_4.11 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_4.12 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_4.13 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_4.14 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.0 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.1 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.2 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.3 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.4 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.5 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.6 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.7 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.8 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]
\[T_5.9 = \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]

D3.8:

\[; s, i; s = i + 1; \{\Sigma\}; g : T_g \vdash \text{store } g : \mathcal{M} [q]' ([q]' \uparrow' \mathcal{1})\]

D3.7:

\[; s, i; s = i + 1; \{\Sigma\}; \vdash \uparrow K^{matC}_2 : \mathcal{M} K^{matC}_2 \mathcal{1}\]

\[; s, i; s = i + 1; \{\Sigma\}; g : T_g \vdash E_{3.7} : T_{3.3}\]
\[
\frac{\vdash s, t; s = i + 1; \{\Sigma\}; f : T_f}{\vdash s, t; s = i + 1; \{\Sigma\}; f : T_f} \quad E_{3.7}
\]

\[
\frac{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c, d : T_d, e_{a_d} d : T_{4.3}}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c, d : T_d} \quad E_{3.6}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}, \Gamma, h : \{r\}, t : T_c, d : T_d, e_{a_d} d : T_{4.3}}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{3.5}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{3.4}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{3.3}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{3.2}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{3.1}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{3.0}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.30}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.32}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.31}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.30}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.31}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.30}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.31}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.30}
\]

\[
\frac{\vdash s, i; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c}{\vdash s, t; s = i + 1; \{\Sigma\}; \Gamma, h : \{r\}, t : T_c} \quad E_{2.31}
\]
D2:
\[
\vdash s; s = 0; \langle \Sigma \rangle; x_1 : T_{i1} \vdash E_2 : T_{i2}
\]

D1.1:
\[
\vdash s; s; \langle \Sigma \rangle; x_2 : T_{i2} \vdash x_2 : T_{i2}
\]

D1:
\[
\vdash s; \langle \Sigma \rangle; (\Gamma), x_1 : T_{i1}, x_2 : T_{i2} \vdash E_1 : T_{i1}
\]

\[\vdash s; \langle \Sigma \rangle; (\Gamma), a : T_i \vdash \langle x_1, x_2 \rangle = a \text{ in } E_1 : T_{i1}\]

D0:
\[
\vdash s; \langle \Sigma \rangle; \langle \Gamma \rangle, x : T_i \vdash x : T_i
\]

Main derivation:
\[
\vdash s; \langle \Sigma \rangle; (\Gamma), x : T_i, u : T_i \vdash u : T_i
\]

\[
\vdash s; \langle \Sigma \rangle; (\Gamma), x : T_i \vdash E_0 : T_{01}
\]

\[\vdash s; \langle \Sigma \rangle; (\Gamma), x : T_i \vdash \lambda u. E_0 : T_0\]

8. Share:
\[
\Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash \theta_{\tau'} e : \tau' \rightsquigarrow e_a
\]
\[
\Sigma; \Gamma, z : \tau \vdash \theta_{\tau'} e[z/x, z/y] : \tau' \rightsquigarrow E_0
\]

\[\text{Share-unit}\]

\[E_0 = \lambda u. E_1\]
\[E_1 = \text{bind } a = \text{coerce}_{1,1,1} z \text{ in } \langle x, y \rangle = a \text{ in } e_a u\]

\[T_0 = [q] 1 \rightarrow M_0 ([q'] ([\tau'])\]

D1:
\[
\vdash s; \langle \Sigma \rangle; (\Gamma), u : [q] 1, x : \langle \tau_1 \rangle, y : \langle \tau_2 \rangle \vdash e_a : T_0
\]

\[\vdash s; \langle \Sigma \rangle; u : [q] 1 \rightarrow u : [q] 1\]

D0:
\[
\vdash s; \langle \Sigma \rangle; a : ([\tau_1] \otimes [\tau_2]) \vdash a : ([\tau_1] \otimes [\tau_2])
\]

Main derivation:
\[
Dc1
\]

\[\vdash s; \langle \Sigma \rangle; : ([\tau]) \vdash z : ([\tau])\]

\[\vdash \langle \Sigma \rangle; : ([\tau]) \vdash \text{coerce}_{1,1,1} z : M_0 ([\tau_1] \otimes [\tau_2])\]

\[\vdash \langle \Sigma \rangle; (\Gamma), z : ([\tau]), u : [q] 1 \vdash E_0 : M_0 [q'] ([\tau'])\]

\[\vdash \langle \Sigma \rangle; (\Gamma), z : ([\tau]) \vdash \lambda u. E_0 : T_0\]

\[\text{coerce}_{1,1,1} : \langle 1 \rangle \rightarrow M_0 ([1] \otimes [1])\]

\[\text{coerce}_{1,1,1} \triangleq \lambda u. \text{ret}([\theta ()], [\theta ()])\]

\[T_{c0} = (1) \rightarrow M_0 ([1] \otimes [1])\]
\[T_{c1} = M_0 ([1] \otimes [1])\]
\[T_{c2} = (1) \otimes (1)\]

Dc1:
\[
\vdash s; u : (1) \vdash ([1] \otimes ([1] \otimes [1])) : T_{c2}
\]

\[\vdash s; u : ([1] \otimes ([1] \otimes [1])) \vdash T_{c2}\]

\[\vdash s; u : (1) \vdash \text{ret}([\theta ()], [\theta ()]) : T_{c1}\]

\[\vdash s; \vdash \lambda u. \text{ret}([\theta ()], [\theta ()]) : T_{c0}\]
\[ \Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash^q e : \tau' \leadsto e_a \quad \tau = \tau_1 \uparrow \tau_2 \quad \tau = \tau_1 = \tau_2 = b \] Share-base

\[ E_0 = \lambda u. E_1 \]

\[ E_1 = \text{bind } a = \text{coerce}_{b, b, b} z \text{ in let } \langle x, y \rangle = a \text{ in } e_a u \]

\[ T_0 = [q] 1 \rightarrow M_0 [q'] \langle \tau' \rangle \]

D1:

\[ \vdots; \vdots; (\Sigma); (\Gamma), u : [q] 1, x : (\tau_1), y : (\tau_2) \vdash e_a : T_0 \quad \vdots; \vdots; (\Sigma); u : [q] 1 \vdash u : [q] 1 \]

D0:

\[ \vdots; \vdots; (\Sigma); a : ((\tau_1) \circ (\tau_2)) \vdash a : ((\tau_1) \circ (\tau_2)) \quad D1 \]

Main derivation:

\[ \vdots; \vdots; (\Sigma); (\Gamma), u : [q] 1, a : ((\tau_1) \circ (\tau_2)) \vdash \text{let } \langle x, y \rangle = a \text{ in } e_a u : [q] 1 \rightarrow M_0 [q'] \langle \tau' \rangle \]

\[ \text{coerce}_{b, b, b} : (b) \rightarrow M_0 ((b) \circ (b)) \]

\[ \text{coerce}_{b, b, b} \triangleq \lambda u. \text{let } ! u' = u \text{ in ret } \langle [u'], [u'] \rangle \]

\[ T_{c0} = (b) \rightarrow M_0 ((b) \circ (b)) \]

\[ T_{c1} = M_0 ((b) \circ (b)) \]

\[ T_{c2} = (b) \circ (b) \]

Dc2:

\[ \vdots; \vdots; u' : b ; \vdash \langle [u'], [u'] \rangle : T_{c2} \]

Dc1:

\[ \vdots; \vdots; u : (b) \vdash ! u = u \text{ in ret } \langle [u'], [u'] \rangle : T_{c1} \]

\[ \vdots; \vdots; \vdash \lambda u. \text{let } ! u' = u \text{ in ret } \langle [u'], [u'] \rangle : T_{c0} \]

\[ \Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash^q e : \tau' \leadsto e_a \]

\[ \tau = L^\hat{p} x'' \]

\[ \tau_1 = L^\hat{p}_1 x'' \quad \tau_2 = L^\hat{p}_2 x'' \quad \tau'' = \tau'_1 \uparrow \tau'_2 \quad \hat{p} = \hat{p}_1 + \hat{p}_2 \]

Share-list

\[ E_0 = \lambda u. E_1 \]

\[ E_1 = \text{bind } a = \text{coerce}_{r, r, r} z \text{ in let } \langle x, y \rangle = a \text{ in } e_a u \]

\[ T_0 = [q] 1 \rightarrow M_0 [q'] \langle \tau' \rangle \]

D1:

\[ \vdots; \vdots; (\Sigma); (\Gamma), u : [q] 1, x : (\tau_1), y : (\tau_2) \vdash e_a : T_0 \quad \vdots; \vdots; (\Sigma); u : [q] 1 \vdash u : [q] 1 \]

D0:

\[ \vdots; \vdots; (\Sigma); a : ((\tau_1) \circ (\tau_2)) \vdash a : ((\tau_1) \circ (\tau_2)) \quad D1 \]

Main derivation:

\[ \vdots; \vdots; (\Sigma); (\Gamma), u : [q] 1, a : ((\tau_1) \circ (\tau_2)) \vdash \text{let } \langle x, y \rangle = a \text{ in } e_a u : [q] 1 \rightarrow M_0 [q'] \langle \tau' \rangle \]

\[ \vdots; \vdots; (\Sigma); (\Gamma), z : (\tau) \vdash \text{coerce}_{r, r, r} z : M_0 ((\tau_1) \circ (\tau_2)) \]

\[ \vdots; \vdots; (\Sigma); (\Gamma), z : (\tau) \vdash E_0 : M_0 [q'] \langle \tau' \rangle \]

\[ \vdots; \vdots; (\Sigma); (\Gamma), z : (\tau) \vdash \lambda u. E_0 : T_0 \]

66
\[
\begin{align*}
\text{coerce}_{\text{L}q, \text{L}p, \tau_1, \text{L}p, \tau_2} : \! \! & : \! \! E \rightarrow M \rightarrow \{t_1 \} \leq \{t_2 \} \\
\text{coerce}_{\text{L}q, \text{L}p, \tau_1, \text{L}p, \tau_2} \triangleq \! \! & \! \! \text{fix} f, \lambda g, \lambda e. \text{let } ! g' = g \text{ in } e; \text{let } \langle \! \langle p, l \rangle \rangle = x \text{ in } E_0
\end{align*}
\]

where

\[E_0 \triangleq \text{release } = p \text{ in } E_1\]

\[E_1 \triangleq \text{match } l \text{ with } |\text{nil} \mapsto E_{2.1} | h :: t \mapsto E_3\]

\[E_{2.1} \triangleq \text{bind } z_1 \text{ = store() in } E_{2.2}\]

\[E_{2.2} \triangleq \text{bind } z_2 \text{ = store() in } E_{2.3}\]

\[E_{2.3} \triangleq \text{ret}\{\langle z_1, \text{nil} \rangle, \langle z_2, \text{nil} \rangle\}\]

\[E_3 \triangleq \text{bind } H = g' \text{ in } E_{3.1}\]

\[E_{3.1} \triangleq \text{bind } o_1 = () \text{ in } E_{3.2}\]

\[E_{3.2} \triangleq \text{bind } T = f \vartriangleright g \langle o_1, t \rangle \text{ in } E_{4}\]

\[E_4 \triangleq \text{let } \langle H_1, H_2 \rangle = H \text{ in } E_5\]

\[E_5 \triangleq \text{let } \langle T_1, T_2 \rangle = T \text{ in } E_6\]

\[E_6 \triangleq T_1 \cdot t_1 \cdot \text{let } \langle p_1, l_1 \rangle = t_1 \text{ in } E_{7.1}\]

\[E_{7.1} \triangleq T_2 \cdot t_2 \cdot \text{let } \langle p_2, l_2 \rangle = t_2 \text{ in } E_{7.2}\]

\[E_{7.2} \triangleq \text{release } = p_1' \text{ in } E_{7.3}\]

\[E_{7.3} \triangleq \text{release } = p_2' \text{ in } E_{7.4}\]

\[E_{7.4} \triangleq \text{bind } o_1 = \text{store() in } E_{7.5}\]

\[E_{7.5} \triangleq \text{bind } o_2 = \text{store() in } E_{8}\]

\[E_8 \triangleq \text{ret}\langle \langle o_1, H_1 :: T_1 \rangle , \langle o_2, H_2 :: T_2 \rangle \rangle\]

\[T_0 = \langle \langle t \rangle \rangle \rightarrow M \rightarrow \{t_1 \} \leq \{t_2 \} \rightarrow M \rightarrow \{L^q \cdot \tau_1 \} \leq \{L^q \cdot \tau_2 \}\]

\[T_1 = \langle \langle t \rangle \rangle \rightarrow M \rightarrow \{t_1 \} \leq \{t_2 \}\]

\[T_1.0 = \exists s . (\langle \phi (\vartriangleright p, s) \rangle 1 \leq \text{L}^* \langle \tau \rangle)\]

\[T_1.1 = \langle\langle\phi (\vartriangleright p, s)\rangle \leq \text{L}^* \langle \tau \rangle)\]

\[T_1.2 = \langle\langle\phi (\vartriangleright p, s)\rangle \leq 1\]

\[T_1.3 = \text{L}^* \langle \tau \rangle\]

\[T_2 = \langle \text{L}^* \langle \tau \rangle) \rightarrow M \rightarrow \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_3 = \text{M} \rightarrow \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_3.1 = \text{M}(\phi (\vartriangleright p, s)) \leq \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_3.11 = \text{M}(\phi (\vartriangleright p, s - 1)) \leq \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_3.12 = \{\phi (\vartriangleright p, s - 1)\} \leq \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_4 = \text{M} \rightarrow \{t_1 \} \leq \{t_2 \}\]

\[T_4.1 = \{t_1 \} \leq \{t_2 \}\]

\[T_5 = \text{M} \rightarrow \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_5.1 = \{\text{L}^q \cdot \tau_1 \} \leq \{\text{L}^q \cdot \tau_2 \}\]

\[T_5.2 = \{\text{L}^q \cdot \tau_1 \} \leq \exists s_1 . (\langle \phi (\vartriangleright p_1, s_1) \rangle 1 \leq \text{L}^* \langle \tau_1 \rangle)\]

\[T_5.21 = \{\phi (\vartriangleright p_1, s_1) \leq 1 \leq \text{L}^* \langle \tau_1 \rangle)\]

\[T_5.22 = \{\phi (\vartriangleright p_1, s_1) \leq 1 \]

\[T_5.23 = \text{L}^* \langle \tau_1 \rangle\]

\[T_5.3 = \{\text{L}^q \cdot \tau_2 \} \leq \exists s_2 . (\langle \phi (\vartriangleright p_2, s_2) \rangle 1 \leq \text{L}^* \langle \tau_2 \rangle)\]

\[T_5.31 = \{\phi (\vartriangleright p_2, s_2) \leq 1 \leq \text{L}^* \langle \tau_2 \rangle)\]
\[
\begin{align*}
T_{3.32} &= [\phi(q \bar{p}_2, s'_2)] 1 \\
T_{3.33} &= L^{\mu_2}(\bar{r}_2) \\
P_1 &= p_1 \downarrow + \phi(q \bar{p}_1, s'_1) \\
P_2 &= p_2 \downarrow + \phi(q \bar{p}_2, s'_2) \\
P_6 &= M_{P_1}([P_1] 1) \\
T_{6.1} &= [P_1] 1 \\
T_7 &= M_{P_2}([P_2] 1) \\
T_{7.1} &= [P_2] 1 \\
T_{8.0} &= M(\bar{g} \downarrow_1) ([L^{\bar{p}_1} \vec{r}_1] \otimes [L^{\bar{p}_2} \vec{r}_2]) \\
T_{8.1} &= M(\bar{g} \downarrow_1 + P_1) ([L^{\bar{p}_1} \vec{r}_1] \otimes [L^{\bar{p}_2} \vec{r}_2]) \\
T_{8.2} &= M(\bar{g} \downarrow_1 + P_1 + P_2) ([L^{\bar{p}_1} \vec{r}_1] \otimes [L^{\bar{p}_2} \vec{r}_2]) \\
T_{8.3} &= M(\bar{p}_2 \downarrow_1 + P_2) ([L^{\bar{p}_1} \vec{r}_1] \otimes [L^{\bar{p}_2} \vec{r}_2]) \\
T_{8.4} &= M 0 ([L^{\bar{p}_1} \vec{r}_1] \otimes [L^{\bar{p}_2} \vec{r}_2]) \\
T_{8.41} &= [L^{\bar{p}_1} \vec{r}_1] \otimes [L^{\bar{p}_2} \vec{r}_2] \\
T_{8.5} &= ([L^{\bar{p}_1} \vec{r}_1] \\
T_{8.51} &= \exists s_1.([\phi(p_1, s_1)] 1 \otimes L(s_1)\{\bar{r}_1\}) \\
T_{8.52} &= ([\phi(p_1, s_1)] 1 \otimes L(s_1)\{\bar{r}_1\}) \\
T_{8.6} &= ([L^{p_2} \vec{r}_2]) \\
T_{8.61} &= \exists s_2.([\phi(p_2, s_2)] 1 \otimes L(s_2)\{\bar{r}_2\}) \\
T_{8.62} &= ([\phi(p_2, s_2)] 1 \otimes L(s_2)\{\bar{r}_2\}) \\
\end{align*}
\]

D1.82:

\[
\begin{align*}
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_2 : \{\bar{r}_2\}, l'_2 : T_{3.33}, o_2 : T_{7.1} \vdash [\{o_2, H_2 \vdash \bar{l}_2\}] : T_{8.62} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_2 : \{\bar{r}_2\}, l'_2 : T_{3.33}, o_2 : T_{7.1} \vdash [\{o_2, H_2 \vdash \bar{l}_2\}] : T_{8.62} \\
\end{align*}
\]

D1.81:

\[
\begin{align*}
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, l'_1 : T_{5.23}, o_1 : T_{6.1} \vdash [\{o_1, H_1 \vdash \bar{l}_1\}] : T_{8.52} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, l'_1 : T_{5.23}, o_1 : T_{6.1} \vdash [\{o_1, H_1 \vdash \bar{l}_1\}] : T_{8.52} \\
\end{align*}
\]

D1.8:

\[
\begin{align*}
D1.81 & \quad D1.182 \\
; s_2, s'_1, s, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.23}, l'_2 : T_{3.33}, o_1 : T_{6.1}, o_2 : T_{7.1} \vdash \langle\langle o_1, H_1 \vdash \bar{l}'_1\rangle, \langle o_2, H_2 \vdash \bar{l}'_2\rangle\rangle : T_{8.41} \\
; s_2, s'_1, s, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.23}, l'_2 : T_{3.33}, o_1 : T_{6.1}, o_2 : T_{7.1} \vdash \text{ret}\langle\langle o_1, H_1 \vdash \bar{l}'_1\rangle, \langle o_2, H_2 \vdash \bar{l}'_2\rangle\rangle : T_{8.4} \\
; s_2, s'_1, s, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.23}, l'_2 : T_{3.33}, o_1 : T_{6.1}, o_2 : T_{7.1} \vdash \text{E}_8 : T_{8.4} \\
\end{align*}
\]

D1.75:

\[
\begin{align*}
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; \vdash \text{store()} : T_7 \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.33}, o_1 : T_{6.1} \vdash \text{bind } o_2 = \text{store()} \text{ in } E_8 : T_{8.3} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.33}, o_1 : T_{6.1} \vdash \text{E}_7.5 : T_{8.3} \\
\end{align*}
\]

D1.74:

\[
\begin{align*}
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; \vdash \text{store()} : T_6 \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.33}, o_1 : T_{6.1} \vdash \text{bind } o_1 = \text{store()} \text{ in } E_7.5 : T_{8.2} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.33}, E_{7.4} : T_{8.2} \\
\end{align*}
\]

D1.73:

\[
\begin{align*}
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; p_2 : T_{5.32} \vdash p_2 : T_{5.32} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.23}, p_2 : T_{5.32}, l'_2 : T_{5.33} \vdash \text{release } = p_2' \text{ in } E_{7.4} : T_{8.1} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, l'_1 : T_{5.23}, p_2 : T_{5.32}, l'_2 : T_{5.33} \vdash \text{E}_7.3 : T_{8.1} \\
\end{align*}
\]

D1.72:

\[
\begin{align*}
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; \vdash \text{store()} : T_8.0 \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, p_1' : l'_1 : p_2' : T_{3.32}, l'_2 : T_{5.33} \vdash \text{E}_7.2 : T_{8.0} \\
; s_2, s'_1, s_1, s; : g' : T_1', f : T_0; H_1 : \{\bar{r}_1\}, H_2 : \{\bar{r}_2\}, p_1' : l'_1 : p_2' : T_{3.32}, l'_2 : T_{5.33} \vdash \text{E}_7.3 : T_{8.0} \\
\end{align*}
\]

68
D1.71:
\[
\vdash s_1', s_1, s; g' : T_1', f : T_0; T_1 : T_5.3, T_3.3 \vdash T_1 : T_5.2
\]
\[
\vdash s_1, s; g' : T_1', f : T_0; H_1 : \{\tau_1\}, H_2 : \{\tau_2\}, T_2; T_5.3, l_1' : T_5.23 \vdash T_2; T_3.3, \text{let}(\langle p_2', l_2' \rangle) = t p_2 \text{ in } E_7 : T_8.0
\]
\[
\vdash s_1, s; g' : T_1', f : T_0; H_1 : \{\tau_1\}, H_2 : \{\tau_2\}, T_2; T_5.3, p_1' : T_5.22, l_1' : T_5.23 + E_7 : T_8.0
\]

D1.61:
\[
\vdash s_1', s_1; g' : T_1', f : T_0; T_1 : T_5.21 + (p_1) : T_3.21
\]
\[
\vdash s_1, s; g' : T_1', f : T_0; H_1 : \{\tau_1\}, H_2 : \{\tau_2\}, T_2; T_5.3, t p_1 : T_5.21 + \text{let}(\langle p_1', l_1' \rangle) = t p_1 \text{ in } E_7 : T_8.0
\]

D1.6:
\[
\vdash s; g' : T_1', f : T_0; T_1 : T_5.2, T_3.3 \vdash T_1 : T_5.2
\]
\[
\vdash s; g' : T_1', f : T_0; p : T_1.2, H_1 : \{\tau_1\}, H_2 : \{\tau_2\}, T_1 : T_5.2, T_1 : T_3.3 \vdash T_3; T_1, \text{let}(\langle p_1', l_1' \rangle) = t p_1 \text{ in } E_7 : T_8.0
\]
\[
\vdash s; g' : T_1', f : T_0; \text{let}(\langle T_1, T_2 \rangle) = T \text{ in } E_0 : T_8.0
\]

D1.5:
\[
\vdash s; g' : T_1', f : T_0; p : T_1.2, H : T_4.1 + H : T_4.1
\]
\[
\vdash s; g' : T_1', f : T_0; p : T_1.2, H : T_4.1, T : T_5.1 + E_4 : T_8.0
\]

D1.4:
\[
\vdash s; g' : T_1', f : T_0; t : L^{\ast-1}\{\tau\}, \alpha : T_{3.12} + \text{let } T = f \langle \alpha, t \rangle \text{ in } E_0 : T_8.0
\]

D1.21:
\[
\vdash s; g' : T_1', f : T_0; p : T_1.2, h : \{\tau\}, t : L^{\ast-1}\{\tau\} + \text{store()} : T_{3.11}
\]
\[
\vdash s; g' : T_1', f : T_0; p : T_1.2, h : \{\tau\}, t : L^{\ast-1}\{\tau\} + \text{bind } \alpha = \text{store()} \text{ in } E_0 : T_{3.12}
\]

D1.13:
\[
\vdash s; g' : T_1', f : T_0; p : T_1.2, h : \{\tau\}, t : L^{\ast-1}\{\tau\} + E_3 : T_{3.1}
\]

D1.14:
\[
\vdash s; g' : T_1', f : T_0; z_2 : [0] + z_2 : [0]
\]
\[
\vdash s; g' : T_1', f : T_0; z_2 : [0] + \text{nil} : L^0(\{\tau_2\})
\]
\[
\vdash s; g' : T_1', f : T_0; z_2 : [0] + \text{nil} : \langle \{z_2, \text{nil}\}, ([s'] 1 \otimes L^s(\{\tau_2\})
\]

D1.13:
\[
\vdash s; g' : T_1', f : T_0; z_1 : [0] + z_1 : [0]
\]
\[
\vdash s; g' : T_1', f : T_0; z_1 : [0] + \text{nil} : L^0(\{\tau_1\})
\]
\[
\vdash s; g' : T_1', f : T_0; z_1 : [0] + \text{nil} : \langle \{z_1, \text{nil}\}, ([s'] 1 \otimes L^s(\{\tau_1\})
\]

D1.13:
D1.12: 
\[ \vdash s; g^f : T^f_1, f : T_0; z_1 : [0] 1, z_2 : [0] 1 : \langle\langle z_1, \text{nil}\rangle, \langle z_2, \text{nil}\rangle \rangle : T_3 : \text{store}() : M 0 [0] 1 \]
\[ \vdash s; g^f : T^f_1, f : T_0; z_1 : [0] 1, z_2 : [0] 1 : \text{bind}_z = \text{store}() \text{ in } E_{2.3} : T_{3.1} \]
\[ \vdash s; g^f : T^f_1, f : T_0; z_1 : [0] 1, z_2 : [0] 1 : E_{2.2} : T_{3.1} \]

D1.13 
\[ \vdash s; g^f : T^f_1, f : T_0; l : T_{1.3} \]
\[ \vdash s; g^f : T^f_1, f : T_0; l : T_{1.3} \]

D1.14 
\[ \vdash s; g^f : T^f_1, f : T_0; z_1 : [0] 1, z_2 : [0] 1 : \text{ret} \langle\langle z_1, \text{nil}\rangle, \langle z_2, \text{nil}\rangle \rangle : T_{3.1} \]
\[ \vdash s; g^f : T^f_1, f : T_0; z_1 : [0] 1, z_2 : [0] 1 : E_{2.3} : T_{3.1} \]

D1.11: 
\[ \vdash s; g^f : T^f_1, f : T_0; l : T_{1.3} \]
\[ \vdash s; g^f : T^f_1, f : T_0; l : T_{1.3} \]

D1.10: 
\[ \vdash s; g^f : T^f_1, f : T_0; p ; T_{1.2} ; l : T_{1.3} \]
\[ \vdash s; g^f : T^f_1, f : T_0; p ; T_{1.2} ; l : T_{1.3} \]

D1: 
\[ \vdash s; g^f : T^f_1, f : T_0; l : T_{1.3} \]

D0.3: 
\[ \vdash s; g^f : T^f_1, f : T_0; p ; T_{1.2} ; l : T_{1.3} \]
\[ \vdash s; g^f : T^f_1, f : T_0; p ; T_{1.2} ; l : T_{1.3} \]

D0.2: 
\[ \vdash s; g^f : T^f_1, f : T_0; p ; T_{1.2} ; l : T_{1.3} \]
\[ \vdash s; g^f : T^f_1, f : T_0; p ; T_{1.2} ; l : T_{1.3} \]

D0.1: 
\[ \vdash s; g^f : T^f_1, f : T_0; x ; T_{1.1} \]
\[ \vdash s; g^f : T^f_1, f : T_0; x ; T_{1.1} \]

D0: 
\[ \vdash s; g^f : T^f_1, f : T_0; y ; T_{1.1} \]
\[ \vdash s; g^f : T^f_1, f : T_0; y ; T_{1.1} \]

\[ \Gamma, x : \tau_1, y : \tau_2 \vdash \gamma^f e : \tau' \rightsquigarrow e_a \quad \tau = \tau_1 \land \tau_2 \quad \tau = \tau_a \wedge \tau_b \quad \tau = \tau_{a'} \wedge \tau_{b'} \quad \tau = \tau_{a''} \wedge \tau_{b''} \]

Share-pair

\[ E_0 = \lambda u. E_1 \]
\[ E_1 = \text{bind } a = \text{coerce}(\tau_a, \tau_b)(\tau_{a'}, \tau_{b'})(\tau_{a''}, \tau_{b''}) \text{ z in let}(\langle x, y \rangle) = a \text{ in } e_a u \]
\[ T_0 = [q] 1 \rightarrow M 0 ([q'] [\tau']) \]

D1: 
\[ \vdash \varepsilon : [\Sigma]; [\Gamma], u : [q] 1, x : \langle \tau_1 \rangle, y : \langle \tau_2 \rangle \rightarrow e_a : T_0 \]
\[ \vdash \varepsilon : [\Sigma]; [\Gamma], u : [q] 1, x : \langle \tau_1 \rangle, y : \langle \tau_2 \rangle \rightarrow e_a : M 0 [q'] 1 \]

D0: 
\[ \vdash \varepsilon : [\Sigma]; a : (\langle \tau_1 \rangle \otimes \langle \tau_2 \rangle) \rightarrow a : (\langle \tau_1 \rangle \otimes \langle \tau_2 \rangle) \]
\[ \vdash \varepsilon : [\Sigma]; u : [q] 1, a : (\langle \tau_1 \rangle \otimes \langle \tau_2 \rangle) \rightarrow \text{let}(\langle x, y \rangle) = a \text{ in } e_a u : [q] \rightarrow M 0 [q] [\tau'] \]
Main derivation:

\[ \cdots ; \cdots ; \vdash_c \tau_0, \tau_0 \vdash \text{coerce}_{(\tau_0, \tau_0, \tau_0', \tau_0'')} \vdash M_0 (\langle \tau_1 \rangle \circ \langle \tau_2 \rangle) \]

\[ \vdash_c \tau_0, \tau_0 \vdash \nu \vdash E_0 \vdash M_0 \langle \nu \rangle \langle \tau_1 \rangle \]

\[ \vdash_c \tau_0, \tau_0 \vdash \lambda \text{let } \nu ! \vdash E_0 \]

\[ \vdash_c \tau_0, \tau_0 \vdash \text{coerce}_{(\tau_0, \tau_0, \tau_0', \tau_0'')} \triangleq 0, g_1, \lambda \text{let } \nu ! \]

where

\[ E_0 \triangleq \text{let } \nu ! g_1 = g_{1_1} \text{ in } E_{1_1} \]

\[ E_1 \triangleq \text{let } \nu ! g_2 = g_{2_1} \text{ in } E_{2_1} \]

\[ E_2 \triangleq \text{bind } \nu ! P_1 = g_{1_2} \text{ in } E_{3_1} \]

\[ E_3 \triangleq \text{bind } P_2 = g_{2_2} \text{ in } E_{4_1} \]

\[ E_4 \triangleq \text{let } \langle p_{1_1}, p_{1_2} \rangle = P_1 \text{ in } E_5 \]

\[ E_5 \triangleq \text{let } \langle p_{2_1}, p_{2_2} \rangle = P_2 \text{ in } E_6 \]

\[ E_6 \triangleq \text{ret } \langle p_{1_1}, p_{2_1}, p_{1_2}, p_{2_2} \rangle \]

D0:

\[ \vdash_c \tau_0, \tau_0 \vdash \text{coerce}_{(\tau_0, \tau_0, \tau_0', \tau_0'')} \vdash 0, \nu \vdash M_0 (\langle \tau_1 \rangle \circ \langle \tau_2 \rangle) \]

\[ \vdash_c \tau_0, \tau_0 \vdash \nu \vdash E_0 \vdash M_0 \langle \nu \rangle \langle \tau_1 \rangle \]

\[ \vdash_c \tau_0, \tau_0 \vdash \lambda \text{let } \nu ! \vdash E_0 \]

\[ \vdash_c \tau_0, \tau_0 \vdash \text{coerce}_{(\tau_0, \tau_0, \tau_0', \tau_0'')} \triangleq 0, g_1, \lambda \text{let } \nu ! \]

D6:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; p_{1_1} \vdash T_{1,11}; p_{1_2} \vdash T_{1,12}; p_{2_1} \vdash T_{2,11}; p_{2_2} \vdash T_{2,12} \vdash \langle p_{1_1}, p_{2_1} \rangle, \langle p_{1_2}, p_{2_2} \rangle : T_{0,61} \]

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; p_{1_1} \vdash T_{1,11}; p_{1_2} \vdash T_{1,12}; p_{2_1} \vdash T_{2,11}; p_{2_2} \vdash T_{2,12} \vdash \text{ret } \langle p_{1_1}, p_{2_1}, p_{1_2}, p_{2_2} \rangle : T_{0,6} \]

D5:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; p_{1_1} \vdash T_{1,11}; p_{1_2} \vdash T_{1,12}; p_{2_1} \vdash T_{2,11}; p_{2_2} \vdash T_{2,12} \vdash E_6 : T_{0,6} \]

D4:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; P_2' \vdash T_{2,1} \vdash P_2' : T_{2,1} \]

D4:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; P_2' \vdash T_{2,1} \vdash \text{ret } \langle p_{1_1}, p_{1_2} \rangle = P_2' \text{ in } E_0 : T_{0,6} \]

D4:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; P_2' \vdash T_{2,1} \vdash E_4 : T_{0,6} \]

D4:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; P_2' \vdash T_{2,1} \vdash \text{bind } P_2 = g_{2_2} \text{ in } E_4 : T_{0,6} \]

D4:

\[ \vdash f : T_0; g_1 \vdash T_{0,32}; g_2 \vdash T_{0,42}; P_2' \vdash T_{2,1} \vdash E_3 : T_{0,6} \]
D2:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; g_1' : T_{0.32}, g_2' : T_{0.42}; p_1 : (\tau_1) \vdash g_1' p_1 : T_1 \\
\vdots \vdots \vdots f &: T_0; g_1' : T_{0.32}, g_2' : T_{0.42}; p_1 : (\tau_1), p_2 : (\tau_2) \vdash \text{bind} P' \equiv g_1' p_1 \text{ in } E_3 : T_{0.6}
\end{align*}
\]

D3:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; g_1' : T_{0.32}, g_2' : T_{0.42}; p_1 : (\tau_1), p_2 : (\tau_2) \vdash g_2 = g_2 \text{ in } E_2 : T_{0.6}
\end{align*}
\]

D1:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; g_1' : T_{0.32}, g_2' : T_{0.41}; g_2 : T_{0.41} \vdash g_2 : T_{0.41}
\end{align*}
\]

D2:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; g_1' : T_{0.32}, g_2' : T_{0.41}; p_1 : (\tau_1), p_2 : (\tau_2) \vdash \text{let} \ g_1' = g_1 \text{ in } E_1 : T_{0.6}
\end{align*}
\]

D0.1:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; g_1 : T_{0.31}, g_2 : T_{0.41}, g_1 : T_{0.31} \vdash g_1 : T_{0.31}
\end{align*}
\]

D0:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; p : T_{0.51} \vdash p : T_{0.51}
\end{align*}
\]

D0.1:

\[
\begin{align*}
\vdots \vdots \vdots f &: T_0; g_1 : T_{0.31}, g_2 : T_{0.41}, p : T_{0.51} \vdash \text{let} \ ((p_1, p_2)) = p \text{ in } E_0 : T_{0.6}
\end{align*}
\]

9. Sub:

\[
\begin{align*}
\Sigma; \Gamma \vdash^\eta e : \tau \rightsquigarrow e_a & \quad \tau <: \tau' \\
\Sigma; \Gamma \vdash^\eta e : \tau' \rightsquigarrow e_a & \quad \text{Sub}
\end{align*}
\]

Main derivation:

\[
\begin{align*}
\vdots \vdots \vdots (\Sigma); (\Gamma) \vdash e_a : [q] 1 \rightarrow M 0 ([q'](\tau)) \\
\vdots \vdots \vdots (\Sigma); (\Gamma) \vdash (\tau) <: (\tau') \quad \text{Lemma [22]} \\
\vdots \vdots \vdots (\Sigma); (\Gamma) \vdash e_a : [q] 1 \rightarrow M 0 ([q'](\tau')) & \quad \text{T-sub}
\end{align*}
\]

10. Super:

\[
\begin{align*}
\Sigma; \Gamma, x : \tau_1 \vdash^\eta e : \tau \rightsquigarrow e_a & \quad \tau'_1 <: \tau_1 \\
\Sigma; \Gamma, x : \tau'_1 \vdash^\eta e : \tau \rightsquigarrow e_a & \quad \text{Super}
\end{align*}
\]

Main derivation:

\[
\begin{align*}
\vdots \vdots \vdots (\Sigma); (\Gamma), x : (\tau_1) \vdash e_a : [q] 1 \rightarrow M 0 ([q'](\tau)) \\
\vdots \vdots \vdots (\Sigma); (\Gamma) \vdash (\tau) <: (\tau_1) \\
\vdots \vdots \vdots (\Sigma); (\Gamma), x : (\tau'_1) \vdash e_a : [q] 1 \rightarrow M 0 ([q'](\tau'_1)) & \quad \text{Lemma [22]} \\
\vdots \vdots \vdots (\Sigma); (\Gamma), x : (\tau'_1) \vdash e_a : [q] 1 \rightarrow M 0 ([q'](\tau'_1)) & \quad \text{T-weaken}
\end{align*}
\]

11. Relax:

\[
\begin{align*}
\Sigma; \Gamma \vdash^\eta e : \tau \rightsquigarrow e_a & \quad q \geq p \quad q - p \geq q' - p' \\
\Sigma; \Gamma \vdash^\eta e : \tau \rightsquigarrow \lambda \alpha . E_0 & \quad \text{Relax}
\end{align*}
\]

where

\begin{align*}
E_0 &= \text{release } a = o \text{ in } E_1 \\
E_1 &= \text{bind } a = \text{store}(o) \text{ in } E_2 \\
E_2 &= \text{bind } b = e_a \text{ a } \text{ in } E_3 \\
E_3 &= \text{release } c = b \text{ in } \text{store } c
\end{align*}

D2:

\[
\begin{align*}
\vdots \vdots \vdots (\Sigma); b : [p'](\tau) \vdash b : [p'](\tau) \\
\vdots \vdots \vdots (\Sigma); c : [\tau] \vdash \text{store } c : M(q - p + p')([q - p + p'](\tau)) \\
\vdots \vdots \vdots (\Sigma); b : [p'](\tau) \vdash E_3 : M(q - p) ([q - p + p'](\tau))
\end{align*}
\]
D1.2: \[ \vdash : \langle \Sigma \rangle ; a : [p] 1 \vdash a : [p] 1 \]

D1.1: \[ \vdash : \langle \Sigma \rangle ; (\Gamma) \vdash e_a : [p] 1 \rightarrow M 0 ([p']{\tau}) \]

D1: \[ \begin{array}{c}
\vdash : \langle \Sigma \rangle ; (\Gamma), a : [p] 1 \vdash e_a a : M 0 ([p']{\tau}) \\
\vdash : \langle \Sigma \rangle ; (\Gamma), a : [p] 1 \vdash E_2 : M(q - p) ([q - p + p']{\tau})
\end{array} \]

D2: \[ \vdash : \langle \Sigma \rangle ; (\Gamma), a : [p] 1 \vdash E_1 : M(q - p + p')([q']{\tau}) \]

D0: \[ \vdash : \langle \Sigma \rangle ; \vdash store() : M p ([p] 1) \]

D0.0: \[ \vdash : \langle \Sigma \rangle ; (\Gamma) \vdash \Gamma : M 0 ([q - p + p']{\tau}) \]

Main derivation:

\[ \begin{array}{c}
\vdash : \langle \Sigma \rangle ; a : [q] 1 \vdash a : [q] 1 \\
\vdash : \langle \Sigma \rangle ; (\Gamma), a : [q] 1 \vdash E_0 : M 0 ([q - p + p']{\tau}) \\
\vdash : \langle \Sigma \rangle ; (\Gamma), a : [q] 1 \vdash E_5 : M 0 ([q']{\tau})
\end{array} \]

\[ \vdash : \langle \Sigma \rangle ; (\Gamma) \vdash \lambda o. E_0 : [q] 1 \rightarrow M 0 ([q']{\tau}) \]

12. Let:

\[ \begin{array}{c}
\Sigma; \Gamma_1 \vdash p^{q - K^{'ext}_3} e_1 : \tau_1 \rightsquigarrow e_{a_1} \\
\Sigma; \Gamma_2, x : \tau_1 \vdash p^{q - K^{'ext}_2} e_2 : \tau_1 \rightsquigarrow e_{a_2}
\end{array} \]

\[ \vdash : \langle \Sigma \rangle; \Gamma_1, \Gamma_2 \vdash \langle q \rangle^{q' \rightarrow q} let \; x = e_1 \; in \; e_2 : \tau \rightsquigarrow \mu \]

where

\( E_1 = \lambda u. E_0 \)
\( E_0 = \text{release} \; - \; u \; \text{in} \; E_1 \)
\( E_1 = \text{bind} \; - \; \tau^{K^{'ext}_3} \; \text{in} \; E_2 \)
\( E_2 = \text{bind} \; a \; \text{store}() \; \text{in} \; E_3 \)
\( E_3 = \text{bind} \; b \; = \; e_{a_1} \; a \; \text{in} \; E_4 \)
\( E_4 = \text{release} \; x \; = \; b \; \text{in} \; E_5 \)
\( E_5 = \text{bind} \; - \; = \; \tau^{K^{'ext}_2} \; \text{in} \; E_6 \)
\( E_6 = \text{bind} \; c \; = \; \text{store}() \; \text{in} \; E_7 \)
\( E_7 = \text{bind} \; d \; = \; e_{a_2} \; c \; \text{in} \; E_8 \)
\( E_8 = \text{release} \; f \; = \; d \; \text{in} \; E_9 \)
\( E_9 = \text{bind} \; - \; = \; \tau^{K^{'ext}_1} \; \text{in} \; E_{10} \)
\( E_{10} = \text{bind} \; g \; = \; \text{store} \; f \; \text{in} \; \text{ret} \; g \)

\( T_0 = [q] 1 \rightarrow M 0 ([q']{\tau}) \)
\( T_{0.1} = [q] 1 \)
\( T_{0.2} = M 0 ([q']{\tau}) \)
\( T_{0.3} = M q ([q']{\tau}) \)
\( T_{0.4} = M(q - K^{'ext}_3) ([q']{\tau}) \)
\( T_{0.5} = M(q - K^{'ext}_2) ([q - K^{'ext}_1] 1) \)
\( T_{0.51} = [q - K^{'ext}_1] 1 \)
\( T_{0.6} = M 0 ([q']{\tau}) \)
\( T_{0.61} = [q] 1 \)
\( T_{0.7} = M p ([q']{\tau}) \)
\( T_{0.8} = M(p - K^{'ext}_3) ([q']{\tau}) \)
\( T_{0.9} = M(p - K^{'ext}_2) ([p - K^{'ext}_1] 1) \)
\( T_{0.91} = [(p - K^{'ext}_1)] 1 \)
\( T_1 = M 0 ([q' + K^{'ext}_2]) ([q]{\tau}) \)
\(T_{1.1} = \|([q' + K_3^{let}])\{\tau\}\)
\(T_{1.2} = M([q' + K_3^{let}])\{([q']\{\tau\}\})\)
\(T_{1.3} = M q'\{([q']\{\tau\}\})\)

D10:
\[\cdots; (\Sigma); g : [q']\{\tau\} \vdash \text{ret } : M 0 [q']\{\tau\}\]

D9:
\[\cdots; (\Sigma); f : \{\tau\} \vdash \text{store } f : T_{1.3}\]
\[\cdots; (\Sigma); f : \{\tau\} \vdash \text{bind } g = \text{store } f \text{ in ret } : T_{1.3}\]
\[\cdots; (\Sigma); f : \{\tau\} \vdash E_{10} : T_{1.3}\]

D8:
\[\cdots; (\Sigma); d : T_{1.1} \vdash d : T_{1.1}\]
\[\cdots; (\Sigma); d : T_{1.1} \vdash \text{release } f = d \text{ in } E_{8} : T_{0.2}\]

D7:
\[\cdots; (\Sigma); (\Gamma)_2), c : T_{0.91} \vdash e_{a2} c : T_{1}\]
\[\cdots; (\Sigma); (\Gamma)_2), c : T_{0.91} \vdash \text{bind } d = e_{a2} c \text{ in } E_{8} : T_{0.2}\]
\[\cdots; (\Sigma); (\Gamma)_2), c : T_{0.91} \vdash E_{7} : T_{0.2}\]

D6:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash \text{store }() : T_{0.9}\]
\[\cdots; (\Sigma); (\Gamma)_2) \vdash E_{5} : T_{0.8}\]

D5:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash \text{bind } = \vdash M K_2^{let} 1\]
\[\cdots; (\Sigma); (\Gamma)_2) \vdash\]

D4:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash\]

D3:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash\]

D2:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash\]

D1:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash\]

D0:
\[\cdots; (\Sigma); (\Gamma)_2) \vdash\]
Main derivation:

\[
\vdots\vdots\vdots(\Sigma);(\Gamma_1),(\Gamma_2),u:T_{0,1}\vdash u:T_{0,1}^D0
\]

\[
\vdots\vdots\vdots(\Sigma);(\Gamma_1),(\Gamma_2),u:T_{0,1}\vdash release=\ -\ u\ in\ E_1:T_{0,2}^D1
\]

\[
\vdots\vdots\vdots(\Sigma);(\Gamma_1),(\Gamma_2),u:T_{0,1}\vdash E_0:T_{0,2}^D2
\]

13. Pair:

\[
\Sigma;x_1:τ_1,x_2:τ_2\vdash_q^{K_{pair}}(x_1,x_2):(τ_1,τ_2)\rightsquigarrow E_t
\]

where

\[
\begin{align*}
E_t &= λu.E_0 \\
E_0 &= release=\ -\ u\ in\ E_1 \\
E_1 &= bind=\ -\ τ_1^{K_{pair}}\ in\ E_2 \\
E_2 &= bind\ a=\ store(x_1,x_2)\ in\ ret\ a \\
T_0 &= ([q+K_{pair}]1\ →\ M\ 0\ ([q]\ τ_1)\ \&\ \tau_2) \\
T_{0,1} &= ([q+K_{pair}]1 \\
T_{0,2} &= M\ 0\ ([q]\ τ_1)\ \&\ \tau_2) \\
T_{0,3} &= M\ ([q+K_{pair})\ ([q]\ τ_1)\ \&\ \tau_2) \\
T_{0,4} &= M\ q\ ([q]\ τ_1)\ \&\ \tau_2)
\end{align*}
\]

D2:

\[
\vdots\vdots\vdots(\Sigma);a:[q]\ τ_1\ \&\ \tau_2\vdash ret\ a:M\ 0\ ([q]\ τ_1)\ \&\ \tau_2
\]

D1:

\[
\vdots\vdots\vdots(\Sigma);x_1:([q]\ τ_1),x_2:([q]\ τ_2)\vdash store(x_1,x_2):T_{0,4}^D2
\]

\[
\vdots\vdots\vdots(\Sigma);x_1:([q]\ τ_1),x_2:([q]\ τ_2)\vdash bind\ a=\ store(x_1,x_2)\ in\ ret\ a:T_{0,4}^D1
\]

Main derivation:

\[
\vdots\vdots\vdots(\Sigma);x_1:([q]\ τ_1),x_2:([q]\ τ_2),u:T_{0,1}\vdash u:T_{0,1}^D0
\]

\[
\vdots\vdots\vdots(\Sigma);x_1:([q]\ τ_1),x_2:([q]\ τ_2),u:T_{0,1}\vdash release=\ -\ u\ in\ E_1:T_{0,2}^D1
\]

\[
\vdots\vdots\vdots(\Sigma);x_1:([q]\ τ_1),x_2:([q]\ τ_2),u:T_{0,1}\vdash E_0:T_{0,2}^D2
\]

\[
\vdots\vdots\vdots(\Sigma);x_1:([q]\ τ_1),x_2:([q]\ τ_2)\vdash λu.E_0:T_{0,3}^D3
\]

14. MatP:

\[
τ=(τ_1,τ_2)
\]

1. \[
Σ,Γ,x_1:τ_1,x_2:τ_2\vdash_q^{K_{matP}}e:τ\rightsquigarrow e_t
\]

\[
Σ,Γ,x:τ\vdash_q^{K_{matP}}match\ x\ with\ (x_1,x_2)\ →\ c:τ\rightsquigarrow e_t
\]

where

\[
\begin{align*}
E_t &= λu.E_0 \\
E_0 &= release=\ -\ u\ in\ E_1 \\
E_1 &= bind=\ -\ τ_1^{K_{matP}}\ in\ E_2 \\
E_2 &= let\ ⟨x_1,x_2⟩=\ x\ in\ E_3 \\
E_3 &= bind\ a=\ store()\ in\ E_4 \\
E_4 &= bind\ b=e_t\ a\ in\ E_5 \\
E_5 &= release\ c=b\ in\ E_6 \\
E_6 &= bind=\ -\ τ_1^{K_{matP}}\ in\ E_7 \\
E_7 &= bind\ d=\ store\ c\ in\ ret\ d \\
T_0 &= [q]1\ →\ M\ 0\ ([q]\ [τ'])
\end{align*}
\]

\[
T_{0,1} = [q]1
\]
$T_{0.2} = \mathbb{M} \mathbb{O} \{q'[r']\}$
$T_{0.3} = \mathbb{M} q \{q'[r']\}$
$T_{0.4} = \mathbb{M} (q - K_1^{matP}) \{q'[r']\}$
$T_{0.5} = \mathbb{M} (q - K_1^{matP}) \{(q - K_1^{matP}) \mathbb{I}\}$
$T_{0.51} = [(q - K_1^{matP}) \mathbb{I}]$
$T_{0.6} = \mathbb{M} 0 \{q'[r']\}$
$T_{0.61} = [(q' + K_2^{matP}) \{r'\}]$
$T_{0.7} = \mathbb{M} (q' + K_2^{matP}) \{q'[r']\}$
$T_{0.71} = [q'[r']]$
$T_{0.8} = \mathbb{M} q' \{q'[r']\}$

D7:

$$\vdots \vdots \vdots (\Sigma); d : [q'[r']] \vdash \text{ret } d : \mathbb{M} 0 \{q'[r']\}$$

D6:

$$\vdots \vdots \vdots (\Sigma); e : [r'] \vdash \text{store } e : \mathbb{M} q' \{q'[r']\} \quad \text{D7}$$
$$\vdots \vdots \vdots (\Sigma); c : [r'] \vdash \text{bind } d = \text{store } c \text{ in ret } d : T_{0.8}$$
$$\vdots \vdots \vdots (\Sigma); c : [r'] \vdash E_7 : T_{0.8}$$

D5:

$$\vdots \vdots \vdots (\Sigma); c : [r'] \vdash \uparrow K_2^{matP} : \mathbb{M} K_1^{matP} \mathbb{I} \quad \text{D6}$$
$$\vdots \vdots \vdots (\Sigma); c : [r'] \vdash \text{bind } = \uparrow K_2^{matP} \text{ in } E_7 : T_{0.7}$$
$$\vdots \vdots \vdots (\Sigma); c : [r'] \vdash E_6 : T_{0.7}$$

D4:

$$\vdots \vdots \vdots (\Sigma); b : T_{0.61} \vdash b : T_{0.61} \quad \text{D5}$$
$$\vdots \vdots \vdots (\Sigma); b : T_{0.61} \vdash \text{release } c = b \text{ in } E_6 : T_{0.2}$$
$$\vdots \vdots \vdots (\Sigma); b : T_{0.61} \vdash E_5 : T_{0.2}$$

D3:

$$\vdots \vdots \vdots (\Sigma); (\Gamma), x_1 : \{r_1\}, x_2 : \{r_2\}, a : T_{0.51} \vdash \epsilon_t \ a : T_{0.6} \quad \text{D4}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x_1 : \{r_1\}, x_2 : \{r_2\}, a : T_{0.51} \vdash \text{bind } b = \epsilon_t \ a \text{ in } E_5 : T_{0.2}$$

D2:

$$\vdots \vdots \vdots (\Sigma); \vdash \text{store }() : T_{0.5} \quad \text{D3}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x_1 : \{r_1\}, x_2 : \{r_2\} \vdash \text{bind } a = \text{store }() \text{ in } E_4 : T_{0.4}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x_1 : \{r_1\}, x_2 : \{r_2\} \vdash E_3 : T_{0.4}$$

D1:

$$\vdots \vdots \vdots (\Sigma); x : \{r\} \vdash x : \{r\} \quad \text{D2}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\} \vdash \text{let } (x_1, x_2) = x \text{ in } E_3 : T_{0.4}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\} \vdash E_2 : T_{0.4}$$

D0:

$$\vdots \vdots \vdots (\Sigma); \vdash \uparrow K_1^{matP} : \mathbb{M} K_1^{matP} \mathbb{I} \quad \text{D1}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\} \vdash \text{bind } = \uparrow K_1^{matP} \text{ in } E_2 : T_{0.3}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\} \vdash E_1 : T_{0.3}$$

Main derivation:

$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\}, u : T_{0.1} \vdash u : T_{0.1} \quad \text{D0}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\}, u : T_{0.1} \vdash \text{release } = u \text{ in } E_1 : T_{0.2}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\}, u : T_{0.1} \vdash E_0 : T_{0.2}$$
$$\vdots \vdots \vdots (\Sigma); (\Gamma), x : \{r\} \vdash \lambda u. E_0 : T_0$$

15. Augment:

$$\Sigma ; \Gamma \vdash q' \ q ; e : \tau \rightsquigarrow e_a$$
$$\Sigma ; \Gamma, x : \tau' \vdash q' \ q ; e : \tau \rightsquigarrow e_a$$

Augment
Proof. Proof by induction on the $\tau <: \tau'$ relation.

1. Base:

   $\vdash b <: b$

2. Pair:

   $\tau_1 <: \tau'_1 \qquad \tau_2 <: \tau'_2$

   $\vdash (\tau_1, \tau_2) <: (\tau'_1, \tau'_2)$

3. List:

   $\tau_1 <: \tau_2 \qquad \overline{\tau'} \geq \overline{\tau}$

   $\vdash L\overline{\tau_1} <: L\overline{\tau_2}$

Lemma 32 (Subtyping preservation). $\forall \tau, \tau'$. $\tau <: \tau' \implies (\tau) <: (\tau')$

A.5.2 Cross-language model: RAMLU to $\lambda$-amor

Definition 33 (Logical relation for RAMLU to $\lambda$-amor).

\[
\begin{align*}
[\text{unit}]_{\lambda} & \triangleq \{(T, s_v, t_v) \mid s_v \in [\text{unit}] \land t_v \in [1] \land s_v = t_v\} \\
[b]_{\lambda} & \triangleq \{(T, s_v, t_v) \mid s_v \in [b] \land t_v \in [b] \land s_v = t_v\} \\
[(\tau_1, \tau_2)]_{\lambda} & \triangleq \{(T, \ell, \langle \ell_1, \ell_2 \rangle) \mid H(\ell) = (s_v_1, s_v_2) \land (T, s_v_1, s_v_2) \in [\tau_1]_{\lambda} \land (T, s_v_2, t_v_2) \in [\tau_2]_{\lambda}\} \\
[L\overline{\tau}]_{\lambda} & \triangleq \{(T, \ell, s_\ell) \mid (T, \ell, s_\ell) \in [L \tau]\}
\end{align*}
\]

where $[L \tau]_{\lambda} \triangleq \{(T, NLL, n)\} \cup \{(T, \ell, s_\ell) \mid H(\ell) = (s_v_1, s_v_2) \land (T, s_v_1, s_v_2) \in [\tau]_{\lambda} \land (T, s_v_2, t_v_2) \in [\tau]_{\lambda}\}$

Definition 34 (Interpretation of typing context).

\[
[\Gamma]_{\lambda} = \{(T, V, \delta_0) \mid \forall x : \tau \in \text{dom}(\Gamma), (T, V(x), \delta_0(x)) \in [\tau]_{\lambda}\}
\]
Proof. We need to prove that \((T, f(x) = e_s, \text{fix} f.\lambda u.\lambda x.e_t) \in [\tau_1 \rightarrow \tau_2]^H \) \(\forall (T', f(x) = e_s, \text{fix} f.\lambda u.\lambda x.e_t) \in [\tau_1 \rightarrow \tau_2]^H \)

**Lemma 36** (Monotonicity for values). \(\forall v^v, t^v, T, \tau, H.\)

\((T, s^v, t^v) \in [\tau]^H \implies \forall T' \leq T. (T', s^v, t^v) \in [\tau]^H \)

**Proof.** Given: \((T, s^v, t^v) \in [\tau]^H\)

To prove: \(\forall T' \leq T. (T', s^v, t^v) \in [\tau]^H\)

This means given some \(T' \leq T\) it suffices to prove that 
\((T', s^v, t^v) \in [\tau]^H\)

By induction on \(\tau\)

1. \(\tau = \text{unit}\):
   - In this case we are given that \((T, s^v, t^v) \in [\text{unit}]^H\)
   - and we need to prove \((T', s^v, t^v) \in [\text{unit}]^H\)
   - We get the desired trivially from Definition \(\ref{def:unit}\)

2. \(\tau = b:\)
   - In this case we are given that \((T, s^v, t^v) \in [b]^H\)
   - and we need to prove \((T', s^v, t^v) \in [b]^H\)
   - We get the desired trivially from Definition \(\ref{def:unit}\)

3. \(\tau = L\vec{p}\tau':\)
   - In this case we are given that \((T, s^v, t^v) \in [L\vec{p}\tau']^H\)
   - Here let \(s^v = \ell_s\) and \(t^v = \langle \ell_t, \ell_{\ell_t} :: l_t \rangle\)
   - and we have \((T, \ell_s, \ell_{t^v} :: l_t) \in [L\tau']^H\) \((\text{MV-L1})\)
   - And we need to prove \((T', \ell_s, \ell_{t^v} :: l_t) \in [L\vec{p}\tau']^H\)
   - Therefore it suffices to prove that \((T', \ell_s, \ell_{t^v} :: l_t) \in [L\tau']^H\)

   We induct on \((T, \ell_s, \ell_{t^v} :: l_t) \in [L\tau']^H\)
   - \(\bullet (T, \text{NULL}, \text{nil}) \in [L\tau']^H:\)
     - In this case we need to prove that \((T', \text{NULL}, \text{nil}) \in [L\tau']^H\)
     - We get this directly from Definition \(\ref{def:unit}\)
   - \(\bullet (T, \ell_s, \ell_{t^v} :: l_t) \in [L\tau']^H:\)
     - Since from (MV-L1) we are given that \((T, \ell_s, \ell_{t^v} :: l_t) \in [L\tau']^H\)
     - therefore from Definition \(\ref{def:unit}\) we have
     \[
     H(\ell_{\ell_t}) = (\ell_{\ell_s}, \ell_{t_s}) \land \langle T, s^v, t^v \rangle \in [\tau']^H \land \langle T, \ell_{\ell_t}, l_t \rangle \in [L \tau']^H
     \] \((\text{MV-L2})\)
     - In this case we need to prove that \((T', \ell_s, \ell_{t^v} :: l_t) \in [L\tau']^H\)
     - From Definition \(\ref{def:unit}\) it further it suffices to prove that
       - \(- H(\ell_{\ell_s}) = (\ell_{\ell_v}, \ell_{t_{\ell_t}}):\)
         - Directly from (MV-L2)
       - \(- (T', s^v, t^v) \in [\tau']^H:\)
         - From (MV-L2) and outer induction
       - \(- (T', \ell_{\ell_t}, l_t) \in [L \tau']^H:\)
         - From (MV-L2) and inner induction

4. \(\tau = (\tau_1, \tau_2):\)
   - In this case we are given that \((T, \ell, \langle v_1^v, v_2^v \rangle) \in [(\tau_1, \tau_2)]^H\)
     - This means from Definition \(\ref{def:pair}\) we have
     \[
     H(\ell) = (\ell_{v_1}, \ell_{v_2}) \land (T, s^v, t^v) \in [\tau_1]_V \land (T, s^v, t^v) \in [\tau_2]_V
     \] \((\text{MV-P0})\)
     - and we need to prove \((T', \ell, \langle v_1^v, v_2^v \rangle) \in [(\tau_1, \tau_2)]^H\)
     - Similarly from Definition \(\ref{def:pair}\) it suffices to prove that
     \[
     H(\ell) = (\ell_{v_1}, \ell_{v_2}) \land (T', s^v, t^v) \in [\tau_1]_V \land (T', s^v, t^v) \in [\tau_2]_V
     \] \((\text{MV-P0})\)
     - We get this directly from (MV-P0), IH1 and IH2

**Lemma 37** (Monotonicity for functions). \(\forall v^v, t^v, T, \tau, H.\)

\((T, f(x) = e_s, \text{fix} f.\lambda u.\lambda x.e_t) \in [\tau_1 \rightarrow \tau_2]^H \implies \forall T' \leq T. (T', f(x) = e_s, \text{fix} f.\lambda u.\lambda x.e_t) \in [\tau_1 \rightarrow \tau_2]^H\)

**Proof.** We need to prove that \((T', f(x) = e_s, \text{fix} f.\lambda u.\lambda x.e_t) \in [\tau_1 \rightarrow \tau_2]^H\)

This means from Definition \(\ref{def:pair}\) it suffices to prove that
∀ v′, t′, T′′ < T′ . (T′′, s, v′, t′) ∈ [τ] V \implies (T′′, e_s, e_t[()/u][v′/x][λx.λu.e/f]) ∈ [τ] E.

This means given some *v′, t′, T′′ < T′ it suffices to prove that
(T′′, e_s, e_t[()/u][v′/x][λx.λu.e/f]) ∈ [τ] E.

(MF0)

Since we are given that (T, f(x) = e_s, fixf.λx.v[x]) ∈ [τ] q/q τ_2 therefore from Definition 33 we have
∀ v′_1, t′_1, T′_1 < T . (T′_1, s, v′_1, t′_1) ∈ [τ] V \implies (T′_1, e_s, e_t[()/u][v′_1/x][λx.λu.e/f]) ∈ [τ] E.

Instantiating with the given *v′, t′, T′ we get the desired

\[ \square \]

Lemma 38 (Monotonicity for expressions). ∀ e_s, e_t, T, τ, H.

(T, e_s, e_t) ∈ [τ] H \implies ∀ T′ ≤ T . (T′, e_s, e_t) ∈ [τ] H.

Proof. To prove: (T′, e_s, e_t) ∈ [τ] H.

This means from Definition 33 it suffices to prove that
∀ H′, v, p, p′, t < T′ . V, H ⊢ p v s, H′ \implies ∃ v_s, t′ v, J, e_c u v_1 p′ s, H′ \land p − p′ ≤ J

(ME0)

Since we are given that (T, e_s, e_t) ∈ [τ] H therefore again from Definition 33 we know that
∀ H′, *v, p, p′, t < T . V, H ⊢ p v s, H′ \implies ∃ v_s, t′ v, J, e_c u v_1 p′ s, H′ \land p − p′ ≤ J.

Instantiating with the given v, t′, p′ and using Lemma 36 we get the desired

\[ \square \]

Lemma 39 (Monotonicity for Γ). ∀ v, t, v, T, τ, H.

(T, V, δ_1) ∈ [Γ] V \implies ∀ T′ ≤ T . (T′, V, δ_1) ∈ [Γ] V.

Proof. To prove: (T′, V, δ_1) ∈ [Γ] V.

From Definition 35 it suffices to prove that
∀ x : τ ∈ dom(Γ). (T′, V(x), δ_1(x)) ∈ [τ] V.

This means given some x : τ ∈ dom(Γ) it suffices to prove that
(T′, V(x), δ_1(x)) ∈ [τ] V.

Since we are given that (T, V, δ_1) ∈ [Γ] V therefore again from Definition 35 we have
∀ x : τ ∈ dom(Γ). (T, V(x), δ_1(x)) ∈ [τ] V.

Instantiating it with the given x and using Lemma 36 we get the desired

\[ \square \]

Lemma 40 (Monotonicity for Σ). ∀ v, t, V, T, τ, H.

(T, δ_s f, δ_t f) ∈ [Σ] V \implies ∀ T′ ≤ T . (T′, δ_s f, δ_t f) ∈ [Σ] V.

Proof. To prove: (T′, δ_s f, δ_t f) ∈ [Σ] V.

From Definition 35 it suffices to prove that
∀ f : (τ_1 → τ_2) ∈ dom(Σ). (T′, δ_s f(f) d_t f(f) d_t f(f)) ∈ [(τ_1 → τ_2)] V.

This means given some f : (τ_1 → τ_2) ∈ dom(Σ) it suffices to prove that
(T′, δ_s f(f) d_t f(f) d_t f(f)) ∈ [(τ_1 → τ_2)] V.

Since we are given that (T, δ_s f, δ_t f) ∈ [Σ] V therefore from Definition 35 we have
∀ f : (τ_1 → τ_2) ∈ dom(Σ). (T, δ_s f(f) d_t f(f) d_t f(f)) ∈ [(τ_1 → τ_2)] V.

Instantiating it with the given f and using Lemma 36 we get the desired

\[ \square \]

Theorem 41 (Fundamental theorem). ∀ Σ, τ, q, q′, e_s, e_t, I, V, H, δ_1, δ_s f, δ_t f, T.

Σ : τ → e_t \land
(T, V, δ_1) ∈ [Γ] V \land (T, δ_s f, δ_t f) ∈ [Σ] V

\implies (T, e_s[()/u][v′/x][λx.λu.e/f]) ∈ [τ] E.

Proof. Proof by induction on Σ; Γ ⊢ q q′ e_s : τ → e_t
1. unit:

$$\Sigma_i: \vdash q^+ \uparrow K^{\text{unit}} \quad \text{unit}$$

where

$$E_i = \lambda u. \text{release} - = u \text{ in } \text{bind} - = \uparrow K^{\text{unit}} \quad \text{in } a = \text{store}(\text{in } \text{ret}(a))$$

$$E_i' = \text{release} - = u \text{ in } \text{bind} - = \uparrow K^{\text{unit}} \quad \text{in } a = \text{store}(\text{in } \text{ret}(a))$$

To prove: $$(T, x\delta_f, E_i (\delta t\delta_f) [\tau]_{\Sigma_i}^V H)$$

This means from Definition $[33]$ we are given some $^{*v, H', \lambda v, r, r', t}$ s.t. $V, H \vdash r' (\lambda v) q \vdash (\lambda v) H$. From (E:Unit) we know that $t = 1$

Therefore it suffices to prove that

(a) $\exists q v_t, t v_f, J, E_i (\lambda u. \text{release}) (\lambda v) q v_t, t v_f \land (T - 1, (\lambda v) t v_t) \in [v]_{\Sigma_i}^V$

We choose $^{+v_t, t v_f, J}$ as $E_i', (\lambda v) K^{\text{unit}}$ respectively

Since from E-app we know that $E_i \downarrow E_i'$, also since $E_i' \downarrow \uparrow K^{\text{unit}} (\lambda v)$ (from E-release, E-bind, E-store, E-return)

Therefore we get the desired from Definition $[34]$.

(b) $r - r' \leq J$:

From (E:Unit) we know that $\exists p.r = q + K^{\text{unit}}, r' = p$ and since we know that $J = K^{\text{unit}}$, therefore we are done

2. base:

$$\Sigma_i: \vdash q^+ \uparrow K^{\text{base}} c : b \Rightarrow E_i \text{ unit}$$

where

$$E_i = \lambda u. \text{release} - = u \text{ in } \text{bind} - = \uparrow K^{\text{base}} \quad \text{in } a = \text{store}(c) \text{ in } \text{ret}(a)$$

$$E_i' = \text{release} - = u \text{ in } \text{bind} - = \uparrow K^{\text{base}} \quad \text{in } a = \text{store}(c) \text{ in } \text{ret}(a)$$

To prove: $$(T, x\delta_f, E_i (\delta t\delta_f) [\tau]_{\Sigma_i}^V H)$$

This means from Definition $[33]$ we are given some $^{*v, H', \lambda v, r, r', t}$ s.t. $V, H \vdash r' c \vdash (\lambda v) H$. From (E:base) we know that $t = 1$

Therefore it suffices to prove that

(a) $\exists q v_t, t v_f, J, E_i (\lambda u. \text{release}) (\lambda v) q v_t, t v_f \land (T - 1, (\lambda v) t v_f) \in [v]_{\Sigma_i}^V$

We choose $^{+v_t, t v_f, J}$ as $E_i', (\lambda v) K^{\text{base}}$ respectively

Since from E-app we know that $E_i \downarrow E_i'$, also since $E_i' \downarrow \uparrow K^{\text{base}} c (\lambda v)$ (from E-release, E-bind, E-store, E-return)

Therefore we get the desired from Definition $[34]$.

(b) $r - r' \leq J$:

From (E:base) we know that $\exists p.r = q + K^{\text{base}}, r' = p$ and since we know that $J = K^{\text{base}}$, therefore we are done

3. var:

$$\Sigma_i: x: \tau \vdash q^+ \uparrow K^{\text{var}} x: \tau \Rightarrow E_i \text{ var}$$

where

$$E_i = \lambda u. \text{release} - = u \text{ in } \text{bind} - = \uparrow K^{\text{var}} \quad \text{in } a = \text{store}(x) \text{ in } \text{ret}(a)$$

$$E_i' = \text{release} - = (\lambda v) x \text{ in } \text{bind} - = \uparrow K^{\text{var}} \quad \text{in } a = \text{store}(x) \text{ in } \text{ret}(a)$$

To prove: $$(T, x\delta_f, E_i (\delta t\delta_f) [\tau]_{\Sigma_i}^V H)$$

This means from Definition $[33]$ we are given some $^{*v, H', \lambda v, r, r', t}$ s.t. $V, H \vdash r' x \vdash (\lambda v) V(x), H$. From (E:Var) we know that $t = 1$

Therefore it suffices to prove that

(a) $\exists q v_t, t v_f, J, E_i (\lambda u. \text{release}) (\lambda v) q v_t, t v_f \land (T - 1, V(x), t v_f) \in [v]_{\Sigma_i}^V$

We choose $^{+v_t, t v_f, J}$ as $E_i', (\lambda v) \delta t(x)$ respectively

Since from E-app we know that $E_i \downarrow E_i'$, also since $E_i' \downarrow \uparrow K^{\text{var}} (\lambda v) \delta t(x)$ (from E-release, E-bind, E-store, E-return)

Therefore we get the desired from Definition $[34]$ and Lemma $[40]$.

(b) $r - r' \leq J$:

From (E:VAR) we know that $\exists p.r = q + K^{\text{var}}, r' = p$ and $J = K^{\text{var}}$, so we are done

4. app:
6. cons:

\[
\tau_1 \eta E_q \eta' \tau_2 \in \Sigma(f) \\
\Sigma; x : \tau_1 \eta q \eta E_{q + K_{app}} f x \cdot \tau_2 \rightsquigarrow E_t
\]

where

\[E_t = \lambda u. E_0\]
\[E_0 = \text{release } \eta = \text{u in bind } \eta = \eta K_{app}\] in bind \(P = \text{store()}\) in \(E_1\)
\[E_1 = \text{bind } f_1 = f(P x)\] in release \(f_2 = f_1\) in bind \(\eta = \eta K_{app}\) in bind \(f_3 = \text{store } f_2\) in ret \(f_3\)

To prove: \((T, f, x, E_t) (\delta, \delta_f) \in [\tau_2]_V^H\)

This means from Definition 33 we are given some
\(\ast v, H', \ast v, r, r', t < T\) s.t. \(V, H \vdash t f x \delta_s f \vdash \ast v, H'\)

and it suffices to prove that

\[\exists t \vdash t v, J.E_t (\delta, \delta_f) \in [\tau_2]_{V^x}^H\] (F-A0)

Since we are given that \((T, \delta_s f, \delta_f) \in [\Sigma]^H\) then from Definition 35 we know that

\[(T, \delta_s f (f) \delta_f (f) \delta_f) \in [\tau_1]_{V^x}^H\] (F-A1)

Since we are given that \((T, V, \delta_t) \in [\Gamma]^H\) then we have

\[(T, T - 1, V(x), \delta_t (x)) \in [\tau_1]_{V^x}^H\] (F-A2)

This means from Lemma 36 we also have \((T - 1, V(x), \delta_t (x)) \in [\tau_1]_{V^x}^H\) (F-A3)

5. nil:

\[
\Sigma; \emptyset \eta q \eta K_{nil} \eta \text{nil} : L^{\overline{\tau}} \rightsquigarrow E_t
\]

where

\[E_t = \lambda u. \text{release } \eta = \text{u in bind } \eta = \eta K_{nil}\] in bind \(a = \text{store()}\) in bind \(b = \text{store}((u, \text{nil}))\) in ret \(b\)

To prove: \((T, nil, E_t) (\delta, \delta_f) \in [L^{\overline{\tau}}]_{V^x}^H\)

This means from Definition 33 we are given some
\(\ast v, H', \ast v, t < T\) s.t. \(\emptyset \eta l \eta' \text{nil} \eta^x \text{v, } H'\)

From (E-NIL) we know that \(\ast v = \text{NULL}, H' = H\) and \(t = 1\) and it suffices to prove that

\(a) \exists t \vdash t v, J.e \vdash t v, \eta \eta J v f \wedge (T - 1, \text{nil}, \eta v J) \in [L^{\overline{\tau}}]\)

From E-bind, E-release, E-return we know that \(\eta v = \langle ()\rangle\) therefore from Definition 33 we get the desired

\(b) p - p' \leq J;\) Here \(p = q + K_{nil}, p' = q + J = K_{nil}\), so we are done

6. cons:

\[
\gamma \beta = (p_1, \ldots, p_k) \\
\Sigma; x : \tau, x_1 : L^{\overline{\tau}} \eta q \eta K_{cons} (x_h, x_1) : L^{\overline{\tau}} \rightsquigarrow E_t
\]

where
To prove: \((T, \text{cons}(x_h, x_i), E_1 () \delta t_f) \in [L_\tau V, H]_E^*
\)

This means from Definition 33 we are given some \(v, H', v, p, p', t < T \text{ s.t. } \emptyset \vdash p', \text{cons}(x_h, x_i)\delta f \psi_t v, H'\)

and it suffices to prove that

\[(a) \exists v, H', v, p, p', t < T \text{ s.t. } \emptyset \vdash p', \text{cons}(x_h, x_i)\delta f \psi_t v, H'\]

From (E-app) of \(\lambda\)-amor we know that \(E_t () \psi E'_t\)

Also from E-release, E-bond, E-store we know that \(v_f = ((), \delta t(x_h) :: \delta t(x_i) \downarrow 2)\)

Therefore it suffices to prove that \((T - t, \ell, ((), \delta t(x_h) :: \delta t(x_i) \downarrow 2)) \in [L_\tau V, H']\)

From Definition 33 it further suffices to prove that

\((T - t, V(x_h), \delta t(x_i) :: \delta t(x_i) \downarrow 2) \in [L_\tau H']\)

Since from (E:CONS) rule of univariate RAML we know that \(H' = H[\ell \mapsto v]\) where \(v = (V(x_h), V(x_i))\)

Therefore it further suffices to prove that

\((T - t, V(x_h), \delta t(x_i)) \in [\tau]_V^H\) and \((T - t, V(x_i), \delta t(x_i) \downarrow 2) \in [L_\tau H']\)

Since we are given that \((T, V, \delta t_f) \in [\Sigma]_V^H\) therefore from Definition 33 and Lemma 36 it means we have

\((T - t, V(x_h), \delta t(x_i)) \in [\tau]_V^H\) (F-C1)

and

\((T - t, V(x_i), \delta t(x_i)) \in [L_\tau \Sigma]_V^H\)

This means we also have \((T - t, V(x_i), \delta t(x_i) \downarrow 2) \in [L_\tau H']\) (F-C2)

Since \(H' = H[\ell \mapsto v]\) where \(v = (V(x_h), V(x_i))\) therefore we also have

We get the desired from (F-C1), (F-C2) and Definition 33.

(b) \(p - p' \leq J\)

From (E:CONS) we know that \(p = q' + K_{cons}\) and \(p' = q'\) for some \(q'\). Also we know that \(J = K_{cons}\).

Therefore we are done.

7. match:

\[
\begin{align*}
\Sigma; \Gamma \vdash & q - K_{mat_N}^{max} e_1 : \tau' \rightsquigarrow e_{a1} \\
\{p_1, \ldots, p_k\}; \Sigma; \Gamma, h : \tau, t : L(q, \bar{p})_\tau \vdash & q + p_1 - K_{mat_C}^{max} e_2 : \tau' \rightsquigarrow e_{a2} \\
\Sigma; \Gamma; x : L_\tau \vdash & q, \text{match} x \text{ with } |nil \mapsto e_1 | h :: \tau \mapsto e_2 : \tau' \rightsquigarrow \lambda u. E_0
\end{align*}
\]

where

\(E_0 = \text{release} \quad \psi t u \quad \text{in } E_{0,1}\)

\(E_{0,1} = x : a. \text{let} \langle x_1, x_2 \rangle = a \quad \text{in } E_{1}\)

\(E_1 = \text{match } x_2 \text{ with } |nil \mapsto E_2 | h :: l \mapsto E_3\)

\(E_2 = \text{bind} \quad \psi t \quad \text{in } E_{2,1}\)

\(E_{2,1} = \text{bind } b :: \text{store}() \quad \text{in } E'_{2,1}\)

\(E_2 = \text{bind } c :: (e_{a1} b) \quad \text{in } E_{2,1}\)

\(E'_{2,1} = \text{release } d : c \quad \text{in } E'_{2,2}\)

\(E_{2,1} = \text{release } d : c \quad \text{in } E'_{2,2}\)

\(E_{2,2} = \text{bind} \quad \psi t \quad \text{in } E_{2,3}\)

\(E_{2,3} = \text{release } x_1 \quad \text{in } store \quad \text{in } E_{2,3}\)

\(E_1 = \text{bind} \quad \psi t \quad \text{in } E_{3,1}\)

\(E_{1,1} = \text{release } x_1 \quad \text{in } E_{3,2}\)

\(E_{1,2} = \text{bind } b :: \text{store}() \quad \text{in } E_{3,3}\)

\(E_{1,3} = \text{bind } t : \text{ret}() \quad \text{in } E_{3,4}\)

\(E_{1,4} = \text{bind } d :: \text{store}() \quad \text{in } E_{3,5}\)

\(E_{1,5} = \text{bind } f :: e_{a2} d \quad \text{in } E_{3,6}\)

\(E_{3,6} = \text{release } g = f \quad \text{in } E_{3,7}\)

\(E_{1,7} = \text{bind} \quad \psi t \quad \text{in } store \quad \text{in } E_{3,7}\)

To prove: \((T, \text{match } x \text{ with } |nil \mapsto e_1 | h :: \tau \mapsto e_2, \lambda u. E_0 () \delta t_f) \in [\tau']_V^H\)

This means from Definition 33 we are given some \(v, H', \psi t v, p, p', t < T \text{ s.t. } V, H' \psi t v \quad \text{match } x \text{ with } |nil \mapsto e_1 | h :: \tau \mapsto e_2 \delta t_f \psi_t v, H'\)

2 cases arise:
(a) $V(x) = \text{NULL}$: 
Since $(T, V, \delta_t) \in \Gamma^{V,H}_\psi$ therefore from Definition 34 and Definition 33 we have 
$$\delta_t(x) = \llll{()}, \text{nil}$$ 

**HH**: $(T - 1, e_1 \delta_x, e_{a_1} (\delta_1 \delta_f)) \in [\tau']^{V,H}_\psi$

This means from Definition 33 we have

$v H_2, t v_1, t v_2, t_1, t V, H \vdash_p \vdash_p t v_1, \delta_v, t v_2, H_1' \implies \exists v_{t_1}, t v_1, J_1, e_{a_1}, \downarrow t v_1, J_1, t v_1, \delta v_1, t v_f_1 \land (T - t_1, s v_1, s v_1) \downarrow [\tau']^{H_1'}_\psi \land p_1 - p'_1 \leq J_1$ (F-RUA-M0)

Since we are given that $V, H \vdash p (\text{match x with } \text{nil} \mapsto e_1 [h : t \mapsto e_2])$ therefore from (E:MatchN) we know that $V, H \vdash p - K_1^{\text{matN}} \vdash e_1 [\downarrow v, \delta v, t v_1] \downarrow s v, H'$ therefore instantiating (F-RUA-M0) with $H', s v, p - K_1^{\text{matN}}, p' + K_2^{\text{matN}}$ we get

$\exists v_{t_1}, t v_1, J_1, e_{a_1}, \downarrow t v_1, J_1, t v_1, \delta v_1, t v_f_1 \land (T - t, s v, t v_f_1) \in [\tau']^{H_1'}_\psi \land p - K_1^{\text{matN}} - p' - p_2^{\text{matN}} \leq J_1$ (F-RUA-M1)

It suﬃces to prove that 

$\exists v_{t_1}, t v_f_1, J, \lambda u. E_0 () \downarrow t v_1, J, J_2, e_{a_2} \downarrow t v_2, J, t v_2, \delta v_2 \land (T - t_1, s v_2, s v_2) \in [\tau']^{H_2'}_\psi \land p_2 - p' \leq J_2$ (F-RUA-M0.0)

Since we are given that $V, H \vdash p (\text{match x with } \text{nil} \mapsto e_1 [h : t \mapsto e_2]) \delta_s \downarrow s v, H'$ therefore from (E:MatchC) we know that $V, H \vdash p - K_1^{\text{matC}}, p' + K_2^{\text{matC}}, \downarrow \delta_s \downarrow s v, H'$ therefore instantiating (F-RUA-M0.0) with $H', s v, p - K_1^{\text{matC}}, p' + K_2^{\text{matC}}, t - 1$ we get

$\exists v_{t_2}, t v_f_2, J, e_{a_2} \downarrow t v_2, J, t v_2, \delta v_2 \land (T - t, s v_2, t v_f_2) \in [\tau']^{H_2'}_\psi \land p_2 - p' \leq J_2$ (F-RUA-M2)

It suﬃces to prove that 

$\exists v_{t_1}, t v_f_1, J, \lambda u. E_0 () \downarrow t v_1, J, J_2, e_{a_2} \downarrow t v_2, J, t v_2, \delta v_2 \land (T - t, s v, t v_f) \in [\tau']^{H_2'}_\psi \land p - p' \leq J$

We choose $t v_1$ as $t v_{t_1}, t v_f$ as $t v_f_1$ and as $J_2 + K_1^{\text{matC}} + K_2^{\text{matC}}$ and we get the desired from E-bind, E-release, E-store and (F-RUA-M2)

(b) $V(x) = \xi_1 :

Since $(T, V, \delta_t) \in \Gamma^{V,H}_\psi$ therefore from Definition 34 and Definition 33 we have 

$$\delta_t(x) = \llll{()}, \text{nil}$$

Let $V' = V \cup \{x \mapsto V(z)\} \cup \{y \mapsto V(z)\}$

$$\delta'_t = \delta_t \cup \{x \mapsto \delta_t(z)\} \cup \{y \mapsto \delta_t(z)\}$$

8. Share:

$$\Sigma; \Gamma, x : t_1, y : t_2 - q, e : t' \leadsto e_0 \quad \tau = \tau_1 \uparrow \tau_2 \quad \tau = \tau_1 = \tau_2 = 1$$

Share-unit

$E_0 = \lambda u. E_1$
$E_1 = \text{bind a = coerces}_{1, 1} z$ in let$\llll{z, y} = a$ in $e_a u$

$\text{coerces}_{1, 1} \triangleq \text{null (let } \llll{()} \text{ in } u \text{ and } a\text{ in } u)$

To prove: $(T, e[z/x, z/y], E_0 () \delta_1 \delta_f) \in [\tau']^{V,H}_\psi$

This means from Definition 33 we are given some $s v, H', t v, p, p', t$ s.t $V, H \vdash p (e[z/x, z/y] \delta_1 \downarrow s v, H')$

And we need to prove 

$\exists v_{t_1}, t v_f, J, E_0 () \downarrow t v_1, J, t v_f \land (T - t, s v, t v_f) \in [\tau']^{H'}_\psi \land p - p' \leq J$

Let

$V' = V \cup \{x \mapsto V(z)\} \cup \{y \mapsto V(z)\}$

$$\delta'_t = \delta_t \cup \{x \mapsto \delta_t(z)\} \cup \{y \mapsto \delta_t(z)\}$$
Since we are given that \( (T, V, \delta_t) \in [\Gamma, z : 1]_{V,H}^{V,H} \) therefore from Definition 34 we also have
\( (T, V', \delta'_t) \in [\Gamma, x : 1]_{V,H}^{V,H} \)

**IH**

\( (T, e, e_a (\delta_x) \delta'_t f) \in [\tau']_{V,H}^{V,H} \)

This means from Definition 33 we have
\[
\forall H_1, e, v, p, t. \quad H_1 \vdash_{\tau_1} e \quad \Rightarrow \quad \exists e_a, \tau = t_1, \tau = \tau_1 \Rightarrow (T - \tau_1, e, \tau) \in [\tau']_{V,H}^{V,H}
\]

Instantiating it with the given \( H', v, p, p', t \) we get the desired

\[
\Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash_{\tau'} e : \tau' \quad \Rightarrow \quad \Sigma; \Gamma, z : \tau \vdash_{\tau'} e[z/x, y/z] : \tau' \quad \Rightarrow \quad E_0
\]

Share-base

\( E_0 = \lambda u. E_1 \)

\( E_1 = \text{bind} a = \text{coerce}_{b,b} z \text{ in let} \langle x, y \rangle = a \text{ in } e_u u \)

\( \text{coerce}_{b,b} \triangleq \lambda u. \text{let} u' = u \text{ in ret} \langle u', u' \rangle \)

Similar reasoning as in the unit case above

\[
\tau = \tau_1 \Downarrow \tau_2 \\
\tau = \lambda \bar{p}. \tau'' \\
\tau = \lambda \bar{p}. \tau'' \\
\tau = \tau_1 \Downarrow \tau_2 \\
\bar{p} = \bar{p}_1 \Downarrow \bar{p}_2
\]

Share-list

\( E_0 = \lambda u. E_1 \)

\( E_1 = \text{bind} a = \text{coerce}_{\tau, \tau_1, \tau_2} z \text{ in let} \langle x, y \rangle = a \text{ in } e_u u \)

\( \text{coerce}_{\tau, \tau_1, \tau_2} \triangleq \lambda x, y. \text{let} g' = g \text{ in } e ; \text{let} \langle p, l \rangle = x \text{ in } E_0 \)

where

\( E_0 \triangleq \text{release} e = p \text{ in } E_1 \)

\( E_1 \triangleq \text{match} \ e \text{ with } [\bar{p} \vdash E_2.1 | h :: t \vdash E_3 \]

\( E_2.1 \triangleq \text{bind} z_1 = \text{store}(v) \text{ in } E_2.2 \)

\( E_2.2 \triangleq \text{bind} z_2 = \text{store}(v) \text{ in } E_2.3 \)

\( E_2.3 \triangleq \text{ret} \langle z_1, \langle z_2, \langle z_2, \langle z_2, \rangle \rangle \rangle \rangle \)

\( E_1 \triangleq \text{bind} H = g' \quad h \text{ in } E_{3.1} \)

\( E_{3.1} \triangleq \text{bind} a_0 = () \text{ in } E_{3.2} \)

\( E_{3.2} \triangleq \text{bind} T = f \quad g \text{ in } E_{4} \)

\( E_{4} \triangleq \text{let} \langle H_1, H_2 \rangle = H \text{ in } E_{5} \)

\( E_{5} \triangleq \text{let} \langle T_1, T_2 \rangle = T \text{ in } E_{6} \)

\( E_{6} \triangleq \text{let} \langle p_1, l_1 \rangle = \langle t_1, p \rangle \text{ in } E_{7.1} \)

\( E_{7.1} \triangleq \text{let} \langle p_2, l_2 \rangle = \langle t_2, p \rangle \text{ in } E_{7.2} \)

\( E_{7.2} \triangleq \text{release} e = p_1 \text{ in } E_{7.3} \)

\( E_{7.3} \triangleq \text{release} e = p_2 \text{ in } E_{7.4} \)

\( E_{7.4} \triangleq \text{bind} a_0 = \text{store}(v) \text{ in } E_{7.5} \)

\( E_{7.5} \triangleq \text{bind} a_0 = \text{store}(v) \text{ in } E_{8} \)

\( E_{8} \triangleq \text{ret} \langle \langle \langle o_1, H_1 :: T_1 \rangle, \langle o_2, H_2 :: T_2 \rangle \rangle \rangle \)

To prove: \( (T, e[z/x, y/z], E_0 (\delta_x) \delta'_t f) \in [\tau']_{V,H}^{V,H} \)

This means from Definition 33 we are given some
\( * v, H', * v, p, p', t < T \text{ s.t. } V, \vdash_{\tau_1} e[z/x, y/z] \delta_{sf} \downarrow t_1 * v, H' \)

And we need to prove
\( \exists e_a, \tau = t_1, \tau = \tau_1 \Rightarrow (T - \tau_1, e, \tau) \in [\tau']_{V,H}^{V,H} \quad \Rightarrow \quad p - p' \leq J \)

Let
\( V' = V \cup \{ x \mapsto V(z) \} \cup \{ y \mapsto V(z) \} \)

\( \delta'_t = \delta_t \cup \{ x \mapsto \delta_t(z) \} \cup \{ y \mapsto \delta_t(z) \} \)

Since we are given that \( (T, V, \delta_t) \in [\Gamma, z : \tau]_{V,H}^{V,H} \) therefore from Definition 34 we also have
\( (T, V', \delta'_t) \in [\Gamma, x : \tau_1, y : \tau_2]_{V,H}^{V,H} \)

**IH**
\( (T, e, e_a (\delta t \delta f)) \in [\tau']_E^H \)

This means from Definition 43 we have
\( \forall H', s v_1, p_1', t_1, V', H \vdash p_1' e \downarrow_{t_1} s v_1, H_1' \implies \exists v_\text{f}, t v_\text{f}, J, e_a (\) \downarrow \downarrow \downarrow J \downarrow t v_\text{f} \land (T - t_1, s v_1, t v_\text{f}) \in [\tau']_V^H \land p_1 - p_1' \leq J \)

Instantiating it with the given \( H', s v, p, p', t \) we get the desired

\[ \tau = \tau_1 \uplus \tau_2 \]
\[ \frac{\Sigma; \Gamma, x : \tau_1, y : \tau_2 \vdash g : \tau' \leadsto e_a}{\Sigma; \Gamma, z : \tau \vdash g[z/x, z/y] : \tau' \leadsto E_0} \]

Share-pair

\[ E_0 = \lambda u. E_1 \]
\[ E_1 = \text{bind } a \equiv \text{coerce}(\tau_a, \tau_0, (\tau_0', \tau_0'), (\tau_0'', \tau_0'')) z \text{ in } \{x, y\} = a \text{ in } e_a u \]

\[ \text{coerce}(\tau_a, \tau_0, (\tau_0', \tau_0'), (\tau_0'', \tau_0'')) \equiv \lambda g_1. \lambda g_2. \lambda p. \text{let } \{p_1, p_2\} = p \text{ in } E_0 \]

where
\[ E_0 \triangleq \text{let } g'_1 = g_1 \text{ in } E_1 \]
\[ E_1 \triangleq \text{let } g'_2 = g_2 \text{ in } E_2 \]
\[ E_2 \triangleq \text{bind } P'_1 = g'_1 p_1 \text{ in } E_3 \]
\[ E_3 \triangleq \text{bind } P'_2 = g'_2 p_2 \text{ in } E_4 \]
\[ E_4 \triangleq \text{let } \{p_1, p_1'\}_B = P'_1 \text{ in } E_5 \]
\[ E_5 \triangleq \text{let } \{p_2, p_2'\}_B = P'_2 \text{ in } E_6 \]
\[ E_6 \triangleq \text{ret } \{p_1, p_1'\}_1, \{p_2, p_2'\}_2 \]

Same reasoning as in the list subcase above

9. Sub:

\[ \frac{\Sigma; \Gamma \vdash g : \tau \leadsto e_a \quad \tau <; \tau'}{\Sigma; \Gamma \vdash g : \tau' \leadsto e_a} \]

To prove: \( (T, e, e_a (\delta t \delta f)) \in [\tau']_E^H \)

\[ \text{IH: } (T, e, e_a (\delta t \delta f)) \in [\tau']_E^H \]

We get the desired from IH and Lemma 43

10. Relax:

\[ \frac{\Sigma; \Gamma \vdash e : \tau \leadsto e_a \quad q \geq p \quad q - p \geq q' - p'}{\Sigma; \Gamma \vdash e : \tau \leadsto E_t} \]

where
\[ E_t = \lambda o. E_0 \]
\[ E_0 = \text{release } - = o \text{ in } E_1 \]
\[ E_1 = \text{bind } a = \text{store}(\) \text{ in } E_2 \]
\[ E_2 = \text{bind } b = e_a a \text{ in } E_3 \]
\[ E_3 = \text{bind } e = b \text{ in } \text{store } c \]

To prove: \( (T, e, E_t (\delta t \delta f)) \in [\tau']_E^H \)

This means from Definition 43 we are given some
\( s v, H', s v, r, r', t <; T \) s.t. \( t_1, s v, H' \)

And it suffices to prove that
\[ \exists v_\text{f}, t v_\text{f}, J, E_t (\) \downarrow \downarrow \downarrow J \downarrow t v_\text{f} \land (T - t_1, s v_1, H') \in [\tau]_V \land r - \tau' \leq J \quad \text{(F-R0)} \]

\[ \text{IH: } (T, e, e_a (\delta t \delta f)) \in [\tau']_E^H \]

This means from Definition 43 we have
\( \forall s v_1, H'_1, r_1, t_1 <; T \cdot V, H \vdash r_1 e \downarrow_{t_1} s v_1, H_1' \implies \exists v_\text{f}', t v_\text{f}', J, e_a (\) \downarrow \downarrow \downarrow J \downarrow t v_\text{f}' \land (T - t_1, s v, t v_\text{f}') \in [\tau]_V \land r - \tau' \leq J' \quad \text{(F-R1)} \]

In order to prove (F-R0) we choose \( t v_\text{t}, t v_\text{f}, J' e_a (\) \downarrow \downarrow \downarrow J' \downarrow t v_\text{f}' \) and we get the desired from E-app, E-release, E-bind, E-store and (F-R1)
11. Super:

\[
\begin{align*}
\Sigma; \Gamma, x : \tau_1 \vdash^q_{q'} e : \tau & \iff e_a & \tau_1' & : \tau_1 < : \tau_1' & \text{Super} \\
\end{align*}
\]

Given: \((T, V, \delta_t) \in [\Gamma, x : \tau_1']^V H\)

To prove: \((T, e, e_a) (\delta_t \delta_{sf}) \in [\tau_1']^V H\)

This means from Definition 33 it suffices to prove that

\[
\forall H', s, v, p, p', t < T. V, H \vdash^p_{p'} e, s, t \vdash v, H' \implies \exists v_1, t v_f, J, e_a (\) \vdash t v_1 \vdash t v_f \land (T - t_1, s v, t v_f) \in [\tau_1']^V H' \land p - p' \leq J
\]

This means given some \(H', s, v, p, p', t < T. V, H \vdash^p_{p'} e, s, t \vdash v, H'\) it suffices to prove that

\[
\exists v_1, t v_f, J, e_a (\) \vdash t v_1 \vdash t v_f \land (T - t_1, s v, t v_f) \in [\tau_1']^V H' \land p - p' \leq J \quad \text{(F-Su0)}
\]

Since we are given that \((T, V, \delta_t) \in [\Gamma, x : \tau_1']^V H\) therefore from Definition 34 we know that \((T, V(x), \delta_t(x)) \in [\tau_1']^V H\)

Therefore from Lemma 42 we know that \((T, V(x), \delta_t(x)) \in [\tau_1']^V H\)

**IH:** \((T, e, e_a) (\delta_t \delta_{sf}) \in [\tau_1']^V H\)

This means from Definition 33 we have

\[
\forall H', s, v, p, p', t_1. V, H \vdash^p_{p'} e, s, t_1 \vdash v, H' \implies \exists v_1, t v_f, J, e_a (\) \vdash t v_1 \vdash t v_f \land (T - t_1, s v, t v_f) \in [\tau_1']^V H' \land p_1 - p'_1 \leq J
\]

Instantiating it with the given \(H', s, v, p, p', t\) we get the desired

12. Let:

\[
\begin{align*}
\Sigma; \Gamma_1 \vdash^q_{q'} e_1 : \tau_1 \iff e_{a1} & \Sigma; \Gamma_2, x : \tau_1 \vdash^p_{q - K^F_{i1} + q'} e_2 : \tau_1 \iff e_{a2} & \Sigma; \Gamma_1, \Gamma_2 \vdash^q_{q'} \text{let } x = e_1 \text{ in } e_2 : \tau \iff E_t
\end{align*}
\]

where

\[
\begin{align*}
E_t & = \lambda u. E_0 \\
E_0 & = \text{release} - = u \text{ in } E_1 \\
E_1 & = \text{bind } - = \gamma_{\kappa^{F_{i1}}} \text{ in } E_2 \\
E_2 & = \text{bind } a = \text{store}(\) \text{ in } E_3 \\
E_3 & = \text{bind } b = e_{a1} a \text{ in } E_4 \\
E_4 & = \text{bind } c = \text{store}(\) \text{ in } E_5 \\
E_5 & = \text{bind } d = e_{a2} c \text{ in } E_6 \\
E_6 & = \text{release } f = d \text{ in } E_7 \\
E_7 & = \text{bind } g = \text{store } f \text{ in ret } g
\end{align*}
\]

To prove: \((T, \text{let } x = e_1 \text{ in } e_2, E_t \) \(\delta_t \delta_{sf}) \in [\tau_1']^V H\)

This means from Definition 33 we are given some

\(s, v, H', s, v, r, r', t < T. s, t \vdash v, H'\) (let \(x = e_1 \text{ in } e_2\)) \(\delta_{sf} \vdash v, H'\)

it suffices to prove that

\[
\exists v_1, t v_f, J, e_a (\) \vdash t v_1 \vdash t v_f \land (T - t_1, s v, t v_f) \in [\tau_1']^V H' \land r - r' \leq J
\]

(F-L0)

Since we are given that \((T, V, \delta_t) \in [\Gamma_1, \Gamma_2]^H\) therefore we know that

\[
\exists v_1, v_2, \delta_1, \delta_2 \text{ s.t. } V = V_1, V_2, \delta_1 = \delta_2, \delta_2 \text{ and } (T, V_1, \delta_1) \in [\Gamma_1]^V H \text{ and } (T, V_2, \delta_2) \in [\Gamma_2]^V H
\]

**IH**

\((T, e_1, e_{a1}) (\delta_t \delta_{sf}) \in [\tau_1']^V H\)

This means from Definition 33 we have

\[
\forall H', s, v_1, p_1, t_1. V, H \vdash^p_{p_1} e_1, \delta_t (s v_1, H') \implies \exists v_1, \psi v_f, J, e_a (\) \vdash \psi v_1 \vdash \psi v_f \land (T - t_1, s v_1, t v_f) \in [\tau_1']^V H' \land p_1 - p'_1 \leq J
\]

(F-L1)

Since we know that \(V, H \vdash^p_{p_1} (\) \(x = e_1 \text{ in } e_2)\) \(\delta_{sf} \vdash v, H'\) therefore from (E:Let) we know that

\[
\exists H', s, v_1, t_1. s, t_1 \vdash v, H' \text{ therefore from (E:Let)} \text{ we know that } \exists H', s, v_1, t_1. s, t_1 \vdash v, H'
\]

86
Instantiating (F-L1) with $H'_1, s, v_1, r - K_{\text{let}}$, $r_1, t_1$ we get

$$\exists v_1, t v_f, J, e_1, e_2 (\gamma \downarrow t v_f \land (T - t_1, s, v_1, t v_f) \in [\tau]_{V,H}^{H'_1} \land r - K_{\text{let}} - r_1 \leq J_1 \quad (F-L1.1)$$

**H2**

$$(T - t_1, e_2, e_2 (\gamma \downarrow t v_f) \in [\tau]_{V,H}^{V_2 \cup \{x \mapsto \gamma v_f\}}, H'_1)$$

This means from Definition $33$ we have

$$\forall H'_2, v_2, p_2, v_2 \prec < T - t_1, V, H \vdash p_2 \downarrow v_2 \leftarrow v_2, H' \implies \exists v_2, t v_f, J, e_2 (\gamma \downarrow v_2 \downarrow J \downarrow v_2 \land (T - t_1 - t_2, s, v, t v_f) \in [\tau]_{V,H}^{H'_2} \land r - K_{\text{let}} - r_2 \leq J_2 \quad (F-L2)$$

Since we know that $V, H \vdash v_r (let x = e_1 in e_2) \delta_s f \downarrow ^t s v, H'$ therefore from (E:Let) we know that

$$\exists H'_2, v_2, t_2 < t - t_1 s.t. V, H \vdash H'_2, v_2, t_2 < T - t_1, s, v, t v_f \in [\tau]_{V,H}^{H'_2} \land r_1 - K_{\text{let}} \leq J_2$$

Instantiating (F-L2) with $H'_2, s, v, r_1 - K_{\text{let}}, r' + K_{\text{let}}, t_2$ we get

$$\exists v_2, t v_f, J, e_2 (\gamma \downarrow v_2 \downarrow ^t v_2 \land (T - t_1 - t_2, s, v, t v_f) \in [\tau]_{V,H}^{H'_2} \land r - K_{\text{let}} - (r' + K_{\text{let}}) \leq J_2 \quad (F-L2.1)$$

In order to prove (F-L0) we choose $t v_1$ as $t v_2, t v_f$ as $t v_f, J$ as $J + J_1 + K_{\text{let}} + K_{\text{let}} + K_{\text{let}}, t$ as $t_1 + t_2 + 1$ and we get the desired from (F-L1.1) and (F-L2.1) and Lemma $36$.

13. Pair:

$$\Sigma; x_1 : \tau_1, x_2 : \tau_2 \vdash ^q s, v \text{ pair } (x_1, x_2) : (\tau_1, \tau_2) \rightsquigarrow E_t$$

where

- $E_0 = \lambda u. E_0$
- $E_0 = \text{release } = u \text{ in } E_1$
- $E_1 = \text{bind } = \gamma ^{K_{\text{pair}}} \text{ in } E_2$
- $E_2 = \text{bind } a = \text{store}(x_1, x_2) \text{ in } \text{ret } a$

Given: $(T, V, \delta_1) \in [\tau_1 : \tau_2]_V^{H}$

To prove: $(T, (x_1, x_2), E_t (\gamma \delta_s f)) \in [(\tau_1, \tau_2)]_{V,H}^{H}$

This means from Definition $33$ it suffices to prove that

$$\forall H', s, v, r', t < T \ s.t. V, H \vdash v_r (x_1, x_2) \downarrow ^t s v, H' \implies \exists v_1, t v_f, J, E_t (\gamma \downarrow v_1 \downarrow ^t v_f \land (T - t, s, v, t v_f) \in [(\tau_1, \tau_2)]_{V,H}^{H'} \land r - r' \leq J \quad (F-P0)$$

This means we need to prove that

- $E_t (\gamma \downarrow v_1 \downarrow ^t v_f, J)$
- From E-app, E-release, E-bind, E-tick, E-store and E-return we know that $t v_1 = E_0, t v_f = (\delta_t (x_1), \delta_t (x_2))$ and $J = K_{\text{pair}}$
- $\forall t \vdash v_r (x_1, x_2) \downarrow ^t s v, H'$

This means given some $H', s, v, r', t < T \ s.t. V, H \vdash v_r (x_1, x_2) \downarrow ^t s v, H'$ it suffices to prove that

$$\exists v_1, t v_f, J, E_t (\gamma \downarrow v_1 \downarrow ^t v_f \land (T - t, s, v, t v_f) \in [(\tau_1, \tau_2)]_{V,H}^{H'} \land r - r' \leq J$$

This means we need to prove that $\exists v_1, t v_f, J$

- From (E:Pair) we know that $\exists p, r = p + K_{\text{pair}}$ and $r' = p$. Since we know that $J = K_{\text{pair}}$, therefore we are done.

14. MatP:

$$\tau = (\tau_1, \tau_2) \quad \Sigma, \Gamma, x_1 : \tau_1, x_2 : \tau_2 \vdash ^q s, v \text{ match } x \text{ with } (x_1, x_2) \rightsquigarrow e : \tau' \rightsquigarrow E_t \quad \text{MatP}$$

where

- $E_0 = \lambda u. E_0$
- $E_0 = \text{release } = u \text{ in } E_1$
- $E_1 = \text{bind } = \gamma ^{K_{\text{pair}}} \text{ in } E_2$
- $E_2 = \text{let } (x_1, x_2) = x \text{ in } E_3$
- $E_3 = \text{bind } a = \text{store}(\ell) \text{ in } E_4$
Given: \((T, V, \delta_t) \in [\Gamma, x : \tau]^H\)

To prove: \((T, (\text{match } x \text{ with } (x_1, x_2) \rightarrow e), E_t (\delta) \delta_{tjr}) \in [\tau]^V,H\)

This means from Definition 33 it suffices to prove that

\(\forall H', s, v, p, p', t < T . V, H \vdash_p (\text{match } x \text{ with } (x_1, x_2) \rightarrow e) \Downarrow t s, v, H' \implies \exists t s, v, J, E_t (\delta) \Downarrow t s, v, J t f v_1 \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v_1) \in [\tau']^H \land p - p' \leq J\)

This means given some \(H', s, v, p, p', t < T \text{ s.t. } V, H \vdash_p (\text{match } x \text{ with } (x_1, x_2) \rightarrow e) \Downarrow t s, v, H'\) it suffices to prove that

\(\exists t s, v, J, E_t (\delta) \Downarrow t s, v, J t f v_1 \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v_1) \in [\tau']^H \land p - p' \leq J\) (F-MP0)

Since we are given that \((T, V, \delta_t) \in [\Gamma, x : \tau]^H\) therefore from Definition 34 and since \(\tau = (\tau_1, \tau_2)\) therefore we know that \((T, V(x), \delta_t(x)) \in [(\tau_1, \tau_2)]^H\)

This means from Definition 33 that \(\exists \delta \text{ s.t. } H(\ell) = (s v_1, s v_2) \land (T, s v_1, t v_1) \in [\tau_1]_V \land (T, s v_2, t v_2) \in [\tau_2]_V\)

**IH:** \((T, e, e_t (\delta) \delta_t \cup \{ x_1 \mapsto t v_1 \} \cup \{ x_2 \mapsto t v_2 \} \delta_{tjr}) \in [\tau']^V_{U \cup \{s v_1 \cup \{x_2 \mapsto s v_2\}\}}, H\)

This means from Definition 33 we have

\(\forall H', s, v, p, p', t < T - 1 . V, H \vdash_p (\exists t v_1, t v_2, t v_3, t v_4) \Downarrow t s, v, H' \implies \exists t v_1, t v_2, t v_3, t v_4, J, e_t (\delta) \Downarrow t s, v, J t f v_1 \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v_1) \in [\tau']^H \land p - p' \leq J\)

Since we are given that \(V, H \vdash_p (\text{match } x \text{ with } (x_1, x_2) \rightarrow e) \Downarrow t s, v, H'\) therefore from (E:MatP) we know that

\(V \cup \{ x_1 \mapsto s v_1 \} \cup \{ x_2 \mapsto s v_2 \}, H \vdash_p (\exists t v_1, t v_2, t v_3, t v_4) \Downarrow t s, v, H'\)

Instantiating it with the given \(H', s, v, p - K_{\text{matP}}^1 + K_{\text{matP}}^2, t - 1\) we get

\(\exists t v_1, t v_2, t v_3, t v_4, J, e_t (\delta) \Downarrow t s, v, J t f v_1 \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v_1) \in [\tau']^H \land p - p' \leq J\) (F-MP1)

In order to prove (F-MP0) we choose \(t v_1 \text{ as } t v_1, t v_2 \text{ as } t v_3, J \text{ as } J_1 + K_{\text{matP}}^1 + K_{\text{matP}}^2 \text{ and } t_1 \text{ as } t - 1\) and it suffices to prove that

- \(E_t (\delta) \Downarrow t s, v, J t f v\): we get the desired from E-app, E-bind, E-release, E-store, E-tick, E-return and (F-MP1)
- \((T - t, s, v, J t f) \in [\tau']^H_{\forall}:
  - \text{From (F-MP1)}
- \(p - p' \leq J:\)
  - \text{We get this directly from (F-MP1)}

15. Augment:

Given: \((T, V \cup \{ x \mapsto s v_2 \}, \delta_t \cup \{ x \mapsto t v_2 \}) \in [\Gamma, x : \tau']^H\)

To prove: \((T, e, e_a (\delta) \delta_t \cup \{ x \mapsto t v_2 \} \delta_{tjr}) \in [\tau']^V_{U \cup \{s v_2\}}, H\)

This means from Definition 33 it suffices to prove that

\(\forall H', s, v, p, p', t < T . V \cup \{ x \mapsto s v_2 \}, H \vdash_p e_a (\delta) \Downarrow t s, v, H' \implies \exists t s, v, J, e_a (\delta) \Downarrow t s, v, J t f v \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v) \in [\tau']^H \land p - p' \leq J\)

This means given some \(H', s, v, p, p', t < T \text{ s.t. } V \cup \{ x \mapsto s v_2 \}, H \vdash_p e_a (\delta) \Downarrow t s, v, H'\) it suffices to prove that

\(\exists t s, v, J, e_a (\delta) \Downarrow t s, v, J t f v \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v) \in [\tau']^H \land p - p' \leq J\) (F-Ag0)

Since we are given that \((T, V \cup \{ x \mapsto s v_2 \}, \delta_t \cup \{ x \mapsto t v_2 \}) \in [\Gamma, x : \tau']^H\) therefore from Definition 34 we know that \((T, V, \delta_t) \in [\Gamma]^H\)

**IH:** \((T, e, e_a (\delta) \delta_{tjr}) \in [\tau]^V,H\)

This means from Definition 33 we have

\(\forall H', s, v, p, p', t < T . V, H \vdash_p e_a (\delta) \Downarrow t s, v, H' \implies \exists t s, v, J, e_a (\delta) \Downarrow t s, v, J t f v \Downarrow t s, v, J t f v \land (T - t, s, v, J t f v) \in [\tau']^H \land p - p' \leq J\) (F-Ag1)
Since we are given $V \cup \{x \mapsto ^x v_x\}, H \vdash_p e_s \downarrow^t v, H'$ and since $x \notin \text{free}(v)$ therefore we also have $V, H \vdash_p e_s \downarrow^t v, H''$

Instantiating (F-Ag1) with the given $H', ^s v, p, p', t$ we get

$\exists v_s, t v_f, J, e_a (\delta t_i \downarrow f v_i \downarrow f t v_f \land (T - t, ^s v, t v_f) \in [\tau]^H_f \land p_i - p'_i \leq J)$

Also since $x \notin \text{free}(c)$ therefore we get

$\exists v_s, t v_f, J, e_a (\delta t \cup \{x \mapsto ^x v_x\}) \delta t_i \downarrow f v_i \downarrow f t v_f \land (T - t, ^s v, t v_f) \in [\tau]^H_f \land p_i - p'_i \leq J$

\[\square\]

**Lemma 42** (Value subtyping lemma). $\forall \tau, \tau', H, ^s v, ^t v, T.$

$\tau <: \tau' \land (T, ^s v, ^t v) \in [\tau]^H_f \implies (T, ^s v, ^t v) \in [\tau']^H_f$

**Proof.** Proof by induction on the subtyping relation of Univariate RAML

1. Unit:

   $\underline{\text{unit} <: \text{unit}}$

   Given: $(T, ^s v, ^t v) \in [\text{unit}]^H_f$
   
   To prove: $(T, ^s v, ^t v) \in [\text{unit}]^H_f$

   Trivial

2. Base:

   $\underline{b <: b}$

   Given: $(T, ^s v, ^t v) \in [b]^H_f$
   
   To prove: $(T, ^s v, ^t v) \in [b]^H_f$

   Trivial

3. Pair:

   $\tau_1 <: \tau'_1 \quad \tau_2 <: \tau'_2 \quad (\tau_1, \tau_2) <: (\tau'_1, \tau'_2)$

   Given: $(T, ^s v, ^t v) \in [(\tau_1, \tau_2)]^H_f$
   
   To prove: $(T, ^s v, ^t v) \in [(\tau'_1, \tau'_2)]^H_f$

   From Definition 33 we know that $^s v = \ell$ s.t.

   $H(\ell) = (^s v_1, ^s v_2) \land (T, ^s v_1, ^t v_1) \in [\tau_1]_V \land (T, ^s v_2, ^t v_2) \in [\tau_2]_V$ \hspace{1cm} (S-P0)

   **IH1** $(T, ^s v_1, ^t v_1) \in [\tau'_1]_V$

   **IH2** $(T, ^s v_2, ^t v_2) \in [\tau'_2]_V$

   Again from Definition 33 it suffices to prove that

   $H(\ell) = (^s v_1, ^s v_2) \land (T, ^s v_1, ^t v_1) \in [\tau'_1]_V \land (T, ^s v_2, ^t v_2) \in [\tau'_2]_V$

   We get this directly from (S-P0), IH1 and IH2

4. List:

   $\tau_1 <: \tau_2 \quad \bar{p} \geq \bar{q} \quad \underline{L^p \tau_1 <: L^q \tau_2}$

   Given: $(T, ^s v, ^t v) \in [\bar{L}^p \tau_1]^H_f$
   
   To prove: $(T, ^s v, ^t v) \in [\bar{L}^q \tau_2]^H_f$

   From Definition 33 we know that $^s v = \ell$ and $^t v = \langle()\rangle$ s.t. $(T, l_s, l_t) \in [L \tau_1]_V$

   Similarly from Definition 33 it suffices to show that

   $(T, l_s, l_t) \in [L \tau_2]_V$

   We induct on $(T, l_s, l_t) \in [L \tau_1]_V$

   - Base case:
     
     In this case $l_s = \text{NULL}$ and $l_t = \text{nil}$:
     
     It suffices to prove that $(T, \text{NULL}, \text{nil}) \in [L \tau_2]_V$

     This holds trivially from Definition 33

   - Inductive case
     
     In this case we have $l_s = \ell$ and $l_t = ^t v_0 :: l_t$:
     
     It suffices to prove that $(T, \ell, ^t v_0 :: l_t) \in [L \tau_2]_V$

     Again from Definition 33 it suffices to show that

89
\[ \exists^* v_{h1}, \tau, \ell, H(\ell) = (v_{h1}, \ell, s) \land (T, v_{h1}, \ell) \in [\tau_2]\nu \land (T, \ell, s_1, l_t) \in [L \tau_2]\nu \]

Since we are given that \((T, \ell, v_{h1} : \ell_t) \in [L \tau_1]\nu\) therefore from Definition 33 we have
\[ \exists^* v_{h1}, \ell, \tau, H(\ell) = (v_{h1}, \ell, s) \land (T, v_{h1}, \ell) \in [\tau_1]\nu \land (T, \ell, s_1, l_t) \in [L \tau_1]\nu \]

(S-L1)

We choose \(v_{h1}\) as \(v_h\) and \(\ell_s\) as \(\ell_s\)

- \(H(\ell) = (v_{h1}, \ell_s)\):
  - Directly from (S-L1)

- \((T, v_{h1}, \ell) \in [\tau_1]\nu\):
  - From IH of outer induction

- \((T, \ell, s_1, l_t) \in [L \tau_2]\nu\):
  - From IH of inner induction

\[ \square \]

Lemma 43 (Expression subtyping lemma). \(\forall \tau, \tau', H, v, \tau, e_1\).

\(\tau \Rightarrow \tau' \land (T, e_1, e_1) \in [\tau']_E^\nu \Rightarrow (T, e_1, e_1) \in [\tau']_E^\nu\)

Proof. From Definition 33 we are given that
\[ \forall H', *v, p, p', t < T, \forall V, H \vdash_p^v e_s \Downarrow^\nu \begin{cases} *v, H' \\ \epsilon \end{cases} \quad \text{if} \quad (T, t, *v, v_f) \in [\tau']_E^\nu \land p - p' \leq J \]

(SE0)

Also from Definition 33 it suffices to prove that
\[ \forall H', *v, p, p', t_1 < T, \forall V, H \vdash_p^v e_s \Downarrow^\nu \begin{cases} *v, H' \\ \epsilon \end{cases} \quad \text{if} \quad (T - t_1, *v, v_f) \in [\tau']_E^\nu \land p - p' \leq J \]

This means given some \(H', *v, p, p', t_1 < T\) s.t. \(V, H \vdash_p^v e_s \Downarrow^\nu *v, H'\) it suffices to prove that
\[ \exists^* v_{i_1}, v_f, J, e_t \Downarrow^\nu *v, v_f \Downarrow^\nu \begin{cases} *v, H' \\ \epsilon \end{cases} \quad \text{if} \quad (T - t_1, *v, v_f) \in [\tau']_E^\nu \land p - p' \leq J \]

We instantiate (SE0) with \(H', *v, p, p', t_1\) and we get
\[ \exists^* v_{i_1}, v_f, J, e_t \Downarrow^\nu *v, v_f \Downarrow^\nu \begin{cases} *v, H' \\ \epsilon \end{cases} \quad \text{if} \quad (T - t_1, *v, v_f) \in [\tau']_E^\nu \land p - p' \leq J \]

(SE1)

We get the desired from (SE1) and Lemma 42

\[ \square \]

A.5.3 Re-derived Univariate RAML’s soundness

Definition 44 (Translation of Univariate RAML stack). \((V : \Gamma)_H \equiv \forall x \in \text{dom}(\Gamma), (V(x))_{H, \Gamma(x)}\)

Definition 45 (Translation of Univariate RAML values).

\[ \left[ *v \right]_{H, \tau} \triangleq \begin{cases} *v & \text{if} \quad \tau = \text{unit} \\ \text{nil} & \text{if} \quad \tau = b \\ \left[ (L \downarrow t)_{H, \tau'} \downarrow t \right] & \text{if} \quad \tau = L \tau' \land *v = \text{NULL} \\ \left[ (H(\ell) \downarrow 1)_{H, \tau_1}, (H(\ell) \downarrow 2)_{H, \tau_2} \right] & \text{if} \quad \tau = (\tau_1, \tau_2) \land *v = \ell \end{cases} \]

Lemma 46 (Irrelevance of \(T\) for translated value). \(\forall^* v, \tau, H, \nu \rightarrow T \vdash \left[ *v \right]_{H, \tau} \in [\nu]\) in \(\lambda\)-amor

Proof. By induction on \(\tau\)

1. \(\tau = \text{unit}:\)
   - To prove: \(\forall T. (\Phi_H(*v : \tau), T, \left[ *v \right]_{H, \tau}) \in \left[ [\text{unit}] \right]\)
     - This means given some \(T\) it suffices to prove that
       \((\Phi_H(*v : \text{unit}), T, \left[ *v \right]_{H, \text{unit}}) \in [1]\)
     - We know that \(\Phi_H(*v : \text{unit}) = 0\) therefore it suffices to prove that
       \((0, T, *v) \in [1]\)
     - Since we know that \(*v \in [\text{unit}]\) therefore we know that \(*v = ()\)
     - Therefore we get the desired directly from Definition 15

2. \(\tau = b: \)
   - To prove: \(\forall T. (\Phi_H(*v : \tau), T, \left[ *v \right]_{H, \tau}) \in \left[ [b] \right]\)
     - This means given some \(T\) it suffices to prove that
       \((\Phi_H(*v : b), T, \left[ *v \right]_{H, \tau}) \in [b]\)

90
We know that $\Phi_H(s^v : b) = 0$ therefore it suffices to prove that 
$(0, T, s^v) \in [b]$ 
From Definition 15 it suffices to prove that 
$(0, T, s^v) \in [b]$ 

Since we know that $s^v \in [b]$ 
Therefore we get the desired directly from Definition 15 

3. $\tau = L^{\mathcal{L}_\tau}$: 

By induction on $s^v$: 

- $s^v = NULL = []$: 
  
  To prove: $\forall T . (\Phi_H(s^v : \tau), T, (\tau)_{H,L,s^v,\tau}) \in \{L^{\mathcal{L}_\tau}\}$ 

  This means given some $T$ it suffices to prove that 
  $(\Phi_H([], []), T, (\tau)) \in \{L^{\mathcal{L}_\tau}\}$ 

  From Definition 15 it suffices to prove that 
  $\exists s'. (0, T, (\tau)) \in \{L^{\mathcal{L}_\tau}\}$ 

  By choosing $s'$ as 0 and it suffices to prove that 
  $(0, T, (\tau)) \in \{L^{\mathcal{L}_\tau}\}$ 

- $s^v = \ell = [\tau v_1, \ldots, \tau v_n]$: 
  
  To prove: $\forall T . (\Phi_H([\tau v_1 \ldots \tau v_n], T, (\tau)_{H,L,s^v,\tau}) \in \{L^{\mathcal{L}_\tau}\}$ 

  This means given some $T$ it suffices to prove that 
  $(\Phi_H([\tau v_1 \ldots \tau v_n], T, (\tau)_{H,L,s^v,\tau}) \in \{L^{\mathcal{L}_\tau}\}$ 

  From Definition 15 it suffices to prove that 
  $\exists s'. (\Phi_H([\tau v_1 \ldots \tau v_n], T, (\tau)_{H,L,s^v,\tau}) \in \{L^{\mathcal{L}_\tau}\}$ 

  By choosing $s'$ as $n$ and it suffices to prove that 
  $(\Phi_H([\tau v_1 \ldots \tau v_n], T, (\tau)_{H,L,s^v,\tau}) \in \{L^{\mathcal{L}_\tau}\}$ 

From Definition 45 we know that $(\tau)_{H,L,s^v,\tau} = \{L^{\mathcal{L}_\tau}\}$ 

From Definition 15 it further suffices to prove that 

$\exists p_1, p_2, p_1 + p_2 \leq (\Phi_H([\tau v_1 \ldots \tau v_n], (H(\tau) \downarrow_1)_{H,L,s^v,\tau}) \in \{L^{\mathcal{L}_\tau}\}$ 

We know that $\Phi_H([\tau v_1 \ldots \tau v_n], (H(\tau) \downarrow_1)_{H,L,s^v,\tau}) = (\Phi(n - 1, \tau q) + \sum_{2 \leq i \leq n} \Phi_H([\tau v_i : \tau']) \land (p_1, T, (\tau)) \in [[\phi(\tau q), n - 1]]$ 

From Definition 15 this means we have 

$\exists s'. ((\Phi(n - 1, \tau q) + \sum_{2 \leq i \leq n} \Phi_H([\tau v_i : \tau']) \land (p_1, T, (\tau)) \in [[\phi(\tau q), n - 1]]$ 

From Definition 15 we have 

$\exists p_1, p_2, p_1 + p_2 \leq (\Phi(n - 1, \tau q) + \sum_{2 \leq i \leq n} \Phi_H([\tau v_i : \tau']) \land (p_1, T, (\tau)) \in [[\phi(\tau q), n - 1]]$ 

In order to prove (L0) we choose $p_1$ as $p_1' + q_1$ and $p_2$ as $p_2' + \Phi_H([\tau v_i : \tau'])$; 

- $p_1 + p_2 \leq (\Phi(n, q) + \sum_{1 \leq i \leq n} \Phi_H([\tau v_i : \tau'])$ 

It suffices to prove that
Lemma 47 (Irrelevance of $T$ for translated $\Gamma$). \( \forall \nu, \tau, \nu H \). 

\( H \models V : \Gamma \) in RAML \( \Rightarrow \forall T . (\Phi_{V,H}(\Gamma), T, (V : \Gamma)_{H}) \in \llbracket \Gamma \rrbracket \) in $\lambda$-amor

Proof. To prove: \( \forall T . (\Phi_{V,H}(\Gamma), T, (V : \Gamma)_{H}) \in \llbracket \Gamma \rrbracket \)

This means given soem $T$ it suffices to prove that $(\Phi_{V,H}(\Gamma), T, (V : \Gamma)_{H}) \in \llbracket \Gamma \rrbracket$

From Definition 16 it suffices to prove that

\[ \exists f : \text{Vars} \rightarrow \text{Dats}. (\forall x \in \text{dom}(\Gamma)). (f(x), T, (V : \Gamma)_{H}(x)) \in \llbracket \Gamma \rrbracket(x)_{x} \Leftrightarrow (\sum_{x \in \text{dom}(\Psi)} f(x) \leq \Phi_{V,H}(\Gamma)) \]

We choose $f(x)$ as $\Phi_{H}(V(x) : \Gamma(x))$ for every $x \in \text{dom}(\Gamma)$ and it suffices to prove that

* (\( \forall x \in \text{dom}(\Gamma) \)). $(\Phi_{H}(V(x) : \Gamma(x)), T, (V : \Gamma)_{H}(x)) \in \llbracket \Gamma \rrbracket(x)_{x}$:

This means given some $x \in \text{dom}(\Gamma)$ it suffices to prove that

$(\Phi_{H}(V(x) : \Gamma(x)), T, (V : \Gamma)_{H}(x)) \in \llbracket \Gamma \rrbracket(x)_{x}$
From Definition [44] it suffices to prove that 
\((\Phi_H(V(x) : \Gamma(x)), T, (V(x))_{H,T(x)}) \in \|\Gamma(x)\|_c\) 

From Lemma [48] we know that 
\((\Phi_H(V(x) : \Gamma(x)), T, (V(x))_{H,T(x)}) \in \|\Gamma(x)\|_c\) 

And finally from Definition [15] we have 
\((\Phi_H(V(x) : \Gamma(x)), T, (V(x))_{H,T(x)}) \in \|\Gamma(x)\|_c\)

- \(\{x : \delta \in \text{dom}(\bf{V}) \}, f(x) \leq \Phi_T(V(x) : \Gamma)(x)\) 
  - Since we know that \(\Phi_T(V(x) : \Gamma) = \sum_{x \in \text{dom}(\bf{V})} \Phi_H(V(x) : \Gamma(x))\) therefore we are done

**Lemma 48** (RAML’s stack and its translation are in the cross-lang relation). \(\forall H, V, \Gamma. \) 
\(H \models V : \Gamma \implies \forall T . (T, V, (V : \Gamma)_H) \in [\Gamma]_V^H\)

**Proof.** Given some \(T\), it suffices to prove that \((T, V, (V : \Gamma)_H) \in [\Gamma]_V^H\)

From Definition [44] it suffices to prove that 
\(\forall x : \tau \in \text{dom}(\Gamma), (T, V(x), (V : \Gamma)_H(x)) \in [\tau]_V^H\)

This means given some \(x : \tau \in \text{dom}(\Gamma)\) and we need to prove that 
\((T, V(x), (V : \Gamma)_H(x)) \in [\tau]_V^H\)

Since we are given that \(H \models V : \Gamma\), it means we have \(\forall x \in \text{dom}(\Gamma). H \models V(x) \in [\Gamma(x)]\)

Therefore we get the desired from Lemma [49]

**Lemma 49** (RAML’s value and its translation are in the cross-lang relation). \(\forall H, \ast_v, \tau. \) 
\(H \models \ast_v \in [\tau] \implies \forall T . (T, \ast_v, (\ast_v)_H, \tau) \in [\tau]_V^H\)

**Proof.** By induction on \(\ast_v\)

1. \(\tau = \text{unit}:
   - To prove: \(\forall T . (T, \ast_v, (\ast_v)_H, \tau) \in [\text{unit}]_V^H\)
     This means given some \(T\), from Definition [45] it suffices to prove that 
     \((T, \ast_v, \ast_v) \in [\text{unit}]_V^H\)
     We get this directly from Definition [33]

2. \(\tau = \mathbf{b}:
   - To prove: \(\forall T . (T, \ast_v, (\ast_v)_H, \tau) \in [\mathbf{b}]_V^H\)
     This means given some \(T\), from Definition [45] it suffices to prove that 
     \((T, \ast_v, \ast_v) \in [\mathbf{b}]_V^H\)
     We get this directly from Definition [33]

3. \(\tau = L_{\ell}^\tau:\]
   - By induction on \(\ast_v\)
     - \(\ast_v = \text{NULL}:
       - To prove: \(\forall T . (T, \text{NULL}, (\ast_v)_H, \tau) \in [\text{NULL}]_V^H\)
         Given some \(T\), from Definition [45] it suffices to prove that 
         \((T, \text{NULL}, (\ast_v)_H) \in [L_{\ell}^\tau]_V^H\)
         We get this directly from Definition [33]

     - \(\ast_v = \ell = [\ast v_1 \ldots v_n]:
       - To prove: \(\forall T . (T, \ell, (\ast_v)_H, \tau) \in [\ell]_V^H\)
         Given some \(T\), from Definition [45] it suffices to prove that 
         \((T, \ell, (\ast_v)_H, \tau) \in [L_{\ell}^\tau]_V^H\)
         From Definition [33] it further suffices to prove that 
         \((T, \ell, (\ast_v)_H, \tau) \in [\ell]_V^H \land (T, (\ell) \downarrow 2, (\ast_v)_H, \tau) \in [L_{\ell}^\tau]_V^H\)
         We get \((T, \ell, (\ast_v)_H, \tau) \in [\ell]_V^H\) from IH of outer induction

4. \(\tau = (\tau_1, \tau_2):
   - To prove: \(\forall T . (T, \ell, (\ast_v)_H, (\tau_1, \tau_2)) \in [(\tau_1, \tau_2)]_V^H\)
     Given some \(T\), from Definition [45] it suffices to prove that 
     \((T, \ell, (\ast_v)_H, (\tau_1, \tau_2)) \in [(\tau_1, \tau_2)]_V^H\)
     From Definition [33] it suffices to prove that 
     \((T, \ell, (\ast_v)_H, (\tau_1, \tau_2)) \in [(\tau_1, \tau_2)]_V^H\)
     We get this directly from IH
Lemma 50. \( \forall s, t, \tau, H, T. \)

\[(T, *s, t, \tau, H, T) \in [\tau]_H^V \implies \tau'_v = (\tau v)_{H, \tau} \]

Proof. Proof by induction on the \([\cdot]_V\) relation

1. \([\text{unit}]_H^V\):
   - Given: \( (T, *s, t, \tau, H, T) \in [\text{unit}]_H^V \)
   - To prove: \( s = (\tau v)_{H, \text{unit}} \)
   - Directly from Definition 45

2. \([b]_H^V\):
   - Given: \( (T, *s, t, \tau, H, T) \in [b]_H^V \)
   - To prove: \( t = (\tau v)_{H, \tau} \)
   - Directly from Definition 45

3. \([\langle\langle \tau_1, \tau_2\rangle\rangle]_V\):
   - Given: \( (T, \ell, \langle\langle \tau_1, \tau_2\rangle\rangle) \in \langle\langle \tau_1, \tau_2\rangle\rangle \)
   - This means from Definition 43 we have
     \( H(\ell) = (\tau_1, \tau_2) \wedge (T, *s, t, \tau_1) \in [\tau_1]_V \wedge (T, *s, t, \tau_2) \in [\tau_2]_V \)
   - To prove: \( \langle\langle \tau_1, \tau_2\rangle\rangle = (\ell)_{H, (\tau_1, \tau_2)} \)
   - From Definition 45 we know that
     \( (\ell)_{H, (\tau_1, \tau_2)} = \langle\langle (H(\ell), \downarrow_1)_{H, \tau_1}, (H(\ell), \downarrow_2)_{H, \tau_2} \rangle\rangle \)
   - From (R0) we know that \( H(\ell) = (\tau_1, \tau_2) \) and \( H(\ell) = (\tau_1, \tau_2) \) therefore we have
     \( (\ell)_{H, (\tau_1, \tau_2)} = \langle\langle (H(\ell), \downarrow_1)_{H, \tau_1}, (H(\ell), \downarrow_2)_{H, \tau_2} \rangle\rangle = \langle\langle \tau_1, \tau_2 \rangle\rangle \)
   - Since from (R0) we know that \( (T, *s, t, \tau_1) \in [\tau_1]_V \) therefore we have
     \( (\tau_1)_{H, \tau_1} = \tau_1 \)
     \( (\tau_2)_{H, \tau_2} = \tau_2 \)
   - Similarly we also have
     \( (\tau_1)_{H, \tau_1} = \tau_1 \)
     \( (\tau_2)_{H, \tau_2} = \tau_2 \)

We get the desired from IH1, IH2 and (R1)

4. \([L\tau']_V\):
   - Given: \( (T, \ell_s, \langle\langle (\ell_s, l_s) \rangle\rangle) \in [L\tau']_V \) where \( (T, \ell_s, l_s) \in [L\tau']_V \)
   - To prove: \( \langle\langle (\ell_s, l_s) \rangle\rangle_{H, \tau} = \langle\langle (\ell_s, l_s) \rangle\rangle_{H, \tau} \)
   - From Definition 45 we know that
     \( (\ell_s, l_s)_{H, \tau} = \langle\langle (\ell_s, l_s) \rangle\rangle_{H, \tau} \)
   - Therefore it suffices to prove that \( \ell_s = (\ell_s)_{H, \tau} \)
   - We induct on \( (T, \ell_s, l_s) \) where \( [L\tau']_V \)
     \( \alpha \) \( (\text{NULL})_V \)
     - In this case we know that \( \ell_s = \text{nil} \)
     - From Definition 45 we get the desired
     \( \beta \) \( (\ell_s) \neq \text{NULL} \)
     - In this case we know that \( \ell_s = 'v_h : l'_s \) s.t
     \( H(\ell) = (\tau', \ell'_s) \wedge (T, *s, t, \tau, \tau') \in \langle\langle (\ell, \ell'_s) \rangle\rangle_{H, \tau} \)
     - We get the desired from Definition 45 of outer induction and IH of inner induction

Definition 51 (Top level RAML program translation). Given a top-level RAML program

\( P \equiv F, e_{\text{main}} \) where \( F \equiv f_1(x) = e_{f_1}, \ldots, f_n(x) = e_{f_n} \) s.t

\[ \Sigma, x : \tau f_1 \vdash_{q_1} e f_1 : \tau' f_1 \]
\[ \ldots \]
\[ \Sigma, x : \tau f_n \vdash_{q_n} e f_n : \tau' f_n \]
\[ \Sigma, \Gamma \vdash_{q} \text{main} : \tau \]

where \( \Sigma = f_1 : \tau f_1 \vdash_{q_1} e f_1 : \tau' f_1, \ldots, f_n : \tau f_n \vdash_{q_n} e f_n : \tau' f_n \)

Translation of \( P \) denoted by \( \mathcal{P} \) is defined as \( F, e \) where

\[ \mathcal{F} = \text{fix} f_1, \lambda u, \lambda x. e_{t_1}, \ldots, \text{fix} f_n, \lambda u, \lambda x. e_{t_n} \) s.t
\[ \Sigma, x : \tau f_1 \vdash_{q_1} e f_1 : \tau' f_1 \implies e_{t_1} \]
\[ \ldots \]
\[ \Sigma, x : \tau f_n \vdash_{q_n} e f_n : \tau' f_n \implies e_{t_n} \]

and

\[ \Sigma, \Gamma \vdash_{q} \text{main} : \tau \implies e \]
Theorem 52 (RAML univariate soundness). \( \forall H, H', V, \Gamma, \Sigma, c, \tau, s, v, p, p', q, q', t. \)
\[ P = F, e \text{ and } \overrightarrow{\text{P}} \text{ be a RAML top-level program and its translation respectively (as defined in Definition 51)} \]
\[ H \models V : \Gamma \land \Sigma, \Gamma \vdash q \quad e : \tau \land V, H \vdash p_s e \downarrow \vdash s, v, H' \]
\[ \implies p - p' \leq (\Phi_{H, V}(\Gamma) + q) - (q' + \Phi_H(s, v, \tau)) \]

Proof. From Definition 51, we are given that
\[ F \triangleq f_1(x) = e_{f_1}, \ldots, f_n(x) = e_{f_n} \text{ s.t.} \]
\[ \Sigma, x : \tau_f \vdash q_f x : \tau_f, \rightsquigarrow e_t, \]
\[ \ldots \]
\[ \Sigma, x : \tau_f \vdash q_f x : \tau_f, \rightsquigarrow e_t \]

Let \( \forall i \in [1 \ldots n], \delta_{sf}(f_i) = (f_i(x) = e_{f_i}) \) and \( \forall i \in [1 \ldots n], \delta_{sf}(f_i) = (\fix f_i, \lambda u. \lambda x. e_t) \)

Claim: \( \forall T \vdash (T, \delta_{sf}, \delta_{tf}) \in [\Sigma]^H \)

Proof.
This means given some \( T \), it suffices to prove that
\( (T, \delta_{sf}, \delta_{tf}) \in [\Sigma]^H \)

We induct on \( T \)

Base case: Trivial

Inductive case:
\( \text{IH: } \forall T' < T \vdash (T', \delta_{sf}, \delta_{tf}) \in [\Sigma]^H \)

From Definition 55, it suffices to prove that
\[ \forall f_i \in \text{dom}(\Sigma), (T, f_i(x) = e_{f_i}, \delta_{sf}, \fix f_i, \lambda u. \lambda x. e_t, \delta_{sf}) \in [\tau_f]^{q_f/q'}_{\tau_f}_H \]

Given some \( f_i \in \text{dom}(\Sigma) \), it suffices to prove that
\[ (T, f_i(x) = e_{f_i}, \delta_{sf}, \fix f_i, \lambda u. \lambda x. e_t, \delta_{sf}) \in [\tau_f]^{q_f/q'}_{\tau_f}_H \]

From Definition 55 it suffices to prove that
\[ \forall v', t v', T' < T \vdash (T', *v', t v', *v') \in [\tau_f]^{q_f/q'}_{\tau_f}_H \]

This means given some \( *v', t v', T' < T \) s.t. \( (T', *v', t v') \in [\tau_f]^{q_f/q'}_{\tau_f}_H \) it suffices to prove that
\[ (T', e_{f_i}, \delta_{sf}, \delta_{sf} f_i() / [v' / x], [\fix f_i, \lambda u. \lambda x. e_t, \delta_{sf} f_i]) \in [\tau_f]^{q_f/q'}_{\tau_f}_H \]

Also since are given \( (T', *v', t v') \) and therefore we have
\[ (T', (x \mapsto *v'), (x \mapsto t v')) \in [x : \tau_f]^{q_f/q'}_{\tau_f}_H \]

Also from IH we have \( (T', \delta_{sf}, \delta_{tf}) \in [\Sigma]^V_H \)

We can apply Theorem 41 to get
\[ (T', e_{f_i}, \delta_{sf}, \delta_{sf} f_i()) \{ x \mapsto t v' \} \delta_{tf} \in [\tau_f]^{q_f/q'}_{\tau_f}_H \]

And this prove (C0)

From Theorem 31, we know that \( \exists e_t \) s.t
\[ \Sigma, \Gamma \vdash q, \quad e : \tau \rightsquigarrow e_t \text{ and } \vdash : (\Sigma); (\Gamma) \vdash e_t : [q] 1 \rightsquigarrow \emptyset 0 [q'] \tau \]

From Lemma 48, we know that \( \forall T \vdash (T, V, (V : \Gamma)_{H'}) \in [\Gamma]^{q_f}_H \)

Also from the Claim proved above we know that \( \forall T \vdash (T, \delta_{sf}, \delta_{tf}) \in [\Sigma]^H \)

Therefore from Theorem 41, we know that \( \forall T \vdash (T, e_{sf}, e_t) (V : \Gamma)_{H'} \delta_{tf} \in [\tau_f]^{q_f/q'}_{\tau_f}_H \)

This means from Definition 53, we have
\[ \forall T \vdash H'_{t}, *v_{s}, t_{p}, (t' < T') \quad V, H \vdash p_{s} e \downarrow s, v_{t} \quad H'_{t} \implies \exists v_{s}, t v_{f}, J, e_{t} (V : \Gamma)_{H'} \delta_{tf} \downarrow \forall \downarrow t v_{f} \downarrow \forall \downarrow t v_{f} \wedge (T - t', *v_{s}, t v_{f}) \in [\tau_f]^{q_f/q'}_{\tau_f}_H \]

(RD-0.0)

We are given that \( V, H \vdash p_{s} e \downarrow s, v, H' \)

Therefore instantiating (RD-0.0) with \( t + 1, H', *v_{s}, p', t \) we get
\[ \exists v_{s}, t v_{f}, J, e_{t} (V : \Gamma)_{H'} \delta_{tf} \downarrow \forall \downarrow t v_{f} \downarrow \forall \downarrow t v_{f} \wedge (1, *v_{s}, t v_{f}) \in [\tau_f]^{q_f/q'}_{\tau_f}_H \quad \text{and} \quad p - p' \leq J \]

(RD-0)
Since from Lemma \[47\] we know that \( \forall T. (\Phi_{V,H}(\Gamma), T, (V : \Gamma)_H) \in [[\Gamma]] \)
Therefore we also have \( (\Phi_{V,H}(\Gamma), t_1 + t_2 + 1, (V : \Gamma)_H) \in [[\Gamma']] \)

Therefore from Theorem \[29\] we get
\[ \exists p_e. (p_e, 1, v_f) \in [[\tau]] \wedge J \leq (q + \Phi_{V,H}(\Gamma)) - (q' + p_e) \quad (RD-1) \]

Since we have \((1, v, v_f) \in [\tau]'_H\) therefore from Lemma \[50\] we know that \( v_f = (v)_H, \tau \)

From Lemma \[66\] we know that \( \forall T. (\Phi_{H}(v : \tau), T, (v)_H, \tau) \in [[\tau]] \)

Therefore we have \( (\Phi_{H}(v : \tau), 1, (v)_H, \tau) \in [[\tau]] \) \( (RD-2) \)

From (RD-1), (RD-2) and Lemma \[61\] we know that \( p_e \geq \Phi_{H}(v : \tau) \)

Since from (RD-1) we know that \( J \leq (q + \Phi_{V,H}(\Gamma)) - (q' + p_e) \) therefore we also have
\( J \leq (q + \Phi_{V,H}(\Gamma)) - (q' + \Phi_{H}(v : \tau)) \quad (RD-3) \)

Finally from (RD-0) and (RD-3) we get the desired.

\[ \blacksquare \]

B. Development of λ-amor (full)

B.1 Syntax

Expressions
\[
e ::= v | e_1 e_2 | \langle e_1, e_2 \rangle | \text{let}(x, y) = e_1 \in e_2 | \langle e, e \rangle | \text{fst}(e) | \text{snd}(e) | \text{inl}(e) | \text{inr}(e) | \text{case } c, e, e, y, e | \text{let! } x = e_1 \in e_2 | e :: e | e ; x. e
\]

Values
\[
v ::= x | e | \lambda x.e | \langle e_1, e_2 \rangle | \langle v, v \rangle | \text{inl}(e) | \text{inr}(e) | \text{nil} | \text{let! } x. e | \text{ret } e | \text{bind } x = e_1 \in e_2 | \text{release } e = e_1 \in e_2 | \text{store } e
\]

(No value forms for \([I \tau]\))

Index
\[
I ::= N | i | I + I | I - I | \sum_{a \leq I} I | \Omega_a I | \lambda x. I | I I
\]

Sort
\[
S ::= \mathbb{N} | \mathbb{R}^+ | S \rightarrow S
\]

Kind
\[
K ::= Type | S \rightarrow K
\]

Types
\[
\tau ::= 1 | b | \tau_1 \rightarrow \tau_2 | \tau_1 \odot \tau_2 | \tau_1 \& \tau_2 | \tau_1 \oplus \tau_2 | \lambda a. \tau | [I \tau] | \mathbb{M} I \tau | \alpha | \forall \alpha : K. \tau | \forall i : S. \tau | \lambda x. \tau | \tau I | L I \tau | \exists i : S. \tau | c \Rightarrow \tau | c \& \tau
\]

Constraints
\[
c ::= I = I | I < I | c \land c
\]

Lin. context
\[
\Gamma ::= . | \Gamma, x : \tau
\]

for term variables

Bounded Lin. context
\[
\Omega ::= . | \Omega, x : a < I \tau
\]

for term variables

Unres. context
\[
\Theta ::= . | \Theta, i : S
\]

for sort variables

Unres. context
\[
\Psi ::= . | \Psi, \alpha : K
\]

for type variables

Definition 53 (Bounded sum of context for dLPCF).
\[
\sum_{a \leq I} \cdot = \cdot
\]

\[\sum_{a \leq I} \Gamma, x : [b < J] \tau = (\sum_{a \leq I} \Gamma), x : [c < \sum_{a < I} J][c] \]

where
\[\tau = \sigma([\sum_{d < a} J[d/a] + b/c] \]

Definition 54 (Bounded sum of multiplicity context).
\[
\sum_{a \leq I} \cdot = \cdot
\]

\[\sum_{a \leq I} \Omega, x : \varphi < J \tau = (\sum_{a \leq I} \Omega), x : c < \sum_{a < I} \tau \sigma\]

where
\[\tau = \sigma([\sum_{d < a} J[d/a] + b/c] \]

Definition 55 (Binary sum of context for dLPCF).
\[
\begin{align*}
\Gamma_1 & = \cdot \\
\Gamma_1 + \Gamma_2 & = \begin{cases} 
\Gamma_2 & (\Gamma_1' + \Gamma_2/x), x : [c < I + J] \tau \\
(\Gamma_1 + \Gamma_2), x : a < I \tau & \Gamma_1 = \Gamma_1', x : [a < I] \tau \land (x : [-]-) \notin \Gamma_2 \\
(\Gamma_1 + \Gamma_2'), x : a < I \tau & \Gamma_1 = \Gamma_1', x : [a < I] \tau \land (x : [-]-) \notin \Gamma_2
\end{cases}
\end{align*}
\]

96
Definition 56 (Binary sum of multiplicity context).

\[\Omega_1 \oplus \Omega_2 \triangleq \begin{cases} 
\Omega_2 
& \Omega_1 = . \\
(\Omega'_1 \oplus \Omega_2/x), x : a \in I, \tau 
& \Omega_1 = \Omega'_1, x : a \in I, \tau \land (x : -) \notin \Omega_2 \\
(\Omega'_1 \oplus \Omega_2), x : a \in I, \tau 
& \Omega_1 = \Omega'_1, x : a \in I, \tau \land (x : -) \notin \Omega_2
\end{cases} \]

Definition 57 (Binary sum of affine context).

\[\Gamma_1 \oplus \Gamma_2 \triangleq \begin{cases} 
\Gamma_2 
& \Gamma_1 = . \\
(\Gamma'_1 \oplus \Gamma_2), x : \tau 
& \Gamma_1 = \Gamma'_1, x : \tau \land (x : -) \notin \Gamma_2
\end{cases} \]
B.2 Typesystem

Typing \( \Theta; \Delta; \; \Gamma \vdash e : \tau \)

\[
\begin{align*}
\text{T-var1} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash c : b \\
\text{T-var2} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash () : 1
\end{align*}
\]

\[
\begin{align*}
\text{T-base} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash e : \tau \\
\text{T-unit} & : \Theta, \Delta \vdash I \geq 1
\end{align*}
\]

\[
\begin{align*}
\text{T-nil} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash \text{nil} : L^0 \tau \\
\text{T-match} & : \Theta, \; \Delta, n = 0 ; \Omega; \Gamma \vdash e_1 : \tau' \\
\text{T-existI} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash e : \tau[n/s] \\
\text{T-existE} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash e : \exists s : S.\tau
\end{align*}
\]

\[
\begin{align*}
\text{T-lam} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash \lambda x. e : (\tau \to \tau_2) \\
\text{T-app} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash e_1 \ e_2 : \tau_2 \\
\text{T-sub} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash e : \tau' \\
\text{T-tensorI} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash e_1 \ e_2 : (\tau_1 \otimes \tau_2) \\
\text{T-tensorE} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash \text{let} \langle x, y \rangle = e \text{ in } e' : \tau
\end{align*}
\]

\[
\begin{align*}
\text{T-withI} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash (e_1, e_2) : (\tau_1 \& \tau_2) \\
\text{T-fst} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash \text{fst}(e) : \tau_1 \\
\text{T-snd} & : \Psi; \Theta; \Delta; \; \Omega; \Gamma \vdash \text{snd}(e) : \tau_2
\end{align*}
\]
Figure 27: Typing rules for \(\lambda\)-amor
Figure 28: Subtyping

\[
\begin{align*}
\Psi; \Theta; \Delta &\vdash \tau <: \tau & \text{sub-refl} \\
\Psi; \Theta; \Delta &\vdash \tau_1 <: \tau_1' & \text{sub-tensor} \\
\Psi; \Theta; \Delta &\vdash \tau_1 \otimes \tau_2 <: \tau_1' \otimes \tau_2' & \text{sub-sum} \\
\Psi; \Theta; \Delta &\vdash \tau_1 <: \tau_1' & \text{sub-sub} \\
\Psi; \Theta; \Delta &\vdash \tau_1 \otimes \tau_2 <: \tau_1' \otimes \tau_2' & \text{sub-with} \\
\Psi; \Theta; \Delta &\vdash \tau <: \tau' & \text{sub-potential} \\
\Psi; \Theta; \Delta &\vdash [n] \tau <: [n'] \tau' & \text{sub-potent} \\
\Psi; \Theta; \Delta &\vdash a <: J <: \tau & \text{sub-exp} \\
\Psi; \Theta; \Delta &\vdash \tau_1 <: \tau_2 & \text{sub-potIndex} \\
\Psi; \Theta; \Delta &\vdash \tau <: \tau' & \text{sub-pot} \\
\Psi; \Theta; \Delta &\vdash \tau <: 0 & \text{sub-potZero} \\
\Psi; \Theta; \Delta &\vdash \tau <: \lambda i. \tau & \text{sub-familyExp} \\
\Psi; \Theta; \Delta &\vdash \sum_{a \lessdot l} K[a] \tau <: \tau & \text{sub-base} \\
\end{align*}
\]

Figure 29: Γ Subtyping for dLPCF

\[
\begin{align*}
\Psi; \Theta; \Delta &\vdash \Gamma \subseteq. \\
x : [a < J] \tau' \in \Gamma_1 & \Psi; \Theta; \Delta; a; a < I \vdash \tau' \subseteq \tau & \text{dlpcf-subBase} \\
\Theta; \Delta &\vdash \Gamma_1 \subseteq \Gamma_2, x : [a < I] \tau & \text{dlpcf-subInd} \\
\end{align*}
\]

Figure 30: Ω Subtyping

\[
\begin{align*}
\Psi; \Theta; \Delta &\vdash \Omega \subseteq. \\
x : a < J & \tau' \in \Omega_1 & \Psi; \Theta; \Delta; a; a < I \vdash \tau' <: \tau & \text{sub-mInd} \\
\Psi; \Theta; \Delta &\vdash \Omega_1 \subseteq \Omega_2, x : a < I \tau & \text{sub-mBase} \\
\end{align*}
\]

Figure 31: Γ Subtyping

\[
\begin{align*}
\Psi; \Theta; \Delta &\vdash \Gamma \subseteq. \\
x : \tau' \in \Gamma_1 & \Psi; \Theta; \Delta \vdash \tau' <: \tau & \text{sub-lBase} \\
\Psi; \Theta; \Delta &\vdash \Gamma_1 \subseteq \Gamma_2, x : \tau & \text{sub-lBase} \\
\end{align*}
\]
Figure 32: Typing rules for sorts

Figure 33: Kind rules for types
B.3 Semantics

Forcing reduction, $e \Downarrow_f v$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1 \Downarrow_t v$, $e_2 \Downarrow_t l$</td>
<td>E-cons</td>
</tr>
<tr>
<td>$e_1 :: e_2 \Downarrow_{t+1+t+2+1} v :: l$</td>
<td>E-match</td>
</tr>
<tr>
<td>$e_1 \Downarrow_t v$, $v_h :: l$</td>
<td>E-matchCons</td>
</tr>
<tr>
<td>$e_2 [v_h/h] [l/t] \Downarrow_t v$</td>
<td>E-exist</td>
</tr>
<tr>
<td>$e_1 \Downarrow_t \lambda x. e'$</td>
<td>E-app</td>
</tr>
<tr>
<td>$e_1 \Downarrow_t (e_2 / x) \Downarrow_t v'$</td>
<td>E-TI</td>
</tr>
<tr>
<td>$e \Downarrow_{t1} (v_1, v_2)$</td>
<td>E-TE</td>
</tr>
<tr>
<td>$\langle \langle v, \Lambda \rangle \rangle \Downarrow_t v$</td>
<td>E-WI</td>
</tr>
<tr>
<td>$\langle \langle v_1, v_2 \rangle \rangle$</td>
<td>E-fst</td>
</tr>
<tr>
<td>$\Downarrow_{t1} \mathsf{inl}(v)$</td>
<td>E-case1</td>
</tr>
<tr>
<td>$\Downarrow_{t1} \mathsf{inr}(v)$</td>
<td>E-case2</td>
</tr>
<tr>
<td>$\Downarrow_{t1} \mathsf{exp} e$</td>
<td>E-expI</td>
</tr>
<tr>
<td>$\Downarrow_{t1} \mathsf{fix} e$</td>
<td>E-fix</td>
</tr>
<tr>
<td>$e \Downarrow_t \mathsf{store}(e)$</td>
<td>E-store</td>
</tr>
</tbody>
</table>

$v \in \{(), \mathit{nil}, \lambda y.e, \Lambda.e, \mathsf{fix} e, \mathit{bind} x = e_1 \in e_2, \mathit{ret} e, \mathit{release} x = e_1 \in e_2, \mathit{store} e\}$

Figure 34: Evaluation rules: pure and forcing
B.4 Model

Definition 58 (Value and expression relation).

\[ \Delta \equiv \{(p, T, (\_))\} \]
\[ \Delta \equiv \{(p, T, v) \mid v \in [\mathbb{B}]\} \]
\[ \Delta \equiv \{(p, T, nil)\} \]
\[ \Delta \equiv \{(p, T, v \cdot l) \mid \exists p_1, p_2, p_1 + p_2 \leq p \land (p_1, T, v) \in [\mathcal{F}] \land (p_2, T, l) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, (v_1, v_2)) \mid \exists p_1, p_2, p_1 + p_2 \leq p \land (p_1, T, v_1) \in [\mathcal{I}] \land (p_2, T, v_2) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, (v_1, v_2)) \mid (p, T, v_1) \in [\mathcal{I}] \land (p, T, v_2) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, v \cdot l) \mid (p, T, v) \in [\mathcal{I}] \lor \{(p, T, \text{in}(l)) \mid (p, T, v) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, \lambda x.e) \mid \forall p', e'.T'<T'.(p', T', e') \in [\mathcal{I}] \Rightarrow (p + p', T', e'/x) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, c) \mid \exists p_0, \ldots, p_{n-1}.p_0 + \ldots + p_{n-1} \leq p \land \forall 0 < i \leq n. (p, T, e) \in [\mathcal{F}[i/a]\} \]
\[ \Delta \equiv \{(p, T, v) \mid \exists p', p' + n \leq p \land (p', T, v) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, v) \mid \forall p', v'.T'<T'.v' \Rightarrow \exists p'.v + p' \leq p + n \land (p', T - T', v') \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, e) \mid \forall i. f I = [\mathcal{F}[i/a]\} \]
\[ \Delta \equiv \{(p, T, v) \mid \exists c. (p, T, v) \in [\mathcal{I}]\} \]
\[ \Delta \equiv \{(p, T, e) \mid \forall v. T'<T.e \downarrow_{T'} v \Rightarrow (p, T - T', v) \in [\mathcal{I}]\} \]

Definition 59 (Interpretation of typing contexts).

\[ [\Gamma]_{\mathcal{E}} \equiv \{(p, T, \gamma) \mid \exists f : \text{Vars} \rightarrow \text{Pets} \land (\forall x \in \text{dom}(\Gamma). (f(x), T, \gamma(x)) \in [\Gamma(x)]_{\mathcal{E}}) \land (\sum_{x \in \text{dom}(\Gamma)} f(x)) \leq p)\} \]

\[ [\Omega]_{\mathcal{E}} \equiv \{(p, T, \delta) \mid \exists f : \text{Vars} \rightarrow \text{Indices} \rightarrow \text{Pets} \land (\forall (x \_a < i \_\Omega). \forall 0 \leq i < I. (f x i, T, \delta(x)) \in [\mathcal{F}[i/a]\} \land (\sum_{x \_a < i \_\Omega} \sum_{0 \leq i < I} f x i) \leq p)\} \]

Definition 60 (Type and index substitutions). \( \sigma : \text{TypeVar} \rightarrow \text{Type}, \iota : \text{IndexVar} \rightarrow \text{Index} \)

Lemma 61 (Value monotonicity lemma). \( \forall p, p', v, \tau, T', T. (p, T, v) \in [\mathcal{I}] \land p \leq p' \land T' \leq T \Rightarrow (p', T', v) \in [\mathcal{I}] \)

Proof. Proof by induction on \( \tau \)

Lemma 62 (Expression monotonicity lemma). \( \forall p, p', v, \tau, T', T. (p, T, e) \in [\mathcal{F}] \land p \leq p' \land T' \leq T \Rightarrow (p', T', e) \in [\mathcal{F}] \)

Proof. From Definition 58 and Lemma 61

Lemma 63 (Lemma for substitution). \( \forall p, \delta, I, \Omega. (p, \delta) \in [\sum_{a \_\Omega} \Omega] \Rightarrow \exists p_0, \ldots, p_{n-1}. p_0 + \ldots + p_{n-1} \leq p \land \forall 0 \leq i < I. (p_i, \delta) \in [\Omega[i/a]\}

Proof. Given: \( (p, \delta) \in [\sum_{a \_\Omega} \Omega] \)

When \( \Omega = \).

The proof is trivial simply choose \( p_i \) as 0 and we are done

When \( \Omega(a) = x_0 : \_a < J_0(a), \ldots, x_n : \_a < J_n(a), \tau_n(a) \)

Therefore from Definition 53 and Definition 59 we have

\[ \exists \sigma : \text{Vars} \rightarrow \text{Indices} \rightarrow \text{Pets} \land (\forall (x_j : \_a < \sum_{a < J} J_a). \sigma \in (\sum_{a < J} \Omega) \land (\sum_{x_j : \_a < \sum_{a < J} J_a} \sigma(x_j(a)) \leq p) \quad \text{(SM0)} \]

To prove the desired, for each \( i \in [0, I - 1] \) we choose

\( p_k \) as \( \sum_{x_j : \_a < J_j(a)} \tau_j(i) \in (\Omega(i)) \sum_{0 \leq k < J_j(a)} f x_j(k + \sum_{d < i} J_d[i/d]) \)

and we need to prove
1. $p_0 + \ldots + p_{i-1} \leq p$.

It suffices to prove that
$$\sum_{0 \leq i < l} \sum_{x_j \in b \in J_i} \tau_j(i) \in \text{dom}(\Omega(i)) \sum_{0 \leq k < J_i} f x_j (k + \sum_{d < i} J_i[d/i]) \leq p$$

We know that $\text{dom}(\sum_{0 \leq i < l} \Omega(i)) = \text{dom}(\Omega)$ and from (SM0) we get the desired.

2. $\forall 0 \leq i < I, (p_i, d_i) \in [\Omega[i/a]]$.

This means given some $0 \leq i < I$, from Definition 59 it suffices to prove that
$$\exists f : \text{Vars} \to \text{Indices} \to \text{Paths}.$$\(\forall (x_j : b \in J_i) \tau_j(i) \in \Omega[i/a], \forall 0 \leq k < J_i(i). (f', x_j, k, \delta(x_j)) \in [\tau_j(i)[k/b]] \land \sum_{x_j : b \in J(i)} (\sum_{0 \leq j < J(i)} f' x k) \leq p_i \) We choose $f'$ s.t.
$$\forall x_j : b \in J(i) \tau_j(i) \in \Omega[i/a], \forall 0 \leq k < J_i(i). f' x_j k = f x_j (k + \sum_{d < i} J[d/i]),$$

And we need to prove:
(a) $\forall (x_j : b \in J(i) \tau_j(i)) \in \Omega[i/a], 0 \leq k < J_i(i). (f' x_j k, \delta(x_j)) \in [\tau_j(i)[k/b]]$.

This means given some $(x_j : b \in J(i) \tau_j(i)) \in \Omega[i/a]$ and some $0 \leq k < J_i(i)$ and it suffices to prove that
$$(f' x_j k, \delta(x_j)) \in [\tau_j(i)[k/b]]$$

This means we need to prove that
$$(f x_j (k + \sum_{d < i} J_i[d/i]), \delta(x_j)) \in [\tau_j(i)[k/b]]$$

Instantiating (SM0) with the given $x_j$ and $(k + \sum_{d < i} J_i[d/i])$ we get
$$f x_j (k + \sum_{d < i} J_i[d/i]), \delta(x_j)) \in [\sigma([k + \sum_{d < i} J_i[d/i]])]$$

And from Definition 54 we get the desired.

(b) $\sum_{x_j : b \in J(i) \tau_j(i) \in \Omega[i/a]} \sum_{0 \leq k < J_i(i)} f' x k \leq p_i$:

It suffices to prove that
$$\sum_{x_j : b \in J(i) \tau_j(i) \in \Omega[i/a]} \sum_{0 \leq k < J_i(i)} f x (k + \sum_{d < i} J_i[d/i]) \leq p_i$$

Since we know that $p_i$ is $\sum_{x_j : b \in J(i) \tau_j(i) \in \Omega[i/a]} \sum_{0 \leq k < J_i(i)} f x (k + \sum_{d < i} J_i[d/i])$ therefore we are done.

\[ \square \]

**Theorem 64** (Fundamental theorem). $\forall \Psi, \Theta, \Delta, \Omega, e, \tau \in \text{Type}.$

$$\Psi : \Theta : \Delta : \Omega \vdash e : \tau \land (p_i, T, \gamma) \in [\Gamma \sigma_i [\epsilon] \land (p_m, T, \delta) \in [\Omega \sigma_i [\epsilon] \land \vdash \Delta \tau \implies (p_i + p_m, T, e, \gamma \delta) \in [\tau \sigma_i [\epsilon].$$

**Proof.** Proof by induction on the typing judgment

1. T-var1:

$$\Psi : \Theta : \Delta : \Omega, x : \tau \vdash x : \tau$$

Given: $(p_i, T, \gamma) \in [\Gamma, x : \tau \sigma_i [\epsilon]$ and $(p_m, T, \delta) \in [\Omega, \sigma_i [\epsilon]$ To prove: $(p_i + p_m, T, x, \delta \gamma) \in [\tau \sigma_i [\epsilon]$ Since we are given that $(p_i, T, \gamma) \in [\Gamma, x : \tau \sigma_i [\epsilon]$ therefore from Definition 59 we know that
$$\exists f, (f(x), T, \gamma(x)) \in [\tau \sigma_i [\epsilon] where f(x) \leq p_i$$

Therefore from Lemma 62 we get $(p_i + p_m, T, x, \delta \gamma) \in [\tau \sigma_i [\epsilon]$.

2. T-var2:

$$\Theta, \Delta \vdash I \geq 1$$

$$\Psi : \Theta : \Delta : \Omega, x : \tau \vdash x : \tau [0/a]$$

Given: $(p_i, T, \gamma) \in [\Gamma, \sigma_i [\epsilon]$ and $(p_m, T, \delta) \in [\Omega, x : \tau \sigma_i [\epsilon]$ To prove: $(p_i + p_m, x, \delta \gamma) \in [\tau[0/a] [\epsilon]$$ Since we are given that $(p_i, T, \delta) \in [\Omega, x : \tau [0/a] \sigma_i [\epsilon]$ therefore from Definition 59 we know that
$$\exists f : \text{Vars} \to \text{Indices} \to \text{Paths}.$$\((f x 0, T, \delta(x)) \in [\tau[0/a] [\sigma_i [\epsilon]) \) where $(f x 0) \leq p_m$

Therefore from Lemma 62 we get $(p_i + p_m, T, x, \delta \gamma) \in [\tau[0/a] [\sigma_i [\epsilon]$.

3. T-unit:

$$\Psi : \Theta : \Delta : \Omega \vdash () : 1$$

T-unit
5. T-nil:

Given: \((p_l, T, \gamma) \in [\Gamma \sigma_i]_\xi, \,(p_m, T, \delta) \in [\Omega \sigma_i]_\xi\) and \(\models \Delta \iota\)

To prove: \((p_l + p_m, T, () \delta \gamma) \in [\mathbf{1} \sigma_i]_\xi\)

From Definition [58] it suffices to prove that
\[ \forall T' \prec T, (.) \mathcal{V}_T (.) \Rightarrow (p_l + p_m, T - T', ()) \in [\mathbf{1}] \]

This means given \(() \mathcal{V}_0 ()\) it suffices to prove that
\(\exists \gamma (p_l + p_m, T, ()) \in [\mathbf{1}]\)

We get this directly from Definition [58]

4. T-base:

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash c : b \quad \text{T-base} \]

Given: \((p_l, T, \gamma) \in [\Gamma, \sigma_i]_\xi, \,(p_m, T, \delta) \in [\Omega, \sigma_i]_\xi\) and \(\models \Delta \iota\)

To prove: \((p_l + p_m, T, c) \in [b]_\xi\)

From Definition [58] it suffices to prove that
\[ \forall v, T' \prec T, c \mathcal{V}_T v \Rightarrow (p_l + p_m, T - T', c) \in [b] \]

This means given some \(v, T' \prec T\) s.t \(c \mathcal{V}_T v\). Also from E-val we know that \(T' = 0\) therefore it suffices to prove that
\((p_l + p_m, T, v) \in [b] \)

From (E-val) we know that \(v = c\) therefore it suffices to prove that
\((p_l + p_m, T, c) \in [b] \)

We get this directly from Definition [58]

5. T-nil:

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \mathbf{nil} : L^0 \tau \quad \text{T-nil} \]

Given: \((p_l, T, \gamma) \in [\Gamma, \sigma_i]_\xi, \,(p_m, T, \delta) \in [\Omega, \sigma_i]_\xi\)

To prove: \((p_l + p_m, T, \mathbf{nil} \delta \gamma) \in [L^0 \tau \sigma_i]_\xi\)

From Definition [15] it suffices to prove that
\[ \forall T' \prec T, \mathbf{v}, \mathbf{nil} \mathcal{V}_T \mathbf{v} \Rightarrow (p_l + p_m, T - T', \mathbf{v}) \in [L^0 \tau \sigma_i] \]

This means given some \(T' \prec T, \mathbf{v}'\) s.t \(\mathbf{nil} \mathcal{V}_T \mathbf{v}'\) it suffices to prove that
\((p_l + p_m, T - T', \mathbf{v}) \in [L^0 \tau \sigma_i] \)

From (E-val) we know that \(T' = 0\) and \(\mathbf{v}' = \mathbf{nil}\), therefore it suffices to prove that
\((p_l + p_m, T, \mathbf{nil}) \in [L^0 \tau \sigma_i] \)

We get this directly from Definition [15]

6. T-cons:

\[ \Psi; \Theta; \Delta; \Omega_1; \Gamma_1 \vdash e_1 : \tau \quad \Psi; \Theta; \Delta; \Omega_2; \Gamma_2 \vdash e_2 : L^n \tau \quad \Theta \vdash n : \mathbb{N} \quad \text{T-cons} \]

Given: \((p_l, T, \gamma) \in [(\Gamma_1 \odot \Gamma_2) \sigma_i]_\xi, \,(p_m, T, \delta) \in [(\Omega) \sigma_i]_\xi\)

To prove: \((p_l + p_m, T, (e_1 : e_2) \delta \gamma) \in [L^{n+1} \tau \sigma_i]_\xi\)

From Definition [58] it suffices to prove that
\[ \forall e', t < T, (e_1 : e_2) \delta \gamma \mathcal{V}_t e' \Rightarrow (p_l + p_m, T - t, e') \in [L^{n+1} \tau \sigma_i] \]

This means given some \(e', t < T\) s.t \((e_1 : e_2) \delta \gamma \mathcal{V}_t e'\), it suffices to prove that
\((p_l + p_m, T - t, e') \in [L^{n+1} \tau \sigma_i] \)

From (E-cons) we know that \(\exists v_f, l. v' = v_f :: l\)

Therefore from Definition [58] it suffices to prove that
\[\exists p_1, p_2, p_1 + p_2 \leq p_l + p_m \land (p_1, T - t - l) \in [\tau \sigma_i] \land (p_2, T - t - l) \in [\tau \sigma_i] \quad \text{(F-C0)}\]

From Definition [59] and Definition [61] we know that \(\exists p_{l_1}, p_{l_2}, p_{l_1} + p_{l_2} = p_l s.t \)
\((p_{l_1}, T, \gamma) \in [(\Gamma_1) \sigma_i]_\xi\) and \((p_{l_2}, T, \gamma) \in [(\Gamma_2) \sigma_i]_\xi\)

Similarly from Definition [59] and Definition [62] we also know that
\(\exists p_{m_1}, p_{m_2}, p_{m_1} + p_{m_2} = p_m s.t \)
\((p_{m_1}, T, \delta) \in [(\Omega_1) \sigma_i]_\xi\) and \((p_{m_2}, T, \delta) \in [(\Omega_2) \sigma_i]_\xi\)

\[ \text{H1:} \]
(p₁ + p₁₉, T, e₁ δγ) ∈ [τ σ₁]ₑ

Therefore from Definition 58 we have
∀₁ < T, e₁ δγ ⊦₁ vₙf → (p₁ + p₁₉, T - t₁, vₙf) ∈ [τ]

Since we are given that (e₁ :: e₂) δγ ⊦₁ vₙf :: l therefore from E-cons we also know that ∃₁l < t. e₁ δγ ⊦₁ vₙf

Therefore we have (p₁ + p₁₉, T - t₁, vₙf) ∈ [τ σ₁]ₑ (F-C1)

IH2:
(p₂ + p₂₉, T, e₂ δγ) ∈ [Lₙ τ σ₂]ₑ

Therefore from Definition 58 we have
∀₂ < T. e₂ δγ ⊦₂ l → (p₂ + p₂₉, T - t₂, l) ∈ [Lₙ τ σ₂]

Since we are given that (e₁ :: e₂) δγ ⊦₂ vₙf :: l therefore from E-cons we also know that ∃₂l < t - t₁. e₂ δγ ⊦₂ l

Since t₂ < t - t₁ < t < T, therefore we have
(p₂ + p₂₉, T - t₂, l) ∈ [Lₙ τ σ₂]ₑ (F-C2)

In order to prove (F-C0) we choose p₁ as p₁ + p₁₉ and p₂ as p₂ + p₂₉, we get the desired from (F-C1), (F-C2) and Lemma 61.

7. T-match:

\[
\begin{array}{c}
\Psi; \Theta; \Delta; \Omega_1; \Gamma_1 :: e :: L \tau \\
\Psi; \Theta; \Delta; n = 1; \Omega_2; \Gamma_2 :: t \rightarrow e :: \tau' \\
\Psi; \Theta; \Delta; \Gamma, \Delta :: e \rightarrow e :: \tau \\
\Psi; \Theta; \Delta; \Gamma :: e :: \tau' \rightarrow \tau \\
\end{array}
\]

T-match

\[
\begin{array}{c}
\Psi; \Theta; \Delta; \Omega_1; \Omega_2; \Gamma_1; \Gamma_2 :: e :: L \tau \\
\Psi; \Theta; \Delta; n = 0; \Omega_2; \Gamma_2 :: e :: \tau' \\
\Psi; \Theta; \Delta :: t :: \tau \\
\Psi; \Theta; \Delta :: \tau \rightarrow \tau :: K \\
\end{array}
\]

Given: (p₁₉, T, γ) ∈ [[(Γ₀) δγ)]ₑ, (p₂, T, δ) ∈ [[Ω₂ σ₂]ₑ

To prove: (p₁ + p₁₉, T, (match e with |nil → e₁ | h :: t → e₂) δγ) ∈ [[τ σ₁]ₑ

From Definition 58 it suffices to prove that
∀₁ < T, vₙf, (match e with |nil → e₁ | h :: t → e₂) δγ ⊦₁ vₙf → (p₁ + p₁₉, T - t₁, vₙf) ∈ [τ σ₁]

This means given some t < T, vₙf s.t (match e with |nil → e₁ | h :: t → e₂) δγ ⊦₁ vₙf it suffices to prove that
(p₁ + p₁₉, T - t₁, vₙf) ∈ [τ σ₁]ₑ (F-M0)

From Definition 58 and Definition 58 we know that ∃p₁, p₁₉, p₁ + p₁₉ = p₁ s.t
(p₁, T, γ) ∈ [[(Γ₀) δγ)]ₑ and (p₂, T, δ) ∈ [[Ω₂ σ₂]ₑ

Similarly from Definition 58 and Definition 58 we also know that
∃p₂, p₂₉, p₂ + p₂₉ = p₂ s.t
(p₂, T, δ) ∈ [[Ω₂ σ₂]ₑ and (p₁₉, T, δ) ∈ [[Ω₁ σ₁]ₑ

IH1
(p₁ + p₁₉, T, e δγ) ∈ [Lₙ τ σ₁]ₑ

This means from Definition 58 we have
∀₁ < T. e δγ ⊦₁ v₁ → (p₁ + p₁₉, T - t₁, v₁) ∈ [Lₙ τ σ₁]

Since we know that (match e with |nil → e₁ | h :: t → e₂) δγ ⊦₁ vₙf therefore from E-match we know that
∃₁l < t. e₁ δγ ⊦₁ v₁

Since t < T, therefore we have (p₁ + p₁₉, T - t₁, v₁) ∈ [Lₙ τ σ₁]

2 cases arise:
(a) v₁ = nil:

In this case we know that n = 0 therefore

IH2
(p₂ + p₂₉, T, e₂ δγ) ∈ [τ σ₂]ₑ

This means from Definition 58 we have
∀₂ < T. e₂ δγ ⊦₂ v₂ → (p₂ + p₂₉, T - t₂, v₂) ∈ [τ σ₂]

Since we know that (match e with |nil → e₁ | h :: t → e₂) δγ ⊦₂ v₂ therefore from E-match we know that
∃₂l < t. e₂ δγ ⊦₂ v₂

Since t₁ < t < T therefore we have
(p₂ + p₂₉, T - t₂, v₂) ∈ [τ σ₂]

And from Lemma 61 we get
(p₂ + p₂₉, p₁ + p₁₉, T - t₁, v₂) ∈ [τ σ₁]ₑ

And finally since p₁ = p₁ + p₁₉ and p₂ = p₁ + p₂₉ therefore we get
(p₁ + p₁₉, T - t₁, v₂) ∈ [τ σ₁]ₑ

And we are done.
(b) $v_1 = v :: t$

In this case we know that $n > 0$ therefore

**IH**

$$(p_2 + p_{m2} + p_1 + p_{m1}, T, e_2 \delta \gamma) \in [\tau', \sigma']_\varepsilon$$

where

$$\gamma' = \gamma \cup \{ h \mapsto v \} \cup \{ t \mapsto t \}$$

and

$$\nu' = \nu \cup \{ I \mapsto n - 1 \}$$

This means from Definition 58 we have

$$\forall t < T . e_2 \delta \gamma' \downarrow_{t_2} v_f \implies (p_2 + p_{m2} + p_1 + p_{m1}, T - t_2, v_f) \in [\tau', \sigma']$$

Since we know that $(e \text{ with } |n| \mapsto e_1 | h :: t \mapsto e_2) \delta \gamma' \downarrow_{t} v_f$ therefore from E-match we know that $\exists t_2 < t . e_2 \delta \gamma' \downarrow_{t} v_f$

Since $t_2 < t < T$ therefore we have

$$(p_2 + p_{m2} + p_1 + p_{m1}, T - t_2, v_f) \in [\tau', \sigma']$$

From Lemma 61 we get

$$(p_2 + p_{m2} + p_1 + p_{m1}, T - t, v_f) \in [\tau', \sigma']_\varepsilon$$

And finally since we have $\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \tau' : K$ therefore we also have

$$(p_1 + p_{m1}, T - t, v_f) \in [\tau', \sigma']_\varepsilon$$

And we are done

8. T-existI:

$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau[n/s] \quad \Theta \vdash n : S}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \exists s : S. \tau}$$

**T-existI**

Given: $(p_1, T, \gamma) \in [\Gamma \sigma]_\varepsilon$, $(p_1, \gamma, \delta) \in [\Omega \sigma]_\varepsilon$

To prove: $(p_1 + p_{m1}, T, e \delta \gamma) \in [\exists s. \tau \sigma]_\varepsilon$

From Definition 58 it suffices to prove that

$$\forall t < T . \forall e. e \delta \gamma \downarrow_{t} v_f \implies (p_1 + p_{m1}, T - t, v_f \delta \gamma) \in [\exists s. \tau \sigma]$$

This means given some $t < T, v_f$ s.t $e \delta \gamma \downarrow_{t} v_f$ it suffices to prove that

$$(p_1 + p_{m1}, T - t, v_f) \in [\exists s. \tau \sigma]$$

From Definition 58 it suffices to prove that

$$\exists s'(p_1 + p_{m1}, T - t, v_f) \in [\tau[n/s] \sigma]$$

**IH**: $(p_1 + p_{m1}, T, e \delta \gamma) \in [\tau[n/s] \sigma]_\varepsilon$

This means from Definition 58 we have

$$\forall t' < T . e \delta \gamma \downarrow_{t'} v_f \implies (p_1 + p_{m1}, T - t', v_f) \in [\tau[n/s] \sigma]$$

Since we are given that $e \delta \gamma \downarrow_{t} v_f$ therefore we get

$$(p_1 + p_{m1}, T - t, v_f) \in [\tau[n/s] \sigma]$$

(F-E0)

To prove (F-E0) we choose $s'$ as $n$ and we get the desired from (F-E1)

9. T-existE:

$$\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : \exists s. \tau \quad \Psi; \Theta, s; \Delta; \Omega_2; \Gamma_2, x : \tau \vdash e' : \tau' \quad \Psi; \Theta; \Delta; \tau' : K}{\Psi; \Theta; \Delta; \Omega_1 \oplus \Omega_2; \Gamma_1 \oplus \Gamma_2 \vdash e; x. e' : \tau'}$$

**T-existE**

Given: $(p_1, T, \gamma) \in [(\Gamma_1 + \Gamma_2) \sigma]_\varepsilon$, $(p_1, \gamma, \delta) \in [(\Omega) \sigma]_\varepsilon$

To prove: $(p_1 + p_{m1}, T, (e; x.e') \delta \gamma) \in [\tau' \sigma]_\varepsilon$

From Definition 58 it suffices to prove that

$$\forall t < T . (e; x.e') \delta \gamma \downarrow_{t} v_f \implies (p_1 + p_{m1}, T - t, v_f) \in [\tau' \sigma]$$

This means given some $t < T, v_f$ s.t $(e; x.e') \delta \gamma \downarrow_{t} v_f$ it suffices to prove that

$$(p_1 + p_{m1}, T - t, v_f) \in [\tau' \sigma]$$

(F-E0)

From Definition 59 and Definition 57 we know that $\exists p_{m1}, p_1, p_{m2}, p_2 = p_1$ s.t

$$(p_1, T, \gamma) \in [(\Gamma_1) \sigma]_\varepsilon$$

and $(p_2, T, \gamma) \in [(\Gamma_2) \sigma]_\varepsilon$

Similarly from Definition 59 and Definition 56 we also know that

$$\exists p_{m1}, p_{m2}, p_{m1} + p_{m2} = p_m \text{ s.t } (p_m, T, \delta) \in [(\Omega) \sigma]_\varepsilon$$

and $(p_{m2}, T, \delta) \in [(\Omega) \sigma]_\varepsilon$
IH1

\((p_{n1} + p_{m1}, T, e \delta \gamma) \in [\exists s. \tau \sigma_1]_E\)

This means from Definition 58 we have
\(\forall t < T. e \delta \gamma \downarrow t, v_1 \Rightarrow (p_{n1}, T - t, v_1) \in [\exists s. \tau \sigma_1]_E\)

Since we know that \((e; x.e') \delta \gamma \downarrow v_f\) therefore from E-existE we know that \(\exists l < t, v_1.e \delta \gamma \downarrow v_1\). Therefore we have
\((p_{n1} + p_{m1}, T - t, v_1) \in [\exists s. \tau \sigma_1]\)

Therefore from Definition 58 we have
\([\exists \gamma'. (p_{n1} + p_{m1}, T - t, v_1) \in [\tau' / s] \sigma_t]\) (F-EE1)

IH2

\((p_{n2} + p_{m2} + p_{n1} + p_{m1}; T, e' \delta' \gamma) \in [\tau' \sigma']_E\)

where \(\delta' = \delta \cup \{x \mapsto e_1\}\) and \(\ell' = \ell \cup \{s \mapsto s'\}\)

This means from Definition 58 we have
\(\forall t < T. e' \delta' \gamma \downarrow t, v_f' \Rightarrow (p_{n2} + p_{m2} + p_{n1} + p_{m1}, T - t, v_f') \in [\tau' \sigma']\)

Since we know that \((e; x.e') \delta \gamma \downarrow v_f\) therefore from E-existE we know that \(\exists l < t. e' \delta' \gamma \downarrow v_f\).

Since \(t_2 < t < T\) therefore we have
\((p_{n2} + p_{m2} + p_{n1} + p_{m1}, T - t, v_f') \in [\tau' \sigma']\)

Since \(p_1 = p_{n1} + p_{n2}\) and \(p_m = p_{m1} + p_{m2}\) therefore we get
\((p_1 + p_m, T - t, v_f') \in [\tau' \sigma']\)

And finally from Lemma 61 and since we have \(\Psi; \Theta; \Delta \vdash \tau': K\) therefore we also have
\((p_1 + p_m, T - t, v_f') \in [\tau' \sigma']_E\)

And we are done.

10. T-lam:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma, x : \tau_1 \vdash e : \tau_2}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \lambda x. e : (\tau_1 \rightarrow \tau_2)} \text{T-lam}
\]

Given: \((p_1, T, \gamma) \in [\Gamma, \sigma_1]_E, (p_m, T, \delta) \in [\Omega \sigma_t]_E\) and \(\vdash \Delta \iota\)

To prove: \((p_1 + p_m, T, (\lambda x.e) \delta \gamma) \in [(\tau_1 \rightarrow \tau_2) \sigma_t]_E\)

From Definition 58 it suffices to prove that
\(\forall t < T, v_f. (\lambda x.e) \delta \gamma \downarrow t, v_f \Rightarrow (p_1 + p_m, T - t, v_f) \in [(\tau_1 \rightarrow \tau_2) \sigma_t]_E\)

This means given some \(t < T, v_f\) s.t. \((\lambda x.e) \delta \gamma \downarrow t, v_f\). From E-val we know that \(t = 0\) and \(v_f = (\lambda x.e) \delta \gamma\).

Therefore we have
\((p_1 + p_m, T, (\lambda x.e) \delta \gamma) \in [(\tau_1 \rightarrow \tau_2) \sigma_t]_E\)

From Definition 58 it suffices to prove that
\(\forall p', e', T' < T. (p_1 + p_m + p', T', e'[e'/x]) \in [\tau_2 \sigma_t]_E\)

This means given some \(p', e', T' < T\) s.t. \((p', T', e'[e'/x]) \in [\tau_1 \sigma_t]_E\) it suffices to prove that
\((p_1 + p_m + p', T', e'[e'/x]) \in [\tau_2 \sigma_t]_E\) (F-L1)

From IH we know that
\((p_1 + p', p_m, T, e \delta \gamma') \in [\tau_2 \sigma_t]_E\)

where \(\gamma' = \gamma \cup \{x \mapsto e'\}\)

Therefore from Lemma 62 we get the desired

11. T-app:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : (\tau_1 \rightarrow \tau_2) \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2 \vdash e_2 : \tau_1}{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash \tau_1 \oplus \Omega_2; \Gamma_1 \oplus \Gamma_2 \vdash e_1 \Theta_2 : \tau_2} \text{T-app}
\]

Given: \((p_1, T, \gamma) \in [(\Gamma_1 \oplus \Gamma_2) \sigma_t]_E, (p_m, \delta) \in [(\Omega_1 \oplus \Omega_2) \sigma_t]_E\) and \(\vdash \Delta \iota\)

To prove: \((p_1 + p_m, T, e_1 e_2 \delta \gamma) \in [\tau_2 \sigma_t]_E\)

From Definition 58 it suffices to prove that
\(\forall t < T, v_f. (e_1 e_2 \delta \gamma) \downarrow t, v_f \Rightarrow (p_m + p_1, T - t, v_f) \in [\tau_2 \sigma_t]_E\)

This means given some \(t < T, v_f\) s.t. \((e_1 e_2 \delta \gamma) \downarrow t, v_f\) it suffices to prove that
(p_m + p_t, T - t, v_f) ∈ [(\tau_2 \sigma_t)] \quad (F-A0)

From Definition[59] and Definition[57] we know that \( \exists p_{i1}, p_{i2}, p_{i1} + p_{i2} = p_t \) s.t (p_{i1}, T, \gamma) ∈ \([\Gamma]_{\sigma t}\) and (p_{i2}, T, \gamma) ∈ \([\Gamma_2]_{\sigma t}\)

Similarly from Definition[59] and Definition[56] we also know that \( \exists p_{m1}, p_{m2}, p_{m1} + p_{m2} = p_m \) s.t (p_{m1}, T, \delta) ∈ \([\Omega_1]_{\sigma t}\) and (p_{m2}, T, \delta) ∈ \([\Omega_2]_{\sigma t}\)

IH1
(p_{i1} + p_{m1}, T, e_1 \delta \gamma) ∈ \([\tau_{1_1} \ldots \tau_2] \sigma t\)

IH2
(p_{i2} + p_{m2}, T - t_1 - 1, e_2 \delta \gamma) ∈ \([\tau_1 \sigma_t]\) \quad (F-A2)

Instantiating (F-A1) with p_{i2} + p_{m2} and e_2 \delta \gamma we get
(p_{i1} + p_{i2} + p_{m2}, T - t_1 - 1, e_2 \delta \gamma / x) ∈ \([\tau_2 \sigma_t]\)

12. T-sub:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \quad \Theta; \Delta \vdash \tau < : \tau'}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau'}
\]

13. T-weaken:

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \quad \Psi; \Theta \vdash \Gamma' < : \Gamma \quad \Psi; \Theta \vdash \Omega' < : \Omega}{\Psi; \Theta; \Delta; \Omega'; \Gamma' \vdash e : \tau}
\]

14. T-tensor:

\[
\frac{\Psi; \Theta; \Delta; \Omega_1; \Gamma_1 \vdash e_1 : \tau_1 \quad \Psi; \Theta; \Delta; \Omega_2; \Gamma_2 \vdash e_2 : \tau_1}{\Psi; \Theta; \Delta; \Omega_1 \oplus \Omega_2; \Gamma_1 \oplus \Gamma_2 \vdash \langle e_1, e_2 \rangle : \langle \tau_1 \oplus \tau_2 \rangle}
\]
From Definition [58] it suffices to prove that
$$\forall t . \langle e_1, e_2 \rangle \Delta \psi_t \langle v_1, v_2 \rangle \Rightarrow (p_1 + p_m, T - t, \langle v_1, v_2 \rangle) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$

This means given some $t < T$ s.t. $\langle e_1, e_2 \rangle \Delta \psi_t \langle v_1, v_2 \rangle$ it suffices to prove that
$$(p_1 + p_m, T - t, \langle v_1, v_2 \rangle) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$  
(F-TI0)

From Definition [59] and Definition [57] we know that $\exists p_{11}, p_{12}, p_{11} + p_{12} = p_1$ s.t
$$(p_{11}, T, \gamma) \in \left[ (\Gamma_1) \sigma \rangle \right] \text{ and } (p_{12}, T, \gamma) \in \left[ (\Omega_1) \sigma \rangle \right]$$  
Similarly from Definition [59] and Definition [56] we also know that $\exists p_{m1}, p_{m2}, p_{m1} + p_{m2} = p_m$ s.t
$$(p_{m1}, T, \delta) \in \left[ (\Omega_1) \sigma \rangle \right] \text{ and } (p_{m2}, T, \delta) \in \left[ (\Omega_2) \sigma \rangle \right]$$

IH1:
$$(p_1 + p_m, T, e_1 \Delta \gamma) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
Therefore from Definition [58] we have
$$\forall t_1 < T . e_1 \Delta \gamma \psi_{t_1} v_{f_1} \Rightarrow (p_1 + p_m, T - t_1, v_{f_1}) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
Since we are given that $\langle e_1, e_2 \rangle \Delta \gamma \psi_t \langle v_1, v_2 \rangle$ therefore from E-TI we know that $\exists t_1 < T. e_1 \Delta \gamma \psi_{t_1} v_{f_1}$
Hence we have $(p_1 + p_m, T - t_1, v_{f_1}) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$  
(F-TI1)

IH2:
$$(p_{12} + p_{m2}, T, e_2 \Delta \gamma) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
Therefore from Definition [58] we have
$$\forall t_2 < T . e_2 \Delta \gamma \psi_{t_2} v_{f_2} \Rightarrow (p_{12} + p_{m2}, T - t_2, v_{f_2}) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
Since we are given that $\langle e_1, e_2 \rangle \Delta \gamma \psi_t \langle v_1, v_2 \rangle$ therefore from E-TI we also know that $\exists t_2 < T. e_2 \Delta \gamma \psi_{t_2} v_{f_2}$
Hence $t_2 < t < T$ therefore we have
$$(p_{12} + p_{m2}, T - t_2, v_{f_2}) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$  
(F-TI2)

Applying Lemma [61] on (F-TI1) and (F-TI2) and by using Definition [15] we get the desired.

15. T-tensorE:

$$\Psi; \Theta; \Delta; \Omega_1; \Gamma_1 \vdash e : (\tau_1 \otimes \tau_2) \quad \Psi; \Theta; \Delta; \Omega_2; \Gamma_2; x : \tau_1; y : \tau_2 \vdash e' : \tau$$
$$\quad \text{T-tensorE}$$

Given: $(p_1, T, \gamma) \in \left[ (\Gamma_1 \otimes \Gamma_2) \sigma \rangle \right], (p_{m1}, T, \delta) \in \left[ (\Omega_1) \sigma \rangle \right]$
To prove: $(p_1 + p_m, T, \delta) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$

From Definition [15] it suffices to prove that
$$\forall t < T, v_f . (let (\langle x, y \rangle) = e \in e') \Delta \psi_t v_f \Rightarrow (p_1 + p_m, T - t, v_f) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
This means given some $t < T, v_f$ s.t. $(let (\langle x, y \rangle) = e \in e') \Delta \psi_t v_f$ it suffices to prove that
$$(p_1 + p_m, T - t, v_f) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$  
(F-TE0)

From Definition [16] and Definition [13] we know that $\exists p_{11}, p_{12}, p_{11} + p_{12} = p_1$ s.t
$$(p_{11}, T, \gamma) \in \left[ (\Gamma_1) \sigma \rangle \right] \text{ and } (p_{12}, T, \gamma) \in \left[ (\Omega_1) \sigma \rangle \right]$$  
Similarly from Definition [59] and Definition [56] we also know that $\exists p_{m1}, p_{m2}, p_{m1} + p_{m2} = p_m$ s.t
$$(p_{m1}, T, \delta) \in \left[ (\Omega_1) \sigma \rangle \right] \text{ and } (p_{m2}, T, \delta) \in \left[ (\Omega_2) \sigma \rangle \right]$$

IH1:
$$(p_1 + p_m, T, e \Delta \gamma) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
This means from Definition [15] we have
$$\forall t_1 < T . e \Delta \psi_{t_1} \langle v_1, v_2 \rangle \Delta \gamma \Rightarrow (p_1 + p_m, T - t_1, \langle v_1, v_2 \rangle) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
Since we know that $(let (\langle x, y \rangle) = e \in e') \Delta \psi_t \langle v_1, v_2 \rangle$ therefore from E-sub ExpE we know that $\exists t_1 < t, v_1, v_2. e \Delta \psi_{t_1} \langle v_1, v_2 \rangle$. Therefore we have
$$(p_1 + p_m, T - t_1, \langle v_1, v_2 \rangle) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
From Definition [15] we know that
$$\exists p_{11}, p_{12}, p_{12} + p_{12} = p_1 + p_{m1} \land (p_1, T, v_1) \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right] \land (p_2, T, v_2) \in \left[ (\tau_2 \sigma) \right]$$  
(F-TE1)

IH2:
$$(p_{12} + p_{m2} + p_1 + p_{m2}, T, e' \Delta \gamma') \in \left[ (\tau_1 \otimes \tau_2) \sigma \rangle \right]$$
where
$$\gamma' = \gamma \cup \{ x \mapsto v_1 \} \cup \{ y \mapsto v_2 \}$$

110
This means from Definition 15 we have
\(\forall t < T . e' \delta' \Downarrow t_2, v_f \implies (p_{t2} + p_{m2} + p_1 + p_2, T - t_2, v_f) \in [T \sigma I]\)

Since we know that \((\text{let}(x, y) = e \in e') \delta' \Downarrow t_2, v_f\) therefore from E-TE we know that \(\exists t_2 < t, e' \delta' \Downarrow t_2, v_f\).

Therefore we have
\((p_{t2} + p_{m2} + p_1 + p_2, T - t_2, v_f) \in [T \sigma I]\)

From Lemma 61 we get
\((p_1 + p_m, T - t, v_f) \in [T \sigma I]_E\)

And we are done

16. T-withI:

\[\text{T-withI:}\]
\[
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_1 : \tau_1 \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e_2 : \tau_1
\]

\[\text{T-withI:}\]

\[
\forall t, e_1, e_2 \delta \Downarrow t, (v_f_1, v_f_2) \implies (p_1 + p_m, T - t, (v_f_1, v_f_2)) \in [(\tau_1 \& \tau_2) \sigma I]_E
\]

This means given \((e_1, e_2) \delta \Downarrow t, (v_f_1, v_f_2)\) it suffices to prove that
\((p_1 + p_m, T - t, (v_f_1, v_f_2)) \in [(\tau_1 \& \tau_2) \sigma I]_E\) (F-WI0)

IH1:

\((p_1 + p_m, T, e_1, \delta) \in [\tau_1 \sigma I]_E\)

Therefore from Definition 15 we have
\(\forall t, e_1 \delta \Downarrow t, v_f_1 \implies (p_1 + p_m, T - t, v_f_1) \in [\tau_1 \sigma I]_E\)

Since we are given that \((e_1, e_2) \delta \Downarrow t, (v_f_1, v_f_2)\) therefore from E-WI we know that \(\exists t_1 < t, e_1 \delta \Downarrow t_1, v_f_1\)

Since \(t_1 < t < T\), therefore we have
\((p_1 + p_m, T - t_1, v_f_1) \in [\tau_1 \sigma I]_E\) (F-WI1)

IH2:

\((p_1 + p_m, T, e_2, \delta) \in [\tau_2 \sigma I]_E\)

Therefore from Definition 15 we have
\(\forall t, e_2 \delta \Downarrow t_2, v_f_2 \implies (p_1 + p_m, T - t_2, v_f_2) \in [\tau_2 \sigma I]_E\)

Since we are given that \((e_1, e_2) \delta \Downarrow t, (v_f_1, v_f_2)\) therefore from E-WI we also know that \(\exists t_2 < t, e_2 \delta \Downarrow t_2, v_f_2\)

Since \(t_2 < t < T\), therefore we have
\((p_1 + p_m, T - t_2, v_f_2) \in [\tau_2 \sigma I]_E\) (F-WI2)

Applying Lemma 61 on (F-WI1) and (F-WI2) we get the desired.

17. T-fst:

\[\text{T-fst:}\]
\[
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (\tau_1 \& \tau_2) \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{fst}(e) : \tau_1
\]

Given: \((p_1, T, \gamma) \in [(\Gamma) \sigma I]_E, (0, T, \delta) \in [\Omega \sigma I]_E\)

To prove: \((p_1 + p_m, T, (\text{fst}(e)) \delta \gamma) \in [\tau_1 \sigma I]_E\)

From Definition 15 it suffices to prove that
\(\forall t, v_f, (\text{fst}(e)) \delta \Downarrow t, v_f \implies (p_1 + p_m, T - t, v_f) \in [\tau_1 \sigma I]_E\)

This means given some \(t < T, v_f\) s.t. \((\text{fst}(e)) \delta \Downarrow t, v_f\) it suffices to prove that
\((p_1 + p_m, T - t, v_f) \in [\tau_1 \sigma I]_E\) (F-F0)

IH

\((p_1 + p_m, T, e, \delta) \in [(\tau_1 \& \tau_2) \sigma I]_E\)

This means from Definition 15 we have
\(\forall t, e \delta \Downarrow t, (v_1, v_2) \delta \gamma \implies (p_1 + p_m, T - t_1, (v_1, v_2)) \in [(\tau_1 \& \tau_2) \sigma I]_E\)

Since we know that \((\text{fst}(e)) \delta \Downarrow t, v_f\) therefore from E-fst we know that \(\exists t_1 < t, v_1, v_2, e \delta \Downarrow t_1, (v_1, v_2)\).

Since \(t_1 < t < T\), therefore we have
\((p_1 + p_m, T - t_1, (v_1, v_2)) \in [(\tau_1 \& \tau_2) \sigma I]_E\)

From Definition 15 we know that
\[(pt + pm, T - t_1, v_1) \in [\tau_1 \sigma t] \]

Finally using Lemma 61 we also have
\[(pt + pm, T - t, v_1) \in [\tau_1 \sigma t] \]

Since from E-fst we know that \(v_f = v_1\), therefore we are done.

18. T-snd:
Similar reasoning as in T-fst case above.

19. T-inl:
\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau_1 & \quad \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{inl}(e) : \tau_1 + \tau_2 \\
\end{align*}
\]

Given: \((p_t, T, \gamma) \in [\Gamma \sigma t]_\times, (0, T, \delta) \in [\Omega \sigma t]_\times\)
To prove: \((pt + pm, T, \text{inl}(e) \delta \gamma) \in [\tau_1 + \tau_2] \sigma t\)

From Definition 15 it suffices to prove that
\[\forall t < T. \text{inl}(e) \delta \gamma \downarrow \text{inl}(v) \implies (pt + pm, T - t, \text{inl}(v)) \in [\tau_1 \tau_2] \sigma t\]

This means given some \(t < T\) s.t. \(\text{inl}(e) \delta \gamma \downarrow \text{inl}(v)\) it suffices to prove that
\[(pt + pm, T - t, \text{inl}(v)) \in [\tau_1 \tau_2] \sigma t\]  \((\text{F-IL0})\)

19.1. \(\text{IH}\):

\[(p_t + pm, T, e_1 \delta \gamma) \in [\tau_1 \sigma t]_\times\]

Therefore from Definition 15 we have
\[\forall t_1 < T. e_1 \delta \gamma \downarrow t_1, v_{f_1} \implies (p_t + pm, T - t_1, v_{f_1}) \in [\tau_1 \sigma t]\]

Since we are given that \(\text{inl}(e) \delta \gamma \downarrow t \text{inl}(v)\) therefore from E-inl we know that \(\exists t_1 < t.e \delta \gamma \downarrow t_1, v\)

Hence we have \((p_t + pm, T - t_1, v) \in [\tau_1 \sigma t]\)

From Lemma 61 we get \((pt + pm, T - t, v) \in [\tau_1 \sigma t]\)

And finally from Definition 15 we get \((\text{F-IL0})\)

20. T-inr:
Similar reasoning as in T-inr case above.

21. T-case:
\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : (\tau_1 + \tau_2) & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2, x : \tau_1 \vdash e_1 : \tau \\
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash \text{case } e, x, e_1, y, e_2 : \tau & \quad \Psi; \Theta; \Delta; \Omega; \Gamma_2, y : \tau_2 \vdash e_2 : \tau \\
\end{align*}
\]

Given: \((p_t, T, \gamma) \in [(\Gamma_1 + \Gamma_2) \sigma t]_\times, (0, T, \delta) \in [\Omega \sigma t]_\times\)
To prove: \((pt + pm, T, \text{case } e, x, e_1, y, e_2, \delta \gamma) \in [\tau \sigma t]_\times\)

From Definition 15 it suffices to prove that
\[\forall t < T, v_f. (\text{case } e, x, e_1, y, e_2) \delta \gamma \downarrow t, v_f \implies (pt + pm, T - t, v_f) \in [\tau \sigma t]\]

This means given some \(t < T, v_f\) s.t. \(\text{case } e, x, e_1, y, e_2, \delta \gamma \downarrow t, v_f\) it suffices to prove that
\[(pt + pm, T - t, v_f) \in [\tau \sigma t]\]  \((\text{F-C0})\)

From Definition 14 and Definition 13 we know that \(\exists p_{11}, p_{12}, p_{p1} + p_{p2} = p_t\) s.t.
\[(p_{11}, T, \gamma) \in [(\Gamma_1) \sigma t]_\times\] and \((p_{12}, T, \gamma) \in [(\Gamma_2) \sigma t]_\times\)

Similarly from Definition 59 and Definition 50 we also know that \(\exists p_{m1}, p_{m2}, p_{m1} + p_{m2} = pm\) s.t.
\[(p_{m1}, T, \delta) \in [(\Omega_1) \sigma t]_\times\] and \((p_{m2}, T, \delta) \in [(\Omega_2) \sigma t]_\times\)

1. \(\text{IH1}\):

\[(p_{11} + pm_1, T, e \delta \gamma) \in [(\tau_1 + \tau_2) \sigma t]_\times\]

This means from Definition 15 we have
\[\forall t' < T. e \delta \gamma \downarrow t, v_1 \delta \gamma \implies (p_{11} + pm_1, T - t', v_1) \in [(\tau_1 + \tau_2) \sigma t]\]

Since we know that \(\text{case } e, x, e_1, y, e_2 \delta \gamma \downarrow t, v_f\) therefore from E-case we know that \(\exists t' < t, v_1, e \delta \gamma \downarrow t', v_1\).

Since \(t' < t < T\), therefore we have
\[(p_{11} + pm_1, T - t', v_1) \in [(\tau_1 + \tau_2) \sigma t]\]

2 cases arise:
(a) \(v_1 = \text{inl}(v)\):

\[(pt + pm_{2p} + pm_1, T - t', e_1 \delta \gamma') \in [\tau \sigma t]_\times\]

where
\[ \gamma' = \gamma \cup \{ x \mapsto v \} \]

This means from Definition 15 we have
\[ \forall t_1 < T \cdot t.e_1 \delta \gamma' \downarrow_{v_f} \iff (p_{t_2} + p_{m_2} + p_{t_1} + p_{m_1}, T - t' - t_1, v_f) \in [\tau \sigma_t] \]

Since we know that (case e, x,e_1, y,e_2) \( \delta \gamma' \downarrow_{v_f} \) therefore from E-case we know that \( \exists t_1 \cdot e_1 \delta \gamma' \downarrow_{v_f} \)

where \( t_1 = t - t' - 1 \).

Since \( t_1 = t - t' - 1 < T - t' \) therefore we have \( (p_{t_2} + p_{m_2} + p_{t_1} + p_{m_1}, T - t' - t_1, v_f) \in [\tau \sigma_t] \)

From Lemma 61 we get \( (p_{t_2} + p_{m_2} + p_{t_1} + p_{m_1}, T - t, v_f) \in [\tau \sigma_t] \)

And we are done

(b) \( v_1 = \text{inr}(v) \)

Similar reasoning as in the inl case above.

22. T-subExpI:

\[
\frac{\Psi; \Theta, a; \Delta, a < I; \Omega; \vdash e : \tau}{\Psi; \Theta, \Delta; \sum_{a < I} \Omega; \vdash! e : ![a < I] \tau} \quad \text{T-subExpI}
\]

Given: \((p_1, \gamma) \in \llbracket \_ \rrbracket \varepsilon, (p_{m}, \delta) \in \llbracket \sum_{a < I} \Omega \rrbracket \varepsilon \) and \( \models \Delta \iota \)

To prove: \((p_1 + p_m, \varepsilon \sigma \gamma) \in \llbracket ![a < I] \tau \rrbracket \varepsilon \)

From Definition 55 it suffices to prove that
\[ \forall t < T, (\varepsilon \delta \gamma \downarrow t, (\varepsilon \delta \gamma) \in \llbracket ![a < I] \tau \rrbracket \varepsilon \]

This means given some \( t < T \cdot \iota \) s.t. \((\varepsilon \delta \gamma \downarrow t) \delta \gamma \) it suffices to prove that
\((p_1 + p_m, T - t, (\varepsilon \delta \gamma) \in \llbracket ![a < I] \tau \rrbracket \varepsilon \)

From Definition 55 it suffices to prove that
\[ \exists p_0, \ldots, p_{l-1}, p_0 + \ldots + p_{l-1} \leq (p_1 + p_m) \land \forall 0 \leq i < l \cdot (p_1, T, (\varepsilon \delta \gamma) \in \llbracket I[i] \rrbracket \varepsilon) \quad \text{(F-SI0)} \]

Since we know that \((p_i, T, \delta) \in \llbracket \sum_{a < \Omega} \sigma_t \rrbracket \varepsilon \) therefore from Lemma 63 we know that
\[ \exists \gamma_0, \ldots, \gamma_{l-1}, \gamma_0 + \ldots + \gamma_{l-1} \leq p_m \land \forall 0 \leq i < l \cdot (p_1, T, \delta) \in \llbracket \Omega[i] \rrbracket \varepsilon \quad \text{(F-SI1)} \]

Instantiating IH with each \( p_0, \ldots, p_{l-1} \) we get
\( (p_0', T, \varepsilon \delta \gamma) \in \llbracket ![0/a] \sigma_t \rrbracket \varepsilon \) and
\[
\ldots
\]
\( (p_{l-1}', T, \varepsilon \delta \gamma) \in \llbracket ![l-1/a] \sigma_t \rrbracket \varepsilon \quad \text{(F-SI2)} \)

Therefore we get (F-SI0) from (F-SI1) and (F-SI2)

23. T-subExpE:

\[
\frac{\Psi; \Theta; a; \Delta; \Omega_1; \Gamma_1 \vdash e : ![a < I] \tau \quad \Psi; \Theta; a; \Delta; \Omega_2; \Gamma_1 \vdash e' : ![a < I] \tau'}{\Psi; \Theta; a; \Delta; \Omega_1 \otimes \Omega_2; \Gamma_1 \otimes \Gamma_2 \vdash \text{let} \ x = e \ in \ e' : ![a < I] \tau'} \quad \text{T-subExpE}
\]

Given: \((p_1, \gamma) \in \llbracket (\Gamma_1 \oplus \Gamma_2) \sigma_t \rrbracket \varepsilon, (p_m, \delta) \in \llbracket (\Omega_1 \oplus \Omega_2) \sigma_t \rrbracket \varepsilon \) and \( \models \Delta \iota \)

To prove: \((p_1 + p_m, \text{let} \ x = e \ in \ e' \ \delta \gamma) \in \llbracket ![a < I] \sigma_t \rrbracket \varepsilon \)

From Definition 55 it suffices to prove that
\[ \forall t < T, (v_f, (\text{let} \ x = e \ in \ e') \delta \gamma \downarrow v_f \implies (p_1 + p_m, T - t, v_f) \in \llbracket ![a < I] \sigma_t \rrbracket \varepsilon \]

This means given some \( t < T \cdot \iota \) s.t. \((\text{let} \ x = e \ in \ e') \delta \gamma \downarrow v_f \) it suffices to prove that
\((p_1 + p_m, T - t, v_f) \in \llbracket ![a < I] \sigma_t \rrbracket \varepsilon \quad \text{(F-SE0)} \)

From Definition 55 and Definition 57 we know that \( \exists p_1, p_2, p_{m_1} + p_{m_2} = p_m \) s.t
\((p_1, T, \gamma) \in \llbracket (\Gamma_1) \sigma_t \rrbracket \varepsilon \) and \((p_2, T, \gamma) \in \llbracket (\Gamma_2) \sigma_t \rrbracket \varepsilon \)

Similarly from Definition 59 and Definition 56 we also know that \( \exists p_{m_1}, p_{m_2}, p_m, p_{m_2} = p_m \) s.t
\((p_{m_1}, T, \delta) \in \llbracket (\Omega_1) \sigma_t \rrbracket \varepsilon \) and \((p_{m_2}, T, \delta) \in \llbracket (\Omega_2) \sigma_t \rrbracket \varepsilon \)

IH1
\((p_1 + p_m, T, e \delta \gamma) \in \llbracket ![a < I] \sigma_t \rrbracket \varepsilon \)

This means from Definition 55 we have
\[ \forall t_1 < T, e \delta \gamma \downarrow_{v_f}, e \delta \gamma \in \llbracket ![a < I] \sigma_t \rrbracket \varepsilon \]

Since we know that \((\text{let} \ x = e \ in \ e') \delta \gamma \downarrow v_f \) therefore from E-subExpE we know that \( \exists t_1 < t, e_{1} \delta \gamma \downarrow_{v_f}, \ e_{1} \delta \gamma \). Therefore we have

113
(p_1 + p_m, T, t, t_1, e_1, \delta' e) \in [\Gamma; \alpha < \tau; \sigma]$

Therefore from Definition 58 we have
\[ \exists p_0, \ldots, p_{t-1}, p_0 + \ldots + p_{t-1} \leq (p_1 + p_m) \land \forall \delta \in I, (p_i, T, t - t_1, e_1, \delta e) \in [\tau[i/a]; \sigma] \quad \text{(F-SE1)} \]

**IH**

(p_2 + p_m + p_0 + \ldots + p_{t-1}, T, t - t_1, e_1, \delta' e) \in [\tau; \sigma]$

where
\[ \delta' = \delta \cup \{ x \mapsto e_1 \} \]

This means from Definition 58 we have
\[ \forall t_2 < T - t_1, e_1, \delta' e \parallel_v v_{t_2} \quad \text{is} \quad (p_2 + p_m + p_0 + \ldots + p_{t-1}, T - t_1 - t_2, v_f) \in [\tau; \sigma] \]

Since we know that (let \( x = e \) in \( e' \)) \( \delta' e \parallel v \) therefore from E-subExpE we know that \( \exists t_2, e', \delta' e \parallel v_{t_2} \) s.t.
\[ t_2 = t - t_1 - 1. \]

Therefore we have
\[ (p_2 + p_m + p_0 + \ldots + p_{t-1}, T - t_1 - t_2, v_f) \in [\tau; \sigma] \]

Since from \( \text{(F-SE1)} \) we know that \( p_0 + \ldots + p_{t-1} \leq p_1 + p_m \) therefore from Lemma 52 we get
\[ (p_2 + p_m + p_1 + p_m, T - t, v_f) \in [\tau; \sigma] \]

And finally since \( p_1 = p_1 + p_2 \) and \( p_m = p_1 + p_m \) therefore we get
\[ (p_1 + p_m, T - t, v_f) \in [\tau; \sigma] \]

And we are done.

24. T-tabs:

\[ \Psi; \alpha : K; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \]

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda \vdash : (\forall \alpha : K; \tau) \quad \text{T-tabs} \]

Given:
\[ (p_1, T, \tau) \in [\Gamma; \alpha], (p_m, T, \sigma) \in [\Omega; \sigma] \text{ and } \vdash \Delta \]

To prove:
\[ (p_1 + p_m, T, \Lambda, \sigma) \in [\forall \alpha : K; \tau] \]

From Definition 58 it suffices to prove that
\[ \forall t < T, e_\Lambda \vdash_t v \implies (p_1 + p_m, T - t, v) \in [\forall \alpha : K; \tau] \]

This means some \( v \) s.t. \( \Lambda, e \vdash \delta \gamma \parallel e \) and from \( \text{E-val} \) we know that \( v = \Lambda, e \vdash \delta \gamma \) and \( t = 0 \) therefore it suffices to prove that
\[ (p_1 + p_m, T, \Lambda, \sigma) \in [\forall \alpha : K; \tau] \]

From Definition 58 it suffices to prove that
\[ \forall \tau', T' < T, (p_1 + p_m, T', e \delta \gamma) \in [\tau; \sigma] \]

This means some \( \tau', T' < T \) it suffices to prove that
\[ (p_1 + p_m, T', e \delta \gamma) \in [\tau; \sigma'] \]

where
\[ \sigma' = \sigma \cup \{ (\alpha \mapsto \tau') \} \]

We get the desired directly from IH

25. T-tapp:

\[ \Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (\forall \alpha : K; \tau) \quad \Psi; \Theta; \Delta \vdash : \Gamma \quad \text{T-tapp} \]

Given:
\[ (p_1, T, \tau) \in [\Gamma; \alpha], (p_m, T, \sigma) \in [\Omega; \sigma] \text{ and } \vdash \Delta \]

To prove:
\[ (p_1 + p_m, T, e \parallel [\tau; \sigma]) \]

From Definition 58 it suffices to prove that
\[ \forall t < T, v, (\tau; \sigma) \parallel_t v, v \implies (p_1 + p_m, T - t, v) \in [\tau; \sigma] \]

This means some \( t < T, v \) s.t. \( (\parallel [\tau; \sigma]) \parallel_t v \) it suffices to prove that
\[ (p_1 + p_m, T - t, v) \in [\tau; \sigma] \]

\[ \text{(F-Tapp)} \]

**IH**

(p_1 + p_m, T, e \delta \gamma) \in [\forall \alpha : K; \tau]$

This means from Definition 58 we have
\[ \forall t_1 < T, e \delta \gamma \parallel_t \Lambda, e \]

Since we know that \( (e \parallel) \delta \gamma \parallel v_f \) therefore from E-tapp we know that \( \exists t_1 < T, e \delta \gamma \parallel_v, \Lambda, e \), therefore we have
\[(p_t + p_m, T - t_1, \Lambda, e) \in [(\forall \alpha, \tau) \; \sigma_\epsilon]\]

Therefore from Definition 55 we have
\[\forall \tau'', T_1 < T - t_1, (p_t + p_m, T - t_1 - T_1, e, \delta \gamma) \in [\tau[\tau''/\alpha] \; \sigma_\epsilon]_\mathcal{E}\]

Instantiating it with the given \(\tau''\) and \(T - t_1 - 1\) we get
\[(p_t + p_m, T - t_1 - 1, e, \delta \gamma) \in [\tau[\tau''/\alpha] \; \sigma_\epsilon]_\mathcal{E}\]

From Definition 55 we know that
\[\forall t_2 < T - t_1 - 1, v'' = T, e, \delta \gamma \downarrow v, \therefore (p_t + p_m, T - t_1 - 1 - t_2, v'') \in [\tau[\tau''/\alpha] \; \sigma_\epsilon]_\mathcal{E}\]

Since we know that \((e[\tau]) \delta \gamma \downarrow v\) therefore from E-tapp we know that \(\exists t_2, e \downarrow v'\) where \(t_2 = t - t_1 - 1\)

Since \(t_2 = t - t_1 - 1 < T - t_1 - 1\), therefore we have
\[(p_t + p_m, T - t_2, v) \in [\tau[\tau''/\alpha] \; \sigma_\epsilon]_\mathcal{E}\]

And we are done.

26. T-ibs:

\[
\Psi_\Theta, i : S; \Delta; \Omega; \Gamma \vdash e : \tau \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda, e : (\forall i : S, \tau) \quad \text{T-ibs}
\]

Given: \((p_t, T, \gamma) \in [\Gamma, \sigma_\epsilon]_\mathcal{E}, (p_m, T, \delta) \in [\Omega, \sigma_\epsilon]_\mathcal{E}\) and \(\models \Delta \; i\)

To prove: \((p_t + p_m, T, \Lambda, e, \delta \gamma) \in [(\forall i : S. \tau) \; \sigma_\epsilon]_\mathcal{E}\)

From Definition 55 it suffices to prove that
\[\forall t < T, v. \Lambda, e \; \delta \gamma \downarrow v, \therefore (p_m + p_t, T - t, v) \in [(\forall i : S. \tau) \; \sigma_\epsilon]_\mathcal{E}\]

This means given some \(t < T\), \(s.t.\) \(\Lambda, e \; \delta \gamma \downarrow v\) and from (E-val) we know that \(v = \Lambda, e \; \delta \gamma\) and \(t = 0\) therefore it suffices to prove that
\[(p_t + p_m, T, \Lambda, e, \delta \gamma) \in [(\forall i : S. \tau) \; \sigma_\epsilon]_\mathcal{E}\]

From Definition 55 it suffices to prove that
\[\forall I. (p_t + p_m, T, e) \in [\tau[I / i] \; \sigma_\epsilon]_\mathcal{E}\]  

This means given some \(I\) it suffices to prove that
\[(p_t + p_m, T, e) \in [\tau[I / i] \; \sigma_\epsilon]_\mathcal{E} \quad \text{(F-TAb0)}
\]

\[\text{IH} \; (p_t + p_m, T, e \; \delta \gamma) \in [\tau[I'] \; \sigma_\epsilon]_\mathcal{E}\]

where
\[\epsilon' = \epsilon \cup \{i \mapsto I\}\]

We get the desired directly from IH

27. T-iapp:

\[
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (\forall i : S, \tau) \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash i : S \\
\Theta \vdash I : S \\
\quad \text{T-iapp}
\]

Given: \((p_t, T, \gamma) \in [\Gamma, \sigma_\epsilon]_\mathcal{E}, (p_m, T, \delta) \in [\Omega, \sigma_\epsilon]_\mathcal{E}\) and \(\models \Delta \; i\)

To prove: \((p_t + p_m, T, e) \downarrow \delta \gamma \in [\tau[I / i] \; \sigma_\epsilon]_\mathcal{E}\)

From Definition 55 it suffices to prove that
\[\forall t < T, v_f. (e[\tau]) \delta \gamma \downarrow v_f, \therefore (p_m + p_t, T - t, v_f) \in [(\tau[I / i]) \; \sigma_\epsilon]_\mathcal{E}\]

This means given some \(t < T, v_f, s.t.\) \((e[\tau]) \delta \gamma \downarrow v_f\) it suffices to prove that
\[(p_m + p_t, T - t, v_f) \in [(\tau[I / i]) \; \sigma_\epsilon]_\mathcal{E} \quad \text{(F-Iapp0)}
\]

\[\text{IH} \; (p_t + p_m, T, e \; \delta \gamma) \in [(\forall i : S. \tau) \; \sigma_\epsilon]_\mathcal{E}\]

This means from Definition 55 we have
\[\forall t_1 < T, v'. e \; \delta \gamma \downarrow v', \therefore (p_m + p_t, T - t_1, v') \in [(\forall i : S. \tau) \; \sigma_\epsilon]_\mathcal{E}\]

Since we know that \((e[\tau]) \delta \gamma \downarrow v\) therefore from (E-iapp) we know that \(\exists t_1 < T, e \; \delta \gamma \downarrow t_1, \Lambda, e, \therefore\) we have
\[(p_t + p_m, T - t_1, \Lambda, e) \in [(\forall i : S. \tau) \; \sigma_\epsilon]_\mathcal{E}\]

Therefore from Definition 55 we have
\[\forall I' , T_1 < T - t_1, (p_t + p_m, T - t_1 - T_1, e \; \delta \gamma) \in [\tau[I' / i] \; \sigma_\epsilon]_\mathcal{E}\]

Instantiating it with the given \(I\) and \(T - t_1 - 1\) we get
\[(p_t + p_m, T - t_1 - 1, e \; \delta \gamma) \in [\tau[I / i] \; \sigma_\epsilon]_\mathcal{E}\]
From Definition \[58\] we know that
\[\forall v', t_2 < T - t_1 - 1. e \delta t_{v_2} v_f \implies (p_1 + p_m, T - t_1 - 1 - t_2, v'') \in [\tau[I/i] \sigma t]\]
Since we know that \((e [\delta t v_f\])\) therefore from E-iapp we know that \(\exists t_2. e \delta t_{v_2} v_f\) where \(t_2 = t - t_1 - 1\)
Since \(t_2 = t - t_1 - 1 < T - t_1 - 1\), therefore we have
\((p_1 + p_m, v_f) \in [\tau[I/i] \sigma t]\)
And we are done.

28. T-CI:

\[\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \quad \text{T-CI}\]

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (c \Rightarrow \tau)}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda. e : (c \Rightarrow \tau)}
\]

Given: \((p_1, T, \gamma) \in [\Gamma \sigma t]_E, (p_m, T, \delta) \in [\Omega \sigma t]_E\) and \(\vdash \Delta \ i\)
To prove: \((p_1 + p_m, T, \Lambda. e \delta \gamma) \in [(c \Rightarrow \tau) \sigma t]_E\)

From Definition \[58\] it suffices to prove that
\(\forall v, t < T . \Lambda. e \delta \gamma \psi_t v \implies (p_m + p_1, T - t, v) \in [(c \Rightarrow \tau) \sigma t]\)
This means given some \(v, t < T\) s.t. \(\Lambda. e \delta \gamma \psi_t v\) and from (E-val) we know that \(v = \Lambda. e \delta \gamma \) and \(t = 0\) therefore it suffices to prove that
\((p_1 + p_m, T, \Lambda. e \delta \gamma) \in [(c \Rightarrow \tau) \sigma t]\)

From Definition \[58\] it suffices to prove that
\(\forall \forall^* T < T . \vdash e \ i \implies (p_1 + p_m, T', e \delta \gamma) \in [\tau \sigma t]_E\)
This means given some \(\forall^* T < T\) s.t. \(\vdash e \ i\) it suffices to prove that
\((p_1 + p_m, T', e \delta \gamma) \in [\tau \sigma t]_E\)

**IH** \((p_1 + p_m, T', e \delta \gamma) \in [\tau \sigma t]_E\)

We get the desired directly from IH

29. T-CE:

\[\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (c \Rightarrow \tau) \quad \Theta; \Delta \vdash e : \tau \quad \text{T-CE}\]

\[
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (c \Rightarrow \tau)}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \Lambda. e : (c \Rightarrow \tau)}
\]

Given: \((p_1, T, \gamma) \in [\Gamma \sigma t]_E, (p_m, T, \delta) \in [\Omega \sigma t]_E\) and \(\vdash \Delta \ i\)
To prove: \((p_1 + p_m, T, e \delta \gamma) \in [(c \Rightarrow \tau) \sigma t]_E\)

From Definition \[58\] it suffices to prove that
\(\forall v_f, t < T . (e [\delta \gamma] v_f) \implies (p_m + p_1, T - t, v_f) \in [(\tau) \sigma t]\)
This means given some \(v_f, t < T\) s.t. \((e [\delta \gamma] v_f)\) it suffices to prove that
\((p_m + p_1, T - t, v_f) \in [(\tau) \sigma t]\)

**IH** \((p_1 + p_m, T, e \delta \gamma) \in [(c \Rightarrow \tau) \sigma t]_E\)

This means from Definition \[58\] we have
\(\forall v', t' < T . e \delta t_{v_1} v' \implies (p_1 + p_m, T - t', v') \in [(c \Rightarrow \tau) \sigma t]_E\)
Since we know that \((e [\delta \gamma] v_f)\) therefore from E-CE we know that \(\exists t' < t. e \delta \gamma \psi_{v'} \Lambda. e\')\), therefore we have
\((p_1 + p_m, T - t', \Lambda. e') \in [(c \Rightarrow \tau) \sigma t]_E\)

Therefore from Definition \[58\] we have
\(\forall t'' < T - t'. \vdash e \ i \implies (p_1 + p_m, T - t' - t'', e \delta \gamma) \in [\tau \sigma t]_E\)
Since we are given \(\Theta; \Delta \vdash e : \sigma t\) and \(\vdash \Delta \ i\). Therefore instantiating it with \(T - t' - 1\) and since we know that \(\vdash e : \sigma t\). Hence we get
\((p_1 + p_m, T - t' - 1, e \delta \gamma) \in [\tau \sigma t]_E\)

This means from Definition \[58\] we have
\(\forall v'_f, t'' < T - t' - 1. (e') \delta t_{v'_f} v'_f \implies (p_m + p_1, v'_f) \in [(\tau) \sigma t]\)
Since from E-CE we know that \((e') \delta t_{v'_f} v'_f\) therefore we know that \(\exists t'' . e' \delta t_{v''} v_f\) s.t. \(t = t' + t'' + 1\)
Therefore instantiating (F-CE1) with the given \(v_f\) and \(t''\) we get
\((p_m + p_1, T - t, v_f) \in [(\tau) \sigma t]\)
and we are done.
30. T-CAndI:

$$\begin{align*}
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \quad \Theta; \Delta \models c}{\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : (c \& \tau)} \quad \text{T-CAndI}
\end{align*}$$

Given: $$(p_1, T, \gamma) \in [\Gamma \sigma_1]_E$$, $$(p_m, T, \delta) \in [\Omega \sigma_1]_E$$
To prove: $$(p_1 + p_m, T, e \& \delta \gamma) \in [c \& \tau \sigma_1]_E$$

From Definition 58 it suffices to prove that
$$\forall v_f, t < T. e \& \delta \gamma \Downarrow v_f \implies (p_1 + p_m, T - t, v_f) \in [c \& \tau \sigma_1]$$

This means given some $$v_f$$, $$t < T$$ s.t. $$e \& \delta \gamma \Downarrow v_f$$ it suffices to prove that $$(p_1 + p_m, T - t, v_f) \in [c \& \tau \sigma_1]$$

From Definition 58 it suffices to prove that
$$\vdash \forall v_f, t < T. e \& \delta \gamma \Downarrow v_f \implies (p_1 + p_m, T - t, v_f) \in [\tau \sigma_1]$$

Therefore from Definition 59 we have
$$\vdash (p_1 + p_m, T - t, v_f) \in [\tau \sigma_1] \quad \text{(F-CAI)}$$

We get the desired from (F-CAI)

31. T-CAndE:

$$\begin{align*}
\frac{\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e : (c \& \tau) \quad \Psi; \Theta; \Delta, c; \Omega; \Gamma_2, x : \tau \vdash e' : \tau'}{
\Psi; \Theta; \Delta, \Omega; \Gamma_1 \oplus \Gamma_2 \vdash \text{clet } x = e \text{ in } e' : \tau'} \quad \text{T-CAndE}
\end{align*}$$

Given: $$(p_1, T, \gamma) \in [\Gamma_1 \oplus \Gamma_2] \sigma_1]_E$$, $$(p_m, T, \delta) \in [\Omega] \sigma_1]_E$$
To prove: $$(p_1 + p_m, T, (\text{clet } x = e \text{ in } e') \delta \gamma) \in [\tau' \sigma_1]_E$$

From Definition 58 it suffices to prove that
$$\forall v_f, t < T. (\text{clet } x = e \text{ in } e') \delta \gamma \Downarrow v_f \implies (p_1 + p_m, T - t, v_f) \in [\tau' \sigma_1]$$

This means given some $$v_f$$, $$t < T$$ s.t. $$(\text{clet } x = e \text{ in } e') \delta \gamma \Downarrow v_f$$ it suffices to prove that $$(p_1 + p_m, T - t, v_f) \in [\tau' \sigma_1]$$

From Definition 59 and Definition 57 we know that $$\exists p_{11}, p_{12}, p_{11} + p_{12} = p_1$$ s.t
$$(p_{11}, T, \gamma) \in [\Gamma_1] \sigma_1]_E \quad \text{and} \quad (p_{12}, T, \gamma) \in [\Gamma_2] \sigma_1]_E$$

Similarly from Definition 59 and Definition 56 we also know that
$$\exists p_{m1}, p_{m2}, p_{m1} + p_{m2} = p_m$$ s.t
$$(p_{m1}, T, \delta) \in [\Omega_1] \sigma_1]_E \quad \text{and} \quad (p_{m2}, T, \delta) \in [\Omega_2] \sigma_1]_E$$

IH1
$$(p_{11} + p_{m1}, T, e \& \delta \gamma) \in [c \& \tau \sigma_1]_E$$

This means from Definition 58 we have
$$\forall t_1 < T. e \& \delta \gamma \Downarrow v_1 \implies (p_{11}, T - t_1 v_1) \in [c \& \tau \sigma_1]_E$$

Since we know that $$(\text{clet } x = e \text{ in } e') \delta \gamma \Downarrow v_f$$ therefore from E-CAndE we know that $$\exists t_1 < t, v_1, e \delta \gamma \Downarrow t_1, v_1$$.

Therefore we have
$$(p_{11} + p_{m1}, T - t_1, v_1) \in [c \& \tau \sigma_1]$$

Therefore from Definition 58 we have
$$\vdash \forall t_1 < T. e \delta \gamma \Downarrow t_1 v_f \implies (p_{11} + p_{m1}, T - t_1 - t_2, v_f) \in [\tau \sigma_1] \quad \text{(F-CAE1)}$$

IH2
$$(p_{12} + p_{m2} + p_{11} + p_{m1}, T - t_1, e' \delta \gamma') \in [\tau' \sigma_1]_E$$

where
$$\gamma' = \gamma \cup \{x \mapsto v_1\}$$

This means from Definition 58 we have
$$\forall t_2 < T. e' \delta \gamma' \Downarrow t_2 v_f \implies (p_{12} + p_{m2} + p_{11} + p_{m1}, T - t_1 - t_2, v_f) \in [\tau' \sigma_1]$$
Since we know that (let x = e in e′) δγ ⊢_tvf therefore from E-CAAndE we know that ∃t_2.e′ δ′γ ⊢_tvf s.t.
t_2 = t - t_1 - 1.
Therefore we have
(p_{t_2} + p_m + p_{t_1} + p_{m_1}, T - t_1 - t_2, vf) ∈ [τ' σΓ']
Since p_t = p_{t_2} + p_m and p_m = p_{m_1} + p_{m_2} therefore we get
(p_{t} + p_m, T - t, vf) ∈ [τ' σΓ']

And we are done.

32. T-fix:

\[
\frac{\Psi; \Theta, b; \Delta, b < L; \Omega, x : a_t \tau[(b + 1 + \tsum {b}\beta b b b + 1, a I)]/b_1, \vdash e : \tau \quad L \geq \tsum {b}b b b 0, 1 I}{\Psi; \Theta, \sum {b}b \Omega_1, e : \tau[0/b]} \quad \text{T-fix}
\]

Given: (p_t, T, γ) ∈ [[L]], (p_m, T, δ) ∈ [Σ_{b < L} Ω σΓ_1]_ε and \(\vdash \Delta t\)
To prove: (p_t + p_m, T, (fixx.e) δγ) ∈ [[τ[0/b]] σΓ_1]_ε

From Definition 58 it suffices to prove that
∀ T', t < T, v_f (fixx.e) δγ ⊢_tvf \(\rightarrow (p_t + p_m, T - t, v_f) \in [τ[0/b]] σΓ_1\)

This means given some t < T, v_f s.t. fixx.e δγ ⊢_tvf therefore it suffices to prove that
(p_t + p_m, T - t, v_f) ∈ [[τ[0/b]] σΓ_1] (F-FX0)

Also from Lemma 63 we know that
∀p_0, ..., p_{t-1}. p_0 + ... + p_{t-1} \leq p_m ∧ \forall 0 \leq i < L. (p_i, δ) ∈ [[Ω[i/a]]_ε

We define

\[
p_N(leaf) \triangleq p^l_{leaf}
n_N(t) \triangleq p^i_t + (\sum_{a < I(t)} p_N((t + 1 + \tsum {b}b b b t + 1, a I(b))))
\]

Claim
∀0 ≤ t < L. (p_N(t), T, e δ'γ) ∈ [[τ[t/b]] σΓ_1]_ε
where
δ' = δ ∪ \{x \mapsto fixx.eδ\}

This means given some t it suffices to prove
(p_N(t), T, e δγ) ∈ [[τ[t/b]] σΓ_1]_ε

We prove this by induction on t
Base case: when t is a leaf node (say l)
It suffices to prove that (p^l_t, T, e δ'γ) ∈ [[τ[l/b]] σΓ_1]_ε

We know that I(l) = 0 therefore from IH (of the outer induction) we get the desired

Inductive case: when t is some arbitrary non-leaf node
From IH we know that
∀a < I(t). (p_N(t'), T, e δ'γ) ∈ [[τ'[t'/b]] σΓ_1]_ε where t' = (t + 1 + \tsum {b}b b b t + 1, a I(b))

Claim
∀τ'. (p_N(t'), T, e δ'γ) ∈ [[τ' σΓ_1]_ε where δ' = δ ∪ \{x \mapsto fixx.eδ\} \(\rightarrow (p_N(t'), T, fixx.e δγ) ∈ [[τ' σΓ_1]_ε\)

Proof is trivial

Therefore we have
∀a < I(t). (p_N(t'), T, fixx.e δγ) ∈ [[τ'[t'/b]] σΓ_1]_ε where t' = (t + 1 + \tsum {b}b b b t + 1, a I(b))

Now from the IH of the outer induction we get
(p^i_t + \sum_{a < I} p_N(t'), T, e δ'γ) ∈ [[τ[t/b]] σΓ_1]_ε

Which means we get the desired i.e
(p_N(t), T, e δ'γ) ∈ [[τ[t/b]] σΓ_1]_ε
Since we have proved
\[ \forall 0 \leq t < L. \ (p_N(t), T, e \delta' \gamma) \in [\tau[t/b]] e \]
where
\[ \delta' = \delta \cup \{ x \mapsto \text{fix} x. e \} \]

Therefore from Definition 59 we have
\[ \forall 0 \leq t < L. \ \forall T'' < T. e \delta' \gamma \downarrow_{T''} v_f \implies (p_N(t), T - T'', v_f) \in [\tau[t/b]] e \]

Instantiating with \( t \) with 0 and since we know that \( \text{fix} x. e \delta' \gamma \downarrow_{T''} v_f \) therefore know that \( \exists T'' < T. e \delta' \gamma \downarrow_{T''} v_f \) where \( T'' = T'' - 1 \)
\[ (p_N(0), T - T'', v_f) \in [\tau[0/b]] e \]

Since \( p_N(0) \leq p_m \) therefore \( p_N(0) \leq p_l + p_m \)
And we get the (F-FX0) from Lemma 61

33. T-ret:

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash \text{ret } c : \mathbb{M} 0 \tau
\end{align*}
\]

T-ret

Given: \( (p_l, T, \gamma) \in [\Gamma \sigma_i] e, (p_m, T, \delta) \in [\Omega \sigma_i] e \) and \( \models \Delta \iota \)
To prove: \( (p_l + p_m, T, \text{ret } e \delta' \gamma) \in [\mathbb{M} 0 \tau] e \)

From Definition 59 it suffices to prove that
\[ \forall t < T. (e \delta) \downarrow_{T} v_f \implies (p_m + p_l, T - t, (e \delta) \gamma) \in [\mathbb{M} 0 \tau] e \]

Since from E-val we know that \( t = 0 \) therefore it suffices to prove that
\( (p_m + p_l, T - t, v_f) \in [\tau \sigma_i] e \)
(F-R0)

\[ \models \]
\[ (p_l + p_m, T, e \delta' \gamma) \in [\tau \sigma_i] e \]

This means given some \( n', t' < T, v_f \) s.t. \( (e \delta) \downarrow_{T} v_f \) it suffices to prove that
\[ \exists p', n' + p' \leq p_l + p_m \wedge (p', T - t', v_f) \in [\tau] \]

From (E-ret) we know that \( n' = 0 \) therefore we choose \( p' \) as \( p_l + p_m \) and it suffices to prove that
\( (p_l + p_m, T - t', v_f) \in [\tau \sigma_i] e \)
(F-R0)

34. T-bind:

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 \vdash e_1 : \mathbb{M} n_1 \tau_1 \\
\Theta \vdash n_1 : \mathbb{R}^+ \\
\Theta \vdash n_2 : \mathbb{R}^+
\end{align*}
\]

T-bind

Given: \( (p_l, T, \gamma) \in [(\Gamma_1 \oplus \Gamma_2) \sigma_i] e, (p_m, T, \delta) \in [(\Omega_1 \oplus \Omega_2) \sigma_i] e \) and \( \models \Delta \iota \)
To prove: \( (p_l + p_m, T, \text{bind } x = e_1) \in [(\mathbb{M}(n_1 + n_2) \tau_2) \sigma_i] e \)

From Definition 59 it suffices to prove that
\[ \forall t < T, v. (\text{bind } x = e_1) \downarrow_{T} v \implies (p_m + p_l, T - t, \text{bind } x = e_1) \in [\mathbb{M}(n_1 + n_2) \tau_2] e \]

This means given some \( t < T, v \) s.t. \( \text{bind } x = e_1 \downarrow_{T} v \) and from E-val we know that \( v = (\text{bind } x = e_1) \downarrow_{T} \delta' \gamma \) and \( t = 0 \). It suffices to prove that
\( (p_l + p_m, T, \text{bind } x = e_1) \in [\mathbb{M}(n_1 + n_2) \tau_2] e \)

This means from Definition 59 it suffices to prove that
\[ \forall s', t' < T, v_f. (\text{bind } x = e_1) \downarrow_{T} v \implies \exists p', s' + p' \leq p_l + p_m + n \wedge (p', T - t', v_f) \in [\tau_2] e \]

This means given some \( s', t' < T, v_f \) s.t. \( \text{bind } x = e_1 \downarrow_{T} v \) and we need to prove that
\( \exists p', s' + p' \leq p_l + p_m + n \wedge (p', T - t', v_f) \in [\tau_2] e \)
(F-B0)

From Definition 59 and Definition 57 we know that \( \exists p_{11}, p_{12} p_{11} + p_{12} = p_l \)
(\( p_{11}, T, \gamma) \in [(\Gamma_1) \sigma_i] e \) and \( (p_{12}, T, \gamma) \in [(\Gamma_2) \sigma_i] e \)
Similarly from Definition 59 and Definition 56 we also know that \( \exists p_{ml}, p_{m2} : p_{ml} + p_{m2} = p_m \) s.t.
\( (p_{ml}, T, \delta) \in [(\Omega_1)_{\sigma l}]_\epsilon \) and \( (p_{m2}, T, \delta) \in [(\Omega_2)_{\sigma l}]_\epsilon \)

**HI**

\( (p_{l1} + p_{ml}, T, e_1 \delta_\gamma) \in [M(n_1) \tau_1 \sigma l]_\epsilon \)

From Definition 55 it means we have
\( \forall t_1 < T . (e_1) \delta_\gamma \downarrow t_1 (e_1) \delta_\gamma \implies (p_{l1} + p_{ml}, T - t_1, (e_1) \delta_\gamma) \in [M(n_1) \tau_1 \sigma l]_\epsilon \)

Since we know that \( (\text{bind} \ x = e_1 \ in \ e_2) \delta_\gamma \downarrow t_\gamma \) from E-bind we know that \( \exists t_1 < t, v_{ml}(e_1) \delta_\gamma \downarrow t_\gamma \)

Since \( t_1 < t' \ < T \), therefore we have \( (p_{l1} + p_{ml}, T - t_1, (e_1) \delta_\gamma) \in [M(n_1) \tau_1 \sigma l]_\epsilon \)

This means from Definition 55 we are given that
\( \forall t'_1 < T - t_1, (e_1) \delta_\gamma \downarrow t'_1 v_1 \implies \exists p'_{l1}, s_1 + p'_{l1} \leq p_{l1} + p_{ml} + n_1 \land (p'_{l1}, T - t_1 - t'_1, v_1) \in [\tau_1 \sigma l]_\epsilon \)

Since we know that \( (\text{bind} \ x = e_1 \ in \ e_2) \delta_\gamma \downarrow t_\gamma \) from E-bind we know that \( \exists t'_1 < t - t_1, (e_1) \delta_\gamma \downarrow t'_1 v_1 \)

This means we have
\( \exists p'_{l1}, s_1 + p'_{l1} \leq p_{l1} + p_{ml} + n_1 \land (p'_{l1}, T - t_1 - t'_1, v_1) \in [\tau_1 \sigma l]_\epsilon \)

**HI2**

\( (p_{l2} + p_{ml} + p'_{l2}, T - t_1 - t'_1, t_2 \delta_\gamma \cup \{x \mapsto v_1\}) \in [M(n_2) \tau_2 \sigma l]_\epsilon \)

From Definition 55 it means we have
\( \forall t_2 < T - t_1 - t'_1, t_2 \delta_\gamma \cup \{x \mapsto v_1\} \downarrow t_2 \ (e_2) \delta_\gamma \cup \{x \mapsto v_1\} \implies (p_{l2} + p_{ml} + p'_{l2} + n_2, T - t_1 - t'_1 - t_2, (e_2) \delta_\gamma \cup \{x \mapsto v_1\}) \in [M(n_2) \tau_2 \sigma l]_\epsilon \)

Since we know that \( (\text{bind} \ x = e_1 \ in \ e_2) \delta_\gamma \downarrow t_\gamma \) from E-bind we know that
\( \exists t_2 < t' - t_1 - t'_1 < T - t_1 - t'_1 \delta_\gamma \cup \{x \mapsto v_1\} \implies (p_{l2} + p_{ml} + p'_{l2} + n_2, T - t_1 - t'_1 - t_2, (e_2) \delta_\gamma \cup \{x \mapsto v_1\}) \in [M(n_2) \tau_2 \sigma l]_\epsilon \)

This means from Definition 55 we are given that
\( \forall t'_2 < T - t_1 - t'_1 - t_2, t'_2 \delta_\gamma \cup \{x \mapsto v_1\} \downarrow t'_2 v_2 \implies \exists p'_{l2}, s_2 + p'_{l2} \leq p_{l2} + p_{ml} + p'_{l2} + n_2 \land (p'_{l2}, T - t_1 - t'_1 - t_2 - t'_2, v_2) \in [\tau_2 \sigma l]_\epsilon \)

Since we know that \( (\text{bind} \ x = e_1 \ in \ e_2) \delta_\gamma \downarrow t_\gamma \) from E-bind we know that \( \exists t'_2 < t' - t_1 - t'_1 - t_2 - t'_2, v_2 \in [\tau_2 \sigma l]_\epsilon \)

This means we have
\( \exists p'_{l2}, s_2 + p'_{l2} \leq p_{l2} + p_{ml} + p'_{l2} + n_2 \land (p'_{l2}, T - t_1 - t'_1 - t_2 - t'_2, v_2) \in [\tau_2 \sigma l]_\epsilon \)

In order to prove (F-B0) we choose \( p' \) as \( p'_{l2} \) and it suffices to prove

(a) \( s' + p'_{l2} \leq p_{l1} + p_{ml} + n \)

Since from (F-B2) we know that
\( s_2 + p'_{l2} \leq p_{l2} + p_{ml} + p'_{l2} + n_2 \)

Adding \( s_1 \) on both sides we get
\( s_1 + s_2 + p'_{l2} \leq p_{l2} + p_{ml} + s_1 + p'_{l2} + n_2 \)

Since from (F-B1) we know that
\( s_1 + p'_{l1} \leq p_{l1} + p_{ml} + n_1 \)

therefore we also have
\( s_1 + s_2 + p'_{l2} \leq p_{l2} + p_{ml} + p'_{l1} + p_{ml} + n_1 + n_2 \)

And finally since we know that \( n = n_1 + n_2 \), \( s' = s_1 + s_2, p_{l1} = p_{l1} + p_{l2} \) and \( p_{ml} = p_{ml} + p_{ml} \) therefore we get the desired

(b) \( (p'_{l2}, T - t_1 - t'_1 - t_2 - t'_2, v_f) \in [\tau_2 \sigma l]_\epsilon \)

From E-bind we know that \( v_f = v_2 \) therefore we get the desired from (F-B2)

35. T-tick:

\[
\Theta \vdash n : \mathbb{R}^+ \\
\underbrace{\Psi ; \Delta ; \Omega ; \Gamma \vdash \vdash : M n 1}_{\text{T-tick}}
\]

Given: \( (p_{l}, T, \gamma) \in [\Gamma \sigma l]_\epsilon, (p_{m}, T, \delta) \in [\Omega \sigma l]_\epsilon \) and \( \models \Delta \iota \)
To prove: \( (p_l + p_m, T, \uparrow^n \delta \gamma) \in [\mathbb{M} n 1 \sigma_i]_\varepsilon \)

From Definition 58 it suffices to prove that
\( \uparrow^n \delta \gamma \iff (p_l + p_m, T, \uparrow^n \delta \gamma) \in [\mathbb{M} n 1 \sigma_i]_\varepsilon \)

It suffices to prove that
\( (p_m + p_l, T, \uparrow^n \delta \gamma) \in [\mathbb{M} n 1 \sigma_i] \)

From Definition 58 it suffices to prove that
\( \forall t'. < T, t'.(\uparrow^n \delta \gamma \downarrow^\varepsilon) \iff \exists p', n' \leq p_l + p_m + n \wedge (p', T - t', (\varepsilon)) \in [1] \)

This means given some \( t' < T, n' \) s.t. \( (\uparrow^n \delta \gamma \downarrow^\varepsilon) \) it suffices to prove that
\( \exists p', n' \leq p_l + p_m + n \wedge (p', T - t', (\varepsilon)) \in [1] \)

From (E-tick) we know that \( n' = n \) therefore we choose \( p' \) as \( p_l + p_m \) and it suffices to prove that
\( (p_l + p_m, T - t', (\varepsilon)) \in [1] \)

We get this directly from Definition 58.

36. T-release:

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma_1 & \vdash e_1 : [n_1] \tau_1 \\
\Psi; \Theta; \Delta; \Omega; \Gamma_2; x : \tau_1 & \vdash e_2 : M(n_1 + n_2) \tau_2 \\
\Theta & \vdash n_1 : R^+ \\
\Theta & \vdash n_2 : R^+ \\
\end{align*}
\]

- T-release:

Given: \( (p_l, T, \gamma) \in [(\Gamma_1 \oplus \Gamma_2)] \sigma_i \) and \( (p_m, T, \delta) \in [(\Omega_1 \oplus \Omega_2)] \sigma_i \) and \( \models \Delta \)

To prove: \( (p_l + p_m, T, \text{release } x = e_1) \in [\mathbb{M}(n_2) \tau_2 \sigma_i]_\varepsilon \)

From Definition 58 it suffices to prove that
\( \text{release } x = e_1 \in e_2 \) \( \uparrow^n \delta \gamma \downarrow e_0 (\text{release } x = e_1 \in e_2 \delta \gamma) \iff (p_m + p_l, \text{release } x = e_1 \in e_2 \delta \gamma) \in [\mathbb{M}(n_2) \tau_2 \sigma_i]_\varepsilon \)

This means given \( \text{release } x = e_1 \in e_2 \) \( \uparrow^n \delta \gamma \downarrow e_0 (\text{release } x = e_1 \in e_2 \delta \gamma) \) it suffices to prove that
\( (p_m + p_l, \text{release } x = e_1 \in e_2 \delta \gamma) \in [\mathbb{M}(n_2) \tau_2 \sigma_i]_\varepsilon \)

This means from Definition 58 it suffices to prove that
\( \forall t'. < T, v_f, s'. \) \( \text{release } x = e_1 \in e_2 \delta \gamma \downarrow s'. v_f \iff (p_m + p_l + n_2 \wedge (p', T - t', v_f) \in [\mathbb{M}(n_2) \tau_2 \sigma_i]) \)

This means given some \( t' < T, v_f, s' \) s.t. \( \text{release } x = e_1 \in e_2 \delta \gamma \downarrow s'. v_f \) and we need to prove that
\( \exists p', s'. v_f \leq p_l + p_m + n_2 \wedge (p', T - t', v_f) \in [\mathbb{M}(n_2) \tau_2 \sigma_i] \)

From Definition 58 and Definition 58 we know that \( \exists p_{l_1}, p_{l_2}, p_{l_1} + p_{l_2} = p_l \) s.t.
\( (p_{l_1}, T, \gamma) \in [(\Gamma_1)] \sigma_i \) and \( (p_{l_2}, T, \gamma) \in [(\Gamma_2)] \sigma_i \)

Similarly from Definition 58 and Definition 58 we also know that \( \exists p_{m_1}, p_{m_2}, p_{m_1} + p_{m_2} = p_m \) s.t.
\( (p_{m_1}, T, \delta) \in [(\Omega_1)] \sigma_i \) and \( (p_{m_2}, T, \delta) \in [(\Omega_2)] \sigma_i \)

\( \mathbf{H1} \)

\( (p_{l_1} + p_{m_1}, T, e_1 \delta \gamma) \in [[n_1] \tau_1 \sigma_i]_\varepsilon \)

From Definition 58 it means we have
\( \forall t_1. (e_1 \delta \gamma \downarrow v_1) \iff (p_{l_1} + p_{m_1}, T - t_1, v_1) \in [[n_1] \tau_1 \sigma_i]_\varepsilon \)

Since we know that \( \text{release } x = e_1 \in e_2 \delta \gamma \downarrow v_f \) therefore from E-rel we know that \( \exists t_1 < t'. (e_1 \delta \gamma \downarrow v_f) \).

This means we have
\( (p_{l_1} + p_{m_1}, T - t_1, v_1) \in [[n_1] \tau_1 \sigma_i]_\varepsilon \)

\( \mathbf{H2} \)

\( (p_{l_2} + p_{m_2} + p_1', T - t_1, v_f \delta \gamma \cup \{x \mapsto v_1\}) \in [\mathbb{M}(n_1 + n_2) \tau_2 \sigma_i]_\varepsilon \)

From Definition 58 it means we have
\( \forall t_2 < T - t_1, (e_2) \delta \gamma \cup \{x \mapsto v_1\} \downarrow_{\tau_2} (e_2) \delta \gamma \cup \{x \mapsto v_1\} \iff (p_{m_2} + p_{l_2} + p_1' + n_2, T - t_1 - t_2, (e_2) \delta \gamma \cup \{x \mapsto v_1\}) \in [\mathbb{M}(n_1 + n_2) \tau_2 \sigma_i]_\varepsilon \)

Since we know that \( \text{release } x = e_1 \in e_2 \delta \gamma \downarrow v_f \) therefore from E-rel we know that
\( \exists t_2 < t - t_1, (e_2) \delta \gamma \cup \{x \mapsto v_1\} \downarrow_{\tau_2} (e_2) \delta \gamma \cup \{x \mapsto v_1\}. \)

This means we have
\( (p_{m_2} + p_{l_2} + p_1' + n_2, T - t_1 - t_2, (e_2) \delta \gamma \cup \{x \mapsto v_1\}) \in [\mathbb{M}(n_1 + n_2) \tau_2 \sigma_i]_\varepsilon \)

This means from Definition 58 we are given that
Lemma 65 (Γ Subtyping: domain containment). \( \forall p, \gamma, \Gamma_1, \Gamma_2 \),
\[ \Psi; \Theta; \Delta \vdash s \in \tau \implies \forall x : \tau \in \Gamma_2, x : \tau' \in \Gamma_1 \wedge \Psi; \Theta; \Delta \vdash \tau' : \tau. \]

**Proof.** Proof by induction on \( \Psi; \Theta; \Delta \vdash s \in \tau \).
Lemma 66 (Ω Subtyping: domain containment). ∀p, γ, Ω₁, Ω₂. 
Ψ; Θ; Δ ⊢ Ω₁ ⊂ Ω₂ → ∀x : a < I τ ∈ Ω₂, x : a < J τ' ∈ Ω₁ ∩ Ψ; Θ; Δ ⊢ Ω₁ ⊂ Ω₂ 
Proof. Proof by induction on Ψ; Θ; Δ ⊢ Ω₁ ⊂ Ω₂
1. sub-mBase:
Ψ; Θ; Δ ⊢ Ω ⊂⊂ → sub-mBase

To prove: ∀x : a < I τ ∈ Ω₂, x : a < J τ' ∈ Ω₁ ∩ Ψ; Θ; Δ ⊢ I ≤ J ∧ Ψ; Θ, a; Δ, a < I ⊢ τ' <: τ

This means given some y : τ ∈ (Γ₂, x : τ) it suffices to prove that 

∀y : τ ∈ (Γ₂, x : τ) it suffices to prove that

The following cases arise:
• y = x:
  In this case we are given that x : τ' ∈ Γ₁ ∧ Ψ; Θ; Δ ⊢ τ' <: τ
  Therefore we are done
• y ≠ x:
  Since we are given that Ψ; Θ; Δ ⊢ Γ₁/x <: Γ₂ therefore we get the desired from IH

Lemma 67 (Γ subtyping lemma). ∀p, γ, Π₁, Π₂, Σ, τ.
Ψ; Θ; Δ ⊢ Π₁ <: Π₂ → Π₁[σ|Γ] ≤ Π₂[σ|Γ]
Proof. Proof by induction on Ψ; Θ; Δ ⊢ Π₁ <: Π₂
1. sub-lBase:
Ψ; Θ; Δ ⊢ Γ₁ <: Γ₂ → sub-lBase

To prove: ∀(p, T, γ) ∈ [Π₁[σ|Γ]]ε, (p, T, γ) ∈ [Π₂[σ|Γ]]ε
This means given some (p, T, γ) ∈ [Π₁[σ|Γ]]ε it suffices to prove that (p, T, γ) ∈ [Π₂[σ|Γ]]ε
From Definition 59 it suffices to prove that
\[ \exists \, [\Gamma, \sigma, i]. (p, T, \gamma(x)) \in [\Gamma(x)]_\epsilon \land (\sum_{x \in \text{dom}(\cdot)} f(x) \leq p) \]

We choose \( f \) as a constant function \( f' = 0 \) and we get the desired

2. sub-lInd:

\[
\frac{x : \tau' \in \Gamma_1 \quad \Psi; \Theta; \Delta \vdash \gamma < : \tau \quad \Psi; \Theta; \Delta \vdash \Gamma_1 / x : \Gamma_2}{\Psi; \Theta; \Delta \vdash \Gamma_1 < : \Gamma_2, x : \tau} \quad \text{(sub-lBase)}
\]

To prove: \( \forall (p, T, \gamma) \in [\Gamma, \sigma]_\epsilon. (p, T, \gamma) \in [\Gamma_2, x : \tau]_\epsilon \)

This means given some \( (p, T, \gamma) \in [\Gamma, \sigma]_\epsilon \) it suffices to prove that \( (p, T, \gamma) \in [\Gamma_2, x : \tau]_\epsilon \)

This means from Definition 59 it suffices to prove that
\[ \exists \, [\Gamma_1, \sigma, i]. (f(x), T, \gamma(x)) \in [\Gamma(x)]_\epsilon \land (\sum_{x \in \text{dom}(\cdot)} f(x) \leq p) \]

Similarly from Definition 59 it suffices to prove that
\[ \exists f' : \text{Vars} \rightarrow \text{Preds}. (\forall y \in \text{dom}(\Gamma_2, x : \tau). (f'(y), T, \gamma(y)) \in [\Gamma(y)]_\epsilon \land (\sum_{y \in \text{dom}(\Gamma_2, x : \tau)} f'(y) \leq p) \]

We choose \( f' \) as \( f \) and it suffices to prove that

(a) \( \forall y \in \text{dom}(\Gamma_2, x : \tau). (f(y), T, \gamma(y)) \in [\Gamma(y)]_\epsilon \)

This means given some \( y \in \text{dom}(\Gamma_2, x : \tau) \) it suffices to prove that
\( (f(y), T, \gamma(y)) \in [\tau_2]_\epsilon \) where say \( \Gamma(y) = \tau_2 \)

From Lemma 65 we know that
\( y : \tau_1 \in \Gamma_1 \land \Psi; \Theta; \Delta \vdash \tau_1 < : \tau_2 \)

By instantiating (L0) with the given \( y \)
\( (f(y), T, \gamma(y)) \in [\tau_1]_\epsilon \)

Finally from Lemma 70 we also get \( (f(y), T, \gamma(y)) \in [\tau_2]_\epsilon \)

And we are done

(b) \( \sum_{y \in \text{dom}(\Gamma_2, x : \tau)} f(y) \leq p \)

From (L1) we know that \( \sum_{x \in \text{dom}(\Gamma_1)} f(x) \leq p \) and since from Lemma 65 we know that \( \text{dom}(\Gamma_2, x : \tau) \subseteq \text{dom}(\Gamma_1) \) therefore we also have
\( \sum_{y \in \text{dom}(\Gamma_2, x : \tau)} f(y) \leq p \)

\[ \square \]

**Lemma 68 (\( \Omega \) subtyping lemma). \( \forall p, \gamma, \Omega_1, \Omega_2, \sigma, i. \)

\( \Psi; \Theta; \Delta \vdash \Omega_1 < : \Omega_2 \implies [\Omega_1, \sigma, i] \subseteq [\Omega_2, \sigma, i] \)

**Proof.** Proof by induction on \( \Psi; \Theta; \Delta \vdash \Omega_1 < : \Omega_2 \)

1. sub-lBase:

\[
\frac{\Psi; \Theta; \Delta \vdash \Omega_1 < : \Omega_2 \quad \Omega_1 \subseteq \Omega_2}{\Psi; \Theta; \Delta \vdash \Omega_1 < : \Omega_2} \quad \text{(sub-mBase)}
\]

To prove: \( \forall (p, T, \gamma) \in [\Omega_1, \sigma, i]_\epsilon. (p, T, \gamma) \in [\Omega_2, x : \tau]_\epsilon \)

This means given some \( (p, T, \gamma) \in [\Omega_1, \sigma, i]_\epsilon \) it suffices to prove that \( (p, T, \gamma) \in [\Omega_2, x : \tau]_\epsilon \)

From Definition 59 it suffices to prove that
\[ \exists f : \text{Vars} \rightarrow \text{Indices} \rightarrow \text{Preds}. (\forall x : \text{a} < i \tau). \epsilon \implies \forall 0 \leq i < I. (f x i, T, \delta(x)) \in [\tau[i/a]]_\epsilon \land (\sum_{x : \text{a} < i \tau} f x i) \leq p \]

We choose \( f \) as a constant function \( f' = 0 \) and we get the desired

2. sub-lInd:

\[
\frac{x : \text{a} < J \tau' \in \Omega_1 \quad \Psi; \Theta; a; \Delta, a < I \vdash \gamma < : \tau \quad \Omega; \Delta \vdash I \leq J \quad \Psi; \Theta; \Delta \vdash \Omega_1 / x : \Omega_2}{\Psi; \Theta; \Delta \vdash \Omega_1 < : \Omega_2, x : \text{a} < I \tau} \quad \text{(sub-mInd)}
\]

To prove: \( \forall (p, T, \gamma) \in [\Omega_2, x : \tau]_\epsilon \)

This means given some \( (p, T, \gamma) \in [\Omega_2, x : \tau]_\epsilon \) it suffices to prove that \( (p, T, \gamma) \in [\Omega_2, x : \tau]_\epsilon \)

This means from Definition 59 we are given that
\[ \exists f : \text{Vars} \rightarrow \text{Preds}. (\forall x : \text{a} < I \tau). \epsilon \implies \forall 0 \leq i < I. (f x i, T, \delta(x)) \in [\tau[i/a]]_\epsilon \quad \text{(L0)} \]
Similarly from Definition 59 it suffices to prove that

\[ \exists y : \forall y \in \{y \mid a < I_y\} \in \Omega_2, x : \tau. \forall 0 \leq i < I_y, (f x i, T, \delta(y)) \in [r_y[i/a]] \wedge (\sum_{y : \exists y \in \{y \mid a < I_y\}} f' y i) \leq p \]

We choose \( f' \) as \( f \) and it suffices to prove that

\[ (\forall (y : a < I_y \tau_y) \in \Omega_2, x : \tau. \forall 0 \leq i < I_y, (f x i, T, \delta(y)) \in [r_y[i/a]]) \]

This means given some \( (y : a < I_y \tau_y) \in \Omega_2, x : \tau \) and some \( 0 \leq i < I_y \) it suffices to prove that

\[ (f x i, T, \delta(y)) \in [r_y[i/a]] \]

From Lemma 65 we know that

\[ y : a < I_y \tau_1 \in \Omega_1 \wedge \Psi; \Theta; \Delta \vdash I_y \leq J_y \wedge \Psi; \Theta; a; \Delta, a < I_y \vdash \tau_1 <: \tau_y \]

Instantiating (L0) with the given \( y \) and \( i \) we get

\[ (f x i, T, \delta(y)) \in [\tau_1[i/a]] \]

Finally using Lemma 70 we also get

\[ (f x i, T, \delta(y)) \in [\tau_1[i/a]] \]

\[ (\sum_{y : a < I_y \tau_y} f' y i) \leq p \]

From Lemma 66 we know that

\[ (\sum_{a < I_y \tau_y} \Omega_2, x : \tau) \]

And since from (L1) we know that

\[ (\sum_{y : a < I_y \tau_y} \Omega_2, x : \tau) \]

Therefore instantiating (F-SL0) with \( p', e''', T''' \) we get

\[ (p + p', T, e'[e'/x]) \in [\tau_2 \sigma_i] \]

From Lemma 70 we get the desired
To prove: \( \forall (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1 \otimes \tau_2) \sigma \rrbracket \implies (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1' \otimes \tau_2') \sigma \rrbracket \)

This means given \( (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1 \otimes \tau_2) \sigma \rrbracket \)

It suffices prove that

\( (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1' \otimes \tau_2') \sigma \rrbracket \)

This means from Definition 58 we are given that

\( \exists p_1, p_2, p_1 + p_2 \leq p \land (p_1, T, v_1) \in \llbracket \tau_1 \sigma \rrbracket \land (p_2, T, v_2) \in \llbracket \tau_2 \sigma \rrbracket \)

Also from Definition 58 it suffices to prove that

\( \exists p_1', p_2', p_1' + p_2' \leq p \land (p_1', T, v_1) \in \llbracket \tau_1' \sigma \rrbracket \land (p_2', T, v_2) \in \llbracket \tau_2' \sigma \rrbracket \)

IH1 \( \llbracket (\tau_1) \sigma \rrbracket \subseteq \llbracket (\tau_1') \sigma \rrbracket \)

IH2 \( \llbracket (\tau_2) \sigma \rrbracket \subseteq \llbracket (\tau_2') \sigma \rrbracket \)

Instantiating \( p_1', p_2' \) with \( p_1, p_2 \) we get the desired from IH1 and IH2

4. sub-with:

\[
\frac{\Psi; \Theta; \Delta \vdash \tau_1 : \tau_1'; \Psi; \Theta; \Delta \vdash \tau_2 : \tau_2'}{\Psi; \Theta; \Delta \vdash \tau_1 \land \tau_2 : \tau_1' \land \tau_2'}
\]

To prove: \( \forall (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1 \land \tau_2) \sigma \rrbracket \implies (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1' \land \tau_2') \sigma \rrbracket \)

This means given \( (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1 \land \tau_2) \sigma \rrbracket \)

It suffices prove that

\( (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1' \land \tau_2') \sigma \rrbracket \)

This means from Definition 58 we are given that

\( (p, T, v_1) \in \llbracket \tau_1 \sigma \rrbracket \land (p, T, v_2) \in \llbracket \tau_2 \sigma \rrbracket \)

(F-SW0)

Also from Definition 58 it suffices to prove that

\( (p, T, v_1) \in \llbracket \tau_1' \sigma \rrbracket \land (p, T, v_2) \in \llbracket \tau_2' \sigma \rrbracket \)

IH1 \( \llbracket (\tau_1) \sigma \rrbracket \subseteq \llbracket (\tau_1') \sigma \rrbracket \)

IH2 \( \llbracket (\tau_2) \sigma \rrbracket \subseteq \llbracket (\tau_2') \sigma \rrbracket \)

We get the desired from (F-SW0), IH1 and IH2

5. sub-sum:

\[
\frac{\Psi; \Theta; \Delta \vdash \tau_1 : \tau_1'; \Psi; \Theta; \Delta \vdash \tau_2 : \tau_2'}{\Psi; \Theta; \Delta \vdash \tau_1 + \tau_2 : \tau_1' + \tau_2'}
\]

To prove: \( \forall (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1 + \tau_2) \sigma \rrbracket \implies (p, T, \langle v_1, v_2 \rangle) \in \llbracket (\tau_1' + \tau_2') \sigma \rrbracket \)

This means given \( (p, T, v) \in \llbracket (\tau_1 + \tau_2) \sigma \rrbracket \)

It suffices prove that

\( (p, T, v) \in \llbracket (\tau_1' + \tau_2') \sigma \rrbracket \)

This means from Definition 58 2 cases arise

(a) \( v = \text{inl}(v') \):

This means from Definition 58 we have \( (p, T, v') \in \llbracket \tau_1 \sigma \rrbracket \)

(F-SW0)

Also from Definition 58 it suffices to prove that

\( (p, T, v') \in \llbracket \tau_1' \sigma \rrbracket \)

IH \( \llbracket (\tau_1) \sigma \rrbracket \subseteq \llbracket (\tau_1') \sigma \rrbracket \)

We get the desired from (F-SW0), IH

(b) \( v = \text{inr}(v') \):

Symmetric reasoning as in the inl case

6. sub-potential:

\[
\frac{\Psi; \Theta; \Delta \vdash \tau : \tau' \quad \Psi; \Theta; \Delta \vdash n : n'}{\Psi; \Theta; \Delta \vdash [n] \tau : [n'] \tau'}
\]

To prove: \( \forall (p, T, v) \in \llbracket [n] \tau \sigma \rrbracket \implies (p, T, v) \in \llbracket [n'] \tau' \sigma \rrbracket \)

This means given \( (p, T, v) \in \llbracket [n] \tau \sigma \rrbracket \) and we need to prove
(p, T, v) ∈ [[n'] τ' σ]\]

This means from Definition 58 we are given

\[ \exists p'.p' + n ≤ p \land (p', T, v) ∈ [[τ' σ]] \]  \hspace{1cm} (F-SP0)

And we need to prove

\[ \exists p''.p'' + n' ≤ p \land (p'', T, v) ∈ [[τ' σ]] \]  \hspace{1cm} (F-SP1)

In order to prove (F-SP1) we choose \( p'' \) as \( p' \)

Since from (F-SP0) we know that \( p' + n \leq p \) and we are given that \( n' \leq n \) therefore we also have \( p' + n' \leq p \)

\[ (p', T, v) ∈ [[τ' σ]] \] we get directly from IH

7. sub-monad:

\[ \Psi; Θ; Δ \vdash τ <; τ' \]  \hspace{1cm} \[ \Psi; Θ; Δ \vdash n \leq n' \]  \hspace{1cm} \text{sub-monad}

To prove: \( ∀(p, T, v) ∈ [[M n τ σ]], (p, T, v) ∈ [[M n' τ' σ]] \)

This means given \( (p, T, v) ∈ [[M n τ σ]] \) and we need to prove 

\( (p, T, v) ∈ [[M n' τ' σ]] \)

This means from Definition 58 we are given

\[ ∀t'' <T_n, n, v'.v' \vdash t'' → \exists p'.p' + n ≤ p + n \land (p', T - t'', v') ∈ [[τ σ]] \]  \hspace{1cm} (F-SM0)

Again from Definition 58 we need to prove that

\[ ∀t'''' <T_n, n, v'''' v'''' \vdash t'''' → \exists p'''.p''' + n' ≤ p + n' \land (p''', T - t'''', v'') ∈ [[τ' σ]] \]  \hspace{1cm} (F-SM1)

This means given some \( t'' <T_n, n, v'' \) s.t. \( v \vdash v'''' v'''' \) it suffices to prove that

\[ \exists p''', n + p'''' ≤ p + n' \land (p''', T - t''', v'') ∈ [[τ' σ]] \]  \hspace{1cm} (F-SM2)

\[ \Psi; Θ; Δ \vdash τ <; τ' \]  \hspace{1cm} \[ \Psi; Θ; Δ \vdash n \leq n' \]  \hspace{1cm} \text{sub-monad}

In order to prove (F-SM1) we choose \( p'''' \) as \( p' \) and we need to prove

(a) \( n + p' \leq p + n' \):

Since we are given that \( n \leq n' \) therefore we get the desired from (F-SM2)

(b) \( (p', T - t''', v') ∈ [[τ' σ]] \)

We get this directly from IH

8. sub-subExp:

\[ \Psi; Θ; Δ, a < J \vdash τ <; τ' \]  \hspace{1cm} \[ \Psi; Θ; Δ \vdash J \leq I \]  \hspace{1cm} \text{sub-subExp}

To prove: \( ∀(p, T, v) ∈ [[a < i τ σ]], (p, T, v) ∈ [[a < i τ' σ]] \)

This means given \( (p, T, !v) ∈ [[a < i τ σ]] \) and we need to prove 

\( (p, T, !v) ∈ [[a < i τ' σ]] \)

This means from Definition 58 we are given

\[ \exists p_0, ..., p_{i-1}, p_0 + ... + p_{i-1} ≤ p \land ∀0 ≤ i < I, (p_i, T, v) ∈ [[τ[i/a]]] \]  \hspace{1cm} (F-SE0)

Again from Definition 58 we need to prove that

\[ \exists p'_0, ..., p'_{j-1}, p'_0 + ... + p'_{j-1} ≤ p \land ∀0 ≤ j < J, (p_j, T, v) ∈ [[τ'[j/a]]] \]  \hspace{1cm} (F-SE1)

In order to prove (F-SE1) we choose \( p'_0, ..., p'_{j-1} \) as \( p_0 ... p_{j-1} \) and we need to prove

(a) \( p_0 + ... + p_{j-1} ≤ p \):

Since we are given that \( J \leq I \) therefore we get the desired from (F-SE0)

(b) \( ∀0 ≤ j < J, (p_j, T, v) ∈ [[τ'[j/a]] σ] \)

We get this directly from IH and (F-SE0)
To prove: $\forall(p, T, v) \in [L^n \tau \sigma_I]$. $(p, T, v) \in [L^n \tau' \sigma_I]$

This means given $(p, T, v) \in [L^n \tau \sigma_I]$ and we need to prove $(p, T, v) \in [L^n \tau' \sigma_I]$

We induct on $(p, T, v) \in [L^n \tau \sigma_I]$

(a) $(p, T, nil) \in [L^0 \tau \sigma_I]$:

We need to prove $(p, T, nil) \in [L^0 \tau' \sigma_I]$

We get this directly from Definition 58

(b) $(p, T, v' :: l') \in [L^{m+1} \tau \sigma_I]$:

In this case we are given $(p, T, v' :: l') \in [L^{m+1} \tau \sigma_I]$ and we need to prove $(p, T, v' :: l') \in [L^{m+1} \tau' \sigma_I]$

This means from Definition 58 we are given

$\exists p_1, p_2, p_1 + p_2 \leq p \land (p_1, T, v') \in [\tau \sigma_I] \land (p_2, T, l') \in [L^m \tau \sigma_I]$ (Sub-List0)

Similarly from Definition 58 we need to prove that

$\exists p_1', p_2', p_1' + p_2' \leq p \land (p_1', T, v') \in [\tau' \sigma_I] \land (p_2', T, l') \in [L^m \tau' \sigma_I]$

We choose $p_1'$ as $p_1$ and $p_2'$ as $p_2$ and we get the desired from (Sub-List0) IH of outer induction and IH of inner induction

To prove: $\forall(p, T, v) \in [\exists s. \tau \sigma_I]$. $(p, T, v) \in [\exists s. \tau' \sigma_I]$

This means given some $(p, T, v) \in [\exists s. \tau \sigma_I]$ we need to prove $(p, T, v) \in [\exists s. \tau' \sigma_I]$

From Definition 58 we are given that

$\exists s'. (p, T, v) \in [\tau \sigma_I s/s']$ (F-exist0)

IH: $[(\tau) \sigma I \cup \{s \mapsto s'\}] \subseteq [(\tau') \sigma I \cup \{s \mapsto s'\}]$

Also from Definition 58 it suffices to prove that

$\exists s''. (p, T, v) \in [\tau' \sigma I s/s']$

We choose $s''$ as $s'$ and we get the desired from IH

To prove: $\forall(p, T, \Lambda \alpha. e) \in [\forall \alpha. \tau_1 \sigma_I]$. $(p, T, \Lambda \alpha. e) \in [\forall \alpha. \tau_2 \sigma_I]$

This means given some $(p, T, \Lambda \alpha. e) \in [\forall \alpha. \tau_1 \sigma_I]$ we need to prove $(p, T, \Lambda \alpha. e) \in [\forall \alpha. \tau_2 \sigma_I]$

From Definition 58 we are given that

$\forall \tau', T' < T . (p, T', c) \in [\tau_1[\tau'/\alpha]]_{\mathcal{E}}$ (F-STF0)

Also from Definition 58 it suffices to prove that

$\forall \tau'', T'' < T . (p, T'', c) \in [\tau_2[\tau''/\alpha]]_{\mathcal{E}}$ (F-STF1)

IH: $[(\tau_1) \sigma I \cup \{\alpha \mapsto \tau''\}] \subseteq [(\tau_2) \sigma I \cup \{\alpha \mapsto \tau''\}]$

Instantiating (F-STF0) with $\tau'', T''$ we get

$(p, T'', c) \in [\tau_1[\tau''/\alpha]]_{\mathcal{E}}$

and finally from IH we get the desired
12. sub-indexPoly:

\[
\Psi; \Theta; i; \Delta \vdash \tau_1 <: \tau_2 \quad \text{sub-indexPoly}
\]

To prove: \( \forall (p, T, \Lambda.i.e) \in \{\{\forall i.\tau_1\} \sigma i\}, (p, T, \Lambda.i.e) \in \{\{\forall i.\tau_2\} \sigma i\} \)

This means given some \((p, T, \Lambda.i.e) \in \{\{\forall i.\tau_1\} \sigma i\}\) we need to prove
\( (p, T, \Lambda.i.e) \in \{\{\forall i.\tau_2\} \sigma i\} \)

From Definition [58] we are given that
\( \forall I, T', (p, T', e) \in [\tau_1[I/i]]_\varepsilon \quad \text{(F-SIF0)} \)

Also from Definition [58] it suffices to prove that
\( \forall I', T'' < T', (p, T'', e) \in [\tau_2[I'/i]]_\varepsilon \)

This means given some \(I', T'' < T\) and we need to prove
\( (p, T'', e) \in [\tau_2[I'/i]]_\varepsilon \quad \text{(F-SIF1)} \)

\( \text{IH: } \{(\tau_1) \sigma i \cup \{i \mapsto I'\}\} \subseteq \{(\tau_2) \sigma i \cup \{i \mapsto I'\}\} \)

Instantiating (F-SIF0) with \(I', T''\) we get
\( (p, T'', e) \in [\tau_1[I'/i]]_\varepsilon \)

and finally from IH we get the desired

13. sub-constraint:

\[
\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 \quad \Theta; \Delta \vdash c_2 \implies c_1 \quad \text{sub-constraint}
\]

To prove: \( \forall (p, T, \Lambda.e) \in \{\{c_1 \Rightarrow \tau_1\} \sigma i\}, (p, T, \Lambda.e) \in \{\{c_2 \Rightarrow \tau_2\} \sigma i\} \)

This means given some \((p, T, \Lambda.e) \in \{\{c_1 \Rightarrow \tau_1\} \sigma i\}\) we need to prove
\( (p, T, \Lambda.e) \in \{\{c_2 \Rightarrow \tau_2\} \sigma i\} \)

From Definition [58] we are given that
\( \forall T' < T . \vdash c_1 \implies (p, T', e) \in [\tau_1 \sigma i]_\varepsilon \quad \text{(F-SC0)} \)

Also from Definition [58] it suffices to prove that
\( \forall T'' < T . \vdash c_2 \implies (p, T'', e) \in [\tau_2 \sigma i]_\varepsilon \)

This means given some \(T'' < T\) s.t. \(\vdash c_2\) and we need to prove
\( (p, T'', e) \in [\tau_2 \sigma i]_\varepsilon \quad \text{(F-SC1)} \)

Since we are given that \(\Theta; \Delta \vdash c_2 \implies c_1\) therefore we know that \(\vdash c_1\)

Hence from (F-SC0) we have
\( (p, T'', e) \in [\tau_1 \sigma i]_\varepsilon \quad \text{(F-SC2)} \)

\( \text{IH: } \{(\tau_1) \sigma i\} \subseteq \{(\tau_2) \sigma i\} \)

Therefore we get the desired from IH and (F-SC2)

14. sub-CAnd:

\[
\Psi; \Theta; \Delta \vdash \tau_1 <: \tau_2 \quad \Theta; \Delta \vdash c_1 \implies c_2 \quad \text{sub-CAnd}
\]

To prove: \( \forall (p, T, v) \in \{\{c_1 \& \tau_1\} \sigma i\}, (p, T, v) \in \{\{c_2 \& \tau_2\} \sigma i\} \)

This means given some \((p, T, v) \in \{\{c_1 \& \tau_1\} \sigma i\}\) we need to prove
\( (p, T, v) \in \{\{c_2 \& \tau_2\} \sigma i\} \)

From Definition [58] we are given that
\( \vdash c_1 \& (p, T, e) \in [\tau_1 \sigma i]_\varepsilon \quad \text{(F-SCA0)} \)

Also from Definition [58] it suffices to prove that
\( \vdash c_2 \& (p, T, e) \in [\tau_2 \sigma i]_\varepsilon \)

Since we are given that \(\Theta; \Delta \vdash c_2 \implies c_1\) and \(\vdash c_1\) therefore we also know that \(\vdash c_2\)
Also from (F-SCA0) we have \((p, T, e) \in [\tau_1 \sigma_i]_\psi\) (F-SCA1)

\(\text{IH: } \llbracket (\tau_1) \sigma_i \rrbracket \subseteq \llbracket (\tau_2) \sigma_i \rrbracket\)

Therefore we get the desired from IH and (F-SCA1)

15. sub-potArrow:

\[
\frac{\Psi; \Theta; \Delta \vdash k'}{\Psi; \Theta; \Delta \vdash \begin{cases} \llbracket (k) (\tau_1 \rightarrow \tau_2) \rrbracket < \llbracket (k') \tau_1 \rightarrow \tau_2 \rrbracket \\ \text{sub-potArrow} \end{cases}}
\]

To prove: \(\forall (p, T, \lambda x. e) \in \llbracket (\llbracket (k) (\tau_1 \rightarrow \tau_2) \rrbracket \sigma_i). (p, T, \lambda x. e) \in \llbracket (\llbracket (k') \tau_1 \rightarrow \tau_2 \rrbracket \sigma_i)\rrbracket\)

This means that some \((p, T, \lambda x. e) \in \llbracket (\llbracket (k') \tau_1 \rightarrow \tau_2 \rrbracket \sigma_i)\rrbracket\) we need to prove \((p, T, \lambda x. e) \in \llbracket (\llbracket (k') \tau_1 \rightarrow \tau_2 \rrbracket \sigma_i)\rrbracket\)

From Definition 58 we are given that \(\exists p', p' + k \leq p \land (p', T, \lambda x. e) \in \llbracket (\tau_1 \rightarrow \tau_2) \sigma_i)\rrbracket\) (F-SPA0)

Again from Definition 58 we know that \(\forall p'', e', T'' \llbracket (p'', T'' e') \in \llbracket (\tau_1) \sigma_i \rrbracket \implies (p' + p'', T', e[e'/x]) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

Also from Definition 58 it suffices to prove that \(\forall p'', e', T'' \llbracket (p'', T'' e') \in \llbracket (\tau_1) \sigma_i \rrbracket \implies (p + p'', T', e[e'/x]) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

This means that some \((p'', e', T'' s.t. (p'', T'', e'') \in \llbracket (\tau_1) \sigma_i \rrbracket \) we need to prove \((p + p'', T'', e[e'/x]) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

Applying Definition 58 on (F-SPA2) we get \(\forall v_f, t'' < T''. e[e'/x] v_f \implies (p + p'', T'', t' - t', v_f) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

This means that some \(v_f, t'' < T''. e[e'/x] v_f\) and we need to prove that \((p + p'', T'' - t', v_f) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

This means From Definition 58 it suffices to prove that \(\exists p'_1, p'_2 + (k + k') \leq (p + p'') \land (p'_1, T'' - t', v_f) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

Also since we are given that \((p'', T'', e'') \in \llbracket (\tau_1) \sigma_i \rrbracket \) we apply Definition 58 on it to obtain \(\forall t < T'', e'' v'' \downarrow v \implies (p'', T'' - t', v') \in \llbracket (\tau_1) \sigma_i \rrbracket\)

Also since we are given that \(e[e'/x] v_f \) therefore we also know that \(\exists t'' < t'' < T''. e'' v'' \downarrow v\)

Instantiating with \(t'', e''\) we get \((p'', T'' - t'', v'') \in \llbracket (\tau_1) \sigma_i \rrbracket\)

Again using Definition 58 we know that we are given \(\exists p'_1, p'_2 + k' \leq p'' \land (p'_1, T'' - t', v') \in \llbracket (\tau_1) \sigma_i \rrbracket\)

Since \((p'_1, T'' - t', v') \in \llbracket (\tau_1) \sigma_i \rrbracket\) therefore from Definition 58 we also have \((p'_1, T'' - t', v'') \in \llbracket (\tau_1) \sigma_i \rrbracket\)

Instantiating (F-SPA1) with \(p'_1, v', T'' - t''\) we get \((p' + p'_1, T'' - t', e[v'/x]) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

From Definition 58 this means that \(\forall t'' < T'' - t', v_f, e[v'/x] v_f \implies (p' + p'_1, T'' - t' - t'', v_f) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

Since we know that \(e[v'/x] v_f\) therefore we also know that \(\exists t'' < t'' < T''. e'' v'' v_f\) s.t. \(t'' + t'' \leq t'\)

Since we already know that \(t'' < t'' < T''. e'' v'' v_f\) therefore we have \(t'' + t'' \leq t'' < T''\).

Instantiating (F-SPA4.1) with \(t''\) we get \((p' + p'_1, T'' - t'' - t'', v_f) \in \llbracket (\tau_2) \sigma_i \rrbracket\)

Since from (F-SPA0) we know that \(p' + k' \leq p\)

And from (F-SPA3) we know that \(p' + k' \leq p''\)

We add the two to get \(p' + p'_1 + k' + k'' \leq p + p''\) (F-SPA6)

In order to prove (F-SPA4) we choose \(p'_2\) as \(p' + p''\)

and we get the desired from (F-SPA6) and (F-SPA5) and Lemma 61.
16. sub-potZero:

\[ \Psi; \Theta; \Delta \vdash \tau <: [0] \tau \quad \text{sub-potZero} \]

To prove: \( \forall (p, T, v) \in [\tau \sigma i], (p, T, v) \in [[0] \tau \sigma i] \)

This means that given \( (p, T, v) \in [\tau \sigma i] \)

And we need to prove \( (p, T, v) \in [[0] \tau \sigma i] \)

From Definition 58 it suffices to prove that \( \exists p', p' + 0 \leq p \land (p', T, v) \in [\tau \sigma i] \)

We choose \( p' \) as \( p \) and we get the desired

17. sub-familyAbs:

\[ \Psi; \Theta; i : S \vdash \tau <: \tau' \quad \text{sub-familyAbs} \]

To prove:

\( \forall f \in [\lambda i : S. \tau \sigma i]. f \in [\lambda i : S. \tau' \sigma i] \)

This means given \( f \in [\lambda i : S. \tau \sigma i] \) and we need to prove \( f \in [\lambda i : S. \tau' \sigma i] \)

This means from Definition 58 we are given

\( \forall I. f I \in [\tau[I/i] \sigma i] \quad (\text{F-SFAbs0}) \)

This means from Definition 58 we need to prove

\( \forall I'. f I' \in [\tau'[I'/i] \sigma i] \)

This further means that given some \( I' \) we need to prove

\( f I' \in [\tau'[I'/i] \sigma i] \quad (\text{F-SFAbs1}) \)

Instantiating (F-SFAbs0) with \( I' \) we get

\( f I' \in [\tau[I'/i] \sigma i] \)

From IH we know that \( [\tau \sigma i \cup \{ i \mapsto I' \ i \}] \subseteq [\tau' \sigma i \cup \{ i \mapsto I' \ i \}] \)

And this completes the proof.

18. Sub-tfamilyApp1:

\[ \Psi; \Theta; \Delta \vdash \lambda i : S. \tau I <: \tau[I/i] \quad \text{sub-familyApp1} \]

To prove: \( \forall (p, T, v) \in [\lambda i : S. \tau I \sigma i], (p, T, v) \in [\tau[I/i] \sigma i] \)

This means given \( (p, T, v) \in [\lambda i : S. \tau I \sigma i] \) and we need to prove \( (p, T, v) \in [\tau[I/i] \sigma i] \)

This means from Definition 58 we are given

\( (p, T, v) \in [\lambda i : S. \tau I \sigma i] \)

This further means that we have

\( (p, T, v) \in f I \tau \) where \( f I = [\tau \sigma[I/i]] \)

This means we have \( (p, T, v) \in [\tau \sigma[I/i]] \)

And this completes the proof.

19. Sub-tfamilyApp2:

\[ \Psi; \Theta; \Delta \vdash \tau[I/i] <: \lambda i : S. \tau I \quad \text{sub-familyApp2} \]

To prove: \( \forall (p, T, v) \in [\tau[I/i] \sigma i], (p, T, v) \in [\lambda i : S. \tau I \sigma i] \)

This means given \( (p, T, v) \in [\tau[I/i] \sigma i] \) and we need to prove

\( (p, T, v) \in [\lambda i : S. \tau I \sigma i] \quad (\text{Sub-tF0}) \)

This means from Definition 58 it suffices to prove that

\( (p, T, v) \in [\lambda i : S. \tau I \sigma i] \)

It further suffices to prove that

\( (p, T, v) \in f I \tau \) where \( f I \tau = [\tau \sigma[I/i]] \)
Lemma 70 (Expression subtyping lemma). \( \forall \Psi, \Theta, \Delta \vdash \tau < : \tau' \implies [\tau \sigma]_{\xi} \subseteq [\tau' \sigma]_{\xi} \)

\[ \Psi; \Theta; \Delta \vdash \sum_{a < I} K! a!_I \tau < : a < I [K] \tau \]

20. sub-bSum:

To prove: \( \forall (p, T, v) \in [\sum_{a < I} K! a!_I \tau \sigma]_{\xi} \implies (p, T, v) \in [a < I [K] \tau \sigma]_{\xi} \)

This means given some \((p, T, v)\) s.t \((p, T, v) \in [\sum_{a < I} K! a!_I \tau \sigma]_{\xi}\) it suffices to prove that \((p, T, v) \in [a < I [K] \tau \sigma]_{\xi}\)

This means from Definition \(58\) we are given that

\[ \exists p', p' + \sum_{a < I} K \leq p \land (p', T, v) \in [a < I [K] \tau \sigma]_{\xi} \]  

(Sub-BS0)

Since \((p', T, v) \in [a < I [K] \tau \sigma]_{\xi}\) therefore again from Definition \(58\) it means that \(\exists e', v = \! v'\) and

\[ \exists p_0, \ldots, p'_{I-1}, p_0 + \ldots + p'_{I-1} \leq p' \land \forall 0 \leq i < I. (p_i, T, e') \in [\tau[i/a] \sigma_i]_{\xi} \]  

(Sub-BS1)

Since \(\forall 0 \leq i < I. \forall i < I. \forall T, v'' < T. v'' \Downarrow T. v' \
\]  

\([\tau[i/a] \sigma_i]_{\xi}\) therefore from Definition \(58\) we have

\[ \forall 0 \leq i < I. \forall T, v''. v'' \Downarrow T \implies (p_i, T, v') \in [\tau[i/a] \sigma_i]_{\xi} \]

(Sub-BS1.1)

Since we know that \(v = \! v'\) therefore it suffices to prove that \((p, T, \! v') \in [a < I [K] \tau \sigma]_{\xi}\)

From Definition \(58\) it further suffices to prove that

\[ \exists p_0, \ldots, p'_{I-1}, p_0 + \ldots + p'_{I-1} \leq p \land \forall 0 \leq i < I. (p_i, T, e') \in [K] \tau[i/a] \sigma_i]_{\xi} \]

We choose \(p_0 = p_0 + K[0/a] \ldots p'_{I-1} = p_{I-1} + K[(I - 1)/a]\) and it suffices to prove that

- \(p_0 + \ldots + p'_{I-1} \leq p\)
  
  We need to prove that
  
  \( (p_0 + K[0/a] \ldots + p_{I-1} + K[(I - 1)/a]) \leq p \)
  
  We get this from (Sub-BS0) and (Sub-BS1)

- \(\forall 0 \leq i < I. (p_i, T, e') \in [K] \tau[i/a] \sigma_i]_{\xi}\):
  
  Given some 0 \(\leq i < I\) it suffices to prove that
  
  \( (p_i, T, e') \in [K] \tau[i/a] \sigma_i]_{\xi}\)

Since \(p_i\) is \(p_i + K[i/a]\) therefore it suffices to prove that

\( (p_i + K[i/a], T, e') \in [K[i/a]] \tau[i/a] \sigma_i]_{\xi}\)

From Definition \(58\) we need to prove that

\[ \forall \forall v''. v'' \Downarrow T. e' \Downarrow v' \implies (p_i + K[i/a], T - t'' \Downarrow v'') \in [K[i/a]] \tau[i/a] \sigma_i]_{\xi}\]

This means given some \(v'' \Downarrow v'\) we need to prove that

\( (p_i + K[i/a], T - t'' \Downarrow v'') \in [K[i/a]] \tau[i/a] \sigma_i]_{\xi}\)

From Definition \(58\) it suffices to prove that

\[ \exists p'', t'' < T. e' \Downarrow v' \implies (t'' \Downarrow v'') \in [\tau[i/a] \sigma_i]_{\xi}\]

We choose \(p''\) as \(p_i\) and we need to prove

\( (p_i, T - t'' \Downarrow v') \in [\tau[i/a] \sigma_i]_{\xi}\)

Instantiating (Sub-BS1.1) with the given \(i\) and \(v', t''\) we get the desired
Proof. From Theorem 64 we know that $(0, t', e') \in [[\tau \alpha t]]$
And finally from Lemma 69 we get the desired.

\[\Box\]

**Theorem 71** (Soundness). \(\forall e, n, n', \tau \in \text{Type}, t.\)
\(\vdash e : M n \tau \land e \Downarrow v \implies n' \leq n\)

**Proof.** From Theorem 64 we know that $(0, t + 1, e) \in [[M n \tau]]$

From Definition 68 this means we have
\(\forall t' < t + 1.e \Downarrow v' \implies (0, t + 1 - t'v') \in [[M n \tau]]\)

From the evaluation relation we know that $e \Downarrow v$ therefore we have
$(0, t + 1, e) \in [[M n \tau]]$

Again from Definition 68 it means we have
\(\forall t'' < t + 1.e \Downarrow v'' \implies \exists p', n' + p' \leq 0 + n \land (p', t + 1 - t''v', v) \in [[\tau]]\)

Since we are given that $e \Downarrow v$ therefore we have
\(\exists p', n' + p' \leq 0 + n \land (p', 1, v) \in [[\tau]]\)

Since $p' \geq 0$ therefore we get $n' \leq n$

\[\Box\]

**Theorem 72** (Soundness). \(\forall e, n, n', \tau \in \text{Type}.\)
\(\vdash e : [n] 1 \rightarrow M 0 \tau \land e () \Downarrow t_1 - \Downarrow v \implies n' \leq n\)

**Proof.** From Theorem 64 we know that $(0, t_1 + t_2 + 2, e) \in [[n] 1 \rightarrow M 0 \tau]$

Therefore from Definition 68 we know that
\(\forall t' < t_1 + t_2 + 2.e \Downarrow v \implies (0, t_1 + t_2 + 2 - t', v) \in [[n] 1 \rightarrow M 0 \tau]\)

(S0)

Since we know that $e () \Downarrow t_1$ therefore from E-app we know that \(\exists \lambda x.e'\)

Instantiating (S0) with $t_1, \lambda x.e'$ we get $(0, t_2 + 2, \lambda x.e') \in [[n] 1 \rightarrow M 0 \tau]\)

This means from Definition 68 we have
\(\forall p', e', t'' < t_2 + 2.(p', t'', e'[e''/x]) \in [[n] 1 \rightarrow M 0 \tau]\)

(S1)

Claim: $\forall t, (I, t, c) \in [[I 1]]$

**Proof:**
From Definition 68 it suffices to prove that
$() \Downarrow v \implies (I, t, v) \in [[I 1]]$

Since we know that $v = ()$ therefore it suffices to prove that
$(I, t, v) \in [[I 1]]$

From Definition 68 it suffices to prove that
\(\exists p', p' + I \leq I \land (p', t, v) \in [1]\)$

We choose $p'$ as 0 and we get the desired

Instantiating (S1) with $n, (), t_2 + 1$ we get $(n, t_2 + 1, e'[()/x]) \in [[M 0 \tau]]$

This means again from Definition 68 we have
\(\forall t' < t_2 + 1.e'[()/x] \Downarrow v' \implies (n, t_2 + 1 - t'v', v') \in [[M 0 \tau]]\)

From E-val we know that $v' = e'[()/x]$ and $t' = 0$ therefore we have
$(n, t_2 + 1, e'[()/x]) \in [[M 0 \tau]]$

Again from Definition 68 we have
\(\forall t' < t_2 + 1.e'[()/x] \Downarrow v'' \implies \exists p', n' + p' \leq 0 + n \land (p', t_2 + 1 - t', v'') \in [[\tau]]\)

Since we are given that $e \Downarrow t_1 - \Downarrow v \implies n' \leq n$

Since $p' \geq 0$ therefore we have $n' \leq n$

\[\Box\]
### B.5 Embedding dlPCF

**Type translation**

\[
\begin{align*}
\langle b \rangle & \quad \vdash \quad b \\
\langle [a < I]_\tau \rightarrow \tau_2 \rangle & \quad \vdash \quad (\tau_1 \llcorner M 0 (\tau_1)) \rightarrow [I] 1 \rightarrow M 0 (\tau_2)
\end{align*}
\]

**Judgment translation**

\[
\Theta; \Delta; \Gamma \vdash_K e_d : \tau \quad \Rightarrow \quad : \Theta; \Delta; ([\Gamma]; \vdash e_a : [K + \text{count}(\Gamma)] 1 \rightarrow M 0 (\tau)]
\]

where

\[
\begin{align*}
\text{count}(.) & \quad = \quad 0 \\
\text{count}(\Gamma, x : [a < I]_\tau) & \quad = \quad \text{count}(\Gamma) + I
\end{align*}
\]

**Definition 73** (Context translation).

\[
\begin{align*}
\langle 0 \rangle & \quad \vdash \quad . \\
\langle \Gamma, x : [a < I]_\tau \rangle & \quad \vdash \quad (\Gamma), x : a < I \rightarrow M 0 (\tau)
\end{align*}
\]

**Expression translation**

\[
\begin{align*}
\Theta; \Delta \models J \geq 0 & \quad \Theta; \Delta \models I \geq 1 & \quad \Theta; \Delta \vdash [0 / a] : \tau & \quad \Theta; \Delta \models [a < I]_\tau \sigma \downarrow \quad \Theta; \Delta \models \Gamma \downarrow
\end{align*}
\]

\[
\Theta; \Delta; \Gamma, x \vdash [a < I]_\tau e : \tau_2 \rightarrow e_t
\]

\[
\Theta; \Delta; \Gamma; x : [a < I]_\tau \vdash J e : \tau_2 \rightarrow e_t
\]

\[
\lambda p_1 \cdot \text{ret} \lambda y_1 \cdot \lambda p_2 \cdot \text{let} ! x = y \text{ in release } = p_1 \text{ in release } = p_2 \text{ in bind } a = \text{store()} \text{ in } e_t a
\]

\[
\Theta; \Delta; \Gamma \vdash J e_1 : ([a < I]_\tau) \rightarrow \tau_2 \rightarrow e_{t_1}
\]

\[
\Theta, a; \Delta, a < I; \Delta \vdash_K e_2 : \tau_1 \rightarrow e_{t_2}
\]

\[
\Gamma' \equiv \Gamma \oplus \sum_{\tau < I} \Delta \quad H \geq J + I + \sum_{\tau < I} K
\]

\[
\Theta; \Delta; \Gamma' \vdash_H e_1 e_2 : \tau_2 \rightarrow
\]

\[
\lambda p \cdot \text{release } = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{t_1} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b \text{ (coerce1 !} e_{t_2} c) \text{ d}
\]

\[
\Theta, b; \Delta, b < L; \Gamma, x : [a < I]_\tau e : \tau \rightarrow e_t
\]

\[
\tau[0/a] \llcorner \mu \quad \Theta, a, b; \Delta, a < I, b < L; \Gamma \vdash [b + 1 + \boxplus^0_b 1, a] / b] : \sigma
\]

\[
\Gamma' \equiv \sum_{\tau < L} \Gamma \quad L, M \geq \sum_{\tau < L} \Delta \quad N \geq M - 1 + \sum_{\tau < L} K
\]

\[
\Theta; \Delta; \Gamma' \vdash_N \text{fix} x.e : \mu \rightarrow E_0
\]

\[
E_0 = \text{fix} Y . E_1
\]

\[
E_1 = \lambda p . E_2
\]

\[
E_2 = \text{release } = p \text{ in } E_3
\]

\[
E_3 = \text{bind } A = \text{store()} \text{ in } E_4
\]

\[
E_4 = \text{let} ! x = (E_{4,1} E_{4,2}) \text{ in } E_5
\]

\[
E_{4,1} = \text{coerce1 !} Y
\]

\[
E_{4,2} = (\lambda u . !()) A
\]

\[
E_5 = \text{bind } C = \text{store()} \text{ in } E_6
\]

\[
E_6 = e_t C
\]

### B.5.1 Type preservation

**Theorem 74** (Type preservation: dlPCF to Λ-amor). If \(\Theta; \Delta; \Gamma \vdash e : \tau\) in dlPCF then there exists \(e'\) such that \(\Theta; \Delta; \Gamma \vdash e : \tau \rightarrow e'\) such that there is a derivation of \(\vdash \Theta; \Delta; ([\Gamma]; \vdash e : [I + \text{count}(\Gamma)] 1 \rightarrow M 0 (\tau)]\) in Λ-amor.

**Proof.** Proof by induction on the \(\Theta; \Delta; \Gamma \vdash e : \tau\).
\[ \text{• var:} \]
\[
\Theta, \Delta :: J \geq 0 \quad \Theta, \Delta :: I \geq 1 \quad \Theta, \Delta :: a \not\in \vartheta \quad \Theta, \Delta :: a < I \sigma \quad \Theta, \Delta :: \Gamma \downarrow \vartheta
\]
\[
\Theta, \Delta :: \Gamma, x :: [a < I] \sigma \vdash \lambda x. \sigma \leftarrow \rho \text{ release } = p \text{ in } \text{bind } = \uparrow^1 \text{ in } x
\]
D2:
\[
\Theta, \Delta :: [a < I] \sigma \vdash \tau \quad \text{Lemma 70}
\]
D1:
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma \vdash x :: M_0 \sigma[0/a] \quad \text{T-var2}
\]
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma, \vdash x :: M_0 \sigma[0/a] \quad \text{Lemma 80}
\]
D0:
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma, \vdash \uparrow^1 :: M_1 \Gamma
\]
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma, \vdash \text{bind } = \uparrow^1 \text{ in } x :: M(I + J + \text{count}(\Gamma)) \sigma[0/a] \quad \text{bind}
\]
Main derivation:
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma, \vdash p :: (I + J + \text{count}(\Gamma)) \Gamma \vdash p :: (I + J + \text{count}(\Gamma)) \Gamma
\]
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma, \vdash \text{release } = p \text{ in } \text{bind } = \uparrow^1 \text{ in } x :: M_0 \rho
\quad \text{T-release}
\]
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] M_0 \sigma, \vdash \lambda x. \sigma \leftarrow \rho \text{ release } = p \text{ in } \text{bind } = \uparrow^1 \text{ in } x :: ((I + J + \text{count}(\Gamma)) \Gamma \vdash \sigma \leftarrow \rho)
\quad \text{T-lam}
\]
\[ \text{• lam:} \]
\[
\Theta, \Delta :: \Gamma, x :: [a < I] \tau_1 + J e :: \tau_2 \Rightarrow e_t
\]
\[
\lambda p_1, \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
\]
E0 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
E1 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
E2 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
E3 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
E4 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
E5 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
E6 = \text{ret } \lambda y. \lambda p_2, \text{let } ! x = y \text{ in } \text{release } = p_1 \text{ in } \text{release } = p_2 \text{ in } \text{bind } a = \text{store}(x) \text{ in } e_t a
T_0 = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_1 \Rightarrow \tau_2
T_{0.1} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_1 \Rightarrow \tau_2
T_{0.2} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_1 = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_2 = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_{2.1} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_{2.2} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_3 = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_{3.1} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_4 = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_{4.1} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_{4.2} = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
T_5 = [J + \text{count}(\Gamma)] \Gamma \vdash M_0 \Gamma \downarrow [\text{a < I}] \Gamma \vdash \tau_2
D6:
\[
\vdots \Theta, \Delta :: a :: [J + \text{count}(\Gamma)] \Gamma \vdash a :: [J + \text{count}(\Gamma)] \Gamma \downarrow \vartheta
\]
D5:
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] \sigma \vdash e_t \quad \text{T-lam}
\]
D4:
\[
\vdots \Theta, \Delta :: \Gamma, x :: [a < I] \sigma \vdash e_t \quad \text{T-lam}
\]
135
D3:
\[ ; \Theta; \Delta; \cdot \vdash \text{store()} : T_{4,1} \quad \text{store} \]
\[ ; \Theta; \Delta; (\Gamma), x : a \leftarrow M 0 \{ t_1 \}; \cdot \vdash E_{4,3} : T_{4,2} \quad \text{bind} \]

D4:
\[ ; \Theta; \Delta; p_2 : T_{3,1} \vdash p_2 : T_{3,1} \]
\[ ; \Theta; \Delta; (\Gamma), x : a \leftarrow M 0 \{ t_1 \}; p_2 : T_{3,1} \vdash E_{4,2} : T_{4,3} \quad \text{bind} \]

D2:
\[ ; \Theta; \Delta; p_1 : T_{0,2} \vdash p_1 : T_{0,2} \]
\[ ; \Theta; \Delta; (\Gamma), x : a \leftarrow M 0 \{ t_1 \}; p_1 : T_{0,2}, p_2 : T_{3,1} \vdash E_{4,1} : T_{4} \quad \text{release} \]

D1:
\[ ; \Theta; \Delta; y : T_{2,1} \vdash y : T_{2,1} \]
\[ ; \Theta; \Delta; (\Gamma); p_1 : T_{0,2}, y : T_{2,1} \vdash E_{4} : T_{4} \quad \text{T-subExpE} \]
\[ ; \Theta; \Delta; (\Gamma); p_1 : T_{0,2}, y : T_{2,1} \vdash E_{3} : T_{3} \quad \text{lam} \]

Main derivation:
\[ ; \Theta; \Delta; (\Gamma); p_1 : T_{0,2} \vdash E_{2} : T_{2} \quad \text{lam} \]
\[ ; \Theta; \Delta; (\Gamma); p_1 : T_{0,2} \vdash E_{1} : T_{1} \quad \text{ret} \]
\[ ; \Theta; \Delta; (\Gamma); \cdot \vdash E_{0} : T_{0,1} \quad \text{lam} \]

• app:
\[ \Theta; \Delta; \Gamma_1 \vdash f e_1 : ([a < I] t_1) \rightarrow \tau_2 \leadsto e_{t_1} \]
\[ \Theta; a; \Delta; a \leftarrow I; \Gamma_2 \vdash K e_2 : \tau_1 \leadsto e_{t_2} \]
\[ \Gamma' \equiv \Gamma_1 \oplus \sum_{a < I} \Gamma_2 \]
\[ H \geq J + I + \sum_{a < I} K \]

\[ \Theta; \Delta; \Gamma' \vdash \text{app} e_1 e_2 : \tau_2 \leadsto \]

\[ \lambda \text{p.release} \leftarrow p \quad \text{in} \quad \text{bind} \ a = \text{store()} \quad \text{in} \quad \text{bind} \ b = e_{t_1} \quad \text{a in} \quad \text{bind} \ c = \text{store()} \quad \text{in} \quad \text{bind} \ d = \text{store()} \quad \text{in} \quad b \left( \text{coerce1} ! e_{t_2} \right) \quad \text{d} \]

\[ E_0 = \lambda \text{p.E_1} \]
\[ E_1 = \text{release} \leftarrow p \quad \text{in} \quad E_2 \]
\[ E_2 = \text{bind} \ a = \text{store()} \quad \text{in} \quad E_3 \]
\[ E_3 = \text{bind} \ b = e_{t_1} \quad \text{a in} \quad E_4 \]
\[ E_4 = \text{bind} \ c = \text{store()} \quad \text{in} \quad E_5 \]
\[ E_5 = \text{bind} \ d = \text{store()} \quad \text{in} \quad b \left( \text{coerce1} e_{t_2} \right) \quad \text{d} \]

\[ T_0 = [H + \text{count}(\Gamma')] \; 1 \leadsto M 0 \{ \tau_2 \} \]
\[ T_{0,1} = [J + I + \sum_{a < I} K + \text{count}(\Gamma_1) + \text{count}(\sum_{a < I} \Gamma_2)] \; 1 \leadsto M 0 \{ \tau_2 \} \]
\[ T_{0,2} = [J + I + \sum_{a < I} K + \text{count}(\Gamma_1) + \text{count}(\sum_{a < I} \Gamma_2)] \]
\[ T_{0,3} = M(\sum_{a < I} K + \text{count}(\Gamma_1) + \text{count}(\sum_{a < I} \Gamma_2)) \; \{ \tau_2 \} \]
\[ T_1 = [(J + \text{count}(\Gamma'))] \; 1 \leadsto M 0 \{ ([a < I] t_1) \rightarrow \tau_2 \} \]
\[ T_{1,1} = [(J + \text{count}(\Gamma'))] \]
\[ T_{1,2} = [J + \sum_{a < I} K + \text{count}(\sum_{a < I} \Gamma_2)] \; \{ \tau_2 \} \]
\[ T_{1,3} = M(\sum_{a < I} K + \text{count}(\sum_{a < I} \Gamma_2)) \; \{ \tau_2 \} \]
\[ T_{1,4} = [\sum_{a < I} K +\text{count}(\sum_{a < I} \Gamma_2)] \; !_{a < I} 1 \quad [\sum_{a < I} (K + \text{count}(\Gamma_2))] \; !_{a < I} 1 \]
\[ T_{1,5} = \text{store}() \quad (K + \text{count}(\Gamma_2)) \; 1 \]
\[ T_{1,6} = M 0 \{ ([a < I] t_1) \rightarrow \tau_2 \} \]
\[ T_2 = [J + \text{count}(\Gamma')] \; 1 \leadsto M 0 \{ !_{a < I} M 0 \{ t_1 \} \} \rightarrow M 0 \{ \tau_2 \} \]
\[ T_{2,1} = [(J + \text{count}(\Gamma'))] \]
\[ T_{2,2} = M 0 \{ !_{a < I} M 0 \{ t_1 \} \} \rightarrow [I] \; 1 \rightarrow M 0 \{ \tau_2 \} \]
\[ T_{2,21} = !_{a < I} M 0 \{ t_1 \} \rightarrow [I] \; 1 \rightarrow M 0 \{ \tau_2 \} \]
\[ T_{2,22} = [I] \; 1 \rightarrow M 0 \{ \tau_2 \} \]
\[ T_3 = M 0 \{ \tau_2 \} \]
\[ T_{3,1} = M I \{ \tau_2 \} \]
\[ T_4 = M 0 \{ \tau_1 \} \]
\[
T_{4.1} = \text{let } M_0 \{ \tau_1 \} \\
T_5 = \left[ (K + \text{count(} \Gamma_2 \text{)}) \right] \text{let } M_0 \{ \tau_1 \} \\
T_{5.0} = \text{let } [ (K + \text{count(} \Gamma_2 \text{)}) \right] \text{let } M_0 \{ \tau_1 \} \\
T_{5.1} = \text{let } [ (K + \text{count(} \Gamma_2 \text{)}) \right] \text{let } lM_0 \{ \tau_1 \} \\
\]

D0.7:
\[
\vdash \Theta; \Delta; \vdash c : T_{1.15} \vdash c : T_{1.15} \quad \text{T-var}
\]

D0.6:
\[
\vdash \Theta; a; \Delta; \vdash e_{t_2} : T_5 \quad \text{IH} \\
\vdash \Theta; \Delta; \vdash \sum_{a < l} [ \Gamma_2 ]; \vdash e_{t_2} : T_{5.0} \quad \text{subExpr} \quad \text{D0.7} \\
\vdash \Theta; \Delta; \vdash \sum_{a < l} [ \Gamma_2 ]; c : T_{1.15} \vdash c : T_{1.15} \quad \text{coerce \!} e_{t_2} c : T_{4.1} \quad \text{Lemma S1} \\
\]

D0.5:
\[
\vdash \Theta; \Delta; \vdash \sum_{a < l} [ \Gamma_2 ]; b : T_{2.21} \vdash b : T_{2.21} \quad \text{D0.6} \\
\vdash \Theta; \Delta; \vdash \sum_{a < l} [ \Gamma_2 ]; b : T_{2.21}, c : T_{1.15} \vdash b (\text{coerce} \! e_{t_2} c) : T_{2.22} \quad \text{T-app}
\]

D0.4:
\[
\vdash \Theta; \Delta; \vdash d : \text{let } [ \Gamma_2 ]; \vdash d : T_3 \quad \text{D0.4} \\
\vdash \Theta; \Delta; \vdash \sum_{a < l} [ \Gamma_2 ]; b : T_{2.21}, c : T_{1.15}, d : \text{let } [ \Gamma_2 ]; \vdash d : T_3 \quad \text{T-app}
\]

D0.21:
\[
\vdash \Theta; \Delta; \vdash T_{1.14} \vdash T_{1.15} \quad \text{sub-bSum}
\]

D0.2:
\[
\vdash \Theta; \Delta; \vdash !() : \text{let } [ \Gamma_2 ]; \vdash !() : T_{1.13} \quad \text{D0.21} \\
\vdash \Theta; \Delta; \vdash \text{let } [ \Gamma_2 ]; \vdash !() : T_{1.13} \quad \text{T-sub} \quad \text{D0.3} \\
\vdash \Theta; \Delta; \vdash \sum_{a < l} [ \Gamma_2 ]; b : T_{2.21} \vdash E_4 : T_{1.12} \quad \text{bind}
\]

D0.12:
\[
\vdash \Theta; \Delta; \vdash a : T_{2.1} \vdash a : T_{2.1} \quad \text{T-var}
\]

D0.11:
\[
\vdash \Theta; \Delta; \vdash e_{t_1} : T_1 \quad \text{IH1}
\]

D0.1:
\[
\vdash \Theta; \Delta; \vdash \text{let } [ \Gamma_1 ]; \vdash e_{t_1} : T_1 \quad \text{D0.11} \\
\vdash \Theta; \Delta; \vdash \text{let } [ \Gamma_1 ]; \vdash e_{t_1} a : T_{2.2} \quad \text{app} \quad \text{D0.2} \\
\vdash \Theta; \Delta; \vdash [ \Gamma_1 ] \vdash \sum_{a < l} [ \Gamma_2 ]; a : T_{2.1} \vdash E_3 : T_{1.12} \quad \text{bind}
\]

D0:
\[
\vdash \Theta; \Delta; \vdash \text{let } [ \Gamma_1 ] \vdash \sum_{a < l} [ \Gamma_2 ]; \vdash E_2 : T_{0.3} \quad \text{bind}
\]

137
D0.0:

\[ \Theta; \Delta \vdash \Gamma' \subseteq \Gamma_1 \oplus \sum_{a < I} \Gamma_a \]

By inversion

\[ \Theta; \Delta \vdash (\Gamma') <: (\Gamma_1 \oplus \sum_{a < I} \Gamma_a) \]

Lemma 77

Main derivation:

\[ \vdots \Theta; \Delta; \vdash \Gamma' \subseteq \Gamma_1 \oplus \sum_{a < I} \Gamma_a \vdash \lambda p. E : T_0.1 \]
\[ \vdots \Theta; \Delta; \vdash \Gamma_1 \oplus \sum_{a < I} (\Gamma_a) ; p : T_0.1 \vdash E_1 : T_0.2 \]
\[ \vdots \Theta; \Delta; \vdash (\Gamma_1) \oplus \sum_{a < I} \Gamma_a ; p : T_0.1 \vdash E_0 : T_0.11 \]
\[ \vdots \Theta; \Delta; \vdash (\Gamma_1) \oplus \sum_{a < I} \Gamma_a ; \vdash E_0 : T_0.11 \]
\[ \vdots \Theta; \Delta; \vdash (\Gamma_1 \oplus \sum_{a < I} \Gamma_a) ; \vdash E_0 : T_0.11 \]
\[ \vdots \Theta; \Delta; \vdash (\Gamma') ; \vdash E_0 : T_0 \]

Lemma 78

D0.0

T-sub,T-weaken

• fix:

\[ \Theta, b; \Delta, b < L; \Gamma, x : [a < I] \sigma \vdash K \ e : \tau \leadsto e ] \]
\[ \tau[0/\alpha] <: \mu \]
\[ \Theta, a, b; \Delta, a < I, b < L \vdash \tau[(b + 1 + \Theta_b^{b+1,a} I)/b] <: \sigma \]
\[ \Gamma' \subseteq \sum_{b < L} \Gamma \]
\[ L, M \geq \Theta_b^{0,1} I \]
\[ N \geq M - 1 + \sum_{b < L} K \]

\[ \Theta; \Delta; \vdash I' \vdash \mu \]

Lemma 76

Lemma 75

Lemma 77

T-fix

\[ E_0 = \text{fix}$Y.\lambda p. E_1 \]
\[ E_1 = \lambda p. E_2 \]
\[ E_2 = \text{release} - = p \in E_3 \]
\[ E_3 = \text{bind} A = \text{store}() \in E_4 \]
\[ E_4 = \text{let} x = (E_4 \cdot E_4.2) \in E_5 \]
\[ E_{4.1} = \text{coerce}1 \ Y' \]
\[ E_{4.2} = (\lambda u. !()) A \]
\[ E_5 = \text{bind} C = \text{store}() \in E_6 \]
\[ E_6 = e_t C \]

\[ \text{cost}(b') \triangleq \]

if \(0 \leq b' < (\Theta_b^{0,1} I(b))\) then

\[ K(b') + I(b') + \text{count}(\Gamma(b')) + (\sum_{a < I(b')} \text{cost}((b' + 1 + \Theta_b^{b+1,a} I(b)))) \]

else \(0\)

\[ \tau'(b') = [\text{cost}(b')] 1 \rightarrow M \{ \tau(b') \} \]

\[ T_0 = \tau'[(b' + 1 + \Theta_b^{b+1,a} I)/b'] \]

\[ T_0 = [(N + \text{count}(\Gamma'))] 1 \rightarrow M \{ \rho \} \]

\[ T_{0.1} = [(M - 1 + \sum_{b < L} K) + \text{count}(\sum_{b < L} \Gamma')] 1 \rightarrow M \{ \rho \{ 0 \} \} \]

\[ b'' = (b' + 1 + \Theta_b^{b+1,a} I) \]

\[ T_{1.0} = \lambda_{a < I(b')} [\text{cost}(b''')] 1 \rightarrow M \{ \tau(b'') \} \]

\[ T_{1.1} = \lambda_{a < I(b')} [\text{cost}(b'')] 1 \rightarrow M \{ \tau(b'') \} \]

\[ T_{1.11} = M \{ \tau(b'') \} \]

\[ T_{1.12} = M \{ \rho \} \]

\[ T_2 = \sum_{a < I(b')} \text{cost}(b'') \]

\[ T_3 = \sum_{a < I} \text{cost}(b'') \]

\[ T_4 = M(K(b') + I(b') + \text{count}(\Gamma(b'))) \{ \tau(b') \} \]

\[ T_{4.1} = M\{ K(b') + I(b') + \text{count}(\Gamma(b')) \} \{ [K(b') + I(b') + \text{count}(\Gamma(b'))] 1 \}

\[ T_{4.2} = ([K(b') + I(b') + \text{count}(\Gamma(b'))]) 1 \]
\(T_5 = [(K(b') + I(b') + count(\Gamma(b'))) \mid 1 \rightarrow M 0 \| \tau(b')]\)
\(T_{\alpha_0} = 1 \rightarrow !a_{<I}1\)
\(T_{\alpha_1} = [0] (1 \rightarrow !a_{<I}1)\)
\(T_{c_1} = [\sum_{a < I} cost(b')] 1 \rightarrow [\sum_{a < I} cost(b')] !a_{<I}1\)

D5.2:
\[\vdash \Theta, b'; \Delta, b' < L; \vdash C : T_{4.2} \vdash C : T_{4.2}\]

D5.10:
\[\vdash \Theta, b'; \Delta, b' < L; \vdash \tau(b') < \sigma\quad \text{Given}\]
\[\vdash \Theta, b'; \Delta, b' < L; \vdash \{\tau(b')\} < \{\sigma\}\]

Lemma D5.10

D5.1:
\[\vdash \Theta, b'; \Delta, b' < L; [\Gamma], x : a_{<I}(b') T_{1.12}; \vdash c_{i : T5}\]

IH D5.10

T-weaken

D5:
\[\vdash \Theta, b'; \Delta, b' < L; [\Gamma], x : a_{<I}(b') T_{1.11}; \vdash c_{i : T5}\]

D5.1 D5.2

D4:
\[\vdash \Theta, b'; \Delta, b' < L; [\Gamma], x : a_{<I}(b') T_{1.11}; C : T_{4.2} \vdash c_{i : C : M 0 \| \tau(b')}\]

app

D3.2:
\[\vdash \Theta, b'; \Delta, b' < L; Y : a_{<I} T_{0.0}; \vdash !Y : T_{1.0}\]

Lemma D3.2

D3.11:
\[\vdash \Theta, b'; \Delta, b' < L; Y : a_{<I} T_{0.0}; \vdash \text{coerce1} (/!Y) : T_1\]

Lemma D3.11

D3.12:
\[\vdash \Theta, b'; \Delta, b' < L; Y : a_{<I} T_{0.0}; \vdash \text{coerce1} (/!Y) : T_1\]

sub-bSum

Dc2:
\[\vdash \Theta, b'; \Delta, b' < L; \vdash c_{i : T_{d_0.1} < T_{c_1}}\]

sub-potArrow

Dc1:
\[\vdash \Theta, b'; \Delta, b' < L; a < I; \vdash () : 1\]

T-unit

\[\vdash \Theta, b'; \Delta, b' < L; u : 1 \vdash () : a_{<I}1\]

T-subExp1,T-weaken

T-lam

Dc:
\[\vdash \Theta, b'; \Delta, b' < L; \vdash c_{i : T_{d_0.1} < T_{c_1}}\]

T-sub

Dc2

\[\vdash \Theta, b'; \Delta, b' < L; \vdash \lambda u(). T_{d_0}\]

\[\vdash \Theta, b'; \Delta, b' < L; \vdash \lambda u(). T_{c_1}\]

\[\vdash \Theta, b'; \Delta, b' < L; \vdash \lambda u(). T_{d_0}\]

\[\vdash \Theta, b'; \Delta, b' < L; \vdash \lambda u(). T_{c_1}\]

\[\vdash \Theta, b'; \Delta, b' < L; \vdash \lambda u(). T_{d_0}\]

\[\vdash \Theta, b'; \Delta, b' < L; \vdash \lambda u(). T_{c_1}\]

D3.11

\[\vdash \Theta, b'; \Delta, b' < L; \vdash A : T_{2} \vdash A : T_{2}\]

var

\[\vdash \Theta, b'; \Delta, b' < L; \vdash A : T_{2} \vdash (\lambda u().) A : T_{3}\]

T-app D3.12

\[\vdash \Theta, b'; \Delta, b' < L; \vdash A : T_{2} \vdash (\lambda u().) A : T_{3}\]

T-sub D3.11

D3.1

\[\vdash \Theta, b'; \Delta, b' < L; Y : a_{<I} T_{0.0}; A : T_{2} \vdash E_{4.1} E_{4.2} : T_{1.1}\]

D3

\[\vdash \Theta, b'; \Delta, b' < L; \vdash [\Gamma], Y : a_{<I} T_{0.0}; A : T_{2} \vdash E_{4} : T_{4}\]

D2

\[\vdash \Theta, b'; \Delta, b' < L; [\Gamma], \vdash \text{store()} : M(\sum_{a < I(b')}\text{cost(b')}) T_{2}\]

\[\vdash \Theta, b'; \Delta, b' < L; [\Gamma], \vdash \text{store()} : M(\sum_{a < I(b')}\text{cost(b')}) T_{2}\]

\[\vdash \Theta, b'; \Delta, b' < L; [\Gamma], \vdash \text{store()} : M(\sum_{a < I(b')}\text{cost(b')}) T_{2}\]
Proof.

Proof by induction on $\Gamma$

**Lemma 76** (Relation b/w dlPCF context and its translation - bounded sum).

**Proof.**

It suffices to prove that $\text{cost}(\Gamma) = (M - 1 + \sum_{b < L} K) + \text{count}(\sum_{b < L} \Gamma)$

From Definition of cost we know that

$\text{cost}(\Gamma) = (\sum_{b < L} I(b') + \sum_{b < L} K(b')) + \sum_{b < L} \text{count}(\Gamma)$

$= (M - 1 + \sum_{b < L} K) + \sum_{b < L} \text{count}(\Gamma)$

**Claim:** $\text{count}(\sum_{b < L} \Gamma)$

**Proof.**

When $x : [a < I] \notin \Gamma$

$\langle \Gamma_1', x : \lfloor a < I \rfloor + \Gamma_2 \rangle = \langle \Gamma_2 \rangle$

**Lemma 77** (T-weaken).

Let $\langle \Gamma_1', x : \lfloor a < I \rfloor + \Gamma_2 \rangle$, $x : [b < J] \tau[I + b/c] = \Gamma_r$

$\langle \Gamma_1', x : \lfloor a < I \rfloor + \Gamma_2 \rangle$, $x : \lfloor a < I \rfloor [a/c] \cup \Gamma_2$, $x : [b < J] \tau[I + b/c] = \Gamma_r$

$\langle \Gamma_1', x : \lfloor a < I \rfloor + \Gamma_2 \rangle$, $x : \lfloor a < I \rfloor \cup \Gamma_2$, $x : [b < J] \tau[I + b/c] = \Gamma_r$

**Lemma 78** (Relation b/w dlPCF context and its translation - bounded sum).

$\forall \Gamma \in \text{dlPCF}$. $\langle \sum_{a < I} \Gamma \rangle = \sum_{a < I} \langle \Gamma \rangle$
Proof. Proof by induction on \( \Gamma \)

\[
\Gamma = (\sum_{a < I} \cdot) = (\cdot) \quad \frac{\text{Definition 53}}{=}.
\]

\[
\Gamma = \sum_{a < I} (\cdot) \quad \frac{\text{Definition 54}}{=}
\]

\( \Gamma = \Gamma', x : [-] - \)

Let \( (\sum_{a < I} (\Gamma', x : [b < J] \sum_{d < a} b/c)) = \Gamma_r \)

\[
\Gamma_r = (\sum_{a < I} (\Gamma'), x : [c < \sum_{a < I} J]) \quad \frac{\text{Definition 53}}{=}
\]

\[
= \sum_{a < I} (\Gamma'), x : [c < \sum_{a < I} J] M 0 (\cdot) \quad \frac{\text{IH}}{=}
\]

\[
= \sum_{a < I} (\Gamma'), x : [b < J] M 0 (\cdot) [\sum_{d < a} J d/a + b/c]) \quad \frac{\text{Lemma 80}}{=}
\]

\[
= \sum_{a < I} (\Gamma'), x : [b < J] M 0 (\cdot) [\sum_{d < a} J d/a + b/c]) \quad \frac{\text{Definition 74}}{=}
\]

\[
\sum_{a < I} (\Gamma', x : [b < J] \sum_{a < I} J d/a + b/c)) \quad \frac{\text{Definition 73}}{=}
\]

Lemma 77 (Relation b/w dlPCF context and its translation - subtyping). \( \forall \Gamma, \Gamma' \in dlPCF \).

\[ \Theta; \Delta \vdash \Gamma_1 \sqsubseteq \Gamma_2 \implies \Theta; \Delta \vdash L \Gamma_1 M < : L \Gamma_2 M \]

Proof. Proof by induction on the \( \Theta; \Delta \vdash \Gamma_1 \sqsubseteq \Gamma_2 \) relation

1. dlpcf-sub-mBase:

\[
\vdash ; \Theta; \Delta \vdash L \Gamma_1 M < : L \Gamma_1 M \quad \frac{\text{sub-mBase}}{=}
\]

2. dlpcf-sub-mInd:

D4:

\[
\vdash ; \Theta; \Delta \vdash \Gamma_1/x < : \Gamma_2 \quad \frac{\text{By inversion}}{=}
\]

D3:

\[
\vdash ; \Theta; \Delta \vdash I \leq J \quad \frac{\text{By inversion}}{=}
\]

D2:

\[
\vdash ; \Theta; a; \Delta, a < I \vdash \tau' < : \tau \quad \frac{\text{By inversion}}{=}
\]

\[
\vdash ; \Theta; a; \Delta, a < I \vdash M 0 (\tau') < : M 0 (\tau) \quad \frac{\text{Lemma 76}}{=}
\]

D1:

\[
\vdash x : [a < J] \tau' \in \Gamma_1 \quad \frac{\text{By inversion}}{=}
\]

Main derivation:

\[
D1 \quad D2 \quad D3 \quad D4 \quad \vdash ; \Theta; \Delta \vdash L \Gamma_1 M < : L \Gamma_2 M \quad \frac{\text{By inversion}}{=}
\]

Lemma 78. \( \forall L, \Gamma. \)

\[
\sum_{a < L} \text{count}(\Gamma) = \text{count}(\sum_{a < L} \Gamma)
\]

Proof. By induction on \( \Gamma \)

\[
\Gamma = \sum_{a < L} \cdot = \sum_{a < L} \cdot \quad \frac{\text{From Definition of count we know that count(.) = 0 therefore}}{=}
\]

\[
\sum_{a < L} \text{count}(.) = 0 \quad \frac{\text{From Definition 54 we know that}}{=}
\]

\[
\sum_{a < L} \cdot = . \quad \frac{\text{Therefore again from Definition of count we know that count(.) = 0}}{=}
\]

And we are done

\[
\Gamma = \Gamma', x : [b < J] \tau
\]
\begin{align*}
\text{count}(\sum_{a \leq L} \Gamma', x : b < J \tau) &= \text{count}(\sum_{a \leq L} \Gamma', x : c < \sum_{d < a} J[d/a] + b/c) \\
&= \text{count}(\sum_{a \leq L} \Gamma') + \sum_{a \leq L} J[d/a] + b/c) \\
&= \sum_{a \leq L} \text{count}(\Gamma') + \sum_{a \leq L} J[d/a] + b/c)
\end{align*}

Definition of \text{count}(\cdot)

Lemma 79 (Subtyping is preserved by translation). \(\Theta; \Delta \vdash \sigma : \tau \Rightarrow \Theta; \Delta \vdash \langle \sigma \rangle : \langle \tau \rangle\)

Proof. By induction on \(\Theta; \Delta \vdash \sigma : \tau\).

1. \([a < I] \sigma_1 \Rightarrow \sigma_2 : [a < J] \tau_1 \Rightarrow \tau_2\):

\[
\begin{array}{l}
\text{IH1}
\Theta; \Delta \vdash \langle \sigma_1 \rangle : \langle \tau_1 \rangle \\
\hline
\Theta; \Delta \vdash \langle \{a < I, J\} \sigma_2 \rangle : \langle \{a < J\} \tau_2 \rangle \\
\end{array}
\]

Main derivation:

\[
\begin{array}{l}
\Theta; \Delta \vdash \langle \sigma_1 \rangle : \langle \tau_1 \rangle \\
\hline
\Theta; \Delta \vdash \langle \{a < I, J\} \sigma_2 \rangle : \langle \{a < J\} \tau_2 \rangle \\
\end{array}
\]

2. \(\tau = [a < I] \tau_1 \Rightarrow \tau_2\):

\[
\begin{array}{l}
\text{IH2}
\Theta; \Delta \vdash \langle \{a < I\} \sigma_1 \rangle : \langle \tau_1 \rangle \\
\hline
\Theta; \Delta \vdash \langle \{a < I, J\} \sigma_2 \rangle : \langle \{a < J\} \tau_2 \rangle \\
\end{array}
\]

Lemma 80 (Index Substitution lemma). \(\forall \tau \in \text{dLCPF}, J.
\langle \tau[J/b]\rangle = \langle \tau \rangle\)

Proof. By induction on \(\tau\).

1. \(\tau = b\):

\[
\begin{array}{l}
\Theta; \Delta \vdash \langle \{a < I\} \sigma_1 \rangle : \langle \tau_1 \rangle \\
\hline
\Theta; \Delta \vdash \langle \{a < I, J\} \sigma_2 \rangle : \langle \tau_2 \rangle \\
\end{array}
\]

2. \(\tau = [a < I] \tau_1 \Rightarrow \tau_2\):

\[
\begin{array}{l}
\text{IH1}
\Theta; \Delta \vdash \langle \{a < I\} \sigma_1 \rangle : \langle \tau_1 \rangle \\
\hline
\Theta; \Delta \vdash \langle \{a < I, J\} \sigma_2 \rangle : \langle \tau_2 \rangle \\
\end{array}
\]

Lemma 81. \(\Psi; \Theta; \Delta; x : a < I \tau; \vdash !x : !a < I \tau\)

Proof.

\[
\begin{array}{l}
\Psi; \Theta; a; \Delta; a < I; x : b < I \tau[\varepsilon[\tau[a+b/a]]] \vdash x : \tau \\
\hline
\Psi; \Theta; \Delta; x : b < I \tau[\varepsilon[\tau[a+b/a]]] \vdash !x : !a < I \tau
\end{array}
\]

Lemma 82. \(\sum_{a < I} x : b < I \tau[\varepsilon[\tau[a+b/a]]] = x_{a < I} \tau\)

Proof. It suffices to prove that

\[
\sum_{a < I} x : b < I \tau[\varepsilon[\tau[a+b/a]]] = x_{a < I} \tau[c/a]
\]

From Definition \(\text{count}(\cdot)\) it suffices to prove that

\[
\sum_{a < I} x : b < I \tau[\varepsilon[\tau[a+b/a]]] = x_{a < I} \tau[c/a]
\]

Again from Definition \(\text{count}(\cdot)\) it suffices to prove that

\[
\tau[\varepsilon[\tau[a+b/a]]] = \tau[c/a]
\]

\[
\tau[c/a][\sum_{d < a} 1[d/a] + b/c] = \tau[c/a][\sum_{d < a} 1[d/a] + b/c]
\]

142
Proof. D2.2
\[
\sum_{\alpha \in I} \tau = \tau
\]

D2.1:
\[
\sum_{\alpha \in I; x : \beta_1} \tau: x : \tau_1 \quad \vdash x : \tau_1
\]

D2:
\[
\sum_{\alpha \in I; f : \beta_1} (\tau_1 \rightharpoonup \tau_2)[a + b/a]; f : \tau_1 \rightharpoonup \tau_2
\]

D0:
\[
\sum_{\alpha \in I; f : \beta_1} (\tau_1 \rightharpoonup \tau_2)[a + b/a]; x : \beta_1 \tau_1[a + b/a]; \vdash f : \tau_1 \rightharpoonup \tau_2
\]

\[\vdash f : \alpha \in I (\tau_1 \rightharpoonup \tau_2), x : \alpha \in I \tau_1; \vdash ! f (x) : \alpha \in I \tau_2\]

Main derivation:
\[
\sum_{\alpha \in I; f : \beta_1} (\tau_1 \rightharpoonup \tau_2)[a + b/a]; x : \beta_1 \tau_1[a + b/a]; \vdash f : \tau_1 \rightharpoonup \tau_2
\]

\[\vdash f : \alpha \in I (\tau_1 \rightharpoonup \tau_2)[a + b/a], x : \alpha \in I \tau_1[a + b/a]; \vdash f : \tau_1 \rightharpoonup \tau_2
\]

Lemma 85. \[\sum_{\alpha \in I} \tau \rightharpoonup \tau_1[a + b/a], x : \beta_1 \tau_1[a + b/a] = f : \tau_1 \rightharpoonup \tau_2, x : \alpha \in I \tau_1\]

Proof. It suffices to prove that
\[\sum_{\alpha \in I} \tau \rightharpoonup \tau_1[a + b/a], x : \beta_1 \tau_1[a + b/a] = f : \tau_1 \rightharpoonup \tau_2, x : \alpha \in I \tau_1\]

From Definition \[\sum_{\alpha \in I} \tau \rightharpoonup \tau_1[a + b/a], x : \beta_1 \tau_1[a + b/a] = f : \tau_1 \rightharpoonup \tau_2, x : \alpha \in I \tau_1[c/a]\]

Again from Definition \[\sum_{\alpha \in I} \tau \rightharpoonup \tau_1[a + b/a], x : \beta_1 \tau_1[a + b/a] = f : \tau_1 \rightharpoonup \tau_2, x : \alpha \in I \tau_1[c/a]\]

So, we are done

\[\sum_{\alpha \in I} \tau \rightharpoonup \tau_1[a + b/a], x : \beta_1 \tau_1[a + b/a] = f : \tau_1 \rightharpoonup \tau_2, x : \alpha \in I \tau_1\]
B.5.2 Cross-language model: dlPCF to λ-amor

**Definition 86** (Logical relation for dlPCF to λ-Amor).

\[
\begin{align*}
[b]_V & \triangleq \{ (t^v, t^v) \mid t^v \in \lfloor b \rfloor \land t^v \in \lfloor b \rfloor \land t^v = t^v \} \\
\lfloor a < I \rfloor \Gamma_1 \rightarrow \tau_2 \}_V & \triangleq \{ (\lambda x_1 \ldots \lambda x_n \ . \_ , \Gamma_1) \mid \forall x_1 \ldots x_n \ . \_ \in \lfloor a < I \rfloor \Gamma_1 \rightarrow \tau_2 \}_V \\
\Theta ; \Delta ; \Gamma \ 
\end{align*}
\]

\[\begin{align*}
\Theta ; \Delta \vdash J_{\tau} \ 
\end{align*}
\]

**Theorem 88** (Fundamental theorem). \( \forall \Theta , \Delta , \Gamma , \tau , e_t , I , \delta_s , \delta_t \),

\[\Theta ; \Delta ; \Gamma \vdash e_t : \tau \rightsquigarrow e_t \land (\delta_s , \delta_t) \in [\Gamma , x]_E \land \delta_s = \delta_t \]

\[\Rightarrow (e_s \delta_s , e_t \delta_t) \in [\tau , I ]_E \]

**Proof.** Proof by induction on the translation relation:

1. var:

\[
\begin{align*}
\Theta ; \Delta ; x : [a < I \tau'] \vdash \lambda \delta \ . \ f , \ E_1 (\delta) & \rightsquigarrow \lambda \delta \ . \ f , \ E_1 (\delta) \\
\Theta ; \Delta ; J , \ x : [a < I \tau'] \vdash \lambda \delta \ . \ f & \rightsquigarrow \lambda \delta \ . \ f \\
\Theta ; \Delta ; \Gamma ; x : [a < I \tau'] \vdash \lambda \delta \ . \ f , \ E_1 (\delta) & \rightsquigarrow \lambda \delta \ . \ f , \ E_1 (\delta)
\end{align*}
\]

This means from Definition 86 we need to prove that

\[\forall x . x \delta_s \downarrow \tau \ 
\]

This means that given some \( \tau \) s.t. \( x \delta_s \downarrow \tau \) it suffices to prove that

\[\exists J' , J'', J' . E_1 (\delta) \vdash J' , J'' \downarrow \tau \land \downarrow \tau \ 
\]

Since we are given that \( (\delta_s , \delta_t) \in [\Gamma , x]_E \) therefore from Definition 87 we know that

\[\forall x : [a < I \tau] \in \text{dom} (\Gamma , x) \land \forall x : [a < I \tau] \in [\tau , I ]_E \]

This means we also have \( (\delta_s (x) , \delta_t (x)) \in [\tau , I ]_E \). This further means that from Definition 86 we have

\[\exists J' , J'' , x \delta_s (x) \downarrow \tau \ 
\]

We instantiate (F-DA-V1) with \( \tau \) and in order to prove (F-DA-V0) we choose \( J' \) as \( J'' \), \( \tau_v \) as \( \tau_v'' \) and \( \tau_v' \) as \( \tau_v'' \) and we get the desired from (F-DA-V1).

2. lam:

\[
\begin{align*}
\Theta ; \Delta ; \Gamma , x : [a < I \tau] \Gamma' \vdash e : \tau_2 \rightsquigarrow e_t \\
\Theta ; \Delta ; \Gamma \vdash \lambda \delta \ . \ f , \ E_1 (\delta) & \rightsquigarrow \lambda \delta \ . \ f , \ E_1 (\delta)
\end{align*}
\]

This means from Definition 88 we need to prove that

\[\forall x . x \delta_s \downarrow \tau \ 
\]

This means that given some \( \tau \) s.t. \( \lambda x . \delta_s \downarrow \tau \) it suffices to prove that

\[\exists J' , J'' , J' . E_1 (\delta) \vdash J' , J'' \downarrow \tau \land \downarrow \tau \ 
\]

We know that \( \tau = \lambda x . \delta_s \). Also from E-app, E-ret we know that \( \downarrow \tau_v = E_2 \) and \( J' = 0 \)

144
Therefore it suffices to show \((\lambda x.e \delta_s, E_2) \in \{(a < I), \tau_1 \rightarrow \tau_2 \}\) 

From Definition \[86\] it further suffices to prove that \(\forall i, i', (e'_i, e'_i) \in [[a < I], \tau_1]_{NE} \implies (e'_i, e'_i) \in [[a < I], \tau_1]_{NE} \) \(\rightarrow\) \((e'_i, e'_i) \in [\tau_2]_{E} \) \((\text{F-DA-L1})\)

This means given some \(e'_i, e'_i\) s.t. \((e'_i, e'_i) \in [[a < I], \tau_1]_{NE}\). We need to prove that \((e'_i, e'_i) \in \tau_2 \) \(\rightarrow\) \((\text{F-DA-L1.1})\)

Since \((e'_i, e'_i) \in [[a < I], \tau_1]_{NE}\) therefore from Definition \[86\] we have

\(\exists e'_i, e'_i = \text{coerce} 1, e'_i \rightarrow \forall 0 < i < I, (e'_i, e'_i) \in [[a < I], \tau_1]_{i} \) \(\rightarrow\) \((\text{F-DA-L1.1})\)

Let \(\delta'_s = \delta_s \cup \{x \mapsto e'_i\}\) and

\(\delta'_t = \delta_t \cup \{x \mapsto e'_i\}\)

From Definition \[77\] we know that

\((\delta'_s, \delta'_t) \in \Gamma, \tau \) \(\rightarrow\) \((\text{F-DA-L2})\)

Therefore from \(\text{II}\) we have

\((e_s, \delta'_s, e_t(\delta'_t)) \in \tau_2 \) \(\rightarrow\) \((\text{F-DA-L3})\)

This means from Definition \[86\] we have

\(\forall v_b, \delta_s \downarrow v_b \implies \exists J_b, \vdash J_b, v_{b}, e_t(\delta'_t) \downarrow J_b \downarrow v_b \land (v_b, v_b) \in \tau_2 \) \((\text{F-DA-L3})\)

Applying Definition \[86\] on \(\text{F-DA-L1.1}\) we need to prove

\(\forall v_f, e_s(\delta_s) \downarrow v_f \implies \exists J_f, \vdash J_f, v_f, E_2[\delta_s] \downarrow v_f \downarrow J_f \downarrow v_f \land (v_f, v_f) \in \tau_2 \) \((\text{F-DA-L4})\)

Therefore instantiating (F-DA-L3) with \(v_f\) and we get the desired

3. app:

\[\Theta; \Delta; \Gamma \vdash_f e_1 : [[a < I], \tau_1] \rightarrow \tau_2 \rightarrow e_{t1}\]

\[\Theta; \Delta, a < I; \Delta \vdash_F e_2 : \tau_1 \rightarrow e_{t2}\]

\[\Gamma = \Sigma a < I \Delta B_1 \]

\[\Theta; \Delta; \Gamma \vdash_H e_1 e_2 : \tau_2 \rightarrow e_{t2}\]

\[\lambda p. \text{release} = p \text{ in } \text{bind } a = \text{store}(i) \text{ in } \text{bind } b = e_{t1} \text{ in } \text{bind } c = \text{store}(i) \text{ in } \text{bind } d = \text{store}(i) \text{ in } b (\text{coerce} 1, e_{t2} c) d\]

\(E_1 = \lambda p. \text{release} = p \text{ in } \text{bind } a = \text{store}(i) \text{ in } \text{bind } b = e_{t1} \text{ in } \text{bind } c = \text{store}(i) \text{ in } \text{bind } d = \text{store}(i) \text{ in } b (\text{coerce} 1, e_{t2} c) d\)

Given: \((\delta_s, \delta_t) \in \Gamma'\)

To prove: \((e_s, e_t(\delta_t)) \in \tau_2 \) \(\rightarrow\)

This means from Definition \[86\] we need to prove that

\(\forall v_f, (e_s(\delta_s) \downarrow v_f \implies \exists J_f, \vdash J_f, v_f, E_1() \downarrow v_f \downarrow J_f \downarrow v_f \land (v_f, v_f) \in \tau_2 \) \((\text{F-DA-A0})\)

This means that given some \(v_f\) and \(e_s(\delta_s) \downarrow v_f \) it suffices to prove that

\(\exists J_f, \vdash J_f, v_f, E_1() \downarrow v_f \downarrow J_f \downarrow v_f \land (v_f, v_f) \in \tau_2 \) \((\text{F-DA-A0})\)

\(\text{H1}\)

\((e_s(\delta_s, e_{t1}(\delta_t)) \in \{\{a < I\], \tau_1 \rightarrow \tau_2 \}\}) \)

This means from Definition \[86\] we have

\(\forall v_{2}, e_s(\delta_s) \downarrow v_{2} \implies \exists J_f, \vdash J_f, v_{t1}, e_{t1}(\delta_t) \downarrow v_{t1} \downarrow J_f \downarrow v_{t1} \land (v_{t1}, v_{t1}) \in \{\{a < I\], \tau_1 \rightarrow \tau_2 \}\}) \)

Since we know that \((e_s(\delta_s, e_{t1}(\delta_t)) \downarrow v_{t1} \downarrow J_f \downarrow v_{t1} \land (v_{t1}, v_{t1}) \in \{\{a < I\], \tau_1 \rightarrow \tau_2 \}\}) \)

Since we know that \((v_{t1}, v_{t1}) \in \{\{a < I\], \tau_1 \rightarrow \tau_2 \}\}) \)

Let \(\lambda x.e_s, \text{and } t_{2} = \lambda x. \lambda p. \text{let}\! x = y \text{ in } e_{t2}\)

Therefore from Definition \[86\] we have

\(\forall e_s(\delta_s, e_{t1}(\delta_t)) \in [[a < I], \tau_1] \rightarrow \tau_2 \rightarrow (e_{s}, e_{t1}(\delta_t)) \in \tau_2 \) \(\rightarrow\)

\(\text{H2}\)

\((e_{s}, e_{t2}(\delta_t)) \in \tau_1 \) \(\rightarrow\)

\(\text{H2}\)

\((e_{s}, e_{t2}(\delta_t)) \in \{\{a \rightarrow 0\}\}) \)

\((e_{s}, e_{t2}(\delta_t)) \in \{\{a \rightarrow 1\}\}) \)
(e_2 \delta_s, e_2 \delta_t(\delta_t) \in [\tau_1 \cup \{a \mapsto I - 1\}]_E \quad (F-DA-A3)

We claim that
\((e_2 \delta_s, \text{coerce } le_{e_2} !() \delta_t) \in [[a < I] \tau_1 \tau]_E\)

From Definition [83] we know that
\(\text{coerce } F \ X \triangleq \) let \(! = F \) in let \(! x = X \) in!(f x)

therefore the desired holds from Definition [86] and (F-DA-A3)

Instantiating (F-DA-A2) with \(e_2 \delta_s, \text{coerce } le_{e_2} !() \delta_t\) we get
\((e_2 \delta_s,\text{coerce } le_{e_2} !() \delta_t) \in [\tau_2 \\tau]_E\) \quad (F-DA-A4)

This further means that from Definition [86] we have
\(\forall x. v_b \circ e_x [e_2 \delta_s / x] \updownarrow x. v_f \implies \exists J_2, \uparrow^x v_b, \uparrow^x v_f. e_x [\text{coerce } le_{e_2} !() \delta_t / x] \updownarrow x. v_b \uparrow x. v_f \wedge (\uparrow^x v_b, \uparrow^x v_f) \in [\tau_2 \\tau]_V\)

Since we know that \((e_1 e_2) \delta_s \downarrow^n x. v_f\) therefore we know that \(\exists x. v_b, x_2 \text{ s.t. } e_b [e_2 \delta_s / x] \downarrow^n x. v_f\). Therefore we have
\(\exists J_2, \uparrow^x v_b, \uparrow^x v_f. e_b [\text{coerce } le_{e_2} !() \delta_t / x] \downarrow x. v_b \uparrow x. v_f \wedge (\uparrow^x v_b, \uparrow^x v_f) \in [\tau_2 \\tau]_V\) \quad (F-DA-A5)

In order to prove (F-DA-A0) we choose \(J' = J_1 + J_2, \uparrow^x v_b\) as \(\uparrow^x v_b\) and \(\uparrow^x v_f\) as \(\uparrow^x v_f\), we get the desired from (F-DA-A1) and (F-DA-A5)

4. fix:

\[\forall \Theta, \delta_s, \Theta, \delta_t: b < L; \Gamma, x: \{a < I\} \tau \vdash e: \tau \triangleright e_t\]
\[\implies \Theta, \delta_s, \Theta, \delta_t: b < L; \Gamma, \tau \vdash \sigma: \tau \triangleright e_t\]
\[\implies \exists \Theta, \delta_s, \Theta, \delta_t: b < L; \Gamma, \tau \vdash \sigma: \tau \triangleright e_t\]
\[\forall \Theta, \delta_s, \Theta, \delta_t: b < L; \Gamma, \tau \vdash \sigma: \tau \triangleright e_t\]

Given: \((\delta_s, \delta_t) \in [\Gamma]_E\)

To prove: \((\text{fix}_x. e_x, \text{fix}_Y. E_1)(\delta_t) \in [\mu \tau]_E\)

This means from Definition [86] we need to prove that
\(\forall \tau. v. \text{fix}_x. e_x \downarrow^n x. v \implies \exists J', \uparrow^x v_b, E_0. \downarrow \uparrow x. v_b \uparrow J' \uparrow J' \uparrow^x v_f \wedge (\uparrow^x v_b, \uparrow^x v_f) \in [\mu \tau]_V\)

This means that given some \(\uparrow^x v_b\) s.t. \(\text{fix}_x. e_x \downarrow^n x. v\) it suffices to prove that
\(\exists J', \uparrow^x v_b, E_0. \downarrow \uparrow x. v_b \uparrow J' \uparrow J' \uparrow^x v_f \wedge (\uparrow^x v_b, \uparrow^x v_f) \in [\mu \tau]_V\) \quad (F-DA-F0)

Claim 1
\(\forall 0 \leq t < L. (e \delta_s, E_1 (\delta_t) \in [\tau[t/b] \tau]_E\)
where \(\delta'_s = \delta_s \cup \{x \mapsto (x \mapsto (\text{fix}_x. e_x) \delta_s)\} \) and \(\delta'_t = \delta_t \cup \{x \mapsto (x \mapsto (\text{fix}_x. E_1) \delta_t)\}\)

We prove this by induction on the recursion tree

Base case: when \(t\) is a leaf node
Since for a leaf node \(I(t) = 0\) and \(x \notin \text{free}(e)\) therefore from IH (outer induction) we get
\((e \delta_s, \delta_t(\delta_t) \in [\tau[t/b] \tau]_E\)

This means from Definition [86] we have
\(\forall \tau. v. e_x \downarrow^n x. v \implies \exists J', \uparrow^x v_b, J. E_1 (\delta_t) \downarrow \uparrow x. v_b \uparrow J' \uparrow J' \uparrow^x v_f \wedge (\uparrow^x v_b, \uparrow^x v_f) \in [\tau[t/b] \tau]_V\) \quad (BC0)

Since we have to prove \((e \delta_s, E_1 (\delta_t) \in [\tau[t/b] \tau]_E\)

Therefore from Definition [86] it suffices to prove that
\(\forall \tau. v. e_x \downarrow^n x. v \implies \exists J', \uparrow^x v_b, J. E_1 (\delta_t) \downarrow \uparrow x. v_b \uparrow J' \uparrow J' \uparrow^x v_f \wedge (\uparrow^x v_b, \uparrow^x v_f) \in [\tau[t/b] \tau]_V\)

146
Lemma 89.

∀Θ, Δ ⊢ τ <: τ' \iff \|τ\|V \subseteq \|τ'\|V

(a) Θ; Δ \vdash τ <: τ' \iff \|τ\|V \subseteq \|τ'\|V
Proof. Proof by simultaneous induction on $\Theta;\Delta \vdash \tau <: \tau'$ and $\Theta;\Delta \vdash [a < I] \tau <: [a < J] \tau'$

Proof of statement (a)

We case analyze the different cases:

1. $\neg \sigma$:

$$
\Theta;\Delta \vdash B <: A \quad \Theta;\Delta \vdash \tau <: \tau'
$$

To prove: $[(A \rightarrow \tau) \tau]_V \subseteq [(B \rightarrow \tau') \tau]_V$

This means we need to prove that

$$\forall(x.e, \lambda x.\lambda p.e) \in [A \rightarrow \tau]_V \quad (x.e, \lambda x.\lambda p.e) \in [B \rightarrow \tau']_E$$

This means given $(\lambda x.e, \lambda y.\lambda p.e)$ and $x \in (e_1) \in [A \rightarrow \tau]_V$, and we need to prove

$$(\lambda x.e, \lambda y.\lambda p.e) \in [B \rightarrow \tau']_V$$

This means from Definition $\backslash [86]$ we are given that

$$\forall e_1, e_1', (e_1, e_1') \in [A \tau]_E \implies (e_1[e_1'/x], e_1[e_1'/y]/(y)|p|) \in [\tau \tau]_E$$

(SV-A0)

And we need to prove that

$$\forall e_1, e_1', (e_1, e_1') \in [B \tau]_E \implies (e_1[e_1'/x], e_1[e_1'/y]/(y)|p|) \in [\tau \tau]_E$$

(SV-A1)

Since we are given that $(e_1, e_1') \in [B \tau]_E$, therefore from IH (Statement (b)) we have $(e_1, e_1') \in [A \tau]_E$.

In order to prove (SV-A1), we instantiate (SV-A0) with $e_1', e_1''$ and we get

$$(e_1[e_1'/x], e_1[e_1'/y]/(y)|p|) \in [\tau \tau]_E$$

Finally from Lemma $[90]$ we get

$$(e_1[e_1'/x], e_1[e_1'/y]/(y)|p|) \in [\tau \tau]_E$$

Proof of statement (b)

$$
\Theta;\Delta \vdash J \leq I \quad \Theta;\Delta \vdash \tau <: \tau'
$$

To prove: $[[a < I] \tau]_E \subseteq [[a < J] \tau']_E$

This means we need to prove that

$$\forall (e_1, e_1') \in [[a < I] \tau]_E \quad (e_1, e_1') \in [[a < J] \tau']_E$$

This means given $(e_1, e_1') \in [[a < I] \tau]_E$ and we need to prove

$$(e_1, e_1') \in [[a < J] \tau']_E$$

This means from Definition $\backslash [86]$ we are given that

$$\exists e_1, e_1' = coerce(E_1, e_1', l)| \land \forall 0 \leq i < I, (e_1, e_1') \in [\tau[l/a] \tau]_E$$

(SNE0)

and we need to prove

$$\exists e_1, e_1' = coerce(E_1, e_1', l)| \land \forall 0 \leq j < J, (e_1, e_1') \in [\tau'[j/a] \tau]_E$$

(SNE1)

In order to prove (SNE1) we choose $e_1'$ as $e_1$ from (SNE0) and we need to prove

$$\forall 0 \leq j < J, (e_1, e_1') \in [\tau'[j/a] \tau]_E$$

This means given some $0 \leq j < J$ and we need to prove that

$$(e_1, e_1') \in [\tau'[j/a] \tau]_E$$

From (SNE0) we get

$$(e_1, e_1') \in [\tau[j/a] \tau]_E$$

And finally from Lemma $[90]$ we get

$$(e_1, e_1') \in [\tau'[j/a] \tau]_E$$

Lemma 90. $\forall \Theta, \Delta, \tau, \tau', e, e_1, l.$

$$\Theta;\Delta \vdash \tau <: \tau' \land [\Delta l] \implies [\tau]_E \subseteq [\tau']_E$$
Proof. Given: \( \Theta; \Delta \vdash \tau <: \tau' \)
To prove: \( [\tau \ell]_E \subseteq [\tau' \ell]_E \)

It suffices to prove that \( \forall (e_s, e_t) \in [\tau \ell]_E, (e_s, e_t) \in [\tau' \ell]_E \)
This means given \( (e_s, e_t) \in [\tau \ell]_E \) it suffices to prove that
(\( (e_s, e_t) \in [\tau' \ell]_E \))

This means from Definition \([86]\) we are given that
\( \forall v_0, e_s \Downarrow^s v_0 \Rightarrow \exists ! J_0, t_0, v_0, e_t \Downarrow^t t_0 \Downarrow^J t_0 \wedge (v_0, t_0, v_0) \in [\tau \ell]_V \) \hspace{1cm} \( \text{(S0)} \)

And it suffices to prove that
\( \forall v, e_s \Downarrow^s v \Rightarrow \exists ! J, t, v_0, e_t \Downarrow^t v_0 \Downarrow^J t \wedge (v, t, v_0) \in [\tau' \ell]_V \)

This means given some \( s \) s.t. \( e_s \Downarrow^s v \) and we need to prove
\( \exists J, t, v_0, e_t \Downarrow^t v_0 \Downarrow^J t \wedge (v, t, v_0) \in [\tau' \ell]_V \) \hspace{1cm} \( \text{(S1)} \)

We get the desired from (S0) and Lemma \([89]\)

\( \square \)

B.5.3 Re-deriving dlPCF’s soundness

Definition 91 (Closure translation).
\[
\begin{align*}
(\langle e, [] \rangle) & \triangleq e \\
(\langle e, [c_1, \ldots, c_n] \rangle) & \triangleq \lambda x_1 \ldots x_n. e \langle c_1 \rangle \ldots \langle c_n \rangle
\end{align*}
\]

Definition 92 (Krivine triple translation).
\[
\begin{align*}
(\langle e, [\rho] \rangle) & \triangleq \langle (e, \rho) \rangle \\
(\langle e, [\rho, \theta] \rangle) & \triangleq \langle (e, \rho), \langle \theta \rangle \rangle
\end{align*}
\]

Lemma 93 (Type preservation for Closure translation). \( \forall \Theta, \Delta, e, \rho, \tau. \)
\( \Theta; \Delta \vdash \lambda x_1 \ldots x_n.e : \tau \implies \Theta; \Delta; \vdash \langle e, [\rho] \rangle : \sigma \)

Proof.
\[
\begin{array}{c}
\Theta; \Delta; x_1 : [a < I_1]\tau_1 \ldots x_n : [a < I_n]\tau_n +_{n=1} \sum_{a < I_1} H_1 + \ldots + \sum_{a < I_n} H_n \\
J' = K + I_1 + \ldots + I_n + \sum_{a < I_1} H_1 + \ldots + \sum_{a < I_n} H_n \\
D0: \\
\Theta; \Delta; \vdash \lambda x_1 \ldots x_n.e : \langle [\rho] \rangle \\
\text{Given: } \Theta; \Delta; \vdash K \lambda x_1 \ldots x_n.e : \langle [\rho, \theta] \rangle \rightarrow [a_1 < I_1]\tau_1 \rightarrow [a_2 < I_2]\tau_2 \rightarrow \ldots [a_n < I_n]\tau_n \rightarrow \sigma
\end{array}
\]

Main derivation:
\[
\begin{array}{c}
D0 \quad D1 \\
\Theta; \Delta; \vdash \langle e, [\rho] \rangle : \sigma \\
\text{Lemma 3.5 of } [9] \\
\text{Definition } [91]
\end{array}
\]

\( \square \)

Theorem 94 (Type preservation for Krivine triple translation). \( \forall \Theta, \Delta, e, \rho, \theta, \tau. \)
\( \Theta; \Delta \vdash \lambda x_1 \ldots x_n.e : \tau \implies \Theta; \Delta; \vdash \langle e, [\rho, \theta] \rangle : \tau \)
Proof.

\[ \Theta; \Delta \vdash_K (e, \rho) : \sigma \quad \Theta; \Delta \vdash_J \theta : (\sigma, \tau) \quad I \geq K + J \]

\[ \Theta; \Delta \vdash_{\ell} (e, \rho, \theta) : \tau \]

Let \( I' = K + J \)

Proof by induction on \( \theta \)

1. Case \( \epsilon \):

Given: \( \Theta; \Delta \vdash I(e, \rho, \epsilon) : \tau \)

To prove: \( \Theta; \Delta; \epsilon \vdash I \)

\[ \begin{aligned}
\Theta; \Delta; \epsilon & \vdash_{\ell} ([e, \rho]) : \tau \quad \text{Lemma}\ 93 \\
\Theta; \Delta; \epsilon & \vdash_{\ell} ([e, \rho, \epsilon]) : \tau \quad \text{Definition}\ 92 \\
\Theta; \Delta & \vdash I(e, \rho, \epsilon) : \tau \\
\end{aligned} \]

Main derivation:

\[ \begin{array}{c}
\Theta; \Delta; \epsilon \vdash_{\ell} ([e, \rho]) : \tau \\
\Theta; \Delta; \epsilon \vdash_{\ell} ([e, \rho, \epsilon]) : \tau \\
\Theta; \Delta & \vdash I(e, \rho, \epsilon) : \tau \\
\end{array} \]

Lemma 3.5 of [9]

D0:

\[ \begin{aligned}
D0.0 & : \Theta; \Delta; \epsilon \vdash_{\ell} ([e, \rho]) : \tau \\
D0.1 & : \Theta; \Delta; \epsilon \vdash_{\ell} ([e, \rho, \epsilon]) : \tau \\
D & : \Theta; \Delta; \epsilon \vdash_{\ell} ([e, \rho, \epsilon]) : \tau \\
\end{aligned} \]

IH

D-app

Lemma 3.5 of [9]

D1:

\[ \begin{aligned}
\Theta; \Delta; \epsilon; d < L_g & \vdash_K C : \gamma \\
\Theta; \Delta; \epsilon & \vdash_{H_g} \theta' : (\mu, \tau) \\
\end{aligned} \]

Lemma 93

D0:

\[ \begin{aligned}
\Theta; \Delta; \epsilon; d < L_g & \vdash_K C : \gamma \\
\Theta; \Delta; \epsilon & \vdash_{H_g} \theta' : (\mu, \tau) \\
\end{aligned} \]

D-app

Lemma 93

D0.1:

\[ \begin{aligned}
\Theta; \Delta; \epsilon & \vdash_{\ell} ([e, \rho]) : \tau \\
\Theta; \Delta; \epsilon; d < L_g & \vdash_K C : \gamma \\
\end{aligned} \]

Lemma 93

D0.0:

\[ \begin{aligned}
\Theta; \Delta; \epsilon; d < L_g & \vdash_K C : \gamma \\
\Theta; \Delta & \vdash_{\ell} ([e, \rho]) : \tau \\
\end{aligned} \]

D-app

Lemma 93

D2:

\[ \begin{aligned}
\Theta; \Delta & \vdash_{\ell} ([e, \rho, \epsilon]) : \tau \\
\end{aligned} \]

By inversion

Lemma 93

2. Case \( \mathcal{C}.\theta' \):

Given: \( \Theta; \Delta \vdash I(e, \rho, \mathcal{C}.\theta') : \tau \)

To prove: \( \Theta; \Delta; \epsilon \vdash I \)

Since \( \theta = \mathcal{C}.\theta' \) therefore from dlPCF’s type rule for \( \mathcal{C}.\theta' \) we know that

\( \sigma = [d < L_g] \gamma \to \mu \)

That is we are given that

\[ \Theta; d, \Delta; d < L_g \vdash_K C : \gamma \\
\Theta; \Delta & \vdash_{H_g} \theta' : (\mu, \tau) \\
\end{aligned} \]

\[ J \geq H_g + \sum_{d < L_g} K_g + L_g \]

\[ \Theta; \Delta & \vdash_{\ell} \mathcal{C}.\theta' : ([d < L_g] \gamma \to \mu, \tau) \]

D2:

\[ \begin{aligned}
\Theta; \Delta & \vdash_{\ell} \mathcal{C}.\theta' : ([d < L_g] \gamma \to \mu, \tau) \\
\end{aligned} \]

By inversion

D1:

\[ \begin{aligned}
\Theta; \Delta; d < L_g & \vdash_K C : \gamma \\
\Theta; \Delta; d < L_g & \vdash_{H_g} \theta' : (\mu, \tau) \\
\end{aligned} \]

Lemma 93

D0:

\[ \begin{aligned}
\Theta; \Delta; d < L_g & \vdash_K C : \gamma \\
\Theta; \Delta; d < L_g & \vdash_{H_g} \theta' : (\mu, \tau) \\
\end{aligned} \]

D-app

Lemma 93

D0.1:

\[ \begin{aligned}
\Theta; \Delta; \epsilon; d < L_g & \vdash_K C : \gamma \\
\Theta; \Delta; \epsilon; d < L_g & \vdash_{H_g} \theta' : (\mu, \tau) \\
\end{aligned} \]

Lemma 93

D0.0:

\[ \begin{aligned}
\Theta; \Delta; \epsilon & \vdash_{\ell} ([e, \rho]) : \tau \\
\Theta; \Delta; \epsilon; d < L_g & \vdash_K C : \gamma \\
\end{aligned} \]

D-app

Lemma 93

D2:

\[ \begin{aligned}
\Theta; \Delta & \vdash_{\ell} ([e, \rho, \epsilon]) : \tau \\
\end{aligned} \]

By inversion

Lemma 93
Definition 95 (Equivalence for \( \lambda \)-amor).

\[
\begin{align*}
\text{v}_1 \overset{s}{\approx}_{aV} \text{v}_2 \iff & \forall i < s . e .1 \downarrow_i v_a \implies e_2 \downarrow k v_b \land v_a \overset{s_{-i}}{\approx}_{aV} v_b \\
& e_1 \overset{s}{\approx}_{aE} e_2 \\
& \forall e', e'', s' < s . e' \overset{s'}{\approx}_{aE} e'' \implies e_1[e'/x] \overset{s'}{\approx}_{aE} e_2[e''/x]
\end{align*}
\]

Lemma 96 (Monotonicity lemma for value equivalence). \( \forall v_1, v_2, s . v_1 \overset{s}{\approx}_{aV} v_2 \implies \forall s' < s . v_1 \overset{s'}{\approx}_{aV} v_2 \)

Proof. Given: \( v_1 \overset{s}{\approx}_{aV} v_2 \)

To prove: \( \forall s' < s . v_1 \overset{s'}{\approx}_{aV} v_2 \)

This means given some \( s' < s \) and it suffices to prove that \( v_1 \overset{s'}{\approx}_{aV} v_2 \)

We induct on \( v_1 \):

1. \( v_1 = () \):

   Since we are given that \( v_1 \overset{s}{\approx}_{aV} v_2 \) therefore we get the desired Directly from Definition 95.

2. \( v_1 = \lambda x . e_1 \):

   Since we are given that \( v_1 \overset{s}{\approx}_{aV} v_2 \) therefore from Definition 95 we are given that

   \( \forall e', e'', s'' < s . e' \overset{s'}{\approx}_{aE} e'' \implies e_1[e'/x] \overset{s''}{\approx}_{aE} e_2[e''/x] \) (M-L0)

   and we need to prove that \( v_1 \overset{s}{\approx}_{aV} v_2 \) therefore again from Definition 95 we need to prove that

   \( \forall e'_1, e''_1, s'_1 < s . e'_1 \overset{s'_1}{\approx}_{aE} e''_1 \implies e_1[e'_1/x] \overset{s'_1}{\approx}_{aE} e_2[e''_1/x] \)

   This means given some \( e'_1, e''_1, s'_1 < s' \) s.t. \( e'_1 \overset{s'_1}{\approx}_{aE} e''_1 \) we need to prove that

   \( e_1[e'_1/x] \overset{s'_1}{\approx}_{aE} e_2[e''_1/x] \)

   Instantiating (M-L0) with \( e'_1, e''_1, s'_1 \) we get \( e_1[e'_1/x] \overset{s'_1}{\approx}_{aE} e_2[e''_1/x] \)

3. \( v_1 = e_1 \):

   Since we are given \( v_1 \overset{s}{\approx}_{aV} v_2 \) therefore from Definition 95 we have

   \( e_1 \overset{s}{\approx}_{aE} e_2 \) where \( v_2 = e_2 \)

   Similarly from Definition 95 it suffices to prove that \( e_1 \overset{s}{\approx}_{aE} e_2 \)

   We get this directly from Lemma 97.

4. \( v_1 = \Lambda e_1 \):

   Similar reasoning as in the \( \lambda x . e_1 \) case

5. \( v_1 = \text{ret} . e_1 \):

   Since we are given \( v_1 \overset{s}{\approx}_{aV} v_2 \) therefore from Definition 95 we have

   \( \forall i < s . v_1 \downarrow_i v_a \implies v_2 \downarrow_i v_b \land v_a \overset{s_{-i}}{\approx}_{aE} v_b \) where \( v_2 = \text{ret} . e_2 \) (MV-R0)

   Similarly from Definition 95 it suffices to prove that

   \( \forall j < s' . v_1 \downarrow_j v_a \implies v_2 \downarrow_j v_b \land v_a \overset{s'_{-i}}{\approx}_{aE} v_b \)
This means given some \(j < s'\) and \(v_1 \downarrow^k v_a\) and it suffices to prove that
\[v_2 \downarrow^k v_b \land v_a \overset{s'-j}{\approx}_{aE} v_b\]

Instantiating (MV-R0) with \(j\) we get \(v_2 \downarrow^k v_b \land v_a \overset{s'-j}{\approx}_{aE} v_b\)

Since we have \(v_a \overset{s'-j}{\approx}_{aE} v_b\) therefore from Lemma 97 we also get \(v_a \overset{s'-j}{\approx}_{aE} v_b\)

6. \(v_1 = \text{bind} \quad \text{e} \quad \text{in} \quad \text{e} \quad \text{release} \quad \text{e} \quad \text{store} \quad \text{e}\)

Similar reasoning as in the \(\text{ret}\) case

7. \(v_1 = (\langle v_{a1}, v_{a2} \rangle)\):
   - From Definition 95 and IH we get the desired

8. \(v_1 = (v_{a1}, v_{a2})\):
   - From Definition 95 and IH we get the desired

9. \(v_1 = \text{inl}(v)\):
   - From Definition 95 and IH we get the desired

10. \(v_1 = \text{inr}(v)\):
    - From Definition 95 and IH we get the desired

\[
\text{Lemma 97 (Monotonicity lemma for expression equivalence). } \forall e_1, e_2, s. \\
e_1 \overset{s}{\approx}_{aE} e_2 \implies \forall s' < s, e_1 \overset{s'}{\approx}_{aE} e_2
\]

\[\text{Proof. Given: } e_1 \overset{s}{\approx}_{aE} e_2\]

To prove: \(\forall s' < s, e_1 \overset{s'}{\approx}_{aE} e_2\)

This means given some \(s' < s\) and we need to prove \(e_1 \overset{s'}{\approx}_{aE} e_2\)

Since we are given \(e_1 \overset{s}{\approx}_{aE} e_2\) therefore from Definition 95 we have
\n\[\forall i < s, e_1 \downarrow_i v_a \implies e_2 \downarrow_i v_b \land v_a \overset{s-i}{\approx}_{aV} v_b \quad \text{(ME0)}\]

Similarly from Definition 95 it suffices to prove that
\n\[\forall j < s', e_1 \downarrow_j v_a \implies e_2 \downarrow_j v_b \land v_a \overset{s'-j}{\approx}_{aV} v_b\]

This means given some \(j < s'\) s.t. \(e_1 \downarrow_j v_a\) and we need to prove
\n\[e_2 \downarrow_j v_b \land v_a \overset{s'-j}{\approx}_{aV} v_b\]

We get the desired from (ME0) and Lemma 96

\[
\text{Lemma 98 (Monotonicity lemma for } \delta \text{ equivalence). } \forall \delta_1, \delta_2, s. \\
\delta_1 \overset{s}{\approx}_{aE} \delta_2 \implies \forall s' < s, \delta_1 \overset{s'}{\approx}_{aE} \delta_2
\]

\[\text{Proof. From Definition 95 and Lemma 97}\]

\[
\text{Theorem 99 (Fundamental theorem for equivalence relation of } \lambda\text{-amor). } \forall \delta_1, \delta_2, e, s. \\
\delta_1 \overset{s}{\approx}_{aE} \delta_2 \implies e\delta_1 \overset{s}{\approx}_{aE} e\delta_2
\]

\[\text{Proof. We induct on } e\]

1. \(e = x\):
   - We need to prove that \(x\delta_1 \overset{s}{\approx}_{aE} x\delta_2\)
   - This means it suffices to prove that \(\delta_1(x) \overset{s}{\approx}_{aE} \delta_2(x)\)
   - We get this directly from Definition 95

2. \(e = \lambda y.e'\):
   - We need to prove that \(\lambda y.e'\delta_1 \overset{s}{\approx}_{aE} \lambda y.e'\delta_2\)
   - This means from Definition 95 it suffices to prove that
   \[
   \forall i < s, \lambda y.e'\delta_1 \downarrow_i v_a \implies \lambda y.e'\delta_2 \downarrow_i v_b \land v_a \overset{s-i}{\approx}_{aV} v_b
   \]
   - This means that given some \(i < s\) s.t. \(\lambda y.e'\delta_1 \downarrow_i v_a\) it suffices to prove that
   \[
   \lambda y.e'\delta_2 \downarrow_i v_b \land v_a \overset{s-i}{\approx}_{aV} v_b \quad \text{(FTE-L0)}
   \]
   - From E-val we know that \(v_a = \lambda y.e'\delta_1\)
   - From (FTE-L0) we need to prove that
(a) \( \lambda y. e' \delta_2 \Downarrow v_b \).
From E-val we know that \( v_b = \lambda y. e' \delta_2 \)

(b) \( v_a \approx_{aV} v_b \).

We need to prove that \( \lambda y. e' \delta_1 \approx_{aV} \lambda y. e' \delta_2 \)

This means from Definition 95 it suffices to prove that

\[ \forall \epsilon', \epsilon_2', s' < s. \epsilon'_1 \approx_{aE} \epsilon'_2 \implies e' \delta_1[\epsilon'_1/y] \approx_{aE} e' \delta_2[\epsilon'_2/y] \]

This further means that given some \( \epsilon'_1, \epsilon'_2, s' \) s.t. \( \epsilon'_1 \approx_{aE} \epsilon'_2 \) it suffices to prove that

\[ e' \delta_1[\epsilon'_1/y] \approx_{aE} e' \delta_2[\epsilon'_2/y] \]

We get this from IH and Lemma 98

3. \( e = \text{fixy}. e' \).

We induct on \( s \)

IH: \( \forall s' < s. \delta_1 \approx_{aE} \delta_2 \implies \text{fixy}. e', \delta_1 \approx_{aE} \text{fixy}. e', \delta_2 \)

To prove: \( \delta_1 \approx_{aE} \delta_2 \implies \text{fixy}. e, \delta_1 \approx_{aE} \text{fixy}. e', \delta_2 \)

This means we are given \( \delta_1 \approx_{aE} \delta_2 \) and we need to prove \( \text{fixy}. e, \delta_1 \approx_{aE} \text{fixy}. e', \delta_2 \)

From Definition 95 it suffices to prove that

\[ \forall i < s. \text{fixy}. e', \delta_1 \Downarrow i, v_a \implies \text{fixy}. e, \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \]

This means given some \( i < s \) s.t. \( \text{fixy}. e', \delta_1 \Downarrow i, v_a \) and we need to prove \( \text{fixy}. e', \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \)

Since we are given that \( \text{fixy}. e', \delta_1 \Downarrow i, v_a \) therefore from E-fix we know that

\[ e'[\text{fixy}. e', \delta_1/y]_1 \Downarrow i-1, v_a \]

Instantiating with \( s - 1 \) and using Lemma 98 we get

\[ e'[\text{fixy}. e', \delta_1/y]_1 \Downarrow s-1, v_a \]

Let

\[ \delta'_1 = \delta_1 \cup \{ y \mapsto \text{fixy}. e, \delta_1 \} \]
\[ \delta'_2 = \delta_2 \cup \{ y \mapsto \text{fixy}. e, \delta_2 \} \]

From Lemma 98 and (F1) we know that \( \delta'_1 \approx_{aE} \delta'_2 \)

Therefore from IH of outer induction we know that we have

\[ e' \delta'_1 \approx_{aE} e' \delta'_2 \]

This means from Definition 95 we know that

\[ \forall i' < (s - 1). e' \delta'_1 \Downarrow i, v_a \implies e' \delta'_2 \Downarrow v_b \land v_a \approx_{aV} v_b \]

Instantiating with \( i - 1 \) and since we know that \( e' \delta'_1 \Downarrow i, v_a \) and therefore we get

\[ e' \delta'_2 \Downarrow v_b \land v_a \approx_{aV} v_b \]

which is the desired.

4. \( e = e_1 e_2 \).

We need to prove that \( e_1 e_2 \delta_1 \approx_{aE} e_1 e_2 \delta_2 \)

This means from Definition 95 it suffices to prove that

\[ \forall i < s. e_1 e_2 \delta_1 \Downarrow i, v_a \implies e_1 e_2 \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \]

This means that given some \( i < s \) s.t. \( e_1 e_2 \delta_1 \Downarrow i, v_a \) it suffices to prove that

\[ e_1 e_2 \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \quad \text{(FTE-A0)} \]

IH1: \( e_1 \delta_1 \approx_{aE} e_1 \delta_2 \)

Therefore from Definition 95 we have

\[ \forall j < s. e_1 \delta_1 \Downarrow j, v'_a \implies e_1 \delta_2 \Downarrow v'_b \land v'_a \approx_{aE} v'_b \quad \text{(FTE-A1)} \]

Since \( (e_1 \delta_1 e_2 \delta_1) \Downarrow i, v_a \) therefore from E-app we know that \( \exists i_1 < i. e_1 \delta_1 \Downarrow i_1, \lambda y. e' \)

Therefore instantiating (FTE-A1) with \( i_1 \) we get \( e_1 \delta_2 \Downarrow v'_b \land v'_a \approx_{aE} v'_b \quad \text{(FTE-A1.1)} \)

Since \( v'_a = \lambda y. e' \) and since \( v'_a \approx_{aE} v'_b \) therefore from Definition 95 we know that \( v'_b = \lambda y. e'' \)

153
Again since $\lambda y.e'^{s'}/\delta' \approx_{aV} \lambda y.e''$ therefore from Definition 95 we know that

\[ \forall e', e', s' < (s - i - 1), e'^{s'}/\delta' \approx_{aV} e'^{s'}/\delta' \iff e'[e'/y] \approx_{aV} e'[e'/y] \]  

(FTE-A2)

IH2: $e_2 \delta_1 \approx_{aE} e_2 \delta_2$

Instantiating (FTE-A2) with $e_2 \delta_1, e_2 \delta_2$ we get

\[ e'[e_2 \delta_1/y] \approx_{aE} e'[e_2 \delta_1/y] \]

From Definition 95 we have

\[ \forall j < (s - i - 1), e'[e_2 \delta_1/y] \approx_j v_a'' \iff e''[e_2 \delta_1/y] \approx_j v_a'' \approx_{aV} v_a'' \]  

(FTE-A2.1)

Since $(e_1 \delta_1 e_2 \delta_1) \downarrow i a_\text{v}$ therefore from E-app we know that $\exists i_2 = i - i_2 - 1.e'[e_2 \delta_1/x] \downarrow i_2 a_\text{v}$

Instantiating (FTE-A2.1) with $i_2$ we get $e''[e_2 \delta_1/y] \approx_{aV} v_a'' \approx_{aV} v_a''$

Since $i = i_1 + i_2 + 1$ therefore this proves (FTE-A0) and we are done.

5. $e = \langle e_1, e_2 \rangle$

We need to prove that $\langle e_1, e_2 \rangle \delta_1 \approx_{aE} \langle e_1, e_2 \rangle \delta_2$

This means from Definition 95 it suffices to prove that

\[ \langle e_1, e_2 \rangle \delta_1 \downarrow v_a \iff \langle e_1, e_2 \rangle \delta_2 \downarrow v_a \approx_{aV} v_a \]

(FTE-T0)

From E-TI we know that $v_a = \langle v_{a_1}, v_{a_2} \rangle$ and $e_1 \delta_1 \downarrow v_{a_1}$ and $e_2 \delta_1 \downarrow v_{a_2}$

IH1: $e_1 \delta_1 \approx_{aE} e_1 \delta_2$

Therefore from Definition 95 we have

\[ \langle e_1, e_2 \rangle \delta_1 \downarrow v_{a_1} \iff e_1 \delta_2 \downarrow v_{a_1} \approx_{aV} v_{a_1} \]

(FTE-T1)

Since we know that $e_1 \delta_1 \downarrow v_{a_1}$ therefore we get

\[ e_1 \delta_2 \downarrow v_{a_1} \approx_{aV} v_{a_1} \]  

(FTE-T1)

IH2: $e_2 \delta_1 \approx_{aE} e_2 \delta_2$

Similarly from Definition 95 we have

\[ \langle e_1, e_2 \rangle \delta_2 \downarrow v_{a_2} \iff e_2 \delta_2 \downarrow v_{a_2} \approx_{aV} v_{a_2} \]

(FTE-T2)

From (FTE-T0) we need to prove

\[ (a) \langle e_1, e_2 \rangle \delta_2 \downarrow v_b: \]

We get this from (FTE-T1), (FTE-T2) and E-TI

\[ (b) v_a \approx_{aV} v_b: \]

Since $i = i_1 + i_2$, $v_a = \langle v_{a_1}, v_{a_2} \rangle$ and $v_b = \langle v_{b_1}, v_{b_2} \rangle$ it suffices to prove that

\[ \langle v_{a_1}, v_{a_2} \rangle \approx_{aV} \langle v_{b_1}, v_{b_2} \rangle \]

From Definition 95 it suffices to prove that

\[ v_{a_1} \approx_{aV} v_{b_1} \text{ and } v_{a_2} \approx_{aV} v_{b_2} \]

We get this from (FTE-T1), (FTE-T2) and Lemma 96

6. $e = \text{let}(x, y) = e_1$ in $e_2$

We need to prove that let\langle x, y \rangle = e_1 in $e_2 \delta_1 \approx_{aE} \text{ let}(x, y) = e_1$ in $e_2 \delta_2$

This means from Definition 95 it suffices to prove that

\[ \langle x, y \rangle = e_1 \text{ in } e_2 \delta_1 \downarrow v_a \iff \text{ let}(x, y) = e_1 \text{ in } e_2 \delta_2 \downarrow v_b \approx_{aV} v_b \]

(FTE-TE0)

IH1: $e_1 \delta_1 \approx_{aE} e_1 \delta_2$

Therefore from Definition 95 we have
\[\forall i < s. e_1 \delta_1 \downarrow_i v_{a1} \Rightarrow e_1 \delta_2 \downarrow v_{b1} \land v_{a1} \approx^s_{aV} v_{b1}\]

Since we know that let \(\langle x, y \rangle = e_1\) in \(e_2 \delta_1 \downarrow_i v_a\) therefore from E-TE we know that \(\exists i_1 < s. e_1 \delta_1 \downarrow_i \langle\langle v'_{a1}, v''_{a2}\rangle\rangle\). Therefore we get
\[e_1 \delta_2 \downarrow v_{b1} \land v_{a1} \approx^s_{aV} v_{b1}\] (FTE-TE1)

Since \(v_{a1} \approx^s_{aV} v_{b1}\) and \(v_{a1} = \langle\langle v'_{a1}, v''_{a2}\rangle\rangle\) therefore from Definition \(\ref{def:approx}\) we have \(v_{b1} = \langle\langle v'_{b1}, v''_{b2}\rangle\rangle\) (FTE-TE1.1)

Let
\[\delta'_1 = \delta_1 \cup \{ x \mapsto \langle\langle v'_{a1}, v''_{a2}\rangle\rangle \}\]
\[\delta'_2 = \delta_2 \cup \{ x \mapsto \langle\langle v'_{b1}, v''_{b2}\rangle\rangle \}\]

**IH2**: \(e_2 \delta'_1 \approx^s_{aE} e_2 \delta'_2\)

Therefore from Definition \(\ref{def:approx}\) we have
\[\forall i < (s - i_1). e_2 \delta'_1 \downarrow_i v_a \Rightarrow e_2 \delta'_2 \downarrow v_{b2} \land v_a \approx^s_{aV} v_{b2}\]

Since we know that let \(\langle x, y \rangle = e_1\) in \(e_2 \delta_1 \downarrow_i v_a\) therefore from E-TE we know that \(\exists i_2 = i - i_1. e_2 \delta'_1 \downarrow_{i_2} v_a\). Therefore we get
\[e_2 \delta'_2 \downarrow v_{b2} \land v_a \approx^s_{aV} v_{b2}\] (FTE-TE2)

This proves the desired.

7. \(e = \langle e_{a1}, e_{a2}\rangle\):

Similar reasoning as in the \(\langle e_{a1}, e_{a2}\rangle\) case above.

8. \(e = \text{fst}(e')\):

We need to prove that \(\text{fst}(e')\delta_1 \approx^s_{aE} \text{fst}(e')\delta_2\)

This means from Definition \(\ref{def:approx}\) it suffices to prove that
\[\forall i < s. \text{fst}(e')\delta_1 \downarrow_i v_a \Rightarrow \text{fst}(e')\delta_2 \downarrow v_b \land v_a \approx^s_{aV} v_b\]

This means that given some \(i < s\) s.t \(\text{fst}(e')\delta_1 \downarrow_i v_a\) it suffices to prove that
\[\text{fst}(e')\delta_2 \downarrow v_b \land v_a \approx^s_{aV} v_b\] (FTE-F0)

Since we know that \(\text{fst}(e')\delta_1 \downarrow_i \langle\langle v_a, -\rangle\rangle\) we get \(\text{fst}(e')\downarrow_i v_a\)

**IH**: \(e' \delta'_1 \approx^s_{aE} e' \delta'_2\)

This means from Definition \(\ref{def:approx}\) we have
\[\forall j < s. e' \delta'_1 \downarrow_j v_{a1} \Rightarrow e' \delta'_2 \downarrow v_{b1} \land v_{a1} \approx^s_{aV} v_{b1}\]

Instantiating with \(i\) we get \(e' \delta'_2 \downarrow v_{b1} \land v_{a1} \approx^s_{aV} v_{b1}\)

Since we know that \(v_{a1} = \langle\langle v_a, -\rangle\rangle\) therefore from Definition \(\ref{def:approx}\) we also know that \(v_{b1} = \langle\langle v_b, -\rangle\rangle\) s.t \(v_a \approx^s_{aV} v_b\)

This proves the desired.

9. \(e = \text{snd}(e')\):

Similar reasoning as in the \(\text{fst}(e')\) case.

10. \(e = \text{inl}(e')\):

We need to prove that \(\text{inl}(e')\delta_1 \approx^s_{aE} \text{inl}(e')\delta_2\)

This means from Definition \(\ref{def:approx}\) it suffices to prove that
\[\forall i < s. \text{inl}(e')\delta_1 \downarrow_i v_a \Rightarrow \text{inl}(e')\delta_2 \downarrow v_b \land v_a \approx^s_{aV} v_b\]

This means that given some \(i < s\) s.t \(\text{inl}(e')\delta_1 \downarrow_i v_a\) it suffices to prove that
\[\text{inl}(e')\delta_2 \downarrow v_b \land v_a \approx^s_{aV} v_b\] (FTE-IL0)

Since we know that \(\text{inl}(e')\delta_1 \downarrow_i v_a\) therefore from E-inl we know that \(v_a = \text{inl}(v'_{a})\) and \(e' \delta_1 \downarrow_i v'_a\)

**IH**: \(e' \delta'_1 \approx^s_{aE} e' \delta'_2\)

This means from Definition \(\ref{def:approx}\) we have
\[\forall j < s. e' \delta'_1 \downarrow_j v_{a1} \Rightarrow e' \delta'_2 \downarrow v_{b1} \land v_{a1} \approx^s_{aV} v_{b1}\]

Instantiating with \(i\) we get \(e' \delta'_2 \downarrow v_{b1} \land v_{a1} \approx^s_{aV} v_{b1}\)

Since \(e' \delta'_2 \downarrow v_{b1}\) therefore from E-inl we have \(\text{inl}(e')\delta_2 \downarrow \text{inl}(v_{b1})\)

And since we know that \(v_{a1} \approx^s_{aV} v_{b1}\) therefore from Definition \(\ref{def:approx}\) we also know that \(\text{inl}(v_{a1}) \approx^s_{aV} \text{inl}(v_{b1})\)

This proves the desired.
11. $e = \text{inr}(e')$:
Similar reasoning as in the $\text{inl}(e')$ case

12. $e = \text{case } e_c, x.e_1, y.e_r$:
We need to prove that $\text{case } e_c, x.e_1, y.e_r \delta_1 \triangleq aE \text{ case } e_c, x.e_1, y.e_r \delta_2$

This means from Definition $[93]$ it suffices to prove that
\[ \forall i < s. \text{case } e_c, x.e_1, y.e_r \delta_1 \vdash v_a \implies \text{case } e_c, x.e_1, y.e_r \delta_2 \vdash v_b \land v_a \approx_{aV} v_b \]

This means that given some $i < s$ s.t $\text{case } e_c, x.e_1, y.e_r \delta_1 \vdash v_a$ it suffices to prove that
$\text{case } e_c, x.e_1, y.e_r \delta_2 \vdash v_b \land v_a \approx_{aV} v_b$ \text{(FTE-C0)}

Since we know that $\text{case } e_c, x.e_1, y.e_r \delta_1 \vdash v_a$ therefore two cases arise:
2 cases arise:
(a) $e_c \delta_1 \vdash \text{inl}(v_{c1})$:

$\text{IH1 } e_c \delta_1 \triangleq_{aE} e_c \delta_2$
This means from Definition $[93]$ we have
\[ \forall j < s. e_c \delta_1 \vdash_j v_{c1} \implies e_c \delta_2 \vdash v_{c2} \land v_{c1} \approx_{aV} v_{c2} \]

Since we know that instantiation with $i_1$ we get $e_c \delta_2 \vdash v_{c2} \land v_{c1} \approx_{aV} v_{c2}$

From Definition $[95]$ we know that $\exists v'_{c2}, v_{c2} = \text{inl}(v'_{c2})$ s.t $v'_{c1} \approx_{aV} v'_{c2}$

$\text{IH2 } e_c \delta_1 [v'_{c1}/x] \approx_{aE} e_c \delta_2 [v'_{c2}/x]$
This means from Definition $[93]$ we have
\[ \forall j < (s - i_1). e_c \delta_1 [v'_{c1}/x] \vdash_j v_{i1} \implies e_c \delta_2 [v'_{c2}/x] \vdash v_{i2} \land v_{i1} \approx_{aV} v_{b} \]

Since we know that instantiation with $i_2$ we get $e_c \delta_2 [v'_{c2}/x] \vdash v_{i2} \land v_{i1} \approx_{aV} v_{b}$

Therefore instantiating with $i_2$ we get $e_c \delta_2 [v'_{c2}/x] \vdash v_{i2} \land v_{i1} \approx_{aV} v_{b}$

This proves the desired
(b) $e_c \delta_1 \vdash \text{inr}(v_{c1})$:
Similar reasoning as in the previous case

13. $e = \text{le'}$:
We need to prove that $\text{le'} \delta_1 \triangleq_{aE} \text{le'} \delta_2$

This means from Definition $[93]$ it suffices to prove that
\[ \forall i < s. \text{le'} \delta_1 \vdash v_a \implies \text{le'} \delta_2 \vdash v_b \land v_a \approx_{aV} v_b \]

This means that given some $i < s$ s.t $\text{le'} \delta_1 \vdash v_a$ it suffices to prove that
$\text{le'} \delta_2 \vdash v_b \land v_a \approx_{aV} v_b$ \text{(FTE-B0)}

From $\text{E-val}$ we know that $v_a = \text{le'} \delta_1$ and $i = 0$

$\text{IH: } \text{le'} \delta_1 \triangleq_{aE} \text{le'} \delta_2$
From (FTE-B0) we need to prove that
(a) $\text{le'} \delta_2 \vdash v_b$:
From $\text{E-val}$ we know that $v_b = \text{le'} \delta_2$

(b) $v_a \approx_{aV} v_b$:
We need to prove that
$\text{le'} \delta_1 \triangleq_{aV} \text{le'} \delta_2$
This means from Definition $[95]$ it suffices to prove that
$\text{le'} \delta_1 \triangleq_{aE} \text{le'} \delta_2$
We get this directly from IH

14. $e = \text{let } x = e'_1 \text{ in } e'_2$:
We need to prove that $\text{let } x = e'_1 \text{ in } e'_2 \delta_1 \triangleq_{aE} \text{let } x = e'_1 \text{ in } e'_2 \delta_2$

This means from Definition $[93]$ it suffices to prove that
\[ \forall i < s. \text{let } x = e'_1 \text{ in } e'_2 \delta_1 \vdash v_a \implies \text{let } x = e'_1 \text{ in } e'_2 \delta_2 \vdash v_b \land v_a \approx_{aV} v_b \]
This means that given some \( i < s \) s.t \( \let!x = e'_i \) in \( e'_2 \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \) it suffices to prove that
\[
\let!x = e'_1 \in e'_2 \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \quad (\text{FTE-BE0})
\]

**IH1:** \( e'_1 \delta_1 \approx_{aE} e'_2 \delta_2 \)
This means from Definition [95] we have
\[
\forall j < s.e'_1 \delta_1 \Downarrow j v_{a11} \implies e'_2 \delta_2 \Downarrow v_{a1} \land v_{a1} \approx_{aV} v_{b11}
\]
Since we know that \( \let!x = e'_1 \) in \( e'_2 \delta_1 \Downarrow i_v a \) therefore from E-val we know that \( \exists i_1.e'_1 \delta_1 \Downarrow i_1 v_{b1} \)

Instantiating with \( i_1 \) we get \( e'_1 \delta_2 \Downarrow v_{b11} \land v_{a11} \approx_{aV} v_{b11} \)
Since we know that \( v_{a11} = e_{b1} \) therefore from Definition [95] we also know that
\[
\let!x = e'_1 \Downarrow v \quad \text{s.t} \quad e_{b1} \approx_{aE} e_{b2}
\]

**IH2:** \( e'_2[e_{b1}/x] \delta_1 \Downarrow s^{-1}_{i1} e'_2[e_{b2}/x] \delta_2 \)
This means from Definition [95] we have
\[
\forall j < s.e'_2[e_{b1}/x] \delta_1 \Downarrow j v_a \implies e'_2[e_{b2}/x] \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b
\]
Since we know that \( \let!x = e'_1 \) in \( e'_2 \delta_1 \Downarrow i_v a \) therefore from E-subExpE we know that \( \exists i_2.e'_1[e_{b1}/x] \delta_1 \Downarrow i_2 v_a \)

Instantiating with \( i_2 \) we get \( e'_2[e_{b2}/x] \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \)
This proves the desired

15. \( e = \Lambda.e' \):
Similar reasoning as in the \( \lambda y.e' \) case

16. \( e = e' [] \):
Similar reasoning as in the app case

17. \( e = \text{ret} e' \):
We need to prove that \( \text{ret} e' \delta_1 \approx_{aE} \text{ret} e' \delta_2 \)
This means from Definition [95] it suffices to prove that
\[
\forall i < s.\text{ret} e' \delta_1 \Downarrow i v_a \implies \text{ret} e' \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b
\]
This means that given some \( i < s \) s.t \( \text{ret} e' \delta_1 \Downarrow i v_a \) it suffices to prove that
\[
\text{ret} e' \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \quad (\text{FTE-R0})
\]
From E-val we know that \( v_a = \text{ret} e' \delta_1 \) and \( i = 0 \)
From (FTE-R0) we need to prove that
(a) \( \text{ret} e' \delta_2 \Downarrow v_b \):
From E-val we know that \( v_b = \text{ret} e' \delta_2 \)
(b) \( v_a \approx_{aV} v_b \):
We need to prove that
\[
\text{ret} e' \delta_1 \approx_{aV} \text{ret} e' \delta_2
\]
This means from Definition [95] it suffices to prove that
\[
\text{ret} e' \delta_1 \Downarrow v_a \implies \text{ret} e' \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b
\]
This further means that given some \( \text{ret} e' \delta_1 \Downarrow v_a \) it suffices to prove that
\[
\text{ret} e' \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \quad (\text{FTE-R1})
\]
From E-return we know that \( k = 0 \) and \( e' \delta_1 \Downarrow v_a \)

**IH:** \( e' \delta_1 \approx_{aE} e' \delta_2 \)
This means from Definition [95] we have
\[
\forall j < s.e' \delta_1 \Downarrow j v_a \implies e' \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b
\]
Since we are given that \( e' \delta_1 \Downarrow v_a \) therefore we get
\( e' \delta_2 \Downarrow v_b \land v_a \approx_{aV} v_b \)
Since \( e' \delta_2 \Downarrow v_b \) therefore from E-return we also have
\[
\text{ret} e' \delta_2 \Downarrow v_b
\]
This proves the desired
18. \( e = \text{bind } x = e_b \) in \( c_e \):

We need to prove that \( \text{bind } x = e_b \) in \( c_e \delta_1 \approx_{aE} \text{bind } x = e_b \) in \( c_e \delta_2 \)

This means from Definition [95] it suffices to prove that

\[ \forall i < s. \text{bind } x = e_b \) in \( c_e \delta_1 \downarrow_i v_a \implies \text{bind } x = e_b \) in \( c_e \delta_2 \downarrow_i v_b \land v_a \approx_{aV} v_b \]

This means that given some \( i < s \) s.t \( \text{bind } x = e_b \) in \( c_e \delta_1 \downarrow_i v_a \) it suffices to prove that

\[ \text{bind } x = e_b \) in \( c_e \delta_2 \downarrow_i v_b \land v_a \approx_{aV} v_b \quad (\text{FTE-B10}) \]

From E-val we know that \( v_a = \text{bind } x = e_b \) in \( c_e \delta_1 \) and \( i = 0 \)

We need to prove

(a) \( \text{bind } x = e_b \) in \( c_e \delta_2 \downarrow_i v_b \):

From E-val we know that \( v_b = \text{bind } x = e_b \) in \( c_e \delta_2 \)

(b) \( v_a \approx_{aV} v_b \):

We need to prove that \( \text{bind } x = e_b \) in \( c_e \delta_1 \approx_{aV} \text{bind } x = e_b \) in \( c_e \delta_2 \)

From Definition [95] it suffices to prove that

\[ \text{bind } x = e_b \) in \( c_e \delta_1 \downarrow_i v_1 \implies \text{bind } x = e_b \) in \( c_e \delta_2 \downarrow_i v_2 \quad (\text{F-B11}) \]

\[ \text{IH1: } c_e \delta_1 \approx_{aE} c_e \delta_2 \]

This means from Definition [95] we have

\[ \forall j < s.e_c \delta_1 \downarrow_j v_{a1} \implies e_b \delta_2 \downarrow j v_b \land v_{a1} \approx_{aV} v_b \]

Since we know that \( \text{bind } x = e_b \) in \( c_e \delta_1 \downarrow_i v_a \) therefore from E-bind we know that \( \exists i_1.e_b \delta_1 \downarrow_{i_1} v_{a1} \)

Instantiating with \( i_1 \) we get \( e_b \delta_2 \downarrow_{i_1} v_{b1} \land v_{a1} \approx_{aV} v_{b1} \)

Since \( v_{a1} \) is a monadic value and \( v_{a1} \downarrow k_1 v'_{a1} \)

Since \( v_{a1} \approx_{aV} v_{b1} \) therefore from Definition [95] we know that

\[ v_{a1} \downarrow k_1 v'_{a1} \implies v_{b1} \downarrow k_1 v'_{b1} \land v_{a1} \approx_{aV} v_{b1} \]

Since we are given that \( v_{a1} \downarrow k_1 v'_{a1} \) there we have

\[ v_{b1} \downarrow k_1 v'_{b1} \land v_{a1} \approx_{aV} v_{b1} \]

\[ \text{IH2: } e_c[e'_{a1}/x] \delta_1 \approx_{aE} e_c[e'_{a2}/x] \delta_2 \]

This means from Definition [95] we have

\[ \forall j < s.e_c[e'_{a1}/x] \delta_1 \downarrow j v_{a2} \implies e_c[e'_{b1}/x] \delta_2 \downarrow j v_b \land v_{a2} \approx_{aV} v_{b2} \]

Since we know that \( \text{bind } x = e_b \) in \( c_e \delta_1 \downarrow_i v_a \) therefore from E-bind we know that \( \exists i_2.e_c[e'_{a1}/x] \delta_1 \downarrow_{i_2} v_{a2} \)

Instantiating with \( i_2 \) we get \( e_c[e'_{b1}/x] \delta_2 \downarrow_{i_2} v_b \land v_{a2} \approx_{aV} v_{b2} \)

From E-bind we know that \( v_{a2} \) is a monadic value and \( v_{a2} \downarrow k_2 v'_{a2} \)

Since \( v_{a2} \approx_{aV} v_{b2} \) therefore from Definition [95] we know that

\[ v_{a2} \downarrow k_2 v'_{a2} \implies v_{b2} \downarrow k_2 v'_{b2} \land v_{a2} \approx_{aV} v_{b2} \]

Since we are given that \( v_{a2} \downarrow k_2 v'_{a2} \) therefore we have

\[ v_{b2} \downarrow k_2 v'_{b2} \land v_{a2} \approx_{aV} v_{b2} \]

This proves the desired

19. \( e = \uparrow^n \)

Trivial

20. \( e = \text{release } e_r = x \) in \( c_e \):

Similar reasoning as in the bind case
Lemma 100 (Equivalence relation of λ-amor is reflexive for values). \( \forall v, s. v \approx_{sE} v \)

Proof. Instantiating Theorem 99 with \( \delta_1 \) and \( \delta_2 \), \( v \) for \( e \) and with the given \( s \) we get \( v \approx_{sE} v \)

From Definition 95 this means we have

\[ \forall i < s, v \downarrow_i v_a \Rightarrow v \downarrow v_b \approx_{s_{i}} v_b \]

Instantiating it with \( i \) as 0 and since we know that \( v \downarrow_0 v \) therefore we get the desired

Lemma 101 (Property of app rule in λ-Amor). \( \forall e_1, e_2, e, s. 
\]

Proof. We get the desired from Theorem 99
\[ e_{t-1} = \langle \lambda x_{n-1} x_n t u \rangle = \lambda p_1 \cdot \lambda y \cdot \lambda p_2 \cdot \text{let} \ x = y \text{ in release} = = p_1 \text{ in release} = = p_2 \text{ in bind} a = \text{store}() \text{ in } e_{t-1} a \]

where

\[ e_{t-1} = \langle \lambda x_{n-1} t u \rangle \]

\[ e_t = \langle t u \rangle \]

\[ e' = \langle (t u) \rangle = \lambda p \cdot \text{release} \ = = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_t a \text{ in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b (\text{coerce}1 \!\! e_u c) \]

where

\[ e_t = \bar{t} \]

\[ e_u = \bar{u} \]

Since we know that \( \|((t u, \rho, e))\| \downarrow v_a \downarrow^j v_1 \) therefore from the reduction rule we know that

\[ \exists j, L, \|() \downarrow^j L \text{ and } \exists j, L (\text{coerce}1!()\|() \downarrow^j v_1 \text{ s.t. } j = j_1 + j_a \]

Similarly from (A1.1) we know that

\[ \|((t, \rho, (u, \rho), e))\| = ((\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_n \rangle) = ((\lambda x_1 \ldots x_n, u) \langle C_1 \rangle \ldots \langle C_n \rangle) \]

Since \( \Theta; \triangle : \vdash ((t, \rho, (u, \rho), e)) : \) therefore from Theorem 74 we know that

\[ \|((t, \rho, (u, \rho), e))\| = ((\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_n \rangle) = (\lambda p \cdot \text{release} \ = = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_t a \text{ in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b (\text{coerce}1 ! e_{u,n} c) \]

where

\[ e_{t,n} = \langle (\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_n \rangle \rangle \]

\[ e_{u,n} = \langle (\lambda x_1 \ldots x_n, u) \langle C_1 \rangle \ldots \langle C_n \rangle \rangle \]

\[ e_{t,n} = \langle (\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_n \rangle \rangle = (\lambda p \cdot \text{release} \ = = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_{t,1} a \text{ in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b (\text{coerce}1 ! e_{t,2,n} c) \]

where

\[ e_{t,1,n} = \langle (\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_{n-1} \rangle \rangle \]

\[ e_{t,2,n} = \langle C_n \rangle \]

\[ e_{t,1,n} = \langle (\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_{n-1} \rangle \rangle = (\lambda p \cdot \text{release} \ = = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_{t,1,n-1} a \text{ in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b (\text{coerce}1 ! e_{t,2,n-1} c) \]

where

\[ e_{t,1,n-1} = \langle (\lambda x_1 \ldots x_n t) \langle C_1 \rangle \ldots \langle C_{n-2} \rangle \rangle \]

\[ e_{t,2,n-1} = \langle C_{n-1} \rangle \]

\[ e_{t,1,2} = \langle (\lambda x_1 \ldots x_n t) \langle C_1 \rangle \rangle = (\lambda p \cdot \text{release} \ = = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_{t,1} a \text{ in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b (\text{coerce}1 ! e_{t,2} c) \]

where

\[ e_{t,1} = \langle (\lambda x_1 \ldots x_n t) \rangle \]

\[ e_{t,2,1} = \langle C_t \rangle \]

\[ e_{t,1} = \langle (\lambda x_1 \ldots x_n t) \rangle = (\lambda p \cdot \text{release} \ = = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_{t,1} a \text{ in } \text{bind } c = \text{store}() \text{ in } e_{t,2} a \]

where
\[ e_{t_2} = (\lambda x_2 \ldots x_n. t) \]

\[ e_{t_n} = (\lambda x_n. t) = \lambda p_1. \text{ret } \lambda y. \lambda p_2. \text{let } ! x_n = y \text{ in release} = p_1 \text{ in release} = p_2 \text{ in bind } a = \text{store()} \text{ in } e_T a \]
where
\[ e_T = \bar{t} \quad (A1.2) \]

Similarly we also have
\[ e_{u_1,n} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_n \rangle) \]
\[ e_{u_1,n} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_n \rangle) = \lambda p. \text{release} = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{u_1,n} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u_2,n} c \rangle) d \]
where
\[ e_{u_1,n} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_n \rangle) \]
\[ e_{u_2,n} = C_n \]

\[ e_{u_1,n} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_n \rangle) = \lambda p. \text{release} = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{u_1,n-1} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u_2,n-1} c \rangle) d \]
where
\[ e_{u_1,n-1} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_{n-2} \rangle) \]
\[ e_{u_2,n-1} = C_{n-1} \]

\[ e_{u_1,n-1} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_{n-2} \rangle) = \lambda p. \text{release} = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{u_1,n-2} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u_2,n-2} c \rangle) d \]
where
\[ e_{u_1,n-2} = ((\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \ldots \langle C_{n-3} \rangle) \]
\[ e_{u_2,n-2} = C_{n-2} \]

\[ e_{u_1,2} = (\lambda x_1 \ldots x_n. u) \langle C_1 \rangle \]
\[ \lambda p. \text{release} = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{u_1,1} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u_2,1} c \rangle) d \]
where
\[ e_{u_1,1} = (\lambda x_1 \ldots x_n. u) \]
\[ e_{u_2,1} = C_1 \]

\[ e_{u_1,1} = (\lambda x_1 \ldots x_n. u) \]
\[ \lambda p_1. \text{ret } \lambda y. \lambda p_2. \text{let } ! x_1 = y \text{ in release} = p_1 \text{ in release} = p_2 \text{ in bind } a = \text{store()} \text{ in } e_{U,1} a \]
where
\[ e_{U,1} = (\lambda x_2 \ldots x_n. u) \]
\[ e_{U,1} = (\lambda x_2 \ldots x_n. u) = \lambda p_1. \text{ret } \lambda y. \lambda p_2. \text{let } ! x_2 = y \text{ in release} = p_1 \text{ in release} = p_2 \text{ in bind } a = \text{store()} \text{ in } e_{U,2} a \]
where
\[ e_{U,2} = (\lambda x_3 \ldots x_n. u) \]
\[ \ldots \]
\[ e_{U,n-1} = (\lambda x_n. u) \]
\[ \lambda p_1. \text{ret } \lambda y. \lambda p_2. \text{let } ! x_n = y \text{ in release} = p_1 \text{ in release} = p_2 \text{ in bind } a = \text{store()} \text{ in } e_{U,n} a \]
where
\[ e_{U,n} = \bar{u} \quad (A1.3) \]

\[ E_0 = \lambda p. \text{release} = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{t,n} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u,n} c \rangle) d \]
\[ e_{v_1} = \text{release} = () \text{ in bind } a = \text{store()} \text{ in bind } b = e_{t,n} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u,n} c \rangle) d \]
\[ E_{0,1} = \text{bind } a = \text{store()} \text{ in bind } b = e_{t,n} a \text{ in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b (\text{coerce}1 \langle e_{u,n} c \rangle) d \]

161
\[
D_{12}:
\]

\[
E_{t,3,1} (\text{coerce}\ 1 \equiv [\text{C}]) (v) \Downarrow E_{t,4,1}
\]

\[
D_{11}:
\]

\[
e_{t,1,2} (v) \Downarrow E_{t,1,2,1} \quad D_{21} \quad D_{22}
\]

\[
E_{t,1,3,1} \Downarrow E_{t,1,3,1}
\]

\[
D_{n-2}:
\]

\[
E_{t, (n-2), 1} (\text{coerce} \ 1 \equiv [\text{C}_{n-2}]) (v) \Downarrow E_{t, (n-1), 1}
\]

\[
e_{t,1,n-3} (v) \Downarrow E_{t,1,n-3,1} \quad D_{31} \quad D_{32}
\]

\[
E_{t,1,n-2,1} \Downarrow E_{t,1,n-2,1}
\]

\[
D_{n-1}:
\]

\[
E_{t, n-1, 1} (\text{coerce} \ 1 \equiv [\text{C}_{n-1}]) (v) \Downarrow E_{t, n, 1}
\]

\[
e_{t,1,n-2} (v) \Downarrow E_{t,1,n-2,1} \quad D_{n-2,1} \quad D_{n-2,2}
\]

\[
E_{t,1,n-1,1} \Downarrow E_{t,1,n-1,1}
\]

\[
D_n:
\]

\[
\tilde{v} / [(\text{C}_{a} (v)) / x_n] \Downarrow \Downarrow \tilde{v}^n L
\]

By inversion

\[
E_{t, n, 1} [\text{coerce} \ 1 \equiv [\text{C}_{a} (v)) / x_n]] (v) \Downarrow / \tilde{v}^n L
\]

\[
E_{t, n, 1} (\text{coerce} \ 1 \equiv [\text{C}_{a} (v)) / x_n] / p_2) \Downarrow \tilde{v}^n L
\]

\[
e_{t,1,n-1} (v) \Downarrow E_{t,1,n-1,1} \quad D_{(n-1),1} \quad D_{(n-1),2}
\]

\[
E_{t,1,n,1} \Downarrow \tilde{v} E_{t,1,n,1}
\]

\[
D_2:
\]

\[
v_a \Downarrow \tilde{v}_1 \quad \text{Given} \quad v_a \overset{\approx}{\equiv} a v_b \quad v_b \Downarrow \tilde{v}_2 \quad v_1 \overset{\approx}{\equiv} a V \quad v_2
\]

Definition \[\text{02}\]

\[\text{T1:}\]

\[
L (\text{coerce} \ 1 e_{a,n} !() (v)) \Downarrow \Downarrow \Downarrow \tilde{v}^n v_b \quad v_a \overset{\approx}{\equiv} a V \quad v_b
\]

\[
E_{0,4}[L / b] [!(v) / c] \Downarrow \tilde{v}^n v_b
\]

\[
E_{0,4}[L / b] \Downarrow \tilde{v}^n v_b
\]

\[\text{T0:}\]

\[
e_{t,1,n} (v) \Downarrow E_{t,1,n,1} \quad D_{n,1} \quad D_{n,2}
\]

\[
E_{t,1,n,1} \Downarrow \tilde{v} E_{t,1,n,1}
\]

\[\text{E-bind}\]

\[\text{D0.0:}\]

\[
e_{t,n} (v) \Downarrow E_{t,n,1} \quad T_0 \quad T_1 \quad D_2
\]

\[
E_{0,2} \Downarrow \tilde{v} v_2
\]

\[\text{E-bind}\]

\[
v_b \Downarrow \tilde{v} v_2
\]

\[\text{E-release}\]

163
Main derivation:

\[
D0.0 \quad \frac{E_0() \Downarrow v_b}{E_0() \Downarrow v_b \Downarrow v_2}
\]

\[
((\lambda x_1 \ldots x_n.t) \ (C_1) \ \ldots \ (C_n)) \ (\lambda x_1 \ldots x_n.u) \ (C_1) \ \ldots \ (C_n) (\Downarrow) \Downarrow v_b \Downarrow v_2
\]

Claim: \( \forall s.coerce1 \! ! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \overset{\approx_1}{\sim} \lambda E \! ! e_{u,n} !() \)

Proof

From Definition 83 it suffices to prove

\( \forall i < s.coerce1 \! ! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \Downarrow v_i \implies \! ! e_{u,n} !() \Downarrow v_2 \wedge v_i \overset{\approx_1}{\sim} \lambda v_2 \)

This further means that given some \( i < s \) s.t. \( \! ! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \Downarrow v_i \) and we need to prove

\( \! ! e_{u,n} !() \Downarrow v_2 \wedge v_i \overset{\approx_1}{\sim} \lambda v_2 \) \tag{C0}

Since we are given that \( \! ! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \Downarrow v_1 \)
This means from Definition 83 we have \( v_1 = ! (C_1) / x_1 \ldots (C_n) / x_n \) !()

Similarly again from Definition 83 we know that \( v_2 = ! (e_{u,n} !()) \)

In order to prove that \( ! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \overset{\approx_1}{\sim} \lambda e_{u,n} !() \)
from Definition 83 it suffices to prove that

\( ! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \overset{\approx_1}{\sim} \lambda (e_{u,n} !()) \)

Using Definition 85 it suffices to prove

\( \forall j < (s-i).! (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \Downarrow v_j \implies (e_{u,n} !()) \Downarrow v_2 \wedge v_1 \overset{\approx_1}{\sim} \lambda v_2 \)

This means given some \( j < (s-i) \) s.t. \( (C_1) / x_1 \ldots (C_n) / x_n \) !() \( \Downarrow v_j \)

it suffices to prove that

\( (e_{u,n} !()) \Downarrow v_2 \wedge v_1 \overset{\approx_1}{\sim} \lambda v_2 \)

From the embedding of d\( i \)-PCF into \( \lambda \)-amor we know that \( v_1 \) is a value of monadic type

Since we know that

\( e_{u,n} = A批准 \)

\( release = p \) in \( bind a = store() \) in \( bind b = e_{u_1,n} a \) in \( bind c = store() \) in \( bind d = store() \) in \( b \ (coerce1 \! ! e_{u_2,n} !()) \)

where

\[ e_{u_1,n} = ((\lambda x_1 \ldots x_n.u) \ (C_1) \ \ldots \ (C_n)) \]

\[ e_{u_2,n} = C_n \]

\[ e_{u,n} !() \Downarrow v_2 \) from E-\( \text{app} \)

\( v_2 = release = () \) in \( bind a = store() \) in \( bind b = e_{u_1,n} a \) in \( bind c = store() \) in \( bind d = store() \) in \( b \ (coerce1 \! ! e_{u_2,n} !()) \)

Now we need to prove that \( v_1 \overset{\approx_1}{\sim} \lambda v_2 \)

From Definition 85 it suffices to prove that

\( v_1 \Downarrow v_a \implies v_2 \Downarrow \overset{\approx_1}{\sim} \lambda v_2 \)

This means given \( v_1 \Downarrow v_a \) it suffices to prove

\( v_2 \Downarrow \overset{\approx_1}{\sim} \lambda v_2 \)

\( v_2 = release = () \) in \( bind a = store() \) in \( bind b = e_{u_1,n} a \) in \( bind c = store() \) in \( bind d = store() \) in \( b \ (coerce1 \! ! e_{u_2,n} !()) \)

\( E_{u_1,n} = bind a = store() \) in \( bind b = e_{u_1,n} a \) in \( bind c = store() \) in \( bind d = store() \) in \( b \ (coerce1 \! ! e_{u_2,n} !()) \)

\( E_{u,n,1} = bind b = e_{u_1,n} () \) in \( bind c = store() \) in \( bind d = store() \) in \( b \ (coerce1 \! ! e_{u_2,n} !()) \)
\[ E_{u,n,1.2} = \text{bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,n} \ c) \ d \]
\[ e_{u,n} = \lambda p. \]
release \( p \) in bind \( a = \text{store}() \) in bind \( b = e_{u,n-1} \ a \) in bind \( c = \text{store}() \) in bind \( d = \text{store}() \) in bind \( d = \text{store}() \) in bind \( d = \text{store}() \) in\( b \ (\text{coerce1 } l_{e_u,n-1} \ c) \ d \)
\[ E_{u,1,n} = \text{release} \ - \ = \ () \text{ in bind } a = \text{store}() \text{ in bind } b = e_{u,1,n} \ a \text{ in bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in} \]
\[ b \ (\text{coerce1 } l_{e_u,1} \ c) \ d \]
\[ E_{u,1,n,2} = \text{bind } b = e_{u,1,n-1} () \text{ in bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,n} \ c) \ d \]
\[ E_{u,1,n,3} = \text{bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,n} \ c) \ d \]
\[ E_{u,1,n,4} = \text{bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,n} \ c) \ d \]
\[ e_{u,1} = \lambda p. \text{release} \ - \ = \ () \text{ in bind } a = \text{store}() \text{ in bind } b = e_{u,1} \ a \text{ in bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,1} \ c) \ d \]
\[ E_{u,1.2.1} = \text{release} \ - \ = () \text{ in bind } a = \text{store}() \text{ in bind } b = e_{u,1} \ a \text{ in bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,1} \ c) \ d \]

\[ E_{u,1.2.2} = \text{bind } b = e_{u,1} \ a \text{ in bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,1} \ c) \ d \]
\[ E_{u,1.2.3} = \text{bind } c = \text{store}() \text{ in bind } d = \text{store}() \text{ in } b \ (\text{coerce1 } l_{e_u,1} \ c) \ d \]
\[ e_{u,1} = \lambda p. \text{ret} \ y. \lambda p. \text{bind} \ a \ (\text{let } x_1 = y \text{ in } \text{release} \ - \ = p_1 \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,2} a) \]
\[ E_{u,1} = \text{ret} \ y. \lambda p. \text{let} ! x_1 = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,2} a \]
\[ E_{u,1.1,1} = \lambda y. \lambda p. \text{let} ! x_1 = y \text{ in } \text{release} \ - \ () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,2} a \]
\[ E_{u,1.1,2} = \text{let} ! x_1 = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,2} a \]
\[ E_{u,1.1,3} = \text{release} \ - \ = () \text{ in } \text{release} \ - \ = () \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,2} a [(\text{coerce1 } l_{e_u,1} () / x_1)] \]
\[ E_{u,1.2} = \text{ret} \ y. \lambda p. \text{let} ! x_2 = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,3} a [(\text{coerce1 } l_{e_u,1} )] / x_1] \]
\[ E_{u,1.2,1} = \lambda y. \lambda p. \text{let} ! x_2 = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,3} a [(\text{coerce1 } l_{e_u,1} ) / x_1] \]
\[ E_{u,1.2,2} = \text{release} \ - \ = () \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,3} a [(\text{coerce1 } l_{e_u,1} ) / x_1] \]
\[ E_{u,1.2} = \text{ret} \ y. \lambda p. \text{let} ! x_3 = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,4} a) \]
\[ E_{u,1.3,1} = \lambda y. \lambda p. \text{let} ! x_3 = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,4} a) \]
\[ S_2 = ([(\text{coerce1 } l_{e_u,1} ) / x_1][[(\text{coerce1 } l_{e_u} ) / x_2]] \]
\[ E_{u,n,1} = \lambda y. \lambda p. \text{let} ! x_n = y \text{ in } \text{release} \ - \ = () \text{ in } \text{release} \ - \ = p_2 \text{ in } \text{bind } a = \text{store}() \text{ in } e_{U,n} a) \]
\[ S_{n-1} = [(\text{coerce1 } l_{e_u,1} ) / x_1] \ldots ([(\text{coerce1 } l_{e_u,1} ) / x_{n-1}]] \]

\[ D_{n-3}: \]
\[ E_{L(n-3),1} (\text{coerce1 } l_{[C_{n-3}]} !()) () \downarrow \downarrow E_{L(n-2),1} \]

\[ D_2: \]
\[ E_{L,1,1,1} (\text{coerce1 } l_{[C_{1}]} !()) () \downarrow \downarrow E_{L,1,2,1} \]

\[ D_1: \]
\[ E_{L,1,1} \downarrow \downarrow E_{L,1,1} \]

\[ D_2: \]
\[ E_{L,1,3,1} \downarrow \downarrow E_{L,1,3,1} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]

\[ D_2: \]
\[ E_{L,1,2,1} \downarrow \downarrow E_{L,1,2,2} \]
Lemma 103

Proof.

We prove this by induction on $\theta$.

From Lemma 100 we get $v_a \approx_{s-i-j-l} aV v'_a$
\[\exists \nu_{\theta_2}, \nu_{\nu_2}. \nu'_{\theta_2} \eta''_{\theta_2}, \nu_{\theta_2}, \phi_{\theta_2}, j''_{\theta_2}. \\]
\[\langle (\nu, \rho, (u, \rho), \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_2 \land (j - j') = (j'' - j'') \land \forall s. \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2} \quad \text{(ET-0)}\]

From IH we know
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) \text{ and } (t, u, \rho, \theta_{\theta_2}) \text{ are well-typed} \land \]
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) \rightarrow (t, u, \rho, \theta_{\theta_2}) \land \langle (t, u, \rho, \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}. \quad \text{(ET-IH)}\]

From Definition \ref{def:termination} and Definition \ref{def:results} we know that
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) = \langle (t, u, \rho, \theta_{\theta_2}) \rangle \land \text{(ET-1)}\]

Since \((t, u, \rho, (u, \rho), \theta_{\theta_2})\) is well typed therefore we know that
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) \land \text{well-typed} \land \]
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) \rightarrow (t, u, \rho, \theta_{\theta_2}) \land \langle (t, u, \rho, \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}. \quad \text{(ET-2)}\]

From Definition \ref{def:termination} and Definition \ref{def:results} it suffices to prove that
\[\exists j''_{\theta_2}, \nu_{\nu_2}. \langle (t, u, \rho, (u, \rho), \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\nu_2} \land (j - j') = (j'' - j'') \land \forall s. \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2} \quad \text{(ET-3)}\]

Since \((t, u, \rho, (u, \rho), \theta_{\theta_2})\) is well typed therefore we know that
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) \land \text{well-typed} \land \]
\[(t, u, \rho, (u, \rho), \theta_{\theta_2}) \rightarrow (t, u, \rho, \theta_{\theta_2}) \land \langle (t, u, \rho, \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}. \quad \text{(ET-p)}\]

Since we are given that \(\langle (t, u, \rho, \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}\) this means from (ET-1.1) we have
\[\langle (t, u, \rho, \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}. \quad \text{also this means we have}\]
\[\langle (t, u, \rho, \theta_{\theta_2}) \rangle \eta''_{\theta_2} \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}. \quad \text{(ET-5)}\]

This means \(\nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}\) and \(\nu_{\theta_2} = \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}\) therefore from Definition \ref{def:termination} we get \(\forall s. \nu_{\theta_1} \tilde{\varepsilon}_{aV} \nu_{\theta_2}. \quad \text{Also from Definition \ref{def:termination} we have}\]
\[j'' - j' = j'' - j' = j'' - j' = j'' - j' = j' \quad \text{(From ET-IH)}\]
Lemma 104 (Cost and size lemma). \( \forall e_s, D_s, E_s. \)

\[
(e_s, e, c) \rightarrow D_s \rightarrow E_s \land D_s \text{ is well-typed} \land E_s \text{ is well-typed} \land \\
\begin{align*}
&\text{let } e_t = \{D_s\} \land e_i \downarrow v_a \downarrow^j v_1 \\
&\implies \exists v_{t_i}. e_t = \{E_s\} \land e_i (\downarrow v_a \downarrow^j v_2) \land \forall s. v_1 = \approx_{a \in E_s} v_2 \land \\
&1. j' = j - \text{with } |D_s| > |E_s| \lor \\
&2. j' = j - 1 \land |E_s| < |D_s| + |e_s| \\
\end{align*}
\]

Proof. We case analyze on the \( D_s \rightarrow E_s \) reduction.

1. App1:

Given \( D_s = (t, u, \rho, \theta) \) and \( E_s = (t, \rho, (u, \rho).\theta) \)

Let \( D_s' = (t, u, \rho, c) \) and \( E_s' = (t, \rho, (u, \rho).c) \)

Since we are given that \( D_s \) is well-typed and \( E_s \) is well-typed therefore from Lemma 105 we also have \( D_s' \) is well-typed and \( E_s' \) is well-typed.

Also since we know that \( e_i \downarrow v_a \downarrow^j v_1 \) therefore from Lemma 106 we also know that \( \exists v_{t_i}. \{E_s\} \downarrow v_{t_i} \downarrow^j v_2 \) s.t. \( \forall s. v_1 = \approx_{a \in E_s} v_2 \)

And finally from Lemma 102 we know that \( \exists v_{t_i}. \{E_s\} \downarrow v_{t_i} \downarrow^j v_2 \) s.t. \( \forall s. v_1 = \approx_{a \in E_s} v_2 \)

\(|D_s| > |E_s|\) holds directly from the Definition of \(|-|\)

2. App2:

Given: \( (\lambda x.t, \rho, c.\theta) \rightarrow (t, c, \rho, \theta) \)

We induct on \( \theta \)

(a) Case \( \theta = c \):

Since we are given that \( D_s \) i.e \( (\lambda x.t, \rho, c.e) \) is well typed

Therefore from Theorem 94 \( (\lambda x.t, \rho, c.e) \) is well-typed

From Definition \( 92 \) \( (\lambda x.t, \rho, c.\theta) \) is well-typed

Again from Definition \( 92 \) \( (\lambda x.t, \rho, \theta) \) is well-typed

From Definition \( 91 \) we have

\((\lambda x_1 \ldots x_n.\lambda x.t) (C_1) \ldots (C_n) (c)\) is well-typed

Therefore from Theorem \( 94 \) we know that

\[
\{D_s\} = \\
((\lambda x_1 \ldots x_n.\lambda x.t) (C_1) \ldots (C_n) (C)) = \\
\lambda \rho. \text{release} \quad \text{ releases } = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_{t_1} \text{ a in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b \text{ (coerce } e_{t_2} \text{ c) d where}
\]

\[
e_{t_1} = ((\lambda x_1 \ldots x_n.\lambda x.t) (C_1) \ldots (C_n))
\]

\[
e_{t_2} = \{C\} \quad (S-A0)
\]

Since we are given that \( \{D_s\} \downarrow v_a \downarrow^j v_1 \)

therefore from the evaluation rules we know that

\[
\{\} (\{C\}/x) (\{C_1\}/x_1) \ldots (\{C_1\}/x_1) \downarrow v_1 \quad (S-A0.1)
\]

Similarly since we are given that \( D_s \) i.e \( (t, c, \rho, e) \) is well-typed

Therefore from Theorem \( 94 \) \( (t, c, \rho, e) \) is well-typed

From Definition \( 92 \) \( \theta, c.\theta \) is well-typed

From Definition \( 91 \) we have \((\lambda x, x_1 \ldots x_n.t) (C) (C_1) \ldots (C_n)\) is well-typed

Therefore from Theorem \( 94 \) we know that

\[
\{E_s\} = \\
((\lambda x, x_1 \ldots x_n.t) (C) (C_1) \ldots (C_n)) = \\
\lambda \rho. \text{release} \quad \text{ releases } = p \text{ in } \text{bind } a = \text{store}() \text{ in } \text{bind } b = e_{t_1} \text{ a in } \text{bind } c = \text{store}() \text{ in } \text{bind } d = \text{store}() \text{ in } b \text{ (coerce } e_{t_2} \text{ c) d where}
\]

\[
e_{t_1} = ((\lambda x, x_1 \ldots x_n.t) (C, (C_1) \ldots (C_{n-1})))
\]

\[
e_{t_2} = \{C\} \quad (S-A1)
\]

From (SA-0.1) we know that
3. Fix:

Also since we know that $D_s$ is well-typed therefore from Lemma 106 we also know that

and then from Lemma 107 we know that $E_s \vdash v_b \parallel^j v_2$ s.t $\forall s.v_1 \not\preceq_{aV} v_2$

$|D_s| > |E_s|$ holds directly from the Definition of $|-|$
4. Var:  
Given: \( D_s = (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \theta) \) and \( E_s = (t_x, \rho_x, \theta) \)  
Let \( D'_s = (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \epsilon) \) and \( E'_s = (t_x, \rho_x, \epsilon) \)  
Since we are given that \( D_s \) and \( E_s \) are well-typed therefore from Lemma 105 we know that \( D'_s \) and \( E'_s \) are well-typed too.

Also since we know that \( e_t() \downarrow \downarrow v_1 \) therefore from Lemma 106 we also know that  
\[ \exists j. \downarrow^j \tau v_1 \]  
From Lemma 111 we know that \( \downarrow^j \varepsilon \varepsilon \)  
And then from Lemma 110 we know that  
\[ \| E_s \| \downarrow \downarrow^j \varepsilon \varepsilon v_2 \text{ s.t } \forall s, v_1 \overset{\delta}{\Rightarrow} s v_2 \]

**Lemma 105** (\( \epsilon \) typing).  
\[ \forall \Theta, \Delta, I, e, \rho, \theta, \Theta; \Delta \vdash_\epsilon (e, \rho, \theta) : - \implies \Theta; \Delta \vdash \epsilon (e, \rho, \epsilon) : - \]

**Proof.** Main derivation:  
\[ \Theta; \Delta \vdash J (e, \rho, \theta) : \tau \quad \text{Given} \]
\[ \Theta; \Delta \vdash J (e, \rho) : \sigma \quad \text{By inversion} \]
\[ \Theta; \Delta \vdash_0 \epsilon : (\sigma, \sigma) \]

**Lemma 106** (\( \epsilon \) reduction).  
\[ (e, \rho, \theta) \text{ is well typed} \land \| (e, \rho, \theta) \| (\) \downarrow \downarrow^j \varepsilon \varepsilon \implies \| (e, \rho, \epsilon) \| (\) \downarrow \downarrow^j \varepsilon \varepsilon \]

**Proof.** Since \( (e, \rho, \theta) \) is well typed therefore from Lemma 105 we also know that \( (e, \rho, \epsilon) \) is well typed  
From Theorem 94 we know that \( \| (e, \rho, \epsilon) \| \) is also well typed  
From Definition 92 we know that \( \| (e, \rho, \epsilon) \| = \| (e, \rho) \| \)  
Let \( \theta = C_1 \ldots C_n \)  
Similarly from Definition 92 we also know that  
\[ \| (e, \rho, \theta) \| = \| (e, \rho, C_1 \ldots C_n) \| = \]
\[ \| (e, \rho, \{ C_1 \}, \ldots, \{ C_n \}) \| = \lambda p. \text{release} = \varepsilon p \in \text{bind} a = \text{store()} \text{ in bind} b = \epsilon_{t_1} a \text{ in bind} c = \text{store()} \text{ in bind} d = \text{store()} \text{ in } b (\text{coerce1} \epsilon_{t_2} c) \text{ d} \]
where  
\[ \epsilon_{t_1} = \| (e, \rho, \{ C_1 \}, \ldots, \{ C_n \}) \| \]
\[ \epsilon_{t_2} = \{ C_n \} \quad (\text{EO}) \]
Since  
\[ \| (e, \rho, \{ C_1 \}, \ldots, \{ C_n \}) \| \downarrow \downarrow^j \varepsilon \varepsilon \]  
therefore we also know that  
\[ \| (e, \rho) \| \downarrow \downarrow^j \varepsilon \varepsilon \]

**Lemma 107** (Lemma for fix : non-empty stack).  
\[ \forall t, \rho, \theta, j, j', j'', v_1, v_2, v_0 \]
\( (\text{fix}_x.t, \rho, \epsilon) \) and \( (t, (\text{fix}_x.t), \rho, \epsilon) \) are well-typed \( (\text{fix}_x.t, \rho, \theta) \) and \( (t, (\text{fix}_x.t), \rho, \theta) \) are well-typed  
\[ \| (\text{fix}_x.t, \rho, \epsilon) \| (\) \downarrow \downarrow^j \varepsilon v_1 \land \| (t, (\text{fix}_x.t), \rho, \epsilon) \| (\) \downarrow \downarrow^j v_1 \varepsilon \varepsilon _{v_1} \overset{\delta}{\Rightarrow} v_2 \land \]
\[ \| (\text{fix}_x.t, \rho, \theta) \| (\) \downarrow \downarrow^j \varepsilon \varepsilon v_1 \varepsilon v_0 \land \]

\[ \exists v_2, j''. \| (t, (\text{fix}_x.t), \rho, \theta) \| (\) \downarrow \downarrow^j v_2 \varepsilon v_2 \varepsilon _{v_2} \overset{\delta}{\Rightarrow} v_0 \varepsilon v_0 \varepsilon (j - j') = (j'' - j''') \]

**Proof.** We prove this by induction on \( \theta \)
1. Case $\theta = c$:
   Directly from given

2. Case $\theta = C.\theta'$:
   Let $\theta' = C_1 \ldots C_n$ and $\theta'' = C'_1 \ldots C'_{n-1}$

Given:

$(\text{fix}x.t, \rho, C.\theta')$ and $(t, (\text{fix}x.t, \rho), C'.\theta')$ are well-typed ∧ 

$}\langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle \downarrow - v_{g_1}$

We need to prove that

$\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle \downarrow - v'' \quad \forall s. \exists v_{g_1} \theta_{g_1} \approx_{\alpha,v} v_{g_2} \land (j - j') = (j'' - j''')$ \quad \text{(ET-0)}$

From IH we know

$(\text{fix}x.t, \rho, C'.\theta'')$ and $(t, (\text{fix}x.t, \rho), C'.\theta'')$ are well-typed,

$\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle \downarrow - \psi j'' \quad \exists v_{g_{11}}$ \quad \text{(ET-0)}$

From Definition 9.1 and Definition 9.2 we know that

$\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle = \langle \langle \text{fix}x.t, \rho \rangle \rangle \ldots \langle \langle C_{n-1} \rangle \rangle \langle \langle C_n \rangle \rangle$ \quad \text{(ET-1)}$

Since $(\text{fix}x.t, \rho, C.\theta')$ is well-typed therefore we know that

$\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle \downarrow - \psi j'' \quad \forall s. \exists v_{g_{11}} \theta_{g_{11}}$ \quad \text{(ET-2)}$

From Definition 9.1 we know that

$\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle = \langle \langle \text{fix}x.t, \rho \rangle \rangle \ldots \langle \langle C_{n-1} \rangle \rangle \langle \langle C_n \rangle \rangle$ \quad \text{(ET-0)}$

Since $(t, (\text{fix}x.t, \rho), C.\theta')$ is well-typed therefore we know that

$\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle \downarrow - \psi j'' \quad \exists v_{g_{12}}$ \quad \text{(ET-2)}$

Therefore it suffices to prove that

$\forall s. \exists v_{g_{12}} \theta_{g_{12}}$ \quad \forall s. \exists v_{g_{11}} \theta_{g_{11}} \approx_{\alpha,v} v_{g_{12}}$ \quad \text{(ET-p)}$

Since we are given that $\langle \langle \text{fix}x.t, \rho, C'.\theta'\rangle \rangle$ this means from (ET-1.1) we have

$\lambda p. \text{release} - = p \text{ in bind } a = \text{store()} \text{ in bind } b = e_{c_1} \text{ a in bind } c = \text{store()} \text{ in bind } d = \text{store()} \text{ in } b' (\text{coerce}1 !e_{c_2} c) d$

where

$e_{c_1} = \langle \langle \text{fix}x.t, \rho \rangle \rangle \ldots \langle \langle C_{n-1} \rangle \rangle \langle \langle C_n \rangle \rangle$

$e_{c_2} = \langle \langle C_n \rangle \rangle$ \quad \text{(ET-1.1)}$

This means

1) $e_{c_1} \downarrow - \psi j'' \quad \forall s. \exists v_{g_{11}} \theta_{g_{11}}$

2) $e_{c_2} \downarrow - \psi y \quad \forall s. \exists v_{g_{12}} \theta_{g_{12}}$ for some $y$ s.t $y + j'' = j''$
Proof. Let \(\theta\). Lemma 109 (Lemma for fix : empty stack). \(\forall t, \rho, \theta\).
\[
\begin{align*}
\| (\text{fix}\, t, \rho, \theta) \| & \quad \text{is well-typed} \\
\| (t, (\text{fix}\, t, \rho), \theta) \| & \quad \text{is well-typed} \\
\| (\text{fix}\, t, \rho, \theta) \| & \quad \text{is well-typed} \\
\end{align*}
\]
Therefore from Theorem \([73]\) we know that
\[
\| (\text{fix}\, t, (C_1, \ldots, C_n), \theta) \| =
\]
\[
\begin{align*}
(\lambda x_1 \ldots x_n. \text{fix}\, t) \| (C_1) \ldots \| (C_n) &=
\end{align*}
\]
\]
\[
\begin{align*}
\text{release} \quad - \quad p \quad \text{in} \quad \text{bind} \quad a \quad = \quad \text{store}(a) \quad \text{in} \quad \text{bind} \quad b \quad = \quad e_1 \quad a \quad \text{in} \quad \text{bind} \quad c \quad = \quad \text{store}(b) \quad \text{in} \quad \text{bind} \quad d \quad = \quad \text{store}(c) \quad \text{in} \quad \text{bind} \quad e \quad = \quad (\text{coerce} \, e_1 \, !e_1 \, c) \quad d
\end{align*}
\]
where
\[
e_1 \quad (\lambda x_1 \ldots x_n. \text{fix}\, t) \| (C_1) \ldots \| (C_n)
\]
\[
e_2 \quad \| (C_n)
\]

Similarly since we know that \(\| (t, (\text{fix}\, t, (C_1, \ldots, C_n)), (C_1, \ldots, C_n), \theta) \| \) is well-typed and
\[
\| (t, (\text{fix}\, t, (C_1, \ldots, C_n)), (C_1, \ldots, C_n), \theta) \| =
\]
\[
\begin{align*}
(\lambda x_1 \ldots x_n. t) \| (\text{fix}\, t, (C_1, \ldots, C_n)) \| (C_1) \ldots \| (C_n) &=
\end{align*}
\]
\[
\begin{align*}
\text{release} \quad - \quad p \quad \text{in} \quad \text{bind} \quad a \quad = \quad \text{store}(a) \quad \text{in} \quad \text{bind} \quad b \quad = \quad e_1 \quad a \quad \text{in} \quad \text{bind} \quad c \quad = \quad \text{store}(b) \quad \text{in} \quad \text{bind} \quad d \quad = \quad \text{store}(c) \quad \text{in} \quad \text{bind} \quad e \quad = \quad (\text{coerce} \, e_1 \, !e_1 \, c) \quad d
\end{align*}
\]
where
\[
e_1 \quad (\lambda x_1 \ldots x_n. t) \| (\text{fix}\, t, (C_1, \ldots, C_n)) \| (C_1) \ldots \| (C_n)
\]
\[
e_2 \quad \| (C_n)
\]

We need to prove that
\[
(\lambda x_1 \ldots x_n. t) \| (\text{fix}\, t, \rho) \| (C_1) \ldots \| (C_n) \| - \| (C_n) \| - v_2
\]
This means it suffices to prove that
\[
(\lambda x_1 \ldots x_n. t) \| (\text{fix}\, t, (C_1, \ldots, C_n)) \| - \| (C_n) \| - v_2
\]
We get this directly from \((F2)\) and Lemma \([100]\)
Lemma 110 (Lemma for var : non-empty stack). \(\forall t, \rho, x, j, j', j'' \in \mathbb{N}, v, w, v_1, v_2, w_1, w_2, \varphi, \psi, \sigma \in \mathcal{P} \). 

\((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma) \) and \((t_x, \rho_x, \psi, \sigma)\) are well-typed 

\(((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)) \downarrow \Leftrightarrow \exists \psi' \psi' \in \mathbb{N} \land \forall s, \varphi_1 \preceq_A V \varphi_2 \land \forall v_1 (j, j', j'') \quad (ET-0)\)

From IH we know 

\(((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)) \downarrow \Leftrightarrow \exists \psi' \psi' \in \mathbb{N} \land \forall s, \varphi_1 \preceq_A V \varphi_2 \land \forall v_1 (j, j', j'') \quad (ET-1)\)

Since \((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)\) is well-typed therefore we know that 

\(((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)) \downarrow \Leftrightarrow \exists \psi' \psi' \in \mathbb{N} \land \forall s, \varphi_1 \preceq_A V \varphi_2 \land \forall v_1 (j, j', j'') \quad (ET-2)\)

From Definition [91] and Definition [92] we know that 

\(((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)) = (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n)) \quad (ET-0)\)

Also since we know that 

\(((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)) = (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n)) \quad (ET-1)\)

Therefore from (ET-2) we have 

\(((x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \sigma)) \downarrow \Leftrightarrow \exists \psi' \psi' \in \mathbb{N} \land \forall s, \varphi_1 \preceq_A V \varphi_2 \land \forall v_1 (j, j', j'') \quad (ET-2)\)
\( v_{922} (\coerce_1 !\epsilon_{12} c) d \downarrow - \psi' - j'' \psi v_{92} \) and \( \forall s.v_{91} \overset{s.a}{\approx} v_{92} \) \hspace{1cm} (ET-p)

Since we are given that \( \llbracket (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \theta') \rrbracket \) \( \downarrow - \psi' \psi v_{91} \) this means from (ET-1.1) we have

\[ \lambda p. \text{release} \neg = p \text{ in } \text{bind } a = \text{store()} \text{ in } \text{bind } b = e_{11} \text{ a in } \text{bind } c = \text{store()} \text{ in } \text{bind } d = \text{store()} \text{ in } b (\coerce_1 !\epsilon_{12} c) d \downarrow - \psi' \psi v_{91} \]

This means

1) \( \llbracket (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \theta') \rrbracket \) \( \downarrow - \psi' \psi v_{91} \) and
2) This means \( v_{911} (\coerce_1 !\epsilon_{12} c) d \downarrow - \psi' \psi v_{91} \) for some \( y \) s.t. \( y + j'' = j'' \)

Since from (ET-2) we have \( \forall s.v_{91} \overset{s.a}{\approx} v_{922} \) \( \land \) and since \( e_{12} = e_{12}' = [C_n] \) therefore from Definition 95 and Lemma 101 we have

\( v_{922} (\coerce_1 !\epsilon_{12} c) d \downarrow - \psi'' \psi v_{92} \) and \( \forall s.v_{91} \overset{s.a}{\approx} v_{92} \)

This means

\[ j'' - j_1' = j'' - j_1'' = j'' - j'' = j'' - j'' = j'' - j'' \ (\text{From IH}) \]

**Lemma 111** (Lemma for var : empty stack). \( \forall t, \rho, \theta, \)

\[ \Theta; \Delta. \vdash_\text{=} \llbracket (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \epsilon) \rrbracket : = \land \]

\[ \Theta; \Delta. \vdash_\text{=} \llbracket (t_x, \rho_x, \epsilon) \rrbracket : = \land \]

\[ \llbracket (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \epsilon) \rrbracket \] \( \downarrow - \psi t \psi v \iff \llbracket (t_x, \rho_x, \epsilon) \rrbracket \) \( \downarrow - \psi' \psi v \)

**Proof.** From Definition 92 we also have

\[ \llbracket (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \epsilon) \rrbracket = \llbracket (x, (t_0, \rho_0) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n)) \rrbracket = \llbracket (\lambda x_1 \ldots x_n.x) \llbracket (t_0, \rho_0) \rrbracket \ldots \llbracket (t_n, \rho_n) \rrbracket \]

Similarly from Definition 92 we also have

\[ \llbracket (t_x, \rho_x, \epsilon) \rrbracket = \llbracket (t_x, \rho_x) \rrbracket \] \hspace{1cm} (S-V1)

Therefore from Theorem 74 we know that

\[ \llbracket (x, (t_1, \rho_1) \ldots (t_x, \rho_x) \ldots (t_n, \rho_n), \epsilon) \rrbracket = \llbracket (\lambda x_1 \ldots x_n.x) \llbracket (t_1, \rho_1) \rrbracket \ldots \llbracket (t_x, \rho_x) \rrbracket \ldots \llbracket (t_n, \rho_n) \rrbracket \]

\[ \lambda p. \text{release} \neg = p \text{ in } \text{bind } a = \text{store()} \text{ in } \text{bind } b = e_{11} a \text{ in } \text{bind } c = \text{store()} \text{ in } \text{bind } d = \text{store()} \text{ in } b (\coerce_1 !\epsilon_{12} c) d \text{ where } \]

\[ e_{11} = \llbracket (\lambda x_1 \ldots x_n.x) \llbracket (t_1, \rho_1) \rrbracket \ldots \llbracket (t_x, \rho_x) \rrbracket \ldots \llbracket (t_{n-1}, \rho_{n-1}) \rrbracket \]

\[ e_{12} = \llbracket (t_n, \rho_n) \rrbracket \] \hspace{1cm} (V4)

Similarly

\[ e_{11} = \lambda p. \text{release} \neg = p \text{ in } \text{bind } a = \text{store()} \text{ in } \text{bind } b = e_{11} a \text{ in } \text{bind } c = \text{store()} \text{ in } \text{bind } d = \text{store()} \text{ in } b (\coerce_1 !\epsilon_{12} c) d \text{ where } \]

\[ e_{11} = \llbracket (\lambda x_1 \ldots x_n.x) \llbracket (t_1, \rho_1) \rrbracket \ldots \llbracket (t_x, \rho_x) \rrbracket \ldots \llbracket (t_{n-2}, \rho_{n-2}) \rrbracket \]

\[ e_{12} = \llbracket (t_{n-1}, \rho_{n-1}) \rrbracket \]

In the same way we have

\[ e_{11} = \lambda p. \text{release} \neg = p \text{ in } \text{bind } a = \text{store()} \text{ in } \text{bind } b = e_{11} a \text{ in } \text{bind } c = \text{store()} \text{ in } \text{bind } d = \text{store()} \text{ in } b (\coerce_1 !\epsilon_{12} c) d \text{ where } \]

\[ e_{11} = \llbracket (\lambda x_1 \ldots x_n.x) \rrbracket \]

\[ e_{12} = \llbracket (t_1, \rho_1) \rrbracket \]

Similarly we also get

\[ e_{11} = \lambda p_1, \text{ret} \lambda y, \lambda p_2, \text{let} x = y \text{ in } \text{release} \neg = p_1 \text{ in } \text{release} \neg = p_2 \text{ in } \text{bind } a = \text{store()} \text{ in } e_{11} a \text{ where } \]

\[ e_{11} = \llbracket (\lambda x_2 \ldots x_n.x) \rrbracket \]
and 
\[ e_{t,n} = \lambda p_1. \text{ret} \, \lambda y. \lambda p_2. \text{let} \, ! x = y \, \text{in} \, \text{release} = p_1 \, \text{in} \, \text{release} = p_2 \, \text{in} \, \text{bind} \, a = \text{store}(\cdot) \, e_{t,n} \, a \]

where 
\[ e_{t,n} = \exists = \lambda p. \text{release} = p \, \text{in} \, \text{bind} = \uparrow^1 \, \text{in} \, x \]

Since we know that 
\[ (\lambda x_1 \ldots x_n. x) \, ((t_0, \rho_0)) \ldots ((t_n, \rho_n)) \, (\lambda y. \lambda p. \text{release}) \]

this means from E-release, E-bind, E-store, E-app that

\[ (\text{bind} = \uparrow^1 \, \text{in} \, ((t_0, \rho_0)) \, (\lambda y. \lambda p. \text{release}) \]

Therefore from E-bind, E-step and E-app we know that 
\[ ((t_0, \rho_0)) \, (\lambda y. \lambda p. \text{release}) \]

\[ \square \]

**Theorem 112** (Rederiving dlPCF’s soundness), \( \forall t, I, \tau, \rho. \)

\[ \vdash (t, \epsilon, \epsilon) : \tau \land (t, \epsilon, \epsilon) \rightarrow (v, \rho, \epsilon) \implies n \leq |t| \ast (I + 1) \]

**Proof.** Let us rename \( t \) to \( t_1 \) and \( v \) to \( t_{n+1} \) then we know that

\[ (t_1, \epsilon, \epsilon) \rightarrow (t_2, \rho_2, \theta_2) \ldots (t_n, \rho_n, \theta_n) \rightarrow (t_{n+1}, \rho, \epsilon) \]

Since we are given that \((t, \epsilon, \epsilon)\) is well-typed therefore from dlPCF’s subject reduction we know that \((t_2, \rho_2, \theta_2)\) to \((t_n, \rho_n, \theta_n)\) and \((t_{n+1}, \rho, \epsilon)\) are all well-typed.

From Theorem 115 we know that \( \forall 1 \leq i \leq n. (t_i, \rho_i, \theta_i) \rightarrow \)

Also from Theorem 114 we know that \( \forall 1 \leq i \leq n. (t_i, \rho_i, \theta_i) \) is well typed

So now we can apply Theorem 104 and from Definition 96 to get

\[ |\vdash (t_1, \epsilon, \epsilon) \land (t_1, \epsilon, \epsilon) \rightarrow (v, \rho, \epsilon) \implies \exists v_d. (v, \rho, \epsilon) \sim_v v_d \]

Next we apply Theorem 110 for every step of the reduction starting from \((t_1, \epsilon, \epsilon)\) and we know that either the cost reduces by 1 and the size increases by \(|t|\) or cost remains the same and the size reduces.

Thus we know that size can vary from \( t \) to 1 and cost can vary from \( j_1 \) to 0. Therefore, the number of reduction steps are bounded by \(|t| \ast (j_1 + 1)\)

From Theorem 111 we know that \( j_1 < I \) therefore we have \( n \leq |t| \ast (I + 1)\)

\[ \square \]

**B.5.4 Cross-language model: Krivine to dlPCF**

**Definition 113** (Cross language logical realtion: Krivine to dlPCF),

\[ (v_k, \rho, \epsilon) \sim_v v_d \triangleq v_d = v_k \rho \]

\[ (e_k, \rho, \theta) \sim_e e_d \triangleq \exists v_d. (v_k, \rho, \theta) \rightarrow (v_k, \rho', \epsilon) \implies \exists v_d. (v_d, \rho') \rightarrow (v_k, \rho', \epsilon) \sim_v v_d \]

**Lemma 114.** \( \forall e_k, \rho, \theta, e_k', \rho', \theta'. \)

\[ (e_k, \rho, \theta) \rightarrow (e_k', \rho', \theta') \implies \exists e_d. (e_k, \rho, \theta) \rightarrow (e_k', \rho', \theta') \]

**Proof.** Given: \( (e_k, \rho, \theta) \rightarrow (e_k', \rho', \theta') \)

To prove: \( \exists e_d. (e_k, \rho, \theta) \rightarrow (e_k', \rho', \theta') \)

Let's assume it takes \( n \) steps for \( (e_k, \rho, \theta) \rightarrow (e_k', \rho', \theta') \)

We induct on \( n \)

**Base case \( n = 1 \)**

1. **App1:**

   In this case we are given \( (t, \rho, \theta) \rightarrow (t, \rho, (u, \rho), \theta) \)

   Let \( \rho = C_{\rho_1} \ldots C_{\rho_n} \) and \( \theta = C_{\theta_1} \ldots C_{\theta_m} \)

   From Definition 92 we know that

   \[ \langle (e_k, \rho, \theta) \rangle = \langle (\lambda x_1 \ldots x_n. t) \rangle (C_{\rho_1}) \ldots (C_{\rho_n}) \langle (C_{\theta_1}) \ldots (C_{\theta_m}) \rangle \]

   From dlPCF’s app we know that

   \[ (\lambda x_1 \ldots x_n. t) \, u \, C_{\rho_1} \ldots C_{\rho_n} \langle C_{\theta_1} \ldots C_{\theta_m} \rangle \rightarrow \]

   \[ \langle C_{\rho_1} \rangle (x_1) \ldots \langle C_{\rho_n} \rangle (x_n) \, u \, \langle (C_{\rho_1}) \rangle \ldots \langle (C_{\rho_n}) \rangle \langle (C_{\theta_1}) \ldots (C_{\theta_m}) \rangle \]

   We choose \( e_d \) as \( (\lambda x_1 \ldots x_n. t) \, u \, \langle (C_{\rho_1}) \rangle \ldots \langle (C_{\rho_n}) \rangle \, u \, \langle (C_{\theta_1}) \rangle \ldots \langle (C_{\theta_m}) \rangle \) and we get the desired from Definition 92

175
2. App2:

In this case we are given \((\lambda x.t, \rho, C, \theta) \rightarrow (t, C, \rho, \theta)\)

Let \(\rho = C_{\rho_1} \ldots C_{\rho_n}\) and \(\theta = C_{\theta_1} \ldots C_{\theta_m}\)

From Definition 92, we know that

\[
\models (\lambda x.t, \rho, C, \theta) = (\lambda x_1 \ldots x_n.\lambda x.t)(C_{\rho_1}) \ldots (C_{\rho_n}) \cdot (C) \cdot (C_{\theta_1}) \ldots (C_{\theta_m})
\]

From dlPCF’s app rule we know that

\[
(\lambda x_1 \ldots x_n.\lambda x.t)(C_{\rho_1}) \ldots (C_{\rho_n}) \cdot (C) \cdot (C_{\theta_1}) \ldots (C_{\theta_m}) \rightarrow_t [C_{\rho_1}]/x_1 \ldots [C_{\rho_n}]/x_n[[C]/x] \ C_{\theta_1} \ldots C_{\theta_m}
\]

We choose \(v_d'\) as \( t_1[[C_{\rho_1}]/x_1] \ldots [[C_{\rho_n}]/x_n][[C]/x] \ C_{\theta_1} \ldots C_{\theta_m}\) and we get the desired from Definition 92.

3. Var:

In this case we are given \((x, (t_0, \rho_0) \ldots (t_n, \rho_n), \theta) \rightarrow (t_x, \rho_x, \theta)\)

Let \(\rho_x = C_{\rho_1} \ldots C_{\rho_n}\) therefore from Definition 92, we know that

\[
\models (x, (t_0, \rho_0) \ldots (t_n, \rho_n)) \ C_{\rho_1} \ldots C_{\rho_n} = (\lambda x_1 \ldots x_n.x.t)(C_{\rho_1}) \ldots (C_{\rho_n} \cdot (C_{\theta_1}) \ldots (C_{\theta_m})
\]

From dlPCF’s app rule we know that

\[
(\lambda x_1 \ldots x_n.x.t)(C_{\rho_1}) \ldots (C_{\rho_n}) \cdot (C_{\theta_1}) \ldots (C_{\theta_m}) \rightarrow (\lambda x_1 \ldots x_n.x.t)(C_{\rho_1}) \ldots (C_{\rho_n} \cdot (C_{\theta_1}) \ldots (C_{\theta_m})
\]

Let \(\rho_x = C_{\rho_1} \ldots C_{\rho_n}\) therefore from Definition 92, we know that

\[
\models (t_0, \rho_0) \ldots (t_n, \rho_n), \theta) \rightarrow (t_x, \rho_x, \theta)
\]

4. Fix:

In this case we are given \((\text{fix}x.t, \rho, \theta) \rightarrow (t, (\text{fix}x.t), \rho, \theta)\)

Let \(\rho = C_{\rho_1} \ldots C_{\rho_n}\) and \(\theta = C_{\theta_1} \ldots C_{\theta_m}\)

From Definition 92, we know that

\[
\models (\text{fix}x.t, \rho, \theta) = (\lambda x_1 \ldots x_n.\text{fix}x.t)(C_{\rho_1}) \ldots (C_{\rho_n}) \cdot ((\text{fix}x.t, \rho)) \cdot (C_{\theta_1}) \ldots (C_{\theta_m})
\]

From dlPCF’s app rule and fix rule we know that

\[
(\lambda x_1 \ldots x_n.\text{fix}x.t)(C_{\rho_1}) \ldots (C_{\rho_n}) \cdot (C_{\theta_1}) \ldots (C_{\theta_m}) \rightarrow \text{fix}x.t[[C_{\rho_1}]/x_1] \ldots [[C_{\rho_n}]/x_n][[\text{fix}x.t, \rho)]/x \ C_{\theta_1} \ldots C_{\theta_m} \rightarrow t [[C_{\rho_1}]/x_1] \ldots [[C_{\rho_n}]/x_n][[\text{fix}x.t, \rho)]/x \ C_{\theta_1} \ldots C_{\theta_m}
\]

We choose \(e'_d\) as \( t_1[[C_{\rho_1}]/x_1] \ldots [[C_{\rho_n}]/x_n][[\text{fix}x.t, \rho)]/x \ C_{\theta_1} \ldots C_{\theta_m}\) and we get the desired from Definition 92.

**Inductive case**

We get this directly from IH and the base case.

---

**Theorem 115** (Fundamental theorem). \(\forall e_k, \rho, \theta. (e_k, \rho, \theta) \sim_e ((e_k, \rho, \theta))\)

*Proof.* From Definition 113, it suffices to prove that

\[
\forall k, k', (e_k, \rho, \theta) \rightarrow (v_k, k', \epsilon) \implies \exists \forall d. v_d \rightarrow v_d \land (v_k, k', \epsilon) \sim_{v} v_d
\]

This means that if given some \(v_k, k', \text{s.t.} (e_k, \rho, \theta) \rightarrow (v_k, k', \epsilon)\) it suffices to prove that

\[
\exists \forall d. v_d \rightarrow v_d \land (v_k, k', \epsilon) \sim_{v} v_d
\]

From Lemma 114, we know that

\[
\exists \forall d. ((e_k, \rho, \theta) \rightarrow e'_d \land e'_d = ((v_k, k', \epsilon))
\]

Let \(k' = C_1 \ldots C_n\) therefore from Definition 92, we know that

\[
((v_k, k', \epsilon)) = (\lambda x_1 \ldots x_n.v_k)(C_1) \ldots (C_n)
\]

Therefore from dlPCF’s app rule we know that

\[
((v_k, k', \epsilon)) \rightarrow v_k[[C_1]/x_1] \ldots [[C_n]/x_n]
\]

We choose \(v_d\) as \( v_k[[C_1]/x_1] \ldots [[C_n]/x_n]\) and we get the desired from Definition 113. 

\(\square\)
C Examples

C.1 Church numerals

Nat = λn.∀α : N → Type.∀C : N → N.
!(∀α.((α jn ⊗ [C jn] 1) → ⊤0)(α (jn + 1)))) → ⊤0((α 0 ⊗ [(C 0 + ... + C (n - 1) + n)] 1) → ⊤0(α n))
e1 ↑ e2 = bind - = ↑ in e1 e2

Type derivation for ⊥

⊥ = Λ.Λ.λf. ret λx. let⟨⟨y1, y2⟩⟩ = x in ret y1 : Nat 0

T0 = ∀α.∀C.(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [0] 1) → ⊤0(α 0))
T0,1 = ∀C.!(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [0] 1) → ⊤0(α 0))
T0,2 = !(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [0] 1) → ⊤0(α 0))
T0,3 = !(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [0] 1) → ⊤0(α 0))
T1 = ⊤0((α 0 ⊗ [0] 1) → ⊤0(α 0))
T2 = (α 0 ⊗ [0] 1)
T2,1 = α 0
T2,2 = [0] 1
T3 = ⊤0(α 0)
T1,1 = (α 0 ⊗ [0] 1) → ⊤0(α 0)
T2 = (α 0 ⊗ [0] 1)

Main derivation:

D0:  

T1;...;f : T0,3, y1 : T2,1, y2 : T2,2 ⊢ ret y1 : M 0 T2,1

D1:

D0     D1

T1;...;f : T0,3, x : T2 ⊢ x : T2

Type derivation for ⊤

⊤ = Λ.Λ.λf. ret λx. let!u = f in let⟨⟨y1, y2⟩⟩ = x in release - = y2 in E1 : Nat 1

where

E1 = bind a = store() in f_u [] ↑↑⟨⟨y1, a⟩⟩

T0 = ∀α : N → Type.∀C : N → Sort.

!(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [C 0 + 1] 1) → ⊤0(α 1))
T0,1 = ∀C.!(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [C 0 + 1] 1) → ⊤0(α 1))
T0,2 = !(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [C 0 + 1] 1) → ⊤0(α 1))
T0,3 = !(∀α.((α jn ⊗ [C jn] 1) → ⊤0(α (jn + 1)))) → ⊤0((α 0 ⊗ [C 0 + 1] 1) → ⊤0(α 1))
T1 = ⊤0((α 0 ⊗ [C 0 + 1] 1) → ⊤0(α 1))
T2 = (α 0 ⊗ [C 0 + 1] 1)
\( T_{2.1} = \alpha 0 \)
\( T_{2.2} = \lbrack C 0 + 1 \rbrack 1 \)
\( T_{3} = \mathbb{M} 0 (\alpha 1) \)
\( TI = \alpha : \text{N} \rightarrow \text{Type}; C : \text{N} \rightarrow \text{Sort} \)

\[
D7: \quad TI; :: f_u : T_{0.4}; y_1 : T_{2.1}, a : [C 0] 1 \vdash \llangle y_1, a \rrangle : (T_{2.1} \otimes [C 0] 1)
\]

\[
D6: \quad TI; :: f_u : T_{0.4}; \vdash f_u [] : T_{0.5}
\]

\[
D5: \quad D6 \quad D7
\]

\[
TI; :: f_u : T_{0.4}; y_1 : T_{2.2}, a : [C 0] 1 \vdash f_u [] \uparrow \llangle y_1, a \rrangle : \mathbb{M} 1 \alpha 1
\]

\[
D4
\]

\[
TI; :: f_u : T_{0.4}; y_1 : T_{2.1}, y_2 : T_{2.2} \vdash \text{bind } a = \text{store}() \text{ in } f_u [] \uparrow \llangle y_1, a \rrangle : \mathbb{M}(C 0 + 1) \alpha 1
\]

\[
D3
\]

\[
TI; :: f_u : T_{0.4}; y_1 : T_{2.1}, y_2 : T_{2.2} \vdash \text{release } x = y_2 \text{ in } E_1 : T_3
\]

\[
D2
\]

\[
TI; :: f_u : T_{0.4}; x : T_2 \vdash x : T_2
\]

\[
TI; :: f_u : T_{0.4}; x : T_2 \vdash \text{let } \llangle y_1, y_2 \rrangle = x \text{ in } E_1 : T_3
\]

\[
D0
\]

Main derivation:

\[
D0 \quad D1
\]

\[
TI; :: f : T_{0.3}, x : T_2 \vdash \text{let } f_u = f \text{ in } \llangle y_1, y_2 \rrangle = x \text{ in } \text{release } y_2 = y_2 \text{ in } E_1 : T_3
\]

\[
TI; :: f : T_{0.3} \vdash \text{let } f_u = f \text{ in } \llangle y_1, y_2 \rrangle = x \text{ in } \text{release } y_2 = y_2 \text{ in } E_1 : T_{1.1}
\]

\[
TI; :: f : T_{0.3} \vdash \text{let } f_u = f \text{ in } \llangle y_1, y_2 \rrangle = x \text{ in } \text{release } y_2 = y_2 \text{ in } E_1 : T_{0.2}
\]

\[
; \alpha : \text{N} \rightarrow \text{Type}; \vdash \lambda f. \text{ret } \alpha x. \text{let } f_u = f \text{ in } \llangle y_1, y_2 \rrangle = x \text{ in } \text{release } y_2 = y_2 \text{ in } E_1 : T_{0.1}
\]

Type derivation for \( \overline{2} \)

\[ \overline{2} = \lambda \lambda. \lambda f. \text{ret } \alpha x. \text{let } f_u = f \text{ in } \llangle y_1, y_2 \rrangle = x \text{ in } \text{release } y_2 = y_2 \text{ in } E_1 : E_2 : \text{Nat} 2 \]

where

\( E_1 = \text{bind } a = \text{store}() \text{ in } f_u [] \uparrow \llangle y_1, a \rrangle \)
\( E_2 = \text{bind } c = \text{store}() \text{ in } f_u [] \uparrow \llangle b, c \rrangle \)

\[
T_0 = \\
\forall \alpha : \text{N} \rightarrow \text{Type}. \forall C. (\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \rightarrow ((\alpha 0 \otimes [C 0 + C 1 + 2] 1) \rightarrow M 0 (\alpha 2))
\]

\[
T_{0.1} = \forall C. (\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \rightarrow ((\alpha 0 \otimes [C 0 + C 1 + 2] 1) \rightarrow M 0 (\alpha 2))
\]

\[
T_{0.2} = (((\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \rightarrow ((\alpha 0 \otimes [C 0 + C 1 + 2] 1) \rightarrow M 0 (\alpha 2)))
\]

\[
T_{0.3} = (((\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \rightarrow ((\alpha 0 \otimes [C 0 + C 1 + 2] 1) \rightarrow M 0 (\alpha 2)))
\]

\[
T_{0.4} = ((\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1)))
\]

\[
T_{0.5} = ((\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha 1))
\]

\[
T_{0.6} = (\alpha 1 \otimes [C 1] 1) \rightarrow M 0 (\alpha 2)
\]

\[
T_1 = M 0 (\alpha 0 \otimes [C 0 + C 1 + 2] 1) \rightarrow M 0 (\alpha 2)
\]

\[
T_{1.1} = ((\alpha 0 \otimes [C 0 + C 1 + 2] 1) \rightarrow M 0 (\alpha 2))
\]
\[ T_2 = (\alpha \otimes [C \ 0 + C \ 1 + 2]) \ 1 \]
\[ T_{2,1} = \alpha \ 0 \]
\[ T_{2,2} = [C \ 0 + C \ 1 + 2] \ 1 \]
\[ T_3 = M(1) \ (\alpha \ 2) \]
\[ T_{3,1} = M(C \ 0 + C \ 1 + 2) \ (\alpha \ 2) \]
\[ TI = \alpha : N \rightarrow Type; C : N \rightarrow Sort \]

\[
D5.22
\[
TI ; : f_u : T_{0,4} ; b : \alpha \ 1, c : [(C \ 1)] \ 1 \vdash \langle b, c \rangle : (\alpha \ 1 \otimes [(C \ 1)] \ 1)
\]

\[
D5.21
\[
TI ; : f_u : T_{0,4} ; : f_u [\] : T_{0,6}
\]

\[
D5.2
\]

\[
D5.21 \quad D5.22
\]

\[
TI ; : f_u : T_{0,4} ; b : \alpha \ 1, c : [(C \ 1)] \ 1 \vdash f_u [\langle b, c \rangle] : T_3
\]

\[
D5.1
\]

\[
D5.1 \quad D5.2
\]

\[
TI ; : f_u : T_{0,4} ; b : \alpha \ 1 \vdash \text{bind} \ c = \text{store}() \ in \ f_u [\langle b, c \rangle] : M(C \ 1 + 1) \ (\alpha \ 2)
\]

\[
D4.12
\]

\[
D4.11
\]

\[
D4.1
\]

\[
D4 \quad D5
\]

\[
TI ; : f_u : T_{0,4} ; y_1 : T_{2,1}, a : [(C \ 0)] \ 1 \vdash \langle y_1, a \rangle : (T_{2,1} \otimes [(C \ 0)] \ 1)
\]

\[
D3.1
\]

\[
D3 \quad D3.2
\]

\[
TI ; : f_u : T_{0,4} ; y_2 : T_{2,2} \vdash y_2 : T_{2,2}
\]

\[
D2
\]

\[
D2 \quad D3
\]

\[
TI ; : f_u : T_{0,4} ; x : T_2 \vdash x : T_2
\]

\[
D1
\]

\[
D1 \quad D2
\]

\[
TI ; ; f : T_{0,3} \vdash f : T_{0,3}
\]
D0.0:

\[
\begin{align*}
T_1; \ldots; f & : T_{0.3}, x : T_2 \vdash \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in bind } b = E_1 \text{ in } E_2 : T_3 \\
T_1; \ldots; f & : T_{0.3} \vdash \Lambda x. \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in bind } b = E_1 \text{ in } E_2 : T_1.1 \\
T_1; \ldots; f & : T_{0.3} \vdash \text{ret } x. \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in bind } b = E_1 \text{ in } E_2 : T_1 \\
T_1; \ldots; & \vdash \lambda f. \text{ret } x. \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in bind } b = E_1 \text{ in } E_2 : T_{0.2}
\end{align*}
\]

Main derivation:

D0.0

\[
\begin{align*}
\vdash \alpha : \mathbb{N} \rightarrow \text{Type} : \vdash \Lambda C. \lambda f. \text{ret } x. \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in bind } b = E_1 \text{ in } E_2 : T_{0.1} \\
\vdash \Lambda \Lambda C. \lambda f. \text{ret } x. \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in bind } b = E_1 \text{ in } E_2 : T_0
\end{align*}
\]

Type derivation for \(\text{succ} : \forall n. [2] \rightarrow M 0 (\text{Nat } n \rightarrow M 0 (\text{Nat } (n + 1)))\)

\[
\begin{align*}
\text{succ} & = \Lambda \lambda p. \text{ret } \Lambda \lambda \alpha. \text{ret } \lambda \alpha. \text{let } f_u = f \text{ in } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 \\
\text{where} & \\
E_0 & = \text{release } = p \text{ in bind } a = E_1 \text{ in } E_2 \\
E_1 & = \text{bind } b = \text{store } () \text{ in bind } b_1 = (\mathbb{N} \leftrightarrow \top^1 \langle a, c \rangle) \text{ in } b_1 \uparrow^1 \langle y_1, b \rangle \\
E_2 & = \text{bind } c = \text{store } () \text{ in ret } f_u \uparrow^1 \langle a, c \rangle \\
T_p & = [2] 1 \\
T_0 & = \forall n. T_p \rightarrow M 0 (\text{Nat } n \rightarrow M 0 (\text{Nat } (n + 1))) \\
T_{0.0} & = T_p \rightarrow M 0 (\text{Nat } n \rightarrow M 0 (\text{Nat } (n + 1))) \\
T_{0.01} & = M 0 (\text{Nat } n \rightarrow M 0 (\text{Nat } (n + 1))) \\
T_{0.1} & = \forall n. M 0 (\text{Nat } n \rightarrow M 0 (\text{Nat } (n + 1))) \\
T_{0.2} & = M 0 (\text{Nat } (n + 1)) \\
T_{0.11} & = \text{Nat } n \\
T_{0.12} & = \\
& \forall \alpha : \mathbb{N} \rightarrow \text{Type} : \forall C. ([\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\text{Nat } j_n + 1))] \\
M 0 (\alpha (j_n + 1))) & \rightarrow M 0 ((\alpha 0 \otimes [C 0 + \ldots + C (n - 1) + n] 1) \rightarrow M 0 (\alpha n)) \\
T_{0.13} & = \forall C. ([\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\text{Nat } j_n + 1))] \\
M 0 ((\alpha 0 \otimes [C 0 + \ldots + C (n - 1) + n] 1) \rightarrow M 0 (\alpha n)) & \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{0.14} & = \forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
M 0 (\alpha 0 \otimes [C 0 + \ldots + C (n - 1) + n] 1) & \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{0.15} & = M 0 (\alpha 0 \otimes [C 0 + \ldots + C (n - 1) + n] 1) \rightarrow M 0 (\alpha n)) \\
T_{0.16} & = M 1 ((\alpha 0 \otimes [C 0 + \ldots + C (n - 1) + n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{0.17} & = (\alpha 0 \otimes [C 0 + \ldots + C (n - 1) + n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{0.18} & = \text{Nat } + 1 \\
T_1 & = \\
& \forall \alpha : \mathbb{N} \rightarrow \text{Type} : \forall C. ([\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\text{Nat } j_n + 1))] \\
M 0 (\alpha 0 \otimes [C 0 + \ldots + C (n) + (n + 1)] 1) & \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.1} & = \forall C. ([\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\text{Nat } j_n + 1))] \\
M 0 ((\alpha 0 \otimes [C 0 + \ldots + C (n) + (n + 1)] 1) & \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.2} & = (\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
M 0 (\alpha 0 \otimes [C 0 + \ldots + C (n) + (n + 1)] 1) & \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.3} & = (\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.31} & = (\forall j_n. (\alpha j_n \otimes [C j_n] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.40} & = M 0 (\alpha 0 \otimes [C 0 + \ldots + C (n) + (n + 1)] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.4} & = (\forall j_n. (\alpha 0 \otimes [C 0 + \ldots + C (n) + (n + 1)] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.41} & = (\forall j_n. (\alpha 0 \otimes [C 0 + \ldots + C (n) + (n + 1)] 1) \rightarrow M 0 (\alpha (j_n + 1))) \\
T_{1.411} & = \alpha 0 \\
T_{1.412} & = ([C 0 + \ldots + C (n) + (n + 1)] 1 \\
T_{1.42} & = M 0 (\alpha (n + 1)) \\
T_{1.43} & = M 0 (\alpha (n + 1)) \\
T_{1.431} & = M 0 (\alpha (n + 1)) \\
T_{1.44} & = M 0 (\alpha (n + 1)) \\
T_{1.45} & = M 0 (\alpha (n + 1)) \\
\text{TI} & = \alpha ; n, C
\end{align*}
\]
D3.1: \[ D1; :: f_u : T_{1.31}; \alpha : \alpha n, c : [(C \ n) \ 1] 1 \vdash f_u \uparrow \rangle \langle \alpha, c \rangle : \mathbb{M}1\alpha(n+1) \]

D3: \[ D1; :: f_u : T_{1.31}; \vdash \text{store() : } \mathbb{M}(C \ n) [(C \ n) \ 1] \quad D3.1 \]

D2.3: \[ D1; :: f_u : T_{1.31}; y_1 : T_{1.44}, b : [n \ast C] \ 1, b_1 : T_{0.16} \vdash b_1 : T_{0.16} \]

D2: \[ D1; :: f_u : T_{1.31}; \vdash \text{store() : } \mathbb{M}(C \ n) [(C \ n) \ 1] \quad D2.1 \]

D2.1: \[ D1; :: f_u : T_{1.31}; \overline{N} : T_{0.11} \vdash \overline{N} \uparrow \downarrow f_u : T_{0.151} \]

D2: \[ D1; :: f_u : T_{1.31}; \vdash \text{store() : } \mathbb{M}(C \ n) [(C \ n) \ 1] \quad D2.1 \]

D1: \[ D1; :: f_u : T_{1.31}; x : T_{1.44} \vdash x : T_{1.44} \]

D1.0: \[ D0 \quad D1 \]

D1.0: \[ D0; :: \overline{N} : T_{0.11}, p : T_{p}, f : T_{1.31}, x : T_{1.41} \vdash \text{let } f_u = f \text{ in let } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 : T_{1.42} \]

D1.1: \[ D1; :: f_u : T_{1.31}; x : T_{1.41} \vdash x : T_{1.41} \]

D1.2: \[ D1; :: f_u : T_{1.31}; y_2 : T_{1.412} \vdash y_2 : T_{1.412} \]

D1.3: \[ D1; :: f_u : T_{1.31}; y_1 : T_{1.41}, y_2 : T_{1.412} \vdash \text{release } = y_2 \text{ in } E_0 : T_{1.42} \]

D1.4: \[ D1; :: f_u : T_{1.31}; p : T_{p} \vdash p : T_{p} \]

D1.5: \[ D1; :: f_u : T_{1.31}; \text{bind } a = E_1 \text{ in } E_2 : T_{1.43} \]

D2: \[ D2; :: f_u : T_{1.31}; \text{bind } b = \text{store() in bind } b_1 = (\overline{N} \uparrow \downarrow f_u) \text{ in } b_1 \uparrow \rangle \langle y_1, b \rangle : T_{1.44} \]

D2.2: \[ D2.2; :: f_u : T_{1.31}; \overline{N} : T_{0.11}, y_1 : T_{1.44}, b : [(C \ 0 + \ldots + C(n-1) + (n))] \ 1, b_1 : T_{0.16} \vdash b_1 \uparrow \rangle \langle y_1, b \rangle : T_{1.44} \]

D2.3: \[ D2.3; :: f_u : T_{1.31}; y_1 : T_{1.44}, b : [(C \ 0 + \ldots + C(n-1) + (n))] \ 1, b_1 : T_{0.16} \vdash b_1 \uparrow \rangle \langle y_1, b \rangle : T_{1.44} \]

D0: \[ D0; :: \overline{N} : T_{0.11}, p : T_{p}, f : T_{1.31} \vdash \text{let } f_u = f \text{ in let } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 : T_{1.42} \]
Main derivation:

\[
\begin{array}{ll}
\vdots: n_1; n_2; p : T_p \vdash \lambda N_1. \text{ret } \lambda A. \lambda f. \text{ret } A x. \text{let } ! f_u = f \text{ in let } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 : T_{0.1} \\
\vdots: n_1; n_2; p : T_p \vdash \lambda N_1. \text{ret } \lambda A. \lambda f. \text{ret } A x. \text{let } ! f_u = f \text{ in let } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 : T_{0.01} \\
\vdots: n_1; n_2; p : T_p \vdash \lambda N_1. \text{ret } \lambda A. \lambda f. \text{ret } A x. \text{let } ! f_u = f \text{ in let } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 : T_{0.0} \\
\vdots: n_1; n_2; p : T_p \vdash \lambda N_1. \text{ret } \lambda A. \lambda f. \text{ret } A x. \text{let } ! f_u = f \text{ in let } \langle y_1, y_2 \rangle = x \text{ in release } = y_2 \text{ in } E_0 : T_0
\end{array}
\]

Type derivation for \( add : \forall n_1, n_2. ((n_1 * 3 + n_1 + 2) \mapsto \mu M (0 (\text{Nat } n_1 \mapsto 0 (\text{Nat } n_2 \mapsto 0 (\text{Nat } (n_1 + n_2)))))) \)

\[
\begin{array}{ll}
add = \lambda A. \lambda p. \text{ret } \lambda N_1, \text{ret } \lambda N_2. E_0 \\
\end{array}
\]

where

\[
\begin{array}{ll}
E_0 = \text{release } = p \text{ in bind } a = E_1 \text{ in } E_2 \\
E_{0.1} = \text{release } = y_2 \text{ in bind } b_1 = (\text{bind } b_2 = \text{store } () \text{ in succ } [] b_2) \text{ in } b_1 \uparrow^1 y_1 \\
E_1 = (\lambda A. \text{let } \langle y_1, y_2 \rangle = t \text{ in } E_{0.1}) \\
E_2 = \text{bind } b = \text{store } () \text{ in } a \uparrow^1 (\langle N_2, b \rangle)
\end{array}
\]

\[
\begin{array}{ll}
T_p = ([n_1 * 3 + n_1 + 2]) \mapsto 1 \\
T_0 = \forall n_1, n_2, T_p \rightarrow M_0 (\text{Nat } n_1 \rightarrow M_0 (\text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2)))) \\
T_{0.1} = \forall n_1, T_p \rightarrow M_0 (\text{Nat } n_1 \rightarrow M_0 (\text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2)))) \\
T_{0.2} = T_p \rightarrow M_0 (\text{Nat } n_1 \rightarrow M_0 (\text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2)))) \\
T_{0.20} = M_0 (\text{Nat } n_1 \rightarrow M_0 (\text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2)))) \\
T_{0.21} = (\text{Nat } n_1 \rightarrow M_0 (\text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2)))) \\
T_{0.3} = M_0 (\text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2))) \\
T_{0.31} = \text{Nat } n_2 \rightarrow M_0 (\text{Nat } (n_1 + n_2)) \\
T_{0.4} = M_1 (\text{Nat } (n_1 + n_2)) \\
T_{0.40} = M_0 (\text{Nat } (n_1 + n_2)) \\
T_{0.5} = M_1 (\text{Nat } (n_1 + n_2)) \\
T_{0.6} = M_1 (\text{Nat } (n_1 + n_2)) \\
T_1 = \\
\forall \alpha : N \rightarrow \text{Type}. \forall C. \forall ! k. ((\alpha k \mapsto [C k] \mapsto 1) \rightarrow M_0 (\alpha (k + 1))) \rightarrow \\
M_0 ((\alpha 0 \mapsto [C 0 + \ldots + C (n_1 - 1) + n_1] \mapsto 1) \rightarrow M_0 (\alpha (n_1))) \\
\alpha f = \lambda k. \text{Nat } (n_2 + k) \\
T_{1.1} = \forall ! C. \forall ! k. ((\alpha f k \mapsto [C k] \mapsto 1) \rightarrow M_0 (\alpha f (k + 1))) \rightarrow \\
M_0 ((\alpha 0 \mapsto [C 0 + \ldots + C (n_1 - 1) + n_1] \mapsto 1) \rightarrow M_0 (\alpha f (n_1))) \\
T_{1.2} = \forall ! C. \forall ! k. ((\alpha f k \mapsto [C k] \mapsto 1) \rightarrow M_0 (\alpha f (k + 1))) \rightarrow \\
M_0 ((\alpha (n_2 + 0) \mapsto [C 0 + \ldots + C (n_1 - 1) + n_1] \mapsto 1) \rightarrow M_0 (\alpha (n_2 + n_1))) \\
T_{1.21} = ! (\forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1)))) \\
T_{1.22} = \forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.23} = \forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.24} = (\forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.241} = (\forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.2411} = (\forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.2412} = (\forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.242} = M_0 (\text{Nat } (n_2 + (k + 1))) \\
T_{1.3} = M_0 (\text{Nat } (n_2 + 0) \mapsto ([n_1 * 3 + n_1] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + n_1))) \\
T_{1.30} = M_0 (\text{Nat } (n_2 + 0) \mapsto ([n_1 * 3 + n_1] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + n_1))) \\
T_{1.31} = (\forall ! k. (\text{Nat } (n_2 + 0) \mapsto [n_1 * 3 + n_1] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + n_1))) \\
T_2 = \text{Nat } n_2 \\
T_3 = (\forall ! k. (\text{Nat } (n_2 + k) \mapsto [C k] \mapsto 1) \rightarrow M_0 (\text{Nat } (n_2 + k + 1))) \\
\end{array}
\]

D3:

\[
\vdots: n_1, n_2; \vdots; \vdash T_1 \mapsto N_1 : T_1 + N_1 : T_1
\]

D2.10:

\[
\begin{array}{ll}
D3 \vdash: n_1, n_2; \vdash (\lambda k. \text{Nat } [n_2 + k]) : N \rightarrow \text{Type} \\
\vdots: n_1, n_2; \vdots; \vdash N_1 : T_1 \mapsto N_1 [] : T_{1.1} \\
\vdots: n_1, n_2; \vdots; \vdash N_1 : T_1 \mapsto N_1 [] : T_{1.2}
\end{array}
\]

182
D2:

\[D2.10\]
\[;n_1, n_2; ;; \vdash (\lambda x. -3) : \mathbb{N} \rightarrow \mathbb{N}\]
\[;n_1, n_2; ;; \overline{N_1} : T_1 + \overline{N_1} \parallel \parallel : T_{1.21}\]

D1.32:

\[;n_1, n_2, k; ;; b_2 : [2] 1 \vdash \text{succ} [b_2] : \mathbb{M} 0 T_3\]

D1.31:

\[;n_1, n_2, k; ;; \vdash \text{store}() : \mathbb{M} 2 [2] 1\]
\[;n_1, n_2, k; ;; \vdash (\text{bind} b_2 = \text{store}() \text{ in } \text{succ} [b_2] ) : \mathbb{M} 2 T_3\]

D1.3:

\[D1.31\]
\[;n_1, n_2, k; ;; y_1 : T_{1.2411}, b_1 : T_3 \vdash b_1 \uparrow^1 y_1 : \mathbb{M} 1 \text{Nat}[n_2 + k + 1]\]
\[;n_1, n_2, k; ;; y_1 : T_{1.2411} \vdash \text{bind} b_1 = (\text{bind} b_2 = \text{store}() \text{ in } \text{succ} [b_2] ) \text{ in } b_1 \uparrow^1 y_1 : \mathbb{M} 0 \text{Nat}[n_2 + k + 1]\]

D1.2:

\[D1.2\]
\[;n_1, n_2, k; ;; y_2 : T_{1.2412} \vdash y_2 : T_{1.2412}\]
\[;n_1, n_2, k; ;; y_2 : T_{1.2411}, y_2 : T_{1.2412} \vdash \text{release} - = y_2 \text{ in } \text{bind} b_1 = (\text{bind} b_2 = \text{store}() \text{ in } \text{succ} [b_2] ) \text{ in } b_1 \uparrow^1 y_1 : \mathbb{M} 0 \text{Nat}[n_2 + k + 1]\]

D1.1:

\[D1.1\]
\[;n_1, n_2, k; ;; t : T_{1.241} \vdash t : T_{1.241}\]
\[;n_1, n_2, k; ;; y_1 : T_{1.2411}, y_2 : T_{1.2412} \vdash t \text{ in } E0.1 : T_{1.242}\]
\[;n_1, n_2, k; ;; \lambda. \text{let}((y_1, y_2)) = t \text{ in } E0.1 : T_{1.24}\]

D1:

\[D1\]
\[;n_1, n_2; ;; \overline{N_1} : T_1, N_2 : T_2 \vdash E_1 : T_{1.30}\]

D2.1:

\[;n_1, n_2; ;; N_2 : T_2, a : T_{1.31}, b : [(n_1 \ast 3 + n_1)] 1 \vdash a \uparrow^1 \langle N_2, b \rangle : T_{0.4}\]

D2.0:

\[;n_1, n_2; ;; \vdash \text{store}() : \mathbb{M}(n_1 \ast 3 + n_1) [(n_1 \ast 3 + n_1)] 1\]
\[;n_1, n_2; ;; \overline{N_1} : T_1, N_2 : T_2, a : T_{1.31} \vdash \text{bind} b = \text{store}() \text{ in } a \uparrow^1 \langle N_2, b \rangle : T_{0.5}\]

D0.2:

\[D2.0\]
\[;n_1, n_2; ;; N_1 : T_1, N_2 : T_2 \vdash E_2 : T_{0.5}\]

D0.1

\[D0.1\]
\[;n_1, n_2; ;; \overline{N_1} : T_1, N_2 : T_2 \vdash E_1 : T_{1.30}\]

D0.0:

\[;n_1, n_2; ;; p : T_p, \overline{N_1} : T_1, N_2 : T_2 \vdash \text{release} - = p \text{ in } \text{bind} a = E_1 \text{ in } E_2 : T_{0.40}\]

183
Main derivation:
\[
\begin{align*}
\vdash n_1, n_2; \ldots ; p : T_p, \overline{N_1} : T_1 \vdash \lambda \overline{N_2}, E_0 : T_{0.31} \\
\vdash n_1, n_2; \ldots ; p : T_p, \overline{N_1} : T_1 \vdash \text{ret } \lambda \overline{N_2}, E_0 : T_{0.3} \\
\vdash n_1, n_2; \ldots ; p : T_p \vdash \lambda \overline{N_1} \cdot \text{ret } \lambda \overline{N_2}, E_0 : T_{0.21} \\
\vdash n_1, n_2; \ldots ; \vdash \lambda p. \text{ret } \lambda \overline{N_1}, \lambda \overline{N_2}, E_0 : T_{0.2} \\
\vdash n_1; \ldots ; \vdash \lambda \Lambda p. \text{ret } \lambda \overline{N_1}, \lambda \overline{N_2}, E_0 : T_{0.1} \\
\vdash \vdots ; \vdash \lambda \Lambda p. \text{ret } \lambda \overline{N_1}, \lambda \overline{N_2}, E_0 : T_0
\end{align*}
\]

Type derivation for \textit{mult}
\[
\text{mult} = \Lambda \Lambda p. \text{ret } \lambda \overline{N_1}, \lambda \overline{N_2}, E_0
\]
where
\[E_0 = \text{release} \quad = p \text{ in bind } a = E_1 \text{ in } E_2\]
\[E_0.1 = \text{release} \quad = y_2 \text{ in bind } b_1 = (\text{bind } b_2 = \text{store } (\text{in add } [] \quad [\text{in } b_1 \quad \uparrow 1 \quad \overline{N_2} \text{ in } b_1 \quad \uparrow 1 \quad y_1)\]
\[E_1 = \overline{N_1} \quad \uparrow 1 \quad \uparrow 1 \quad \text{!}(\lambda \Lambda \text{let } (y_1, y_2) = t \text{ in } E_0.1)\]
\[E_2 = \text{bind } b = \text{store } (\text{in } a \quad \uparrow 1 \quad \uparrow (\overline{0}, b)\]

\[
\begin{align*}
T_p &= ([n_1 \cdot (n_2 \cdot 3 + n_2 + 4) + n_1 + 2] \quad 1 \\
T_0 &= \forall n_1, n_2. T_p \rightarrow M(0 \cdot (\text{Nat } n_1 \rightarrow M(0 \cdot (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_1 \cdot n_2))))) \\
T_{0.1} &= \forall n_2. T_p \rightarrow M(0 \cdot (\text{Nat } n_1 \rightarrow M(0 \cdot (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_1 \cdot n_2))))) \\
T_{0.2} &= T_p \rightarrow M(0 \cdot (\text{Nat } n_1 \rightarrow M(0 \cdot (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_1 \cdot n_2))))) \\
T_{0.21} &= M(0 \cdot (\text{Nat } n_1 \rightarrow M(0 \cdot (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_1 \cdot n_2))))) \\
T_{0.22} &= (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_1 \cdot n_2))))) \\
T_{0.3} &= M(0 \cdot (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_1 \cdot n_2)))) \\
T_{0.31} &= (\text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot n_2))) \\
T_{0.4} &= M(1 \cdot (\text{Nat } n_1 \cdot n_2)) \\
T_{0.5} &= M(n_1 \cdot (n_2 \cdot 3 + n_2 + 4) + n_1 + 1) \cdot (\text{Nat } n_1 \cdot n_2)) \\
T_{0.6} &= M(n_1 \cdot (n_2 \cdot 3 + n_2 + 4) + n_1 + 2) \cdot (\text{Nat } n_1 \cdot n_2)) \\
T_1 &= \forall a : N \rightarrow \text{Type} \cdot \forall \text{C}.!(\exists j_n . ((\alpha \cdot j_n \otimes [C \cdot j_n]) \rightarrow M(0 \cdot (\alpha \cdot (j_n + 1))))) \rightarrow \infty \\
M(0 \cdot (\alpha \rightarrow [(C \cdot 0 + \ldots + C \cdot (n_1 - 1) + n_1)] \cdot 1) \rightarrow M(0 \cdot (\alpha \cdot n_1)) \\
\lambda a. = \lambda \text{Nat } n_2 \cdot \lambda \text{n} \\
T_{1.1} &= \forall a. \lambda \text{n}(a \cdot j_n \otimes [C \cdot j_n]) \rightarrow M(0 \cdot (a \cdot (j_n + 1)))) \rightarrow \infty \\
M(0 \cdot (a \cdot 0 \rightarrow [(C \cdot 0 + \ldots + C \cdot (n_1 - 1) + n_1)] \cdot 1) \rightarrow M(0 \cdot (a \cdot n_1)) \\
T_{1.2} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot j_n \otimes [C \cdot j_n]) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (j_n + 1)))) \rightarrow \infty \\
M(0 \cdot (\text{Nat } n_2 \cdot 0 \rightarrow [(C \cdot 0 + \ldots + C \cdot (n_1 - 1) + n_1)] \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot n_1)) \\
T_{1.21} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot j_n \otimes [C \cdot j_n]) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (j_n + 1)))) \rightarrow \infty \\
M(0 \cdot (\text{Nat } n_2 \cdot 0 \rightarrow [(C \cdot 0 + \ldots + C \cdot (n_1 - 1) + n_1)] \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot n_1)) \\
T_{1.22} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot j_n \otimes [n_2 \cdot 3 + n_2 + 4]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (j_n + 1)))) \rightarrow \infty \\
T_{1.23} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot j_n \otimes [n_2 \cdot 3 + n_2 + 4]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (j_n + 1)))) \rightarrow \infty \\
T_{1.24} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot k \otimes [n_2 \cdot 3 + n_2 + 4]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.241} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot k \otimes [n_2 \cdot 3 + n_2 + 4]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.2412} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot k \otimes [n_2 \cdot 3 + n_2 + 4]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.242} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot k \otimes [n_2 \cdot 3 + n_2 + 4]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.243} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot 0 \rightarrow [(n_2 \cdot 3 + n_2 + 4)]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.3} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot 0 \rightarrow [(n_2 \cdot 3 + n_2 + 4)]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.30} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot 0 \rightarrow [(n_2 \cdot 3 + n_2 + 4)]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_{1.31} &= \forall a. \lambda \text{n}(\text{Nat } n_2 \cdot 0 \rightarrow [(n_2 \cdot 3 + n_2 + 4)]) \cdot 1) \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1)))) \\
T_2 &= \text{Nat } n_2 \\
T_3 &= \text{Nat } n_2 \rightarrow M(0 \cdot (\text{Nat } n_2 \cdot (k + 1))))
\]

D3:
\[
\begin{align*}
\vdash n_1, n_2; \ldots ; \overline{N_1} : T_1 \vdash \overline{N_1} : T_1
\end{align*}
\]
\[
\begin{align*}
D0.0 \\
\vdots : n_1, n_2 ; \vdots : p : T_p \vdash p : T_p & \quad D0 \\
\vdots : n_1, n_2 ; \vdots : p : T_p, N_1 : T_1, N_2 : T_2 \vdash \text{release - = p in bind a = E1 in } E_2 : T_{0.4}
\end{align*}
\]

Main derivation:

\[
\begin{align*}
D0.0 \\
\vdots : n_1, n_2 ; \vdots : p : T_p, \lambda x. N_1 : \lambda x. T_1 \vdash \lambda x. N_2, E_0 : T_{0.31} \\
\vdots : n_1, n_2 ; \vdots : p : T_p \vdash \lambda x. N_2, E_0 : T_{0.3} \\
\vdots : n_1, n_2 ; \vdots : p : T_p \vdash \lambda x. N_1, \lambda x. N_2, E_0 : T_{0.22} \\
\vdots : n_1, n_2 ; \vdots : p : T_p \vdash \lambda x. N_1, \lambda x. N_2, E_0 : T_{0.21} \\
\vdots : n_1, \vdots : \lambda x. \lambda y. \text{ret } \lambda x. N_1, \lambda x. N_2, E_0 : T_{0.2} \\
\vdots : \lambda x. \lambda y. \text{ret } \lambda x. N_1, \lambda x. N_2, E_0 : T_0 \\
\vdots : \vdots : \vdots : \lambda x. \lambda y. \text{ret } \lambda x. N_1, \lambda x. N_2, E_0 : T_0
\end{align*}
\]

Type derivation for \( \exp \)

\[
\begin{align*}
\exp : \forall n_1, n_2. [\sum_{i \in \{0, n_2 - 1\}} (\lambda k. (n_1 \times (n_1^k \times 3 + n_1^k + 4) + n_1 + 4)) (i)] + n_2 + 2] & \rightarrow \top \\
M_0 (\Nat n_1 \rightarrow M_0 (\Nat n_2 \rightarrow M_0 (\Nat (n_1^{n_2})))) \\
\end{align*}
\]

\[
\begin{align*}
\exp = \lambda \lambda \exp. \text{ret } \lambda x. N_2, E_0 \\
\end{align*}
\]

where

\[
\begin{align*}
E_0 = \text{release - = p in bind a = E1 in } E_2 \\
E_{0.1} = \text{release - = y2 in bind b1 = (bind b2 = store ( in mult [1] b2 \uparrow N_1)} \text{ in b1 \uparrow y1} \\
E_1 = N_2 \uparrow N_1 \uparrow \lambda x. \text{let } \langle y_1, y_2 \rangle \rightarrow t \text{ in } E_0 \uparrow 1) \\
E_2 = \text{bind b = store 1 in a \uparrow \langle \top, b \rangle} \\
P = \sum_{i \in \{0, n_2 - 1\}} (\lambda k. (n_1 \times (n_1^k \times 3 + n_1^k + 4) + n_1 + 4)) (i) + n_2 + 2 \\
T_0 = \[P \] 1 \\
T_1 = \[P \uparrow 1 \] 1 \\
T_{0.1} = \forall n_1, n_2. (\lambda x. N_1 \rightarrow M_0 (\Nat n_1 \rightarrow M_0 (\Nat n_2 \rightarrow M_0 (\Nat (n_1^{n_2})))) \\
T_{0.2} = \forall n_2, \lambda x. N_2, E_0 \rightarrow M_0 (\Nat n_1 \rightarrow M_0 (\Nat n_2 \rightarrow M_0 (\Nat (n_1^{n_2})))) \\
T_{0.3} = \forall n_2, \lambda x. N_2, E_0 \rightarrow M_0 (\Nat n_1 \rightarrow M_0 (\Nat n_2 \rightarrow M_0 (\Nat (n_1^{n_2})))) \\
T_{0.4} = \forall n_1, \lambda x. N_1, \lambda x. N_2, E_0 \rightarrow M_0 (\Nat n_1 \rightarrow M_0 (\Nat n_2 \rightarrow M_0 (\Nat (n_1^{n_2})))) \\
T_{0.5} = \forall n_1, \lambda x. N_1, \lambda x. N_2, E_0 \rightarrow M_0 (\Nat n_1 \rightarrow M_0 (\Nat (n_1^{n_2}))) \\
T_{0.6} = \forall n_1, \lambda x. N_1, \lambda x. N_2, E_0 \rightarrow M_0 (\Nat (n_1^{n_2}))) \\
T_1 = \forall n_1. \Nat (n_1^{n_2}) \\
T_{1.1} = \forall \alpha : N \rightarrow T_{\text{type}}. \forall \exp. (!\forall \alpha j_n. ((\alpha \times j_n \times \Nat j_n) \rightarrow M_0 (\alpha (j_n + 1)))) \rightarrow \top \\
M_0 ((\alpha \times [C 0 + \ldots + C (n_2 - 1) + n_2] \rightarrow M_0 (\alpha n_2)) \\
\end{align*}
\]

\[
\begin{align*}
\forall n_1, \lambda x. \Nat (n_1^{n_2}) \\
T_{1.2} = \forall \exp. (!\forall \alpha j_n. ((\alpha j_n \times [C 0 + \ldots + C (n_2 - 1) + n_2]) \rightarrow M_0 (\alpha n_2)) \\
M_0 ((\alpha \times [C 0 + \ldots + C (n_2 - 1) + n_2] \rightarrow M_0 (\Nat (n_1^{n_2})))) \\
\end{align*}
\]

\[
\begin{align*}
\forall \exp. (!\forall \alpha j_n. ((\alpha j_n \times [\Nat (n_1^{n_2}) \times (n_1 \times (n_1^{n_2} \times 3 + n_1^{n_2} + 4) + n_1 + 4)] \rightarrow M_0 (\Nat (n_1^{n_2} + 1)))) \rightarrow \top \\
M_0 ((\Nat (n_1^{n_2}) \times [C 0 + \ldots + C (n_2 - 1) + n_2]) \rightarrow M_0 (\Nat (n_1^{n_2} + 1))) \\
\end{align*}
\]

\[
\begin{align*}
P = (\lambda k. (n_1 \times (n_1^k \times 3 + n_1^k + 4) + n_1 + 4)) 0 + \ldots + (\lambda k. (n_1 \times (n_1^k \times 3 + n_1^k + 4) + n_1 + 4)) (n_2 - 1) + n_2 \\
\end{align*}
\]
\[ D_3: \]
\[
\vdash n_1, n_2, \ldots; \vdash T_2 : T_1 \]

D2.1:
\[
\vdash n_1, n_2, \ldots; \vdash (\lambda_x. k. \text{Nat}[n_2^k]) : \mathbb{N} \to \text{Type}
\]
\[
\vdash n_1, n_2, \ldots; \vdash \text{Nat} : T_1 + \text{Nat} \mathcal{T} : T_1.1
\]

D2:
\[
\vdash n_1, n_2, \ldots; \vdash (\lambda_x. k. (n_1^k * 3 + n_1^k + 4) + n_1 + 2)) : \mathbb{N} \to \mathbb{N}
\]
\[
\vdash n_1, n_2, \ldots; \vdash \text{Nat} : T_2 : T_1 + \text{Nat} \mathcal{T} : T_2.1
\]

D1.32
\[
\vdash n_1, n_2, k, \ldots; \vdash b_2 : \{(n_1 * (n_1^k + 3 + n_1^k + 4) + n_1 + 2)) \} \vdash \text{mult} \mathcal{T} \vdash b_2 \downarrow T_1 \text{Nat}_1 : \mathbb{M} 1 T_3
\]

D1.31
\[
\vdash n_1, n_2, k, \ldots; \vdash \text{store}() : \mathbb{M}((n_1 * (n_1^k + 3 + n_1^k + 4) + n_1 + 2)) \}} \vdash \text{mult} \mathcal{T} \vdash b_2 \downarrow T_1 \text{Nat}_1 : \mathbb{M} 1 T_3
\]

D1.3
\[
\vdash n_1, n_2, k, \ldots; \vdash y_1 : T_1.241, b_1 : T_3 \vdash (\text{bind } b_2 = \text{store}() \text{ in } \text{mult} \mathcal{T} \vdash b_2 \downarrow T_1 \text{Nat}_1) : \mathbb{M}(n_1 * (n_1^k + 3 + n_1^k + 4) + n_1 + 3) T_3
\]

D1.2:
\[
\vdash n_1, n_2, k, \ldots; \vdash y_2 : T_1.2412 \vdash y_2 : T_1.2412
\]

D1.3
\[
\vdash n_1, n_2, k, \ldots; \vdash y_1 : T_1.241, y_2 : T_1.2412 \vdash \text{release } = y_2 \text{ in } \text{bind } b_1 = (\text{bind } b_2 = \text{store}() \text{ in } \text{mult} \mathcal{T} \vdash b_2 \downarrow T_1 \text{Nat}_1) \text{ in } b_1 \uparrow T_1 \text{Nat}_1 : \mathbb{M} 0 \text{Nat}_2^{(k + 1)}
\]

D1.1
\[
\vdash n_1, n_2, k, \ldots; \vdash t : T_1.241 \vdash t : T_1.241
\]

D2
\[
\vdash n_1, n_2, \ldots; \vdash (\text{A.M. let } \langle y_1, y_2 \rangle = t \text{ in } E_0.1) : T_1.241
\]

D0.1:
\[
\vdash n_1, n_2, \ldots; \vdash \text{Nat}_1 : T_1 \vdash \text{Nat}_1 \mathcal{T} \vdash \text{Nat}_1 \mathcal{T} : T_1.30
\]

D2.1:
\[
\vdash n_1, n_2, \ldots; \vdash \text{Nat}_2 : T_2, a : T_1.31, b : T_0 \vdash a \downarrow T_0.4
\]
C.2 Map

\[\text{map} : \forall n, c.(\tau_1 \rightarrow M c \tau_2) \rightarrow L^n([c] \tau_1) \rightarrow M 0 (L^n \tau_2)\]

\[\text{map} \equiv \text{fix} f. \Lambda. \Lambda. \lambda g.l \! et ! g_u = g \text{ in } E_0\]

\[E_0 = \text{match } t \text{ with } | \text{nil} \mapsto E_{0.1} | h :: t \mapsto E_{0.2} \]

\[E_{0.1} = \text{ret } \text{nil}\]

\[E_{0.2} = \text{release } h_e = h \text{ in } E_{0.3}\]

\[E_{0.3} = \text{bind } h_n = g_u h_e \text{ in } E_{0.4}\]

\[E_{0.4} = \text{bind } t_n = f \times \times ! g_u \text{ in ret } h_n :: t_n\]

\[E_1 = \Lambda. \Lambda. \lambda g.l \! et ! g_u = g \text{ in } E_0\]

\[E_2 = \lambda g.l \! et ! g_u = g \text{ in } E_0\]

\[E_3 = \text{let } g_u = g \text{ in } E_0\]

\[T_0 = \forall n, c.(\tau_1 \rightarrow M c \tau_2) \rightarrow L^n([c] \tau_1) \rightarrow M 0 (L^n \tau_2)\]

\[T_1 = ! (\tau_1 \rightarrow M c \tau_2) \rightarrow L^n([c] \tau_1) \rightarrow M 0 (L^n \tau_2)\]

\[T_{1.1} = (\tau_1 \rightarrow M c \tau_2)\]

\[T_{1.2} = L^n([c] \tau_1)\]

\[T_{1.3} = M 0 (L^n \tau_2)\]

D1.2:

\[; n, c, i; n = i + 1; f : T_0, g_u : T_{1.1}; h_n : \tau_2, t_n : L^i \tau_2 \vdash \text{ret } h_n :: t_n : M 0 L^n \tau_2\]

D1.1:

\[; n, c, i; n = i + 1; f : T_0, g_u : T_{1.1}; h_n : \tau_2 \vdash f \times \times ! g_u \text{ in } E_{0.4} : M 0 L^n \tau_2\]
C.3 Append

\[ \text{append} : \forall s_1, s_2, L^{s_1}[1] \tau \rightarrow L^{s_2} \tau \rightarrow M_0 (L^{s_1+s_2} \tau) \]

\[ \text{append} \triangleq \text{fix}f.\Delta.\lambda l_2.E_0 \]

\[ E_0 = \text{match } l_1 \text{ with } \mid \text{nil} \mapsto E_0.1 \mid h :: t \mapsto E_0.2 \]

\[ E_{0.1} = \text{ret } \text{nil} :: l_2 \]

\[ E_{0.2} = \text{release } h_c = h \text{ in } \text{bind } t_c = f[[ ]] \mid t_2 \text{ in } E_{0.3} \]

\[ E_{0.3} = \text{bind } = = \uparrow^1 \text{ in ret } h_c :: t_c \]

Typing derivation

\[ E_0 = \text{match } l_1 \text{ with } \mid \text{nil} \mapsto E_0.1 \mid h :: t \mapsto E_0.2 \]

\[ E_{0.1} = \text{ret } \text{nil} :: l_2 \]

\[ E_{0.2} = \text{release } h_c = h \text{ in } \text{bind } t_c = f[[ ]] \mid t_2 \text{ in } E_{0.3} \]

\[ E_{0.3} = \text{bind } = = \uparrow^1 \text{ in ret } h_c :: t_c \]

\[ T_0 = \forall s_1, s_2, L^{s_1}[1] \tau \rightarrow L^{s_2} \tau \rightarrow M_0 (L^{s_1+s_2} \tau) \]

\[ T_1 = L^{s_1}[1] \tau \rightarrow L^{s_2} \tau \rightarrow M_0 (L^{s_1+s_2} \tau) \]

\[ T_{1.1} = L^{s_1}[1] \tau \]

\[ T_{1.2} = L^{s_2} \tau \]

\[ T_{1.3} = M_0 (L^{s_1+s_2} \tau) \]

\[ T_2 = L^{s_2} \tau \rightarrow M_1 (L^{s_1+s_2} \tau) \]

D1.0:

\[ \vdash n, c, i; n = i + 1; f : T_0, g : T_{1.1}; h_c : \tau, t : L^{[i]} \tau \vdash (g_\tau, h_c) : M \epsilon T_2 \]

D1.1:

\[ \vdash n, c, i; n = i + 1; f : T_0, g : T_{1.1}; h_c : \tau, t : L^{[i]} \tau \vdash E_{0.3} : M \epsilon T_2 \]

D1:

\[ \vdash n, c, i; n = i + 1; f : T_0, g : T_{1.1}; h_c : \tau, t : L^{[i]} \tau \vdash E_{0.2} : M \epsilon T_2 \]

D0:

\[ \vdash n, c; n = 0; f : T_0, g : T_{1.1}; \vdash \text{cons } n : L^{s_2} \tau \]

\[ \vdash n, c; n = 0; f : T_0, g : T_{1.1}; \vdash \text{ret } \text{nil} :: M \epsilon T_2 \]

\[ \vdash n, c; n = 0; f : T_0, g : T_{1.1}; \vdash E_{0.1} : M \epsilon T_2 \]

Main derivation:

\[ \vdash n, c; f : T_0, g : T_{1.1} \vdash g : T_{1.0} \]

D0

D1

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash l : T_{1.0} \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_3 : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_0 : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash \text{cons } n : L^{s_2} \tau \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash \text{ret } \text{nil} :: M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.1} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.2} : M \epsilon T_2 \]

\[ \vdash n, c; f : T_0, g : T_{1.1}, l : T_{1.2} \vdash E_{0.3} : M \epsilon T_2 \]
D0:

\[ s_1, s_2; s_1 = 0; f : T_0; l_2 : T_{1,2} \vdash E_2 : L^{s_1+s_2} \tau \]

Main derivation:

\[ s_1, s_2; s_1 = 0; f : T_0; l_2 : T_{1,2} \vdash \text{ret } l_2 : \text{M}0(L^{s_1+s_2}\tau) \]

\[ s_1, s_2; s_1 = 0; f : T_0; l_1 : T_{1,1} \vdash l_1 : T_{1,1} \]

\[ s_1, s_2; s_1 = 0; f : T_0; l_2 : T_{1,2} \vdash E_0 : \text{M}0(L^{s_1+s_2}\tau) \]

C.4 Eager functional queue

enqueue : \forall m, n. [3] 1 \rightarrow \tau \rightarrow L^n([2] \tau) \rightarrow L^m\tau \rightarrow \text{M}0(L^{n+1}([2] \tau) \otimes L^m\tau)

enqueue \triangleq \lambda \lambda p a l_1 l_2. \text{release } = p \text{ in bind } x = \text{store } a \text{ in bind } = \text{↑}^1 \text{ in ret }\langle\langle x :: l_1, l_2\rangle\rangle

Typing derivation for enqueue

\[ T_0 = \forall m, n. [3] 1 \rightarrow \tau \rightarrow L^n([2] \tau) \rightarrow L^m\tau \rightarrow \text{M}0(L^{n+1}([2] \tau) \otimes L^m\tau) \]

\[ T_1 = [3] 1 \rightarrow \tau \rightarrow L^n([2] \tau) \rightarrow L^m\tau \rightarrow \text{M}0(L^{n+1}([2] \tau) \otimes L^m\tau) \]

\[ T_2 = \tau \rightarrow L^n([2] \tau) \rightarrow L^m\tau \rightarrow \text{M}0(L^{n+1}([2] \tau) \otimes L^m\tau) \]

\[ T_3 = L^n([2] \tau) \]

\[ T_4 = L^m\tau \]

\[ T_5 = \text{M}0(L^{n+1}([2] \tau) \otimes L^m\tau) \]

\[ T_6 = \text{M}1(L^{n+1}([2] \tau) \otimes L^m\tau) \]

\[ T_7 = \text{M}3(L^{n+1}([2] \tau) \otimes L^m\tau) \]

\[ \text{enqueue } = \lambda \lambda p a l_1 l_2. \text{release } = p \text{ in bind } x = \text{store } a \text{ in bind } = \text{↑}^1 \text{ in ret }\langle\langle x :: l_1, l_2\rangle\rangle \]

\[ E_1 = \lambda \lambda p a l_1 l_2. \text{release } = p \text{ in bind } x = \text{store } a \text{ in bind } = \text{↑}^1 \text{ in ret }\langle\langle x :: l_1, l_2\rangle\rangle \]

\[ E_2 = \text{release } = p \text{ in bind } x = \text{store } a \text{ in bind } = \text{↑}^1 \text{ in ret }\langle\langle x :: l_1, l_2\rangle\rangle \]

\[ E_3 = \text{ret }\langle\langle x :: l_1, l_2\rangle\rangle \]

D2:

\[ m, n; \vdash x : [2] \tau, l_1 : L^n([2] \tau), l_2 : L^m\tau \vdash E_5 : T_4 \]

D1:

\[ m, n; \vdash \text{↑}^1 : \text{M}11 \]

D0:

\[ m, n; \vdash a : \tau \vdash \text{store } a : \text{M}2([2] \tau) \]

Main derivation:

\[ m, n; \vdash p : T_{1,0} \vdash p : T_{1,0} \]

\[ m, n; \vdash p : T_{1,0}, a : \tau, l_1 : L^n([2] \tau), l_2 : L^m\tau \vdash E_2 : T_4 \]

\[ m, n; \vdash E_1 : T_1 \]

\[ m, n; \vdash \vdash \text{enqueue } : T_0 \]

\[ Dq : \forall m, n. (m + n > 0) \Rightarrow [1] 1 \rightarrow L^m([2] \tau) \rightarrow L^n\tau \rightarrow \text{M}0((m' + n' + 1) = (m + n)) \land (L^{m'}[2] \tau \otimes L^{n'}\tau)) \]

\[ Dq \triangleq \Lambda \Lambda \lambda l_1 l_2. \text{match } l_2 \text{ with } \text{nil } \mapsto E_1 \mid h_2 : l_2 \mapsto E_2 \]

\[ E_1 = \text{bind } l_1 = M \llbracket l_1 \text{ nil in match } l_1 \text{ with } \text{nil } \mapsto \text{h}_2 : l'_2 \mapsto E_{1,1} \]

\[ E_{1,1} = \text{release } = p \text{ in bind } = \text{↑}^1 \text{ in ret }\langle\langle \text{nil}, l'_2\rangle\rangle \]

\[ E_2 = \text{release } = p \text{ in bind } = \text{↑}^1 \text{ in ret }\langle\langle l_1, l'_2\rangle\rangle \]

190
Typing derivation for dequeue $Dq$

$T_0 = \forall m, n. (m + n > 0) \Rightarrow [[1] 1 \rightarrow L_m^2[2] \tau \rightarrow L_n^2 \tau \rightarrow$

$M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{T_1 = (m + n > 0) \Rightarrow [[1] 1 \rightarrow L_m^2[2] \tau \rightarrow L_n^2 \tau}$

$M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{T_2 = [[1] 1 \rightarrow L_m^2[2] \tau \rightarrow L_n^2 \tau \rightarrow M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{T_{2:1} = L_m^2[2] \tau \tau \rightarrow M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{T_{3:1} = L_n^2 \tau \rightarrow M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{T_{3:1} = L_m^2[2] \tau \rightarrow M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{T_{3} = (\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)) [m/m'] \tau L_{m' + n'} \tau}$

$T_0 = (m + n + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)[0/m'][i/n']$

$T_7 = (L_m^2[2] \tau \otimes L_n^2 \tau)$

$E_0 = \Lambda \Lambda \Lambda \Lambda \lambda l_1 l_2, \text{match } l_2 \text{ with } \text{nil} \mapsto E_1 \; l_2 \mapsto l_2' \mapsto E_2$

$E_{0:1} = \lambda p \; l_1 l_2. \text{match } l_2 \text{ with } \text{nil} \mapsto E_1 \; l_2 \mapsto l_2' \mapsto E_2$

$E_{0:2} = \text{match } l_2 \text{ with } \text{nil} \mapsto E_1 \; h_2 \mapsto l_2' \mapsto E_2$

$E_1 = \text{bind } t = M \frac{\exists m', n'. (m' + n' + 1) = (m + n) \& (L_m^2[2] \tau \otimes L_n^2 \tau)}{E_1.1 = \text{release } \rightarrow = p \text{ in } \text{bind } x = \uparrow \text{ in } \Lambda \; \text{ret } \langle \text{nil}, l_1' \rangle}$

$E_{2:0} = \text{release } \rightarrow = p \text{ in } \text{bind } x = \uparrow \text{ in } \Lambda \; \text{ret } \langle l_1, l_2' \rangle$

D1.3:

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; h_2 : \tau, l_2' : L^i \tau, l_1 : T_{2:1} \vdash \langle l_1, l_2' \rangle : T_{5:2}$

D1.2:

$\vdash m, n, i; (j + 1 = n), (m + n) > 0 \vdash (m + i + 1) = (m + n)$

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; h_2 : \tau, l_2' : L^i \tau, l_1 : T_{2:1} \vdash \Lambda \; \langle l_1, l_2' \rangle : T_{5:1}$

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; h_2 : \tau, l_2' : L^i \tau, l_1 : T_{2:1} \vdash \Lambda \; \langle l_1, l_2' \rangle : T_{5}$

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; h_2 : \tau, l_2' : L^i \tau, l_1 : T_{2:1} \vdash \text{bind } = \uparrow \text{ in } \Lambda \; \langle l_1, l_2' \rangle : T_{4:1}$

D1.1:

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; h_2 : \tau, l_2' : L^i \tau, l_1 : T_{2:1} \vdash \text{bind } = \uparrow \text{ in } \Lambda \; \langle l_1, l_2' \rangle : T_{4}$

D1:

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; p : [1] 1 \vdash p : [1] 1$

$\vdash m, n, i; (j + 1 = n), (m + n) > 0; h_2 : \tau, l_2' : L^i \tau, l_1 : T_{2:1} \vdash \Lambda \; \langle l_1, l_2' \rangle : T_{4}$

D0.05:

$\vdash m, n, i; (n = 0), (i + 1 = m), (m + n) > 0, (0 + u + 1) = (m + n) ; h_2 \tau, l_2' : L^i \tau \vdash \langle \text{nil}, l_2' \rangle : T_7$

D0.04:

$\vdash m, n, i; (n = 0), (i + 1 = m), (m + n) > 0 \vdash (0 + i + 1) = (m + n)$

$\vdash m, n, i; (n = 0), (i + 1 = m), (m + n) > 0; h_2 \tau, l_2' : L^i \tau \vdash \Lambda \; \langle \text{nil}, l_2' \rangle : T_9$

$\vdash m, n, i; (n = 0), (i + 1 = m), (m + n) > 0; h_2 \tau, l_2' : L^i \tau \vdash \Lambda \; \langle \text{nil}, l_2' \rangle : T_5$

$\vdash m, n, i; (n = 0), (i + 1 = m), (m + n) > 0; h_2 \tau, l_2' : L^i \tau \vdash \Lambda \; \langle \text{nil}, l_2' \rangle : T_4$

D0.03:

$\vdash m, n, i; (n = 0), (i + 1 = m), (m + n) > 0; h_2 \tau, l_2' : L^i \tau \vdash \text{bind } = \uparrow \text{ in } \Lambda \; \langle \text{nil}, l_2' \rangle : T_{4:1}$
Move \(m, n, L^m([2] \tau) \Rightarrow L^n \tau \Rightarrow \emptyset 0 (L^{m+n} \tau)\)

\(Move \triangleq \text{fixM} \Delta \Lambda \lambda l_1 l_2.\text{match} l_1 \text{ with } |nil \mapsto E_1 | h_1 :: l'_1 \mapsto E_2\)

\(E_1 = \text{ret}(l_2)\)

\(E_2 = \text{release} - h \text{ in } \text{bind} - = \uparrow^2 \text{ in } M [\ ] l'_1 (h_1 :: l_2)\)

Typing derivation for \(Move\)

\(T_0 = \forall m, n, L^m([2] \tau) \Rightarrow L^n \tau \Rightarrow \emptyset 0 (L^{m+n} \tau)\)

\(T_1 = L^m([2] \tau) \Rightarrow L^n \tau \Rightarrow \emptyset 0 (L^{m+n} \tau)\)

\(T_{1,1} = L^m([2] \tau)\)

\(T_{1,2} = L^m \Rightarrow \emptyset 0 (L^{m+n} \tau)\)

\(T_{2} = L^n \Rightarrow \emptyset 0 (L^{m+n} \tau)\)

\(T_3 = \emptyset 0 (L^{m+n} \tau)\)

\(T_3,1 = \emptyset 0 (L^{m+n+1} \tau)\)

\(T_3,2 = \emptyset 2 (L^{m+n} \tau)\)

\(E_0 = \text{fixM} \Delta \Lambda \lambda l_1 l_2.\text{match} l_1 \text{ with } |nil \mapsto E_1 | h_1 :: l'_1 \mapsto E_2\)

\(E_{0,0} = \Delta \Lambda \lambda l_1 l_2.\text{match} l_1 \text{ with } |nil \mapsto E_1 | h_1 :: l'_1 \mapsto E_2\)

\(E_{0,1} = \lambda \lambda l_1 l_2.\text{match} l_1 \text{ with } |nil \mapsto E_1 | h_1 :: l'_1 \mapsto E_2\)

\(E_{0,2} = \text{match} l_1 \text{ with } |nil \mapsto E_1 | h_1 :: l'_1 \mapsto E_2\)

\(E_1 = \text{ret}(l_2)\)

\(E_2 = \text{release} - h \text{ in } \text{bind} - = \uparrow^2 \text{ in } M [\ ] l'_1 (h_1 :: l_2)\)

\(E_{2,1} = \text{bind} - = \uparrow^2 \text{ in } M [\ ] l'_1 (h_1 :: l_2)\)

\(E_{2,2} = \emptyset 2 (l'_1 (h_1 :: l_2)\}

\(D_3: m, n, i; i + 1 = m; M : T_0; l'_1 : L^m([2] \tau), l_2 : T_2 ; M [\ ] l'_1 (h_1 :: l_2) : T_4\)

\(D_2: m, n, i; i + 1 = m; M : T_0; l'_1 : L^m([2] \tau), l_2 : T_2 ; M [\ ] l'_1 (h_1 :: l_2) : T_3\)

\(D_1: m, n, i; i + 1 = m; M : T_0; h_1 : [2] \tau \vdash h_1 : [2] \tau\)
C.5 Okasaki’s implicit queue

Typing rules for value constructors and case analysis

\[
\begin{align*}
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash C0 : Queue \tau & \quad \text{ \(T-C0\)} \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : \tau & \quad \text{ \(T-C1\)} \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : [1] 1 \to M0(\tau \otimes Queue (\tau \otimes \tau)) & \quad \text{ \(T-C2\)} \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash C2 e : Queue \tau & \quad \text{ \(T-C3\)} \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : [2] 1 \to M0((\tau \otimes \tau) \otimes Queue (\tau \otimes \tau)) & \quad \text{ \(T-C4\)} \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash e : [1] 1 \to M0(((\tau \otimes \tau) \otimes Queue (\tau \otimes \tau)) \otimes \tau) & \quad \text{ \(T-C5\)} \\
\Psi; \Theta; \Delta; \Omega; \Gamma \vdash C1 e : Queue \tau & \quad \text{ \(T-caseIQ\)}
\end{align*}
\]
$|C_4\ x\mapsto$

\[
\begin{align*}
\text{bind } p' &= \text{store()} \\
\text{ret } C_5\ (\lambda p''). \\
\text{= } \text{release } p' \text{ in } \text{= } \text{release } p'' \text{ in } \\
\text{bind } p'' &= \text{store()} \text{ in } \text{let } \langle f, m \rangle = x' \ p'' \text{ in } \\
\text{ret } \langle \langle f, m \rangle, a \rangle \end{align*}
\]

$|C_5\ x\mapsto$

\[
\begin{align*}
\text{bind } p' &= \text{store()} \\
\text{bind } x' &= x' \ p' \text{ in } \\
\text{let } \langle f, m \rangle &= x' \text{ in } \text{let } \langle f, m \rangle = fm \text{ in } \\
\text{ret } (C_4\ (\lambda p''). \\
\text{bind } m' &= \text{snoc } p' \ m \text{ in } \text{ret } \langle f, m' \rangle)
\end{align*}
\]

Listing 4: snoc function

\[
E_{0.0} = \text{= } \text{release } p \text{ in } E_{0.1}
\]

\[
E_{0.1} = \text{= } \text{= } \uparrow; E_{0.2}
\]

\[
E_{0.2} = \text{case } q \text{ of } C_0 \mapsto E_0 | C_1 \mapsto E_1 | C_2 \mapsto E_2 | C_3 \mapsto E_3 | C_4 \mapsto E_4 | C_5 \mapsto E_5
\]

\[
E_0 = \text{ret } (C_1\ a)
\]

\[
E_1 = \text{ret } C_4\ (\lambda p''., \text{ret } \langle \langle x, a \rangle, C_0 \rangle)
\]

\[
E_2 = \text{bind } p' = \text{store()} \text{ in } E_{2.1}
\]

\[
E_{2.1} = \text{bind } x' = x' \ p' \text{ in } E_{2.2}
\]

\[
E_{2.2} = \text{let } \langle f, m \rangle = x' \text{ in } E_{2.3}
\]

\[
E_{2.3} = \text{ret } (C_3\ (\lambda p''. \langle \langle f, m \rangle, a \rangle))
\]

\[
E_3 = \text{bind } p' = \text{store()} \text{ in } E_{3.1}
\]

\[
E_{3.1} = \text{bind } x' = x' \ p' \text{ in } E_{3.2}
\]

\[
E_{3.2} = \text{let } \langle f, m \rangle = x' \text{ in } E_{3.3}
\]

\[
E_{3.3} = \text{let } \langle f, m \rangle = fm \text{ in } E_{3.31}
\]

\[
E_{3.31} = \text{bind } p_0 = \text{store()} \text{ in } E_{3.4}
\]

\[
E_{3.4} = \text{ret } C_2\ (\lambda p''. E_{3.41})
\]

\[
E_{3.41} = \text{= } \text{release } p_0 \text{ in } \text{= } \text{release } p'' = \text{store()} \text{ in } E_{3.42}
\]

\[
E_{3.42} = \text{bind } m' = \text{snoc } p'' \ m \ (r, a) \text{ in } \text{ret } \langle f, m' \rangle
\]

\[
E_4 = \text{bind } p' = \text{store()} \text{ in } E_{4.1}
\]

\[
E_{4.1} = \text{ret } C_5\ (\lambda p''. E_{4.11})
\]

\[
E_{4.11} = \text{= } \text{release } p' \text{ in } \text{= } \text{release } p'' = \text{bind } p'' = \text{store()} \text{ in } E_{4.12}
\]

\[
E_{4.12} = \text{bind } p'' = \text{store()} \text{ in } \text{let } \langle f, m \rangle = x' \ p'' \text{ in } E_{4.13}
\]

\[
E_{4.13} = \text{ret } \langle \langle f, m \rangle, a \rangle
\]

\[
E_5 = \text{bind } p' = \text{store()} \text{ in } E_{5.1}
\]

\[
E_{5.1} = \text{bind } x' = x' \ p' \text{ in } E_{5.2}
\]

\[
E_{5.2} = \text{let } \langle f, m \rangle = x' \text{ in } E_{5.3}
\]

\[
E_{5.3} = \text{let } \langle f, m \rangle = fm \text{ in } E_{5.4}
\]

\[
E_{5.4} = \text{ret } (C_4\ (\lambda p''. \text{bind } m' = \text{snoc } p'' \ m \text{ in } \text{ret } \langle f, m' \rangle))
\]

$T_0 = [2] \ 1 \mapsto \forall \alpha. \text{Queue } \alpha \rightarrow \alpha \rightarrow \text{M}0 \ 0 \ 0 \text{Queue } \alpha$

$T_0 = \text{M}0 \ 0 \text{Queue } \alpha$

$T_1 = \text{M}1 \ 0 \text{Queue } \alpha$

$T_2 = \text{M}2 \ 0 \text{Queue } \alpha$

$T_3 = \text{M}0 \ (\alpha \odot \text{Queue } (\alpha \odot \alpha))$

$T_{3.1} = (\alpha \odot \text{Queue } (\alpha \odot \alpha))$

$T_{3.2} = \text{Queue } (\alpha \odot \alpha)$

$T_4 = \text{M}0 \ (\alpha \odot \text{Queue } (\alpha \odot \alpha) \odot \alpha)$

$T_{4.1} = (\alpha \odot \text{Queue } (\alpha \odot \alpha) \odot \alpha)$

$T_{4.2} = \alpha \odot \text{Queue } (\alpha \odot \alpha)$

$T_{4.3} = \text{Queue } (\alpha \odot \alpha)$

$T_5 = [2] \ 1 \mapsto \text{M}0 \ (\alpha \odot \alpha) \odot \text{Queue } (\alpha \odot \alpha)$

$T_{5.1} = \text{M}0 \ (\alpha \odot \alpha) \odot \text{Queue } (\alpha \odot \alpha)$

$T_{5.2} = (\alpha \odot \alpha) \odot \text{Queue } (\alpha \odot \alpha)$

$T_{5.3} = (\alpha \odot \alpha)$

$T_{5.4} = \text{Queue } (\alpha \odot \alpha)$

194
\[ T_0 = [1] \vdash M_0 ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha) \otimes \alpha) \]
\[ T_{0,1} = M_0 ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha) \otimes \alpha) \]
\[ T_{0,2} = ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha) \otimes \alpha) \]
\[ T_{0,3} = (\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha) \]
\[ T_{0,4} = (\alpha \otimes \alpha) \]
\[ T_{0,5} = \text{Queue} (\alpha \otimes \alpha) \]
\[ T_7 = M_0 (\alpha \otimes \text{Queue} (\alpha \otimes \alpha)) \]
\[ T_{7,1} = M_1 (\alpha \otimes \text{Queue} (\alpha \otimes \alpha)) \]
\[ T_{7,2} = M_2 (\alpha \otimes \text{Queue} (\alpha \otimes \alpha)) \]
\[ T_8 = M_0 ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha)) \]
\[ T_{8,1} = ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha)) \otimes \alpha \]
\[ T_9 = M_0 ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha)) \]
\[ T_{9,1} = ((\alpha \otimes \alpha) \otimes \text{Queue} (\alpha \otimes \alpha)) \]

D5.5:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; a : \alpha, f : T_{0,4}, m : T_{0,5}, p'' : [2] 1 \vdash S p'' \parallel m \langle r, a \rangle : M_0 (\text{Queue} (\alpha \otimes \alpha)) \\
\alpha; :: S : T_{0,6}; a : \alpha, f : T_{0,4}, m : T_{0,5}, p'' : [2] 1 \vdash \text{let} (\langle f, m \rangle) = f m \text{ in } E_{6,4} : T_0 \\
\end{array}
\]

D5.4:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; a : \alpha, f : T_{0,4}, m : T_{0,5}, p'' : [2] 1 \vdash S p'' \parallel m \langle r, a \rangle : M_0 (\text{Queue} (\alpha \otimes \alpha)) \\
\end{array}
\]

D5.3:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; f m : T_{0,5} \vdash \text{let} (\langle f, m \rangle) = f m \text{ in } E_{6,4} : T_0 \\
\end{array}
\]

D5.2:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; x' : T_{0,6} \vdash x' : T_{0,6} \\
\end{array}
\]

D5.1:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; x : T_{0,6}, p' : [1] 1 \vdash x p' : T_{6,1} \\
\end{array}
\]

D5:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; a : \alpha, x : T_{0,6}, p : [1] 1 \vdash \text{let} x' = x p' \text{ in } E_{5,2} : T_0 \\
\end{array}
\]

D4.5:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; a : \alpha, x : T_{0,4}, f : T_{3,3}, m : T_{3,4} \vdash \text{let} (\langle f, m \rangle, a) : T_{8,1} \\
\end{array}
\]

D4.4:
\[
\begin{array}{l}
\alpha; :: S : T_{0,6}; a : \alpha, x : T_{0,4}, p'' : [2] 1 \vdash x p'' : T_{5,1} \\
\end{array}
\]
D4.1:
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5 \vdash \text{store}() : \mathbb{M} 2 ([2 \ 1]) \]
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5 \vdash \text{bind} p'' = \text{store}() \text{ in let } \langle f, m \rangle = p'' \text{ in } E_{4.13} : T_{8.2} \]
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5 \vdash E_{4.12} : T_{8.2} \]

D4.2:
\[ \alpha::\varepsilon::S:T_0;\varepsilon:p':[1 \ 1] \vdash p'' : [1 \ 1] \]
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5,p' : [1 \ 1] \vdash - = \text{release} p'' \text{ in } E_{4.12} : T_{8.1} \]

D4.11:
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5,p' : [1 \ 1] \vdash - = \text{release} p'' \text{ in } - = \text{release} p'' \text{ in } E_{4.12} : T_{8} \]

D4.3:
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5,p' : [1 \ 1] \vdash - = \text{release} p'' \text{ in } - = \text{release} p'' \text{ in } E_{4.12} : T_{8} \]

D4.4:
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5 \vdash \text{store}() : \mathbb{M} 2 ([2 \ 1]) \]
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5 \vdash \text{bind} p'' = \text{store}() \text{ in let } \langle f, m \rangle = p'' \text{ in } E_{4.13} : T_{8.2} \]
\[ \alpha::\varepsilon::S:T_0;\varepsilon:a:a,x:T_5 \vdash E_{4.12} : T_{8.2} \]
D1.31:  
\[
\frac{\alpha; \alpha; S : T_0; q : \text{Queue} \alpha, a : \alpha, f : \alpha, m : T_{32} \vdash \text{store}\(\alpha\)}{\alpha; \alpha; S : T_0; p' : [0] \vdash x, p' : T_4}
\]

D3.4:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha, f : \alpha, m : T_{43}, r : \alpha \vdash \text{bind}\(\alpha\)}{\alpha; \alpha; S : T_0; q : \text{Queue} \alpha, a : \alpha, f : \alpha, m : T_{43}, r : \alpha \vdash E_{3.31} : T_1}
\]

D3.3:  
\[
\frac{\alpha; \alpha; S : T_0; f : m : T_{42} \vdash f : T_{42}}{\alpha; \alpha; S : T_0; a : \alpha, f : m : T_{43}, r : \alpha \vdash \text{let}\(\alpha, f, m\) = f m \in E_{3.31} : T_1}
\]

D3.2:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha, x' : T_{41} \vdash x': T_{41}}{\alpha; \alpha; S : T_0; a : \alpha, f : m : T_{42}, r : \alpha \vdash E_{3.3} : T_1}
\]

D3.1:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha, x : [0] \vdash \text{M} \(\alpha \otimes \text{Queue} \(\alpha \otimes \alpha \)) \vdash \alpha}{\alpha; \alpha; S : T_0; a : \alpha, x : [0] \vdash \text{M} \(\alpha \otimes \text{Queue} \(\alpha \otimes \alpha \)) \vdash E_{3.1} : T_1}
\]

D3:  
\[
\frac{\alpha; \alpha; S : T_0; \vdash \text{store}\(\alpha\)}{\alpha; \alpha; S : T_0; a : \alpha, x : [0] \vdash \text{M} \(\alpha \otimes \text{Queue} \(\alpha \otimes \alpha \)) \vdash E_{3} : T_1}
\]

D2.1:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha, x : [1] \vdash \text{M} \(\alpha \otimes \text{Queue} \(\alpha \otimes \alpha \)) \vdash E_{2.1} : T_0}{\alpha; \alpha; S : T_0; a : \alpha, x : [1] \vdash \text{M} \(\alpha \otimes \text{Queue} \(\alpha \otimes \alpha \)) \vdash E_{2} : T_1}
\]

D2:  
\[
\frac{\alpha; \alpha; S : T_0; \vdash \text{store}\(\alpha\)}{\alpha; \alpha; S : T_0; a : \alpha, x : [1] \vdash \text{M} \(\alpha \otimes \text{Queue} \(\alpha \otimes \alpha \)) \vdash E_{2} : T_1}
\]

D1:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha, x : \alpha \vdash C_4 \(\lambda p. \text{ret}\(\alpha, x, a\), C_0\)) \vdash \text{Queue} \alpha}{\alpha; \alpha; S : T_0; a : \alpha, x : \alpha \vdash \text{ret} C_4 \(\lambda p. \text{ret}\(\alpha, x, a\), C_0\)) \vdash E_{1} : T_1}
\]

D0:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha \vdash C1 \(\alpha\) \vdash \text{Queue} \alpha}{\alpha; \alpha; S : T_0; a : \alpha \vdash \text{ret}\(\alpha, C_1 a\) \vdash \text{M} \(\text{Queue} \alpha\)}
\]

D0.2:  
\[
\frac{\alpha; \alpha; S : T_0; a : \alpha \vdash E_{0.2} : T_1}{\alpha; \alpha; S : T_0; a : \alpha \vdash E_{0} : T_1}
\]
D0.1: \[
\begin{align*}
\alpha;::;S:T_0;0;::;\vdash & \uparrow1:\underline{\mathbb{M}}\mathbf{1} & \text{D0.2} \\
\alpha;::;S:T_0;0;::;q:\mathsf{Queue}\;\alpha,a:\alpha\vdash & E_0;\vdash T_2
\end{align*}
\]
Main derivation:
\[
\begin{align*}
\alpha;::;S:T_0;0;::;p:[2]1\vdash & [2]1 & \text{D0.1} \\
\alpha;::;S:T_0;0;::;p:[2]1,q:\mathsf{Queue}\;\alpha,a:\alpha\vdash & E_0;\vdash T_0 \\
\vdots;::;\vdash & \text{fixf.}\lambda p.\Lambda.\lambda q.E_0;\vdash T_0
\end{align*}
\]
\[\text{head : [3] } 1 \rightarrow \forall \alpha . \mathsf{Queue}\;\alpha \rightarrow \mathbb{M} \ 0\ \alpha\]
\[\text{head } \triangleq \lambda p.\Lambda.\lambda q.\]
\[\text{bind ht } = \text{headTail } p \quad q \;\text{in ret fst} (ht)\]

Listing 5: head function

\[E_0 = \text{bind ht } = \text{headTail } p \quad q \;\text{in } E_1\]
\[E_1 = \text{ret} (\text{fst} (ht))\]

\[T_0 = [3] \ 1 \rightarrow \forall \alpha . \mathsf{Queue}\;\alpha \rightarrow \mathbb{M} \ 0\ \alpha\]

D0:
\[
\begin{align*}
\alpha;::;::;q:\mathsf{Queue}\;\alpha,ht:(\alpha \otimes \mathsf{Queue}\;\alpha)\vdash & \text{fst} (ht):\alpha \\
\alpha;::;::;q:\mathsf{Queue}\;\alpha,ht:(\alpha \otimes \mathsf{Queue}\;\alpha)\vdash & \text{ret} (\text{fst} (ht)):\mathbb{M} \ 0\ \alpha \\
\alpha;::;::;q:\mathsf{Queue}\;\alpha,ht:(\alpha \otimes \mathsf{Queue}\;\alpha)\vdash & E_1: \mathbb{M} \ 0\ \alpha
\end{align*}
\]
Main derivation:
\[
\begin{align*}
\alpha;::;::;q:\mathsf{Queue}\;\alpha\vdash & \text{headTail } p \quad q : \mathbb{M} \ 0\ (\alpha \otimes \mathsf{Queue}\;\alpha) & \text{D0} \\
\alpha;::;::;q:\mathsf{Queue}\;\alpha\vdash & \text{bind ht } = \text{headTail } p \quad q \;\text{in } E_1 : \mathbb{M} \ 0\ \alpha \\
\alpha;::;::;p:[3]1,q:\mathsf{Queue}\;\alpha\vdash & E_0 : \mathbb{M} \ 0\ \alpha \\
\vdots;::;\vdash & \text{\lambda p.}\Lambda.\lambda q.E_0;\vdash T_0
\end{align*}
\]
\[\text{tail : [3] } 1 \rightarrow \forall \alpha . \mathsf{Queue}\;\alpha \rightarrow \mathbb{M} \ 0\ (\mathsf{Queue}\;\alpha)\]
\[\text{tail } \triangleq \lambda p.\Lambda.\lambda q.\]
\[\text{bind ht } = \text{headTail } p \quad q \;\text{in ret } \text{snd} (ht)\]

Listing 6: tail function

\[E_0 = \text{bind ht } = \text{headTail } p \quad q \;\text{in } E_1\]
\[E_1 = \text{ret} (\text{snd} (ht))\]

\[T_0 = [3] \ 1 \rightarrow \forall \alpha . \mathsf{Queue}\;\alpha \rightarrow \mathbb{M} \ 0\ (\mathsf{Queue}\;\alpha)\]

D0:
\[
\begin{align*}
\alpha;::;::;q:\mathsf{Queue}\;\alpha,ht:(\alpha \otimes \mathsf{Queue}\;\alpha)\vdash & \text{snd} (ht) : \mathsf{Queue}\;\alpha \\
\alpha;::;::;q:\mathsf{Queue}\;\alpha,ht:(\alpha \otimes \mathsf{Queue}\;\alpha)\vdash & \text{ret} (\text{snd} (ht)) : \mathbb{M} \ 0\ (\mathsf{Queue}\;\alpha) \\
\alpha;::;::;q:\mathsf{Queue}\;\alpha,ht:(\alpha \otimes \mathsf{Queue}\;\alpha)\vdash & E_1 : \mathbb{M} \ 0\ (\mathsf{Queue}\;\alpha)
\end{align*}
\]
Main derivation:
\[
\begin{align*}
\alpha;::;::;q:\mathsf{Queue}\;\alpha\vdash & \text{headTail } p \quad q : \mathbb{M} \ 0\ (\alpha \otimes \mathsf{Queue}\;\alpha) & \text{D0} \\
\alpha;::;::;q:\mathsf{Queue}\;\alpha\vdash & \text{bind ht } = \text{headTail } p \quad q \;\text{in } E_1 : \mathbb{M} \ 0\ (\mathsf{Queue}\;\alpha) \\
\alpha;::;::;p:[3]1,q:\mathsf{Queue}\;\alpha\vdash & E_0 : \mathbb{M} \ 0\ (\mathsf{Queue}\;\alpha) \\
\vdots;::;\vdash & \text{\lambda p.}\Lambda.\lambda q.E_0;\vdash T_0
\end{align*}
\]
headTail : [3] 1 \rightarrow \forall \alpha. Queue \alpha \rightarrow M \alpha (\alpha \odot Queue \alpha)
headTail \triangleq \text{fix } HT.\lambda p.\Lambda \lambda q.
\text{~} = \text{release } p \text{ in } - = \uparrow^1, \text{ ret }
case q \text{ of }
| C0 \mapsto \text{fix } x.x
| C1 \mapsto \text{ret } x, C0
| C2 \mapsto 
\text{bind } p' = \text{store }() \text{ in } \text{bind } p_0 = \text{store }() \text{ in }
\text{bind } x' = x \ p' \text{ in let } \langle f, m \rangle = x' \text{ in }
\text{ret } (\langle f, (C4 \ (\lambda p''. - = \text{release } p_0 \text{ in } - = \text{release } p'' \text{ in } \text{bind } p_r = \text{store }() \text{ in } HT. p_r [] m )\rangle)
| C3 \mapsto 
\text{bind } p' = \text{store }() \text{ in } \text{bind } p_0 = \text{store }() \text{ in }
\text{bind } x' = x \ p' \text{ in let } \langle f, m, r \rangle = x' \text{ in let } \langle f, m \rangle = f m \text{ in }
\text{ret } (\langle f, (C5 \ (\lambda p''. - = \text{release } p_0 \text{ in } - = \text{release } p'' \text{ in }
\text{bind } p'' = \text{store }() \text{ in } \text{bind } ht = HT. p'' [] m \text{ in ret } \langle ht, r \rangle)\rangle)
| C4 \mapsto 
\text{bind } p' = \text{store }() \text{ in } \text{bind } x' = x \ p' \text{ in let } \langle f, m \rangle = x' \text{ in let } \langle f, (C4 \ (\lambda (p', C2 \ (\lambda p''. \text{ret } \langle f, (o, C2) \rangle)))\rangle
| C5 \mapsto 
\text{bind } p' = \text{store }() \text{ in } \text{bind } x' = x \ p' \text{ in let } \langle f, m \rangle = x' \text{ in let } \langle f, m \rangle = f m \text{ in let } \langle f_1, f_2 \rangle = f \text{ in }
\text{ret } (\langle f_1, (C3 \ (\lambda p''. \text{ret } \langle f_2, m \rangle), r')\rangle)

Listing 7: head and tail function

E_{0.0} = \text{fix } HT. \lambda p. \Lambda \lambda q. E_{0.1}
E_{0.1} = - = \text{release } p \text{ in } - = \uparrow^1; E_{0.2}
E_{0.2} = \text{case } q \text{ of } | C0 \mapsto E_0 | C1 \mapsto E_1 | C2 \mapsto E_2 | C3 \mapsto E_3 | C4 \mapsto E_4 | C5 \mapsto E_5
E_0 = \text{fix } x.x
E_1 = \text{ret } x, C0
E_2 = \text{bind } p' = \text{store }() \text{ in } E_{2.0}
E_{2.0} = \text{bind } p_0 = \text{store }() \text{ in } E_{2.1}
E_{2.1} = \text{bind } x' = x \ p' \text{ in } E_{2.11}
E_{2.11} = \text{let } \langle f, m \rangle = x' \text{ in } E_{2.2}
E_{2.2} = \text{ret } (\langle f, (C4 \ (\lambda (p'', E_{2.3})\rangle))
E_{2.3} = - = \text{release } p_0 \text{ in } E_{2.4}
E_{2.4} = - = \text{release } p'' \text{ in } E_{2.5}
E_{2.5} = \text{bind } p_r = \text{store }() \text{ in } HT. p_r [] m
E_1 = \text{bind } p' = \text{store }() \text{ in } E_{3.0}
E_{3.0} = \text{bind } p_0 = \text{store }() \text{ in } E_{3.1}
E_{3.1} = \text{bind } x' = x \ p' \text{ in } E_{3.11}
E_{3.11} = \text{let } \langle f, m, r \rangle = x' \text{ in } E_{3.12}
E_{3.12} = \text{let } \langle f, m \rangle = f m \text{ in } E_{3.2}
E_{3.2} = \text{ret } (\langle f, E_{3.3})\rangle
E_{3.3} = C5 \ (\lambda p''. E_{3.31})
E_{3.4} = - = \text{release } p_0 \text{ in } E_{3.41}
E_{3.41} = \text{release } p'' \text{ in } E_{3.5}
E_{3.5} = \text{bind } p''' \text{ in } E_{3.6}
E_{3.6} = \text{bind } ht = HT. p''' [] m \text{ in ret } \langle ht, r \rangle
E_4 = \text{bind } p' = \text{store }() \text{ in } E_{4.1}
E_{4.1} = \text{bind } x' = x \ p' \text{ in } E_{4.2}
E_{4.2} = \text{let } \langle f, m \rangle = x' \text{ in } E_{4.3}
E_{4.3} = \text{let } \langle f_1, f_2 \rangle = f \text{ in } E_{4.4}
E_{4.4} = \text{ret } (\langle f_1, C2 \ (\lambda p''. \text{ret } \langle f_2, m \rangle)))
E_5 = \text{bind } p' = \text{store }() \text{ in } E_{5.1}
E_{5.1} = \text{bind } x' = x \ p' \text{ in } E_{5.2}
E_{5.2} = \text{let } \langle f, m, r \rangle = x' \text{ in } E_{5.3}
E₅.₃ = let \( \langle f, m \rangle = fm \) in E₅.₄
E₅.₄ = let \( \langle f₁, f₂ \rangle = f \) in E₅.₅
E₅.₅ = ret \( \langle f₁, \langle C₃ (λp''', \text{ret} \langle \langle f₂, m \rangle, r \rangle) \rangle \rangle \)

\[ T₀,₀ = [3] \, 1 \leadsto ∀α. \text{Queue} \, α \hookrightarrow M₀ (α \otimes \text{Queue} \, α) \]
\[ T₀,₂ = [1] \, 1 \leadsto M₀ (α \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₂₁ = M₀ (α \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₂₂ = (α \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₂₃ = \text{Queue} \, (α \otimes α) \]
\[ T₀,₃ = [0] \, 1 \leadsto M₀ ((α \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₃₁ = M₀ ((α \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₃₂ = ((α \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₃₃ = (α \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₃₄ = \text{Queue} \, (α \otimes α) \]
\[ T₀,₄ = [2] \, 1 \leadsto M₀ ((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₄₁ = M₀ ((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₄₁₁ = M₁ ((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₄₁₃ = M₃ ((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₄₂ = ((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \]
\[ T₀,₄₃ = (α \otimes α) \]
\[ T₀,₄₄ = \text{Queue} \, (α \otimes α) \]
\[ T₀,₅ = [1] \, 1 \leadsto M₀ (((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₅₁ = M₀ (((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₅₁₁ = M₁ (((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₅₁₂ = M₂ (((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₅₁₃ = M₃ (((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \otimes α) \]
\[ T₀,₅₂ = ((α \otimes α) \otimes \text{Queue} \, (α \otimes α)) \otimes α \]
\[ T₀,₅₃ = (α \otimes α) \]
\[ T₀,₅₄ = (α \otimes α) \]
\[ T₀,₅₅ = \text{Queue} \, (α \otimes α) \]
\[ T₀ = M₀ (α \otimes \text{Queue} \, α) \]
\[ T₁ = M₁ (α \otimes \text{Queue} \, α) \]
\[ T₂ = M₂ (α \otimes \text{Queue} \, α) \]

D₅.₅₁:

\[
\begin{align*}
α; \vdots; HT : T₀,₀ ; f₂ : α, m : T₀,₅₅, r : α, p''' : [0] \quad & \vdash \text{ret} \langle \langle f₂, m \rangle, r \rangle \rangle T₀,₃₁ \\
α; \vdots; HT : T₀,₀ ; f₂ : α, m : T₀,₅₅, r : α \vdash (λp''', \text{ret} \langle \langle f₂, m \rangle, r \rangle) : T₀,₃ \\
α; \vdots; HT : T₀,₀ ; f₂ : α, m : T₀,₅₅, r : α \vdash (C₃ (λp''', \text{ret} \langle \langle f₂, m \rangle, r \rangle)) : \text{Queue} \, α
\end{align*}
\]

D₅.₅:

\[
\begin{align*}
α; \vdots; HT : T₀,₀ ; f₁ : α \vdash f₁ : α & \quad \text{D₅.₅₁} \\
α; \vdots; HT : T₀,₀ ; f₁ : α, f₂ : α, m : T₀,₅₅, r : α \vdash \langle f₁, (C₃ (λp''', \text{ret} \langle \langle f₂, m \rangle, r \rangle)) \rangle : α \otimes \text{Queue} \, α \\
α; \vdots; HT : T₀,₀ ; f₁ : α, f₂ : α, m : T₀,₅₅, r : α \vdash \text{ret} \langle f₁, (C₃ (λp''', \text{ret} \langle \langle f₂, m \rangle, r \rangle)) \rangle : T₁
\end{align*}
\]

D₅.₄:

\[
\begin{align*}
α; \vdots; HT : T₀,₀ ; f : T₀,₅₄ \vdash f : T₀,₅₄ & \quad \text{D₅.₅} \\
α; \vdots; HT : T₀,₀ ; f : T₀,₅₄, m : T₀,₅₅, r : α \vdash \text{let} \langle \langle f₁, f₂ \rangle \rangle = f \text{ in } E₅.₅ : T₁ \\
α; \vdots; HT : T₀,₀ ; f : T₀,₅₄, m : T₀,₅₅, r : α \vdash E₅.₅ : T₁
\end{align*}
\]

D₅.₃:

\[
\begin{align*}
α; \vdots; HT : T₀,₀ ; f m : T₀,₅₃ \vdash f m : T₀,₅₃ & \quad \text{D₅.₄} \\
α; \vdots; HT : T₀,₀ ; f m : T₀,₅₃, r : α \vdash \text{let} \langle \langle f, m \rangle \rangle = f m \text{ in } E₅.₄ : T₁ \\
α; \vdots; HT : T₀,₀ ; f m : T₀,₅₃, r : α \vdash E₅.₃ : T₁
\end{align*}
\]
D5.2: \[
\frac{\alpha; ; ; HT : T_0.0 ; x' : T_0.52 \vdash x' : T_0.52}{D5.3}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; x' : T_0.52 \vdash \text{let} \langle fm, r \rangle = x' \text{ in } E_{0.3} : T_1}{\alpha; ; ; HT : T_0.0 ; x' : T_0.52 \vdash E_{5.2} : T_1}
\]
D5.1: \[
\frac{\alpha; ; ; HT : T_0.0 ; x : T_0.5, p : [1] \vdash x : T_0.51}{D5.2}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; x : T_0.5, p : [1] \vdash \text{bind} x = x' \text{ in } E_{5.2} : T_1}{\alpha; ; ; HT : T_0.0 ; x : T_0.5, p : [1] \vdash E_{5.1} : T_1}
\]
D5: \[
\frac{\alpha; ; ; HT : T_0.0 ; \vdash \text{store}() : M \{1 \} [1]}{D5.1}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; x : T_0.5 \vdash E_{5} : T_2}{\alpha; ; ; HT : T_0.0 ; x : T_0.5 \vdash E_{5} : T_2}
\]
D4.41: \[
\frac{\alpha; ; ; HT : T_0.0 ; f_2 : \alpha, m : T_0.44, p'' : [1] \vdash \text{ret} \langle f_2, m \rangle : T_{0.21}}{\alpha; ; ; HT : T_0.0 ; f_2 : \alpha, m : T_0.44 \vdash (\lambda p''. \text{ret} \langle f_2, m \rangle) : T_{0.2}}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; f_2 : \alpha, m : T_0.44 \vdash C2 (\lambda p''. \text{ret} \langle f_2, m \rangle) : \text{Queue} \alpha}{\alpha; ; ; HT : T_0.0 ; f_2 : \alpha, m : T_0.44 \vdash C2 (\lambda p''. \text{ret} \langle f_2, m \rangle) : \text{Queue} \alpha}
\]
D4.4: \[
\frac{\alpha; ; ; HT : T_0.0 ; f_1 : \alpha \vdash f_1 : \alpha}{D4.1}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; f_1 : \alpha, f_2 : \alpha, m : T_0.44 \vdash \langle \langle f_1, \lambda f_2 \rangle (\lambda p''. \text{ret} \langle f_2, m \rangle) \rangle : \alpha \odot \text{Queue} \alpha}{\alpha; ; ; HT : T_0.0 ; f_1 : \alpha, f_2 : \alpha, m : T_0.44 \vdash \lambda f_1, \lambda f_2 \langle \langle f_1, \lambda f_2 \rangle (\lambda p''. \text{ret} \langle f_2, m \rangle) \rangle : T_{0.0}}
\]
D4.3: \[
\frac{\alpha; ; ; HT : T_0.0 ; f : T_{0.43} \vdash f : T_{0.43}}{D4.4}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; f : T_{0.43}, m : T_0.44 \vdash \langle \langle f_1, f_2 \rangle \rangle = f \text{ in } E_{4.4} : T_0}{\alpha; ; ; HT : T_0.0 ; f : T_{0.43}, m : T_0.44 \vdash E_{4.4} : T_0}
\]
D4.2: \[
\frac{\alpha; ; ; HT : T_0.0 ; x' : T_{0.42} \vdash x' : T_{0.42}}{D4.3}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; x' : T_{0.42} \vdash \text{let} \langle f, m \rangle = x' \text{ in } E_{4.3} : T_0}{\alpha; ; ; HT : T_0.0 ; x' : T_{0.42} \vdash E_{4.3} : T_0}
\]
D4: \[
\frac{\alpha; ; ; HT : T_0.0 ; \vdash \text{store}() : M \{2 \} [2]}{D4.1}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; x : T_0.4 \vdash E_{4} : T_2}{\alpha; ; ; HT : T_0.0 ; x : T_0.4 \vdash E_{4} : T_2}
\]
D3.61: \[
\frac{\alpha; ; ; HT : T_0.0 ; r : \alpha, ht : T_{0.53} \vdash \text{ret} \langle ht, r \rangle : T_{0.51}}{\alpha; ; ; HT : T_0.0 ; r : \alpha, ht : T_{0.53} \vdash \text{ret} \langle ht, r \rangle : T_{0.51}}
\]
D3.6: \[
\frac{\alpha; ; ; HT : T_0.0 ; m : T_{0.54}, r : \alpha, p'' : [3] \vdash HT \ p'' : [m : M \{0 \} T_{0.53}]}{D3.61}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; m : T_{0.54}, r : \alpha, p'' : [3] \vdash \text{bind} ht = HT \ p'' : [m : M \{0 \} T_{0.53} \text{ in ret} \langle ht, r \rangle : T_{0.51}}{\alpha; ; ; HT : T_0.0 ; m : T_{0.54}, r : \alpha, p'' : [3] \vdash E_{3.6} : T_{0.51}}
\]
D3.5: \[
\frac{\alpha; ; ; HT : T_0.0 ; \vdash \text{store}() : [3] \{3\} \{1\}}{D3.6}
\]
\[
\frac{\alpha; ; ; HT : T_0.0 ; m : T_{0.54}, r : \alpha \vdash \text{bind} p'' = \text{store}() \text{ in } E_{3.6} : T_{0.5111}}{\alpha; ; ; HT : T_0.0 ; m : T_{0.54}, r : \alpha \vdash E_{3.5} : T_{0.513}}
\]
References


