TOKYO INSTITUTE OF TECHNOLOGY

MASTER THESIS

Translation Between Effect Instances and Multi-prompt Control Operators

Author:
Kazuki IKEMORI

Student Number:
21M30057

email:
ikemori.k.aa@m.titech.ac.jp

Supervisor:
Prof. Hidehiko MASUHARA

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Programming Research Group
Department of Mathematical and Computing Science

February 28, 2023

Declaration of Authorship

I, Kazuki IKEMORI, declare that this thesis titled, “Translation Between Effect Instances and Multi-prompt Control Operators” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

TOKYO INSTITUTE OF TECHNOLOGY

Abstract

School of Computing
Department of Mathematical and Computing Science

Master of Science

Translation Between Effect Instances and Multi-prompt Control Operators

by Kazuki IKEMORI

The labeled variations of algebraic effect handlers and control operators are an extended mechanism for handling different instances of the same computational effects, such as mutable cells. These mechanisms are supported in several functional programming languages, including Koka, Eff, OCaml, and Racket.

In previous work, it is shown that the expressive power between algebraic effect handlers and delimited control operators is equivalent. This is shown by the macro translations that convert from a program with one mechanism to the other and the other way around based on the syntactic structure. Using the relationship, we can develop the theory and implementation of one mechanism in terms of the other. As an example, we can derive a program transformation of one mechanism from that of the other. As a different example, we can simulate one mechanism in terms of the other without rebuilding the type system and evaluator.

While it is known that unlabeled algebraic effect handlers and delimited control operators are equally expressive, the relationship between labeled versions is not known. Our goal is to clarify the relationship between algebraic effect handlers with effect instances and delimited control operators with prompt tags. In particular, we show the equivalence of expressive power between the two calculi. Following the same approach of unlabeled versions, we define macro translations between effect instances and multi-prompt control operators and prove the type and meaning preservation properties. In the process, we define the novel type system of delimited control operators with prompt tags and prove its type soundness.

Acknowledgements

First of all, I would like to express gratitude to my supervisor, Professor Hidehiko Masuhara, for providing huge support. What I appreciate the most is that he always stood aside of me when I went down a blind alley and gave me sharp and solid advice. He taught me not only how to get a mindset of a good researcher and being one, but also how to do decent research.

I would also like to extend my appreciation to Professor Youyou Cong for guiding me into the long journey of functional programming languages and formal verification. She gave me valuable discussions and thick support that I always looked back to when I got lost. In particular, she introduced type-and-effect systems to me, which became my precious treasure that has opened the door to formal verification area. Moreover, she provided me the opportunities to meet researchers all over the world and that brought good (side) effects and inspirations.

I would like to express my gratitude to Daan Leijen who is the collaborator of our TFP'22 paper. He gave me many supports and opportunities to discuss our research and introduced the Koka language that is a cool programming language with algebraic effect handlers to me.

I am grateful to the members of the Programming Research Group, especially Yudai Tanabe and Yusuke Izawa, for supports and valuable comments. They gave me many comments and the supports that have practical tips. Unfortunately, we couldn't spend much time for discussions together due to the COVID-19 situations, yet I appreciate them for sparing time to provide perspectives on my thesis.

I would like to thank my parents for respecting and supporting my decisions no matter how hard they are with unconditional love.

Lastly, I thank my pets, Jam and Berry. They maintained my health.

Contents

List of Figures

Chapter 1

Introduction

Programmers need mechanisms to perform and handle computational effects such as exceptions and mutable cells. These mechanisms need to differentiate instances of the same computational effects.

In functional programming languages, there are several uniform mechanisms for handling computational effects. One of them is the algebraic effect handlers [Plotkin and Pretnar, 2009; Pretnar, 2015], which is a generalization of exception handlers in that the computation can resume after performing an effect. Another mechanism is the delimited control operators [Felleisen, 1988; Danvy and Filinski, 1990], which have a longer history than the effect handlers.

When these mechanisms are extended with *labels*, they can further express different instances of the same effect. These labels are called *effect instances* in the context of the effect handlers [Bauer and Pretnar, 2013; Biernacki et al., 2020; Xie, Cong, et al., 2022], and *prompt tags* in the context of control operators [Felleisen, 1988; Danvy and Filinski, 1990; Gunter, Rémy, and Riecke, 1995]. There are several programming languages that support these mechanisms. For instance, Koka [Xie, Cong, et al., 2022] and an older version of Eff [Bauer and Pretnar, 2013] support effect handlers with effect instances, and OCaml [Kiselyov, Shan, and Sabry, 2006] and Racket[1] support tagged (aka multi-prompt) control operators.

Our goal is to clarify the relationship between algebraic effect handlers with effect instances and delimited control operators with prompt tags. In particular, we show the equivalence of expressive power between these calculi (i.e., those with labels). In the case of unlabeled algebraic effect handlers and delimited control operators, it is known that the two mechanisms are equally expressive [Piróg, Polesiuk, and Sieczkowski, 2019]. This is shown by the existence of macro translations [Felleisen, 1991], which convert between the two mechanisms based on the syntactic structure. Following the approach of Pirog et al., we define macro translations between the two calculi and prove the type and meaning preservation properties. By doing this, we can develop the theory and implementation of one mechanism in terms of the other.

[1] https://docs.racket-lang.org/reference/cont.html

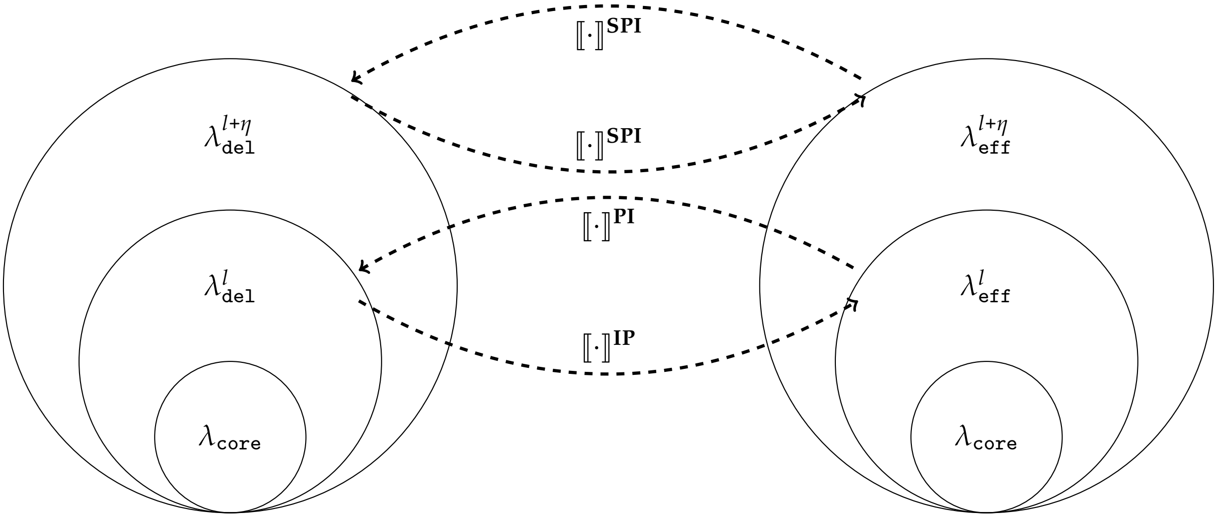
$\llbracket\cdot\rrbracket^{\mathbf{SPI}}$
$\lambda^{l+\eta}_{\texttt{del}}$
$\llbracket\cdot\rrbracket^{\mathbf{SPI}}$
$\lambda^{l+\eta}_{\texttt{eff}}$
$\llbracket\cdot\rrbracket^{\mathbf{PI}}$
$\lambda^{l}_{\texttt{del}}$
$\lambda^{l}_{\texttt{eff}}$
$\llbracket\cdot\rrbracket^{\mathbf{IP}}$
$\lambda_{\texttt{core}}$
$\lambda_{\texttt{core}}$

FIGURE 1.1: Overview of contributions

In this thesis, we make the following contributions, as depicted in Figure 1.1.

- We formalize two calculi $\lambda^{l}_{\texttt{eff}}$ and $\lambda^{l}_{\texttt{del}}$ that have static effect instances and prompt tags respectively, and show the equivalence of expressive power between them through macro translations $\llbracket\cdot\rrbracket^{\mathbf{PI}}$ and $\llbracket\cdot\rrbracket^{\mathbf{IP}}$.

- We formalize two calculi $\lambda^{l+\eta}_{\texttt{eff}}$ and $\lambda^{l+\eta}_{\texttt{del}}$ that have dynamically generated effect instances and prompt tags respectively, and show the equivalence of expressive power between them through macro translations $\llbracket\cdot\rrbracket^{\mathbf{SPI}}$ and $\llbracket\cdot\rrbracket^{\mathbf{SIP}}$.

- In the process, we define the novel type system of delimited control operators with prompt tags and prove type soundness.

The rest of the thesis is structured as follows. In Chapter 2, we show examples of algebraic effect handlers with effect instances and delimited control operators with prompt tags. In Chapter 3, we formalize $\lambda_{\texttt{core}}$, a calculus that has polymorphism, subtyping, and a row-based effect system. Then in Chapters 4 and 5, we formalize various extensions of $\lambda_{\texttt{core}}$ and discuss their relationships. We discuss related work in Chapter 6 and we conclude the thesis in Chapter 7.

The proofs of our calculi can be found in Appendices A - G.

Chapter 2

Background

2.1 Algebraic Effect Handlers

Algebraic effect handlers [Pretnar, 2015; Plotkin and Pretnar, 2009] are a generalization of exception handlers. The mechanism consists of operations that cause effects and handlers that determine the interpretation of operations. When an operation performs an effect, it is handled by the innermost handler surrounding it. Then, the handler executes the corresponding operation clause.

As an example, let us consider a reader effect that represents a read-only state.

$$
\begin{array}{lll}
 & \textbf{handle do } () \textbf{ with } \{x, r.r\ 1; x.x + 1\} & \\
\rightarrow^* & r\ 1 & (r = \lambda z.\textbf{handle } z \textbf{ with } \{x, r.r\ 1; x.x + 1\}) \\
\rightarrow & \textbf{handle } 1 \textbf{ with } \{x, r.r\ 1; x.x + 1\} & \\
\rightarrow & x + 1 & (x = 1) \\
\rightarrow & 2 &
\end{array}
$$

The reduction goes as follows. First, the operation **do** () is handled by the surrounding handler. The handler executes the corresponding operation clause $x, r.r\ 1$, which says it resumes evaluation with the value 1 using the continuation r. The continuation r represents the rest of the computation from the operation call of the handled expression to the innermost handler. The value 1 is the current state of the read-only state. Next, the application $r\ 1$ reduces to **handle** 1 **with** $\{x, r.r\ 1; x.x + 1\}$. Then, the handler executes the return clause $x.x + 1$, which specifies the post-processing of the value that the handled expression reduces to. Therefore, the whole program reduces to 2.

Using effect handlers, one can define many other computational effects, including exceptions, mutable cells, and non-deterministic effect.

2.2 Delimited Control Operators

Delimited control operators [Gunter, Rémy, and Riecke, 1995; Felleisen, 1988; Filinski, 1994] are a more traditional tool for encoding effects. The mechanism consists of control operators that capture the current continuations and delimiters that delimit continuations. There are several variants of delimited control operators. Among them, we focus on the $\texttt{shift}_0$ and `dollar` ($\langle \cdot \mid \cdot \rangle$) operators. The $\texttt{shift}_0$ operators captures the continuation up to the innermost `dollar` operator surrounding it. The `dollar` operator delimits the extent of the continuation captured by $\texttt{shift}_0$.

As an example, let us consider the following expression, which again encodes a reader effect.

$$
\begin{array}{lll}
 & \langle \mathbf{shift}_0\, k.\, k \mid x.\, x + 1\rangle\ 1 & \\
\rightarrow^* & k\ 1 & (k = \lambda z.\langle z \mid x.\, x + \mathtt{1}\rangle) \\
\rightarrow & \langle 1 \mid x.\, x + \mathtt{1}\rangle & \\
\rightarrow & x + \mathtt{1} & (x = \mathtt{1}) \\
\rightarrow & \mathtt{2} &
\end{array}
$$

The reduction goes as follows. First, the $\mathtt{shift}_0$ operator captures the continuation k up to the innermost `dollar` operator, and executes the body k. Second, the application k `1` reduces to $\langle 1 \mid x.\, x + 1\rangle$. Then, the `dollar` operator executes the return clause $x.x + \mathtt{1}$, which specifies the post-processing of the value that the handled expression reduces to since the body of the `dollar` expression is a value `1`. Thus, the whole program reduces to `2`.

2.3 Algebraic Effect Handlers with Effect Instances

Plain algebraic effect handlers are inconvenient to handle the different instances of the same effect because any operation is handled by the innermost corresponding handler. In other words, there is no way to handle an operation using an outer handler. Let us explain the limitation through the following expression, which has two operations and two different handlers for the same reader effect.

$$
\begin{array}{ll}
 & \mathbf{handle}\ \mathbf{handle}\ \mathbf{do}\ \mathtt{()} + \mathbf{do}\ \mathtt{()}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \\
\rightarrow^* & \mathbf{handle}\ r\ 1\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \quad (r = \lambda z.\mathbf{handle}\ z + \mathbf{do}\ \mathtt{()}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}) \\
\rightarrow & \mathbf{handle}\ \mathbf{handle}\ 1 + \mathbf{do}\ \mathtt{()}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \\
\rightarrow^* & \mathbf{handle}\ r\ 1\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \quad (r = \lambda z.\mathbf{handle}\ 1 + z\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}) \\
\rightarrow^* & \mathbf{handle}\ \mathbf{handle}\ 1 + 1\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \\
\rightarrow^* & \mathtt{2}
\end{array}
$$

In the above reduction steps, both calls **do** `()` are handled by the innermost handler, whose operation clause $x, r.r\ \mathtt{1}$ is executed. Hence, the expression is reduced to `2` $(= \mathtt{1} + \mathtt{1})$ instead of `3` $(= \mathtt{1} + \mathtt{2})$. There is a restriction that the interpretation of operations is not given by the outer handler.

One solution to the limitation is to extend algebraic effect handlers with *effect instances* [Bauer and Pretnar, 2013; Biernacki et al., 2020; Xie, Cong, et al., 2022]. This allows us to associate every operation with a specific handler, which is not necessarily the innermost one.

Below is an expression that uses effect instances l and l', which associates the operations $\mathbf{do}\langle l\rangle$ `()` and $\mathbf{do}\langle l'\rangle$ `()` with the inner and outer handler, respectively.

$$
\begin{array}{ll}
 & \mathbf{handle}\langle l'\rangle\ \mathbf{handle}\langle l\rangle\ \mathbf{do}\langle l\rangle\ \mathtt{()} + \mathbf{do}\langle l'\rangle\ \mathtt{()}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \\
\rightarrow^* & \mathbf{handle}\langle l'\rangle\ r\ \mathtt{1}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \quad (r = \lambda z.\mathbf{handle}\langle l\rangle\ z + \mathbf{do}\langle l'\rangle\ \mathtt{()}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}) \\
\rightarrow & \mathbf{handle}\langle l'\rangle\ \mathbf{handle}\langle l\rangle\ \mathtt{1} + \mathbf{do}\langle l\rangle\ \mathtt{()}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \\
\rightarrow^* & r\ \mathtt{2} \quad (r = \lambda z.\mathbf{handle}\langle l'\rangle\ \mathbf{handle}\langle l\rangle\ \mathtt{1} + z\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\}) \\
\rightarrow & \mathbf{handle}\langle l'\rangle\ \mathbf{handle}\langle l\rangle\ \mathtt{1} + \mathtt{2}\ \mathbf{with}\ \{x, r.r\ \mathtt{1}; x.x\}\ \mathbf{with}\ \{x, r.r\ \mathtt{2}; x.x\} \\
\rightarrow^* & \mathtt{3}
\end{array}
$$

In the above reduction steps, the operation $\mathbf{do}\langle l\rangle$ () is handled by the inner handler, and the operation $\mathbf{do}\langle l'\rangle$ () is handled by the outer handler. Hence, the whole expression is reduced to 3.

2.4 Delimited Control Operators with Prompt Tags

Similar to the plain effect handlers, plain delimited control operators ($\texttt{shift}_0$/dollar) are insufficient to write a complex control flow program, because any $\texttt{shift}_0$ operator captures the continuation up to the innermost dollar operator. In other words, there is no way to capture the continuation up to the outer dollar operator. Let us explain the limitation through the following expression, which has two $\texttt{shift}_0$ operators and two dollar operators.

$$\begin{array}{lll}
 & \langle\langle \mathbf{shift}_0\, k.\, k + \mathbf{shift}_0\, k.\, k \mid x.\, x\rangle\ \texttt{1} \mid x.\, \lambda y.x\rangle\ \texttt{2} & \\
\to^* & \langle k\ \texttt{1} \mid x.\, \lambda y.x\rangle\ \texttt{2} & (k = \lambda z.\langle z + \mathbf{shift}_0\, k.\, k \mid x.\, x\rangle) \\
\to & \langle\langle 1 + \mathbf{shift}_0\, k.\, k \mid x.\, x\rangle \mid x.\, \lambda y.x\rangle\ \texttt{2} & \\
\to^* & \langle k\ \texttt{1} \mid x.\, \lambda y.x\rangle\ \texttt{2} & (k = \lambda z.\langle 1 + \mathbf{shift}_0\, k.\, k \mid x.\, x\rangle) \\
\to & \langle\langle 1 + 1 \mid x.\, x\rangle \mid x.\, \lambda y.x\rangle\ \texttt{2} & \\
\to^* & \texttt{2} &
\end{array}$$

In the above reduction steps, both $\texttt{shift}_0$ operators capture the continuation up to the innermost dollar operator and the application $k\,1$ is executed. Hence, the expression is reduce to 2 $(= \texttt{1} + \texttt{1})$ instead of 3 $(= \texttt{1} + \texttt{2})$. There is a restriction that $\texttt{shift}_0$ operators cannot capture the continuation up to the outer dollar operator.

One solution of the limitation is to extend delimited control operators with *prompt tags* [Gunter, Rémy, and Riecke, 1995; Kiselyov, Shan, and Sabry, 2006]. The idea is to associate $\texttt{shift}_0$ operators with the corresponding dollar operators using prompt tags, just like how we extended algebraic effect handlers with effect instances.

Below is a program that uses an prompt tags l and l', which associates the control operators $\mathbf{shift}_0\langle l\rangle\ k.\, k$ and $\mathbf{shift}_0\langle l'\rangle\ k.\, k$ with the inner and outer dollar operators.

$$\begin{array}{lll}
 & \langle\langle \mathbf{shift}_0\langle l\rangle\ k.\, k + \mathbf{shift}_0\langle l'\rangle\ k.\, k \mid x.\, x\rangle_l\ \texttt{1} \mid x.\, x\rangle_{l'}\ \texttt{2} & \\
\to^* & \langle k\ \texttt{1} \mid x.\, x\rangle_{l'}\ \texttt{2} & (k = \lambda z.\langle z + \mathbf{shift}_0\langle l'\rangle\ k.\, k \mid x.\, x\rangle_l) \\
\to & \langle\langle 1 + \mathbf{shift}_0\langle l'\rangle\ k.\, k \mid x.\, x\rangle_l \mid x.\, x\rangle_{l'}\ \texttt{2} & \\
\to^* & k\ \texttt{2} & (k = \lambda z.\langle\langle 1 + z \mid x.\, x\rangle_l \mid x.\, x\rangle_{l'}) \\
\to & \langle\langle 1 + 2 \mid x.\, x\rangle \mid x.\, x\rangle & \\
\to^* & \texttt{3} &
\end{array}$$

In the above reduction steps, the $\mathbf{shift}_0\langle l\rangle\ k.\, k$ operator captures the continuation up to the inner dollar operator, and the $\mathbf{shift}_0\langle l'\rangle\ k.\, k$ operator captures the continuation up to the outer dollar operator. Hence, the whole expression is reduced to 3.

2.5 Macro Translation between Algebraic Effect handlers and Delimited Control Operators

The concept of macro expressibility of Felleisen, 1991 allows us to compare the expressive power between different languages. Intuitively, macro expressibility is defined as follows. Suppose we have two languages, L and L_+. Here, L_+ is an extension of L with some additional feature. Now, if there exists a syntax-directed, meaning-preserving translation from L_{+} to L, we say that L_{+} is macro expressible by L. We call the translation a macro translation, and we view its existence as a witness that the two languages have the equal expressive power. Thus, we can show the equivalence between two calculi by defining a pair of macro translations between them.

There are several studies that use the macro expressibility to compare the relative expressiveness of algebraic effect handlers and delimited control operators. Forster et al., 2016 show that the expressive power between algebraic effect handlers and delimited control operators is equivalent in an untyped setting. Piróg, Polesiuk, and Sieczkowski, 2019 extend the results of Forster et al., 2016 to a typed setting.

In this thesis, we extend the macro translations of Pirog et al. with effect instances and prompt tags. In the following chapters, we formalize individual calculi and discuss the properties of our macro translations.

Chapter 3

Core Calculus

In this chapter, we present $\lambda_{\texttt{core}}$, which serves as the basis of the effectful calculi discussed in later chapters. The core calculus has subtyping, polymorphism and row based effect systems [Leijen, 2014; Leijen, 2017; Xie, Brachthauser, et al., 2020; Hillerström and Lindley, 2016; Convent et al., 2020; Piróg, Polesiuk, and Sieczkowski, 2019]. Note that $\lambda_{\texttt{core}}$ is based on the core language of Piróg, Polesiuk, and Sieczkowski, 2019. In the following sections, we first show a syntax of kinds, types and expressions (Sect. 3.1). Next, we show kinding rules (Sects. 3.2). And then, we show equivalence rules, subtyping rules and typing rules (Sects. 3.3-3.5). Lastly, we show the operational semantics (Sect. 3.6). We highlight the changes in the syntax and typing from Piróg, Polesiuk, and Sieczkowski, 2019 with gray background.

3.1 Syntax

Kind	κ	$::=$	$\mathbf{T}$	(value type)
		$\mid$	$\mathbf{E}$	(effect type)
		$\mid$	$\mathbf{R}$	(effect row type)
Type	σ, τ	$::=$	α	(type variable)
		$\mid$	$\sigma \to_{\rho} \sigma$	(function type)
		$\mid$	$\forall \alpha :: \kappa . \sigma$	(quantified type)
Effect row	ρ	$::=$	ι	(empty effect row)
		$\mid$	$\epsilon \cdot \rho$	(extended effect row)
Expression	e	$::=$	v	(value)
		$\mid$	$e\ e$	(application)
Value	v	$::=$	x	(variable)
		$\mid$	$\lambda x.e$	(lambda abstraction)
Type variable env	Δ	$::=$	$\emptyset \mid \Delta, \alpha :: \kappa$	
Variable env	Γ	$::=$	$\emptyset \mid \Gamma,\ x : \sigma$	
Label env	Σ	$::=$	$\emptyset \mid \Sigma, l$	

Type variables $\ni \alpha, \beta, \ldots$ Expression variables $\ni x, y, \ldots$ Labels $\ni l, l_1, \ldots$

FIGURE 3.1: Syntax of $\lambda_{\texttt{core}}$

Kinds, Types and Effect rows. We define the syntax of kinds, types, and effect rows of $\lambda_{\texttt{core}}$ in Figure 3. Similar to the core language of Pirog et al., kinds include value types **T**, effect types **E**, and effect row types **R**. Types include type variables α, function types $\sigma \to_\rho \sigma$, and quantified types $\forall \alpha :: \kappa . \sigma$. A function type $\sigma_1 \to_\rho \sigma_2$ says the type of input is σ_1, the type of output is σ_2, and the body of a function may perform computational effects ρ. Following [Biernacki et al., 2020; Xie, Brachthauser, et al., 2020], we define an effect row as either an empty row ι, a type variable α with an effect row kind **R**, or an extension $\epsilon \cdot \rho$ of an effect row ρ with an effect ϵ. An effect includes a label, which is a uniform representation effect instances and prompt tags. Unlike on Pirog et al., an effect row cannot be extended with a type variable that has kind **E**. We will discuss the reason in Section 4.1.4. As a convention, we will use α and β for type variables with a value type kind.

Expressions and Values. We define the syntax of expressions and values of $\lambda_{\texttt{core}}$ in Figure 3. Similar to the core language of Pirog et al., expressions include applications $e\ e$ and values v. Values also include variables x and lambda abstractions $\lambda x.e$. As a convention, we will use x and y for variables.

3.2 Kinding Rules

$$\boxed{\Delta \mid \Sigma \vdash \tau :: \kappa}$$

$$\frac{\alpha :: \kappa \in \Delta}{\Delta \mid \Sigma \vdash \alpha :: \kappa}\ [\textsc{KVar}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Sigma \vdash \rho :: \mathbf{R} \qquad \Delta \mid \Sigma \vdash \tau_2 :: \mathbf{T}}{\Delta \mid \Sigma \vdash \tau_1 \to_\rho \tau_2 :: \mathbf{T}}\ [\textsc{KArrow}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau :: \mathbf{T}}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa . \tau :: \mathbf{T}}\ [\textsc{KGen}] \qquad \frac{}{\Delta \mid \Sigma \vdash \iota :: \mathbf{R}}\ [\textsc{KEmpty}]$$

$$\frac{\Delta \mid \Sigma \vdash \epsilon :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \rho :: \mathbf{R}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho :: \mathbf{R}}\ [\textsc{KRow}]$$

FIGURE 3.2: Kinding rules of $\lambda_{\texttt{core}}$

We define the kinding rules of $\lambda_{\texttt{core}}$ in Figure 3.2. We use a kinding judgment of the form $\Delta \mid \Sigma \vdash \tau :: \kappa$. The judgment states that a type τ has kind κ under type variable environment Δ and label environment Σ.

Kinding rules are identical to Pirog et al. The KGEN, KVAR, and KARROW rules are standard. The KEMPTY rule states that an empty effect row has kind **R**. The KROW rule states that an extension $\epsilon \cdot \rho$ has kind **R**, where ϵ has kind **E** and ρ has kind **R**.

3.3 Equivalence Rules

$$\boxed{\Delta \mid \Sigma \vdash \sigma \equiv \sigma'}$$

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa}{\Delta \mid \Sigma \vdash \sigma \equiv \sigma}\ [\textsc{ERefl}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_2^1 \equiv \tau_1^1 \qquad \Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \tau_1^2 \equiv \tau_2^2}{\Delta \mid \Sigma \vdash \tau_1^1 \rightarrow_{\rho_1} \tau_2^1 \equiv \tau_1^2 \rightarrow_{\rho_2} \tau_2^2}\ [\textsc{EArrow}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_1 \equiv \tau_2}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa.\tau_1 \equiv \forall \alpha :: \kappa.\tau_2}\ [\textsc{EGen}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \equiv \epsilon \cdot \rho_2}\ [\textsc{ERow}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon_1 :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \epsilon_2 :: \mathbf{E} \qquad \lceil \epsilon_1 \rceil \neq \lceil \epsilon_2 \rceil}{\Delta \mid \Sigma \vdash \epsilon_1 \cdot \epsilon_2 \cdot \rho_1 \equiv \epsilon_2 \cdot \epsilon_1 \cdot \rho_2}\ [\textsc{ESwap}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \rho_2 \equiv \rho_3}{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_3}\ [\textsc{ETrans}]$$

FIGURE 3.3: Equivalence rules of $\lambda_{\texttt{core}}$

We define the equivalence rules of $\lambda_{\texttt{core}}$ in Figure 3.3. We use an equivalence judgment of the form $\Delta \mid \Sigma \vdash \sigma \equiv \sigma'$. The judgment states that type σ and σ' are equivalent under type variable environment Δ and label environment Σ. Note that these rules do not exist in the calculus of Pirog et al.

The ESwap rule is the most interesting rule. It checks if effects ϵ_1 and ϵ_2 have kind of $\mathbf{R}$ and effect rows ρ_1 and ρ_2 are equivalent under type variable environment Δ and label environment Σ. The meta function $\lceil \cdot \rceil$ extracts a label from an effect ϵ. It also checks if labels $\lceil \epsilon_1 \rceil$ and $\lceil \epsilon_2 \rceil$ are distinct. Then, it derives that swapped effects $\epsilon_1 \cdot \epsilon_2 \cdot \rho_1$ and $\epsilon_2 \cdot \epsilon_2 \cdot \rho_2$ are equivalent under type variable environment Δ and label environment Σ. We will discuss the reason why we can only apply the ESwap rule to effects that have different heads in Section 4.1.4.

The other rules are standard. The ERefl and ETrans rules correspond to reflexivity and transitivity, respectively. The EArrow rule states that two arrow types are equivalent if they have the same input type, output type, and effect row. The EGen states that types τ_1 and τ_2 are equivalent under type variable environment $\Sigma, \alpha :: \kappa$ extended with $\alpha :: \kappa$. The ERow rule concludes that extensions of effect rows ρ_1 and rho_2 by epsilon are equivalent if ρ_1 and ρ_2 are equivalent.

3.4 Subtyping Rules

$$\boxed{\Delta \mid \Sigma \vdash \sigma <: \sigma'}$$

$$\frac{\Delta \mid \Sigma \vdash \sigma_1 \equiv \sigma_2}{\Delta \mid \Sigma \vdash \sigma_1 <: \sigma_2}\ [\textsc{SRefl}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_1^1 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \tau_2^1 <: \tau_2^2}{\Delta \mid \Sigma \vdash \tau_1^1 \rightarrow_{\rho_1} \tau_2^1 <: \tau_1^2 \rightarrow_{\rho_2} \tau_2^2}\ [\textsc{SArrow}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_1 <: \tau_2}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa . \tau_1 <: \forall \alpha :: \kappa . \tau_2}\ [\textsc{SGen}] \qquad \frac{\Delta \mid \Sigma \vdash \rho :: \mathbf{R}}{\Delta \mid \Sigma \vdash \iota <: \rho}\ [\textsc{SEmpty}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 <: \epsilon \cdot \rho_2}\ [\textsc{SRow}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \rho_2 <: \rho_3}{\Delta \mid \Sigma \vdash \rho_1 <: \rho_3}\ [\textsc{STrans}]$$

FIGURE 3.4: Subtyping rules of $\lambda_{\texttt{core}}$

We now define the subtyping rules of $\lambda_{\texttt{core}}$ in Figure 3.4. We use a subtyping judgment of the form $\Delta \mid \Sigma \vdash \sigma <: \sigma'$. The judgment states that types σ and σ' have subtyping relation under type variable environment Δ and label environment Σ.

The subtyping rules are based on Pirog et al. The SARROW states that their input types, output types, and effect rows are in the subtyping relation; note that the relation is contravariant in the input types. The SGEN rule states that the type τ_2 subsumes the type τ_1 under type variable environment Δ extended with a type variable $\alpha :: \kappa$. The SEMPTY rule concludes that the effect row ρ subsumes all empty effect rows. The SROW rule concludes that extensions of effect rows ρ_1 and ρ_2 by epsilon are in the subtyping relation if ρ_1 and ρ_2 are in the subtyping relation. The STRANS rule is for transitivity.

One thing to note here is that the SREFL rule has an extra premise compared to the original rule by Pirog et al. This allow us to derive, for instance, $\Delta \mid \Sigma \vdash \epsilon_2 \cdot \epsilon_1 \cdot \rho <: \epsilon_1 \cdot \epsilon_2 \cdot \rho$ using $\Delta \mid \Sigma \vdash \epsilon_2 \cdot \epsilon_1 \cdot \rho \equiv \epsilon_1 \cdot \epsilon_2 \cdot \rho$, which intuitively holds. Without the additional premise, we cannot derive the above subtyping relation as the original rules of Pirog et al. do not allow swapping of effects.

3.5 Typing Rules

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho}$$

$$\frac{x : \tau \in \Gamma}{\Delta \mid \Gamma \mid \Sigma \vdash x : \tau / \iota}\;[\textsc{Var}]$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e_1 : \tau_1 \rightarrow_{\rho} \tau / \rho \qquad \Delta \mid \Gamma \mid \Sigma \vdash e_2 : \tau_1 / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash e_1\, e_2 : \tau / \rho}\;[\textsc{App}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \tau_1 \rightarrow_{\rho} \tau_2 / \iota}\;[\textsc{Abs}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Gamma \mid \Sigma \vdash e : \tau / \iota \qquad \kappa \in \{\mathbf{T}, \mathbf{R}\}}{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall \alpha :: \kappa . \tau / \iota}\;[\textsc{Gen}]$$

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \forall \alpha :: \kappa . \tau / \rho \qquad \kappa \in \{\mathbf{T}, \mathbf{R}\}}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau\{\sigma / \alpha\} / \rho}\;[\textsc{Inst}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \tau_1 / \rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau_2 / \rho_2}\;[\textsc{Sub}]$$

FIGURE 3.5: Typing rules of $\lambda_{\texttt{core}}$

We define the typing rules of $\lambda_{\texttt{core}}$ in Figure 3.5. We use a typing judgment of the form $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$. The judgment states that expression e has type τ under type variable environment Δ, variable environment Γ, and label environment Σ, and may perform effects ρ.

The typing rules are based on Pirog et al. The VAR, APP, ABS, INST, and SUB are completely standard and almost identical to Pirog et al. The only difference from their original typing rules is that the kind of the type variable in the GEN and INST rules is restricted to **T** or **R**. Note that the APP rule requires the function, the argument, and the function's body to have the same effect.

3.6 Operational Semantics

Evaluation context

Pure evaluation context	F	$::=$	$\square \mid e\ F \mid F\ v$
Evaluation context	E	$::=$	$\square \mid e\ E \mid E\ v$

Reduction rules

$$\frac{}{(\lambda x.e)\ v \mapsto e\{v/x\}}\ [\text{BETA}] \qquad \frac{e \mapsto e'}{E[e] \rightarrow E[e']}\ [\text{STEP}]$$

FIGURE 3.6: Evaluation contexts and Reduction rules of $\lambda_{\texttt{core}}$

We define the operational semantics of $\lambda_{\texttt{core}}$ in Figure 3.6. The semantics is based on the call-by-value evaluation strategy, and is associated with an inductive definition of evaluation contexts. The definition is the completely standard and identical to Pirog et al.'s. As the reduction rules, we define the standard BETA rule for reducing an application, as well as the STEP rule to reduce an expression in an evaluation context.

Chapter 4

Static Effect Instances and Prompt Tags

In this chapter, we present two extensions of the core calculus and a pair of macro translations between the two extended calculi. One of the calculus is called $\lambda^{l}_{\texttt{eff}}$, which features algebraic effect handlers and effect instances. The other is called $\lambda^{l}_{\texttt{del}}$, which features delimited control operators and prompt tags. Note that we treat both effect instances and prompt tags as second-class values (i.e., we cannot pass them to functions or return them from functions). In the following sections, we first give a formalization of the two calculi (Sects. 4.1-4.2) We then define the macro translation and prove the type and meaning preservation properties (Sect. 4.3).

4.1 $\lambda^{l}_{\texttt{eff}}$: Algebraic Effect Handlers with Static Effect Instances

In this section, we define $\lambda^{l}_{\texttt{eff}}$, which is a calculus that extends $\lambda_{\texttt{core}}$ with algebraic effect handlers and effect instances. In the following sections, we define the syntax, the typing rules, and the operational semantics by extending those of $\lambda_{\texttt{core}}$ (Sects. 4.1.1-4.1.3). Next, we explain the reason why the ESWAP rule cannot be applied to all effect (Sect. 4.1.4). Lastly, we show type soundness (Sect. 4.1.5). Note that $\lambda^{l}_{\texttt{eff}}$ is based on the calculi of Pirog et al.

4.1.1 Syntax

$$
\begin{array}{llcll}
\text{Kind} & \kappa & ::= & \ldots & \\
 & & \mid & \mathbf{L} & \text{(label type)} \\
\text{Type} & \sigma,\tau & ::= & \cdots & \\
 & & \mid & \ell & \text{(label)} \\
\text{Effect} & \epsilon & ::= & \exists_{\ell}\ \Delta'.\tau_1 \Rightarrow \tau_2 & \\
\text{Label} & \ell & ::= & l & \text{(static label)} \\
\text{Expression} & e & ::= & \ldots & \\
 & & \mid & \mathbf{do}\langle \ell \rangle\ v & \text{(do)} \\
 & & \mid & \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} & \text{(handle)}
\end{array}
$$

FIGURE 4.1: Syntax of $\lambda^{l}_{\texttt{eff}}$ (extensions)

Kinds, Types and Effects We define the syntax of kinds, types, and effects of $\lambda^{l}_{\texttt{eff}}$ in Figure 4.1. For brevity, we only show the changes to the $\lambda_{\texttt{core}}$ syntax. Kinds are extended with label type **L**. Types are extended with label types ℓ. Effects only take the form $\exists_{\ell}\ \Delta'.\tau_1 \Rightarrow \tau_2$ that is generalized by type variable environment Δ'. It means that the types of an operation's argument and return value are τ_1 and τ_2 and it must be handled by a handler whose label is ℓ. Labels are chosen from a set of statically defined labels Σ.

Expressions We define the syntax of expressions of $\lambda^{l}_{\texttt{eff}}$ in Figure 4.1. Expressions include operations $\mathbf{do}\langle\ell\rangle\ v$ and handlers $\mathbf{handle}\langle l\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$. An operation $\mathbf{do}\langle\ell\rangle\ v$ is for performing an effect with a label ℓ. It is passed a single argument v and must be handled by a handler with a label ℓ. A handler $\mathbf{handle}\langle l\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$ is for handling an effect with a static label l. Here, It means that $x, r.e_h$ is the interpretation of an operation with a static label l and $x.e_r$ is a return clause.

4.1.2 Kinding, Equivalence, and Typing Rules

$$\boxed{\Delta \mid \Sigma \vdash \tau :: \kappa}$$

$$\frac{l \in \Sigma}{\Delta \mid \Sigma \vdash l :: \mathbf{L}}\ [\textsc{KLabel}]$$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta, \Delta' \mid \Sigma \vdash \tau_2 :: \mathbf{T} \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}\ [\textsc{KIEff}]$$

$$\boxed{\Delta \mid \Sigma \vdash \rho \equiv \rho'}$$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau_1 \equiv \tau_1' \qquad \Delta, \Delta' \mid \Sigma \vdash \tau_2 \equiv \tau_2' \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 \equiv \exists_\ell\, \Delta'.\tau_1' \Rightarrow \tau_2'}\ [\textsc{EIEff}]$$

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho}$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1)/\iota \qquad \Delta \mid \Sigma \vdash \delta :: \Delta' \qquad \Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \delta(\tau_2)/(\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota}\ [\textsc{Do}]$$

$$\frac{\begin{array}{c}\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \qquad \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \qquad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{Handle}]$$

FIGURE 4.2: Kinding, Equivalence, and Typing Rules of $\lambda^{l}_{\texttt{eff}}$ (extensions)

Kinding Rules We define the kinding rules of $\lambda^{l}_{\texttt{eff}}$ in Figure 4.2. The KLABEL rule states that a static label has kind $\mathbf{L}$. The KIEFF rule is extended with a label from the kinding rule of the Pirog et al.'s. It states that an effect $\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2$ has kind $\mathbf{E}$, where both of the τ_1 and τ_2 have kind $\mathbf{T}$ and the label ℓ has kind $\mathbf{L}$.

Note that we can define a polymorphic effect using a type variable environment Δ'. Let us show an example of a polymorphic effect.

$$\exists_\ell\, \alpha :: \mathbf{T}.\alpha \Rightarrow \alpha$$

This effect signature says that the argument of the operation can be an arbitrary type α, and that the handler for this effect returns a value of the same type α.

Equivalence Rules We define the equivalence rules of $\lambda^{l}_{\texttt{eff}}$ in Figure 4.2. The EIEFF rule is used to determine if two effects are equivalent or not. It states that two effect types with the same label ℓ are equivalent if the input and output types of the operation are equivalent.

Typing Rules We define the typing rules of $\lambda^{l}_{\texttt{eff}}$ in Figure 4.2. These rules are based on Pirog et al. The DO rule takes care of an operation labeled with ℓ. It states that the argument of the operation has the type $\delta(\tau_1)$, the operation is interpreted as a value that has the type $\delta(\tau_2)$, and the operation introduces the effect $\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2$ indexed by the label ℓ. The type $\delta(\tau_1)$ and $\delta(\tau_2)$ are instantiated by type variable substitution δ. Note that the second premise of the DO rule ensure the well-formedness of the type variable substitution δ. Following Pirog et al., we define the well-formedness of type variable substitution as follows:

Definition 1 (Well-formedness of type variable substitution).
$\Delta \mid \Sigma \vdash \delta :: \Delta' \iff \mathbf{dom}(\delta') = \mathbf{dom}(\Delta') \;\wedge\; \forall \alpha \in \mathbf{dom}(\delta), \Delta \mid \Sigma \vdash \delta(\alpha) :: \Delta'(\alpha)$

The HANDLE rule discharges an effect indexed by the label l. It requires that the expressions e_h and e_r have the same type τ_r and the expression e surrounded by the handler performs the effect $(\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$. Then, it concludes that the expression $\mathbf{handle}\langle l\rangle\; e\; \mathbf{with}\; \{x, r.e_h; x.e_r\}$ has the type τ and performs the residual effects ρ.

4.1.3 Operational Semantics

Evaluation Context

$$E \;::=\; \cdots \mid \mathbf{handle}\langle l\rangle\; E\; \mathbf{with}\; \{x, r.e_h; x.e_r\}$$

Reduction Rules

$$\frac{}{\mathbf{handle}\langle l\rangle\; v\; \mathbf{with}\; \{x, r.e_h; x.e_r\} \mapsto e_r\{v/x\}}\;[\text{ERETURN}]$$

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z.\mathbf{handle}\langle l\rangle\; E[z]\; \mathbf{with}\; \{x, r.e_h; x.e_r\}}{\mathbf{handle}\langle l\rangle\; E[\mathbf{do}\langle l\rangle\; v]\; \mathbf{with}\; \{x, r.e_h; x.e_r\} \mapsto e_h\{v/x\}\{v_c/r\}}\;[\text{EHANDLE}]$$

FIGURE 4.3: Evaluation contexts and reduction rules of $\lambda^{l}_{\texttt{eff}}$ (extensions)

We define the operational semantics of $\lambda^{l}_{\texttt{eff}}$ in Figure 4.3. The semantics is based on Pirog et al. The ERETURN rule states that a handler construct reduces to the return clause $e_r\{v/x\}$ if the handled expression is a value. The EHANDLE rule states that an operation $\mathbf{do}\langle l\rangle\; v$ is handled by the innermost handler indexed with the same label l. The premise $l \notin \lceil E \rceil$ states that there is no handler indexed by the label l in the evaluation context E. Below is the definition of the meta function $\lceil\cdot\rceil$, which extracts labels.

Definition 2 (Label Extraction).

$$\begin{array}{l}
\lceil \square \rceil \;::=\; \emptyset \qquad \lceil E\, e \rceil \;::=\; \lceil E \rceil \qquad \lceil v\, E \rceil \;::=\; \lceil E \rceil \\
\lceil \mathbf{handle}\langle l\rangle\; E\; \mathbf{with}\; \{x, r.e_h; x.e_r\} \rceil \;::=\; l, \lceil E \rceil \\
\lceil \epsilon \cdot \rho \rceil \;::=\; \lceil \epsilon \rceil, \lceil \rho \rceil \qquad \lceil \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 \rceil \;::=\; \ell
\end{array}$$

4.1.4 Restrictions of The ESwap Rule and The Syntax of Effect Rows

As we mentioned in Section 3.3, we restrict the ESwap rule and syntax of effect rows. We can only apply the ESwap rule to effect rows with two different labels at the head in order to maintain type soundness. We explain the problem using the following expression. To understand why this restriction is necessary, let us observe the following reduction steps.

$$\begin{array}{ll} & \textbf{handle}\langle l\rangle\ \textbf{handle}\langle l\rangle\ \textbf{do}\langle l\rangle\ ()\ \textbf{with}\ \{x, r.r\ (x+1); x.x\}\ \textbf{with}\ \{x, r.r\ 1; x.x\} \\ \rightarrow & r\ (x+1) \quad (r = \lambda z.\textbf{handle}\langle l\rangle\ \textbf{handle}\langle l\rangle\ z\ \textbf{with}\ \{x, r.r\ (x+1); x.x\}\ \textbf{with}\ \{x, r.r\ 1; x.x\}) \\ \not\rightarrow & \end{array}$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon_1 :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \epsilon_2 :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon_1 \cdot \epsilon_2 \cdot \rho_1 \equiv \epsilon_2 \cdot \epsilon_1 \cdot \rho_2}\ [\text{BadESwap}]$$

The reduction gets stuck because we cannot add a unit value and an integer. This means we should not accept the expression as a well-typed expression. Now, suppose we have replaced ESWap with BadESwap, which allows swapping of any labels, including the same ones. The rule can be applied to all effect rows where the two labels at the head are not necessarily different. With BadESwap, the above expression is judged well-typed. The reasoning goes as follows. First, we can derive $\emptyset \mid \emptyset \mid l \vdash \textbf{do}\langle l\rangle\ () : \texttt{int}/(\exists_l\, \emptyset.() \Rightarrow \texttt{int}) \cdot (\exists_l\, \emptyset.\texttt{int} \Rightarrow \texttt{int})$ by the Do and Sub rules. Next, we can apply the BadESwap and Sub rules to it and derive $\emptyset \mid \emptyset \mid l \vdash \textbf{do}\langle l\rangle\ () : \texttt{int}/(\exists_l\, \emptyset.\texttt{int} \Rightarrow \texttt{int}) \cdot (\exists_l\, \emptyset.() \Rightarrow \texttt{int})$. Thus, we can apply the Handle rule to it and obtain the following wrong typing judgement.

$$\emptyset \mid \emptyset \mid l \vdash \textbf{handle}\langle l\rangle\ \textbf{handle}\langle l\rangle\ \textbf{do}\langle l\rangle\ ()\ \textbf{with}\ \{x, r.r\ (x+1); x.x\}\ \textbf{with}\ \{x, r.r\ 1; x.x\} : \texttt{int}/\iota$$

To reject expressions like the above one, we restrict, the ESawp rule to effect rows with two different labels. With this restriction, we cannot derive $\emptyset \mid \emptyset \mid l \not\vdash \textbf{handle}\langle l\rangle\ \textbf{do}\langle l\rangle\ ()\ \textbf{with}\ \{x, r.r\ (x+1); x.x\}$ because the type of the first input of the effect must be int in order to apply the Handle rule.

Note that we cannot extend effect rows with a type variable that has kind $\mathbf{E}$. The reason is that we cannot extract labels from type variables that have kind $\mathbf{E}$ using $\lceil\cdot\rceil$.

4.1.5 Type Soundness

We prove the type soundness of $\lambda^{l}_{\texttt{eff}}$ by showing the progress and preservation theorems [Wright and Felleisen, 1994]. The details of the proof can be found in AppendixC.

Theorem 1 (Preservation of $\lambda^{l}_{\texttt{eff}}$).
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\iota$ and $e \rightarrow e'$ then $\emptyset \mid \emptyset \mid \Sigma \vdash e' : \tau/\iota$.

Theorem 2 (Progress of $\lambda^{l}_{\texttt{eff}}$).
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau/\iota$ then e is a value or there exists e' such that $e \rightarrow e'$.

4.2 $\lambda^{l}_{\texttt{del}}$: Delimited Control Operators with Static Prompt Tags

In this section, we define $\lambda^{l}_{\texttt{del}}$, which is a calculus that extends $\lambda_{\texttt{core}}$ with delimited control operators and prompt tags. In the following sections, we define the syntax, the typing rules, and the operational semantics (Sects. 4.2.1-4.2.3). Lastly, we show type soundness (Sect. 4.2.4). Note that $\lambda^{l}_{\texttt{del}}$ is based on the calculi of Pirog et al.

4.2.1 Syntax

$$
\begin{array}{llcll}
\text{Kind} & \kappa & ::= & \ldots & \\
 & & \mid & \mathbf{L} & \text{(label type)} \\
\text{Type} & \sigma, \tau & ::= & \cdots & \\
 & & \mid & \ell & \text{(label)} \\
\text{Effect} & \epsilon & ::= & \exists_{\ell}\, \Delta'.\, \tau / \rho & \\
\text{Label} & \ell & ::= & l & \text{(static label)} \\
\text{Expression} & e & ::= & \ldots & \\
 & & \mid & \mathbf{shift}_0 \langle \ell \rangle\ k.\, e & (\text{shift}_0) \\
 & & \mid & \langle e \mid x.\, e \rangle_l & \text{(dollar)}
\end{array}
$$

FIGURE 4.4: Syntax of $\lambda^{l}_{\texttt{del}}$ (extensions)

Kinds, Types, and Effects We define the syntax of kinds, types, and effects of $\lambda^{l}_{\texttt{del}}$ in Figure 4.4. Similar to the $\lambda^{l}_{\texttt{eff}}$ syntax, kinds and types are extended with label type **L** and label ℓ, respectively. Effects only include $\exists_{\ell}\, \Delta'.\, \tau/\rho$ that is also generalized by type variable environment Δ' similar to the $\lambda^{l}_{\texttt{eff}}$. It means that the τ and ρ are the output type and effect row of the continuation captured by a shift operator indexed by ℓ. As in $\lambda^{l}_{\texttt{eff}}$, labels are chosen from a set of statically defined labels Σ.

Expressions We define the syntax of expressions of $\lambda^{l}_{\texttt{del}}$ in Figure 4.4. Expressions include the shift operators $\mathbf{shift}_0\langle \ell \rangle\ k.\, e$ and the dollar operators $\langle e \mid x.\, e_r \rangle_l$. A shift operator $\mathbf{shift}_0\langle \ell \rangle\ k.\, e$ captures continuations. The expression e is the body of the shift operator and function k is a continuation captured by the shift operator. A dollar operator $\langle e \mid x.\, e_r \rangle_l$ delimits the continuation with label l. The expression e is the body of the dollar operator and expression $x.e_r$ is the return clause.

4.2.2 Kinding, Equivalence, and Typing Rules

$$\boxed{\Delta \mid \Sigma \vdash \tau :: \kappa}$$

$$\frac{l \in \Sigma}{\Delta \mid \Sigma \vdash l :: \mathbf{L}}\ [\textsc{KLabel}]$$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau :: \mathbf{T} \qquad \Delta, \Delta' \mid \Sigma \vdash \rho :: \mathbf{R} \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_{\ell}\, \Delta'.\, \tau / \rho :: \mathbf{E}}\ [\textsc{KMEff}]$$

$$\boxed{\Delta \mid \Sigma \vdash \rho \equiv \rho'}$$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau \equiv \tau' \qquad \Delta, \Delta' \mid \Sigma \vdash \rho \equiv \rho' \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_{\ell}\, \Delta'.\, \tau / \rho \equiv \exists_{\ell}\, \Delta'.\, \tau' / \rho'}\ [\textsc{EMEff}]$$

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma / \rho}$$

$$\frac{\begin{array}{c} \Delta \mid \Sigma \vdash \tau' :: \mathbf{T} \quad \Delta \mid \Sigma \vdash \ell :: \mathbf{L} \quad \Delta, \Delta' \mid \Sigma \vdash \rho' <: \rho \\ \Delta, \Delta' \mid \Gamma, k : \tau' \to_{\rho} \tau \mid \Sigma \vdash e : \tau / \rho' \quad \Delta \mid \Sigma \vdash (\exists_{\ell}\, \Delta'.\, \tau / \rho) \cdot \rho' :: \mathbf{R} \end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle \ell \rangle\, k.\, e : \tau' / (\exists_{\ell}\, \Delta'.\, \tau / \rho) \cdot \rho'}\ [\textsc{Shift}_0]$$

$$\frac{\begin{array}{c} \Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_{l}\, \Delta'.\, \tau / \rho) \cdot \delta(\rho) \quad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau) / \delta(\rho) \end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_{l} : \delta(\tau) / \delta(\rho)}\ [\textsc{Dollar}]$$

FIGURE 4.5: Kinding, Equivalence, Typing rules of $\lambda^{l}_{\texttt{del}}$ (extensions)

Kinding Rules We define the kinding rules of $\lambda^{l}_{\texttt{del}}$ in Figure 4.5. The KLABEL rule is identical to the $\lambda^{l}_{\texttt{eff}}$. The KMEFF rule is rule is extended with a label from the kinding rule of the Pirog et al.'s. It states that an effect $\exists_{\ell}\, \Delta'.\, \tau / \rho$ has kind **E** if the type τ has kind **T** and the effect row has kind **R**. As in $\lambda^{l}_{\texttt{eff}}$, we can define a polymorphic effect using a type variable environment Δ'.

Equivalence Rules We define the equivalence rules for $\lambda^{l}_{\texttt{del}}$ in Figure 4.5. The EMEFF rule is used to determine if two effects are equivalent or not similar to the $\lambda^{l}_{\texttt{del}}$. It states that two effect types with the same label ℓ is equivalent if the output type of a continuation and effect row are equivalent.

Typing Rules We define the typing rules of $\lambda^{l}_{\texttt{del}}$ in Figure 4.5. The SHIFT$_0$ rule states that a shift operator introduces an effect indexed by the label l. The effect has information of the output type and effect row of the delimited continuation k. The effect row ρ' represents the effects of the body of a shift operator and has to be subsumed by the effect row of the continuation k. This is because a shift operator reduces to the body of itself. The DOLLAR rule states that a dollar operator discharges the effect indexed by the label l. It requires that the body of a dollar operator introduces the effect $\exists_l\, \Delta'.\, \tau/\rho$ and also requires the tail of an effect row is instantiated by type variable substitution.

4.2.3 Operational Semantics

Evaluation Context

$$E \quad ::= \quad \cdots \mid \langle E \mid x.\, e\rangle_l$$

Reduction Rules

$$\frac{}{\langle v \mid x.\, e_r\rangle_l \mapsto e_r\{v/x\}}\ [\text{ERETURN}]$$

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z.\langle E[z] \mid x.\, e_r\rangle_l}{\langle E[\mathbf{shift}_0\langle l\rangle\ k.\, e] \mid x.\, e_r\rangle_l \mapsto e\{v_c/x\}}\ [\text{EDOLLAR}]$$

FIGURE 4.6: Evaluation contexts and Reduction rules of $\lambda^{l}_{\texttt{del}}$

We define the operational semantics of $\lambda^{l}_{\texttt{del}}$ in Figure 4.6. The ERETURN rule states that a dollar construct reduces to the return clause $x.e_r$ if the body of the dollar expression is a value. The EDOLLAR rule states that the body of the shift operator $\mathbf{shift}_0\langle l\rangle\ k.\, e$ is executed and passed the continuation delimited by the innermost dollar expression. As in the EHANDLER rule in $\lambda^{l}_{\texttt{eff}}$, the premise $l \notin \lceil E \rceil$ states that there is no dollar expression indexed by the label l in the environment context E. We define the meta function $\lceil \cdot \rceil$ as follows:

Definition 3 (Label Extraction)**.**

$$\lceil \square \rceil \ ::= \ \emptyset \qquad \lceil E\ e \rceil \ ::= \ \lceil E \rceil \qquad \lceil v\ E \rceil \ ::= \ \lceil E \rceil \qquad \lceil \langle E \mid x.\, e\rangle_l \rceil \ ::= \ l, \lceil E \rceil$$
$$\lceil \epsilon \cdot \rho \rceil \ ::= \ \lceil \epsilon \rceil, \lceil \rho \rceil \qquad \lceil \exists_\ell\, \Delta'.\, \tau/\rho \rceil \ ::= \ \ell$$

4.2.4 Type Soundness

We prove the type soundness of $\lambda^{l}_{\texttt{del}}$ in a similar way to that of $\lambda^{l}_{\texttt{eff}}$. The details of the proof can be found in Appendix B.

Theorem 3 (Preservation of $\lambda^{l}_{\texttt{del}}$)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\iota$ and $e \rightarrow e'$ then $\emptyset \mid \emptyset \mid \Sigma \vdash e' : \tau/\iota$.

Theorem 4 (Progress of $\lambda^{l}_{\texttt{del}}$)**.**
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau/\iota$ then e is a value or there exists e' such that $e \rightarrow e'$.

4.3 Translation Between Static Effect Instances and Prompt Tags

In this section, we define $[\![\cdot]\!]^{\mathbf{PI}}$ and $[\![\cdot]\!]^{\mathbf{IP}}$, which is a pair of macro translations between $\lambda^{l}_{\texttt{eff}}$ and $\lambda^{l}_{\texttt{del}}$. They extend the macro translations of Pirog et al. with labels. In the following sections, we first define the macro translation from $\lambda^{l}_{\texttt{del}}$ to $\lambda^{l}_{\texttt{eff}}$ and prove the type and meaning preservation properties (Sect. 4.3.1). we next define the macro translation in the opposite direction (i.e., from $\lambda^{l}_{\texttt{eff}}$ to $\lambda^{l}_{\texttt{del}}$) and prove the type and meaning preservation properties (Sect. 4.3.2). The details of proofs can be found in Appendix D.

4.3.1 Translation from Static Prompt Tags to Effect Instances

$$\begin{array}{lcl}
[\![\mathbf{shift}_0\langle \ell\rangle\ k.\ e]\!]^{\mathbf{PI}} & = & \mathbf{do}\langle \ell\rangle\ \lambda k.[\![e]\!]^{\mathbf{PI}} \\
[\![\langle e \mid x.\ e_r\rangle_l]\!]^{\mathbf{PI}} & = & \mathbf{handle}\langle l\rangle\ [\![e]\!]^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.[\![e_r]\!]^{\mathbf{PI}}\} \\
[\![\exists_\ell\ \Delta'.\ \tau/\rho]\!]^{\mathbf{PI}} & = & \exists_\ell\ \alpha :: \mathbf{T}.(\forall\Delta'.(\alpha \to_{[\![\rho]\!]^{\mathbf{PI}}} [\![\tau]\!]^{\mathbf{PI}}) \to_{[\![\rho]\!]^{\mathbf{PI}}} [\![\tau]\!]^{\mathbf{PI}}) \Rightarrow \alpha
\end{array}$$

FIGURE 4.7: Macro Translation from $\lambda^{l}_{\texttt{del}}$ to $\lambda^{l}_{\texttt{eff}}$

In Figure 4.7, we define $[\![\cdot]\!]^{\mathbf{PI}}$, a macro translation from $\lambda^{l}_{\texttt{del}}$ to $\lambda^{l}_{\texttt{eff}}$. The only difference from the original macro translation is that each translation rule is extended by a label.

First, we explain the translation for the shift operator. A shift operator indexed by the label ℓ is translated to an operation indexed by the same label ℓ because they introduce an effect. After the translation, the body of shift becomes a lambda abstraction that takes a continuation k.

Next, we explain the translation for the dollar operator. A dollar operator indexed by the label l is translated to a handler expression indexed by the same label l. The operation clause $x, r.x\ r$ means that the delimited continuation r is passed to the operation's argument x. And then, the body of a shift operator receives the continuation through the operation clause of the handler. The return clause of a dollar simply becomes the return clause of a handler.

In the addition to the translation for expressions, we define a translation for effect types. An effect type indexed by the label l of $\lambda^{l}_{\texttt{del}}$ is translated to an effect type indexed by the same label l of $\lambda^{l}_{\texttt{eff}}$. The input type of the effect $\forall\Delta'.(\alpha \to_{[\![\rho]\!]^{\mathbf{PI}}} [\![\tau]\!]^{\mathbf{PI}}) \to_{[\![\rho]\!]^{\mathbf{PI}}} [\![\tau]\!]^{\mathbf{PI}}$ represents the type of an operation's argument (i.e., the type of $\lambda k.[\![e]\!]^{\mathbf{PI}}$). In particular, $(\alpha \to_{[\![\rho]\!]^{\mathbf{PI}}} [\![\tau]\!]^{\mathbf{PI}})$ represents the type of the continuation k, where α is the type of the shift expression.

We prove the type and meaning preservation properties of the macro translation. The first theorem states that the translation preserves typability and the second theorem states that the translation preserves meaning.

Theorem 5 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ then $\Delta \mid [\![\Gamma]\!]^{\mathbf{PI}} \mid \Sigma \vdash [\![e]\!]^{\mathbf{PI}} : [\![\tau]\!]^{\mathbf{PI}}/[\![\rho]\!]^{\mathbf{PI}}$.

Theorem 6 (Translation preserves meaning)**.**
If $e \to e'$ then $[\![e]\!]^{\mathbf{PI}} \to^{+} [\![e']\!]^{\mathbf{PI}}$.

It is important to track prompt tags in the type system of $\lambda^{l}_{\texttt{del}}$ in order to naturally extend the typed macro translation of Pirog et al. with labels. If the type system of $\lambda^{l}_{\texttt{del}}$

did not track prompt tags (as the calculus of Kiselyov, Shan, and Sabry, 2006), we would need to synthesize an appropriate effect and assign it to each translated expression to track effect instances by the type system of $\lambda^{l}_{\mathtt{eff}}$. In particular, it is hard to assign the effects of continuations from the point where the operation is performed.

4.3.2 Translation from Static Effect Instances to Prompt Tags

$$
\begin{aligned}
\llbracket \mathbf{do}\langle \ell \rangle\ v \rrbracket^{\mathbf{IP}} &= \mathbf{shift}_0\langle \ell \rangle\ k.\ \lambda h.h\ \llbracket v \rrbracket^{\mathbf{IP}}\ (\lambda x.k\ x\ h) \\
\llbracket \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{IP}} &= \langle \llbracket e \rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}} \rangle_l\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}}) \\
\llbracket \exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2 \rrbracket^{\mathbf{IP}} &= \exists_\ell\ \alpha :: \mathbf{T}, \beta :: \mathbf{R}.\ ((\forall \Delta'.\llbracket \tau_1 \rrbracket^{\mathbf{IP}} \rightarrow_\iota (\llbracket \tau_2 \rrbracket^{\mathbf{IP}} \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha)/\beta
\end{aligned}
$$

FIGURE 4.8: Macro Translation from $\lambda^{l}_{\mathtt{eff}}$ to $\lambda^{l}_{\mathtt{del}}$

In Figure 4.8, we define $\llbracket \cdot \rrbracket^{\mathbf{IP}}$, a macro translation from $\lambda^{l}_{\mathtt{eff}}$ to $\lambda^{l}_{\mathtt{del}}$. Similar to $\llbracket \cdot \rrbracket^{\mathbf{PI}}$, we extend the original macro translation rules with labels.

We first explain the macro translation for operations. An operation indexed by the label ℓ is translated to a shift operator indexed by the same label ℓ. In particular, the argument of the operation is translated to the body of the shift operator. The meaning of the operation is determined by the surrounding handler, hence the body of the shift operator is a lambda abstraction that receives a handler h. The expression $\lambda x.k\ x\ h$ is a continuation from the point where an operation is performed to the handler surrounding the operations.

Next, we explain the macro translation for handlers. A handler expression indexed by the label l is translated to the application of a dollar operator and operation clause. To determine the interpretation of operations appearing in the handled expression, the dollar operator is applied to a lambda abstraction that represents the operation clause. The handled expression and return clause of a handler simply become the body and return clause of a dollar operator, respectively.

We also define a macro translation for effect types. An effect type indexed by the label l of $\lambda^{l}_{\mathtt{eff}}$ is translated to an effect type indexed by the same label l of $\lambda^{l}_{\mathtt{del}}$. The type of the effect type $\forall \Delta'.\llbracket \tau_1 \rrbracket^{\mathbf{IP}} \rightarrow_\iota (\llbracket \tau_2 \rrbracket^{\mathbf{IP}} \rightarrow_\beta \alpha) \rightarrow_\beta \alpha$ represents the type of the body of the translated shift operator. The argument types $\llbracket \tau_1 \rrbracket^{\mathbf{IP}}$ and $\llbracket \tau_2 \rrbracket^{\mathbf{IP}} \rightarrow_\beta \alpha$ correspond to the types of $\llbracket v \rrbracket^{\mathbf{IP}}$ and $\lambda x.k\ x\ h$, respectively. While the effect row of a continuation in $\lambda^{l}_{\mathtt{eff}}$ is determined by the surrounding handler, the effect row of a continuation in $\lambda^{l}_{\mathtt{del}}$ is determined by a shift operator that introduces an effect. Thus, we generalize the effect row of the effect type by the type variable β in order to preserve typability.

An interesting fact is that meaning preservation is not as straightforward as the macro translation $\llbracket \cdot \rrbracket^{\mathbf{PI}}$. This is the case for the macro translation of Pirog et al.'s as well. Intuitively, we need apply η-expansion to the body of the expression $\lambda x.k\ x\ h$. Below, we explain the problem in detail using a concrete reduction sequence.

$$\text{General context} \quad C \ ::= \ \square \mid C\ e \mid e\ C \mid \lambda x.C \mid \langle C \mid x.\ e_r \rangle_l \mid \langle e \mid x.\ C \rangle_l \mid \textbf{shift}_0\langle \ell \rangle\ k.\ C$$

$$\frac{e_1 \mapsto e_2}{C[e_1] \to_i C[e_2]}\ [\text{GSTEP}]$$

FIGURE 4.9: General context and General step

$$
\begin{aligned}
& [\![\textbf{handle}\langle l\rangle\ \textbf{do}\langle l\rangle\ 1\ \textbf{with}\ \{x, r.r\ x; x.x\}]\!]^{\mathbf{IP}} && (4.1)\\
=\ & \langle \textbf{shift}_0\langle l\rangle\ k.\ \lambda h.h\ 1\ (\lambda x.k\ x\ h) \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x) && (4.2)\\
\to\ & (\lambda h.h\ 1\ (\lambda x.k\ x\ h))\{v_c'/k\}\ (\lambda x.\lambda r.r\ x) \qquad v_c' = \lambda z.\langle z \mid x.\ \lambda h.x\rangle_l && (4.3)\\
=\ & (\lambda h.h\ 1\ (\lambda x.v_c'\ x\ h))\ (\lambda x.\lambda r.r\ x) && (4.4)\\
\to\ & (\lambda x.\lambda r.r\ x)\ 1\ (\lambda x.v_c'\ x\ (\lambda x.\lambda r.r\ x)) && (4.5)\\
\to\ & (\lambda r.r\ 1)\ (\lambda x.v_c'\ x\ (\lambda x.\lambda r.r\ x)) && (4.6)\\
\to\ & (\lambda x.v_c'\ x\ (\lambda x.\lambda r.r\ x))\ 1 && (4.7)\\
=\ & (\lambda x.\ \lambda z.\langle z \mid x.\ \lambda h.x\rangle_l\ x\ (\lambda x.\lambda r.r\ x)\ )\ 1 && (4.8)\\
\neq\ & (\lambda z.\langle z \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x))\ 1 && (4.9)\\
=\ & [\![(\lambda z.\textbf{handle}\langle l\rangle\ z\ \textbf{with}\ \{x, r.r\ x; x.x\})\ 1]\!]^{\mathbf{IP}} && (4.10)
\end{aligned}
$$

The expression 4.1 is translated to the expression 4.2 by $[\![\cdot]\!]^{\mathbf{IP}}$. Then, it reduces to the expression 4.8 via reduction rules. The body of the expression 4.8 with gray background is not identical to the expression 4.9 translated from $\lambda z.\textbf{handle}\langle l\rangle\ z\ \textbf{with}\ \{x, r.r\ x; x.x\})\ 1$. More specifically, the former has an eta redex.

Following Pirog et al., we solve the problem by defining general contexts and GSTEP in Figure 4.9 to deal with the problem. Using this definition, we can reduce the expression 4.13 to the expression 4.14. Now, we can obtain $(\lambda z.\langle z \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x))\ 1$ by applying α-conversion to the expression 4.14. Note that $(\lambda z.\langle z \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x))\ 1$ is translated from $(\lambda z.\textbf{handle}\langle l\rangle\ z\ \textbf{with}\ \{x, r.r\ x; x.x\})\ 1$.

$$
\begin{aligned}
& [\![\textbf{handle}\langle l\rangle\ \textbf{do}\langle l\rangle\ 1\ \textbf{with}\ \{x, r.r\ x; x.x\}]\!]^{\mathbf{IP}} && (4.11)\\
=\ & \langle \textbf{shift}_0\langle l\rangle\ k.\ \lambda h.h\ 1\ (\lambda x.k\ x\ h) \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x) && (4.12)\\
\to^*\ & (\lambda x.\ \lambda z.\langle z \mid x.\ \lambda h.x\rangle_l\ x\ (\lambda x.\lambda r.r\ x)\ )\ 1 && (4.13)\\
\to_i\ & (\lambda x.\langle x \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x))\ 1 && (4.14)\\
\overset{\alpha}{=}\ & (\lambda z.\langle z \mid x.\ \lambda h.x\rangle_l\ (\lambda x.\lambda r.r\ x))\ 1 && (4.15)\\
=\ & [\![(\lambda z.\textbf{handle}\langle l\rangle\ z\ \textbf{with}\ \{x, r.r\ x; x.x\})\ 1]\!]^{\mathbf{IP}} && (4.16)
\end{aligned}
$$

We prove the type and meaning preservation properties as the following theorems. The first theorem is almost identical to Theorem 5. The second theorem uses the GSTEP relation instead of the STEP relation.

Theorem 7 (Translation preserves typability).
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\Delta \mid [\![\Gamma]\!]^{\mathbf{IP}} \mid \Sigma \vdash [\![e]\!]^{\mathbf{IP}} : [\![\tau]\!]^{\mathbf{IP}} / [\![\rho]\!]^{\mathbf{IP}}$.

Theorem 8 (Translation preserves meaning).
If $e \to e'$ then $[\![e]\!]^{\mathbf{IP}} \to_i^+ [\![e']\!]^{\mathbf{IP}}$.

Chapter 5

Dynamically Generated Effect Instances and Prompt Tags

In this chapter, we present different extensions of $\lambda^{l}_{\texttt{eff}}$ and $\lambda^{l}_{\texttt{del}}$ and a pair of macro translations between the two extended calculi. One of the calculi is called $\lambda^{l+\eta}_{\texttt{eff}}$, which features algebraic effect handlers and effect instances. The other is called $\lambda^{l+\eta}_{\texttt{del}}$, which features delimited control operators and prompt tags. As in $\lambda^{l}_{\texttt{eff}}$ and $\lambda^{l}_{\texttt{del}}$, we treat effect instances and prompt tags as second-class values. In the following sections, we first give a formalization of the two calculi (Sects. 5.1-5.2). Then, we define the macro translation and prove the type and meaning preservation properties (Sect. 5.3).

5.1 $\lambda^{l+\eta}_{\texttt{eff}}$: Algebraic Effect Handlers with Dynamically Generated Effect Instances

In this section, we define $\lambda^{l+\eta}_{\texttt{eff}}$ which is a calculus that extends $\lambda^{l}_{\texttt{eff}}$ with dynamically generated effect instances. In the following sections, we define the syntax, the typing rules, and the operational semantics by extending that of $\lambda^{l}_{\texttt{eff}}$ (Sects. 5.1.1-5.1.3). Next, we show type soundness (Sect. 5.1.4). Note that we extend $\lambda^{l}_{\texttt{eff}}$ with dynamically generated effect instances based on the calculus of Biernacki et al., 2020.

5.1.1 Syntax

$$
\begin{array}{llcll}
\text{Effect label} & \ell & ::= & \ldots & \\
 & & \mid & \eta & \text{(handler label)} \\
\text{Expression} & e & ::= & \ldots & \\
 & & \mid & e[\ell] & \text{(label application)} \\
 & & \mid & \textbf{handle}\langle\eta\rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} & \text{(labeled handler)} \\
\text{Value} & v & ::= & \ldots & \\
 & & \mid & \Lambda\eta.e & \text{(label abstraction)}
\end{array}
$$

FIGURE 5.1: Syntax of $\lambda^{l+\eta}_{\texttt{eff}}$ (extensions)

We define the syntax of effect labels, expressions, values in Figure 5.1. Effect labels are extended with type variables that have kind **L**. Expressions include label abstractions

$\Lambda\eta.e$, label applications $e[\ell]$ and labeled handlers $\mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$. A label application $e[\ell]$ is for applying the expression e to the effect label ℓ. A labeled handler $\mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$ introduces the effect label η into the handled expression e. Values include label abstractions $\Lambda\eta.e$. A label abstraction $\Lambda\eta.e$ generalizes the expression e over the type variable η of kind $\mathbf{L}$.

5.1.2 Typing Rules

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho}$$

$$\frac{\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau/\iota}{\Delta \mid \Gamma \mid \Sigma \vdash \Lambda\eta.e : \forall\eta :: \mathbf{L}.\tau/\iota}\ [\textsc{LAbs}]$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall\eta :: \mathbf{L}.\tau/\iota \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Gamma \mid \Sigma \vdash e[\ell] : \tau\{\ell/\eta\}/\iota}\ [\textsc{LApp}]$$

$$\frac{\begin{array}{c}\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau/(\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho \qquad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r/\rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r/\rho}\ [\textsc{LHandle}]$$

FIGURE 5.2: Typing Rules of $\lambda^{l+\eta}_{\texttt{eff}}$ (extensions)

We define the typing rules of $\lambda^{l+\eta}_{\texttt{eff}}$ in Figure 5.2. These rules are based on the calculus of Biernacki et al.. The LABS rule states that the expression e can be generalized over the type variable η if it is a pure expression. The LAPP rule states that the expression e can be applied to the effect label ℓ if it is polymorphic over the effect label. The LHANDLE is almost the same as the HANDLE rule in Figure 4.2. The only difference from the HANDLE is that the handled expression e is typed under the type variable environment $\Delta, \eta :: \mathbf{L}$ so that a new effect instance η is introduced into type variable environment Δ. Note that the operation clause and return clause are typed under the type variable environment Δ that does not have the label $\eta :: \mathbf{L}$. This is necessary for preventing names from escaping their scope. In particular, it makes eta available only within the lexical scope of the handled expression e.

5.1.3 Operational Semantics

Evaluation Context

$$E \quad ::= \quad \cdots \mid E[\ell]$$

Reduction Rules

$\boxed{\Sigma \vdash \ e \mapsto e' \ \dashv \Sigma'}$

$$\frac{}{\Sigma \vdash \ (\Lambda\eta.e)[\ell] \mapsto e\{\ell/\eta\} \ \dashv \Sigma} \ [\text{EL}\textsc{App}]$$

$$\frac{l \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta = \{l/\eta\}}{\Sigma \vdash \ \textbf{handle}\langle\eta\rangle \ e \ \textbf{with} \ \{x, r.e_h; x.e_r\} \mapsto \textbf{handle}\langle l\rangle \ \delta(e) \ \textbf{with} \ \{x, r.e_h; x.e_r\} \ \dashv \Sigma'} \ [\text{E}\textsc{Gen}\text{L}\textsc{abel}]$$

FIGURE 5.3: Evaluation contexts and reduction rules of $\lambda^{l}_{\texttt{eff}}$ (extensions)

We define the operational semantics of $\lambda^{l+\eta}_{\texttt{eff}}$ in Figure 5.3. We use a reduction judgment of the form $\Sigma \vdash e \mapsto e' \dashv \Sigma'$. The judgment states that under the label environment Σ, we reduce the expression e to the expression e' and generate the new label environment Σ'. The semantics is based on Biernacki et al., but we extend them with a effect label environment Σ to preserve types. We discuss the extended reduction judgement in Section 5.1.4.

The most interesting rule is EGENLABEL. The rule states that the type variable η is replaced by the label l that is dynamically generated at runtime and the label environment is extended with the label l. Note that the substitution for η does not apply to the operation clause and return clause as they do not involve η.

5.1.4 Type Soundness

As we mentioned in Section 5.1.3, we must extend the reduction judgment with effect label environment Σ in order to prove type soundness. Intuitively, a well-typed expression may be judged to an ill-typed expression that escapes a lexical scope handler label if we do not generate an effect label at runtime. We illustrate the problem using the following expression which is borrowed from Biernacki et al. Suppose that $h_{id} = x, r.r\ x$ and $r_{id} = x.x$.

$$\begin{aligned}
& \textbf{handle}\langle\eta\rangle \ \textbf{handle}\langle\eta'\rangle \ \textbf{do}\langle\eta\rangle \ \lambda\_.\textbf{do}\langle\eta'\rangle \ \texttt{1} \ \textbf{with}\{h_{id}; r_{id}\} \ \textbf{with}\{h_{id}; r_{id}\} && (5.1)\\
\rightarrow \ & (r\ x)\{v_c/r\}\{(\lambda\_.\textbf{do}\langle\eta'\rangle \ \texttt{1})/x\} && (5.2)\\
& \quad v_c = \lambda z.\textbf{handle}\langle\eta\rangle \ \textbf{handle}\langle\eta'\rangle \ z \ \textbf{with}\{h_{id}; r_{id}\} \ \textbf{with}\{h_{id}; r_{id}\} && (5.3)\\
= \ & (\lambda z.\textbf{handle}\langle\eta\rangle \ \textbf{handle}\langle\eta'\rangle \ z \ \textbf{with}\{h_{id}; r_{id}\} \ \textbf{with}\{h_{id}; r_{id}\}) \ \lambda\_.\textbf{do}\langle\eta'\rangle \ \texttt{1} && (5.4)
\end{aligned}$$

The expression 5.1 is well-typed. By applying a reduction rule EHANDLE, We can reduce it to the expression 5.4. However, we cannot derive $\emptyset \mid \emptyset \mid \emptyset \nvdash \lambda\_.\mathbf{do}\langle \eta' \rangle\ \mathtt{1}$ since the type variable η' escapes from its lexical scope. Hence, the expression 5.4 is ill-typed.

To solve this problem, we extend the reduction judgment with effect label environment Σ. We apply the EGENLABEL rule to the expression 5.5. Then, we reduce it to the expression 5.10. Now, we can derive $\emptyset \mid \emptyset \mid l, l' \vdash \lambda\_.\mathbf{do}\langle l' \rangle\ \mathtt{1} : \forall \alpha :: \mathbf{T}.\alpha \to_\rho \mathtt{int} / \iota$ using generated labels at runtime. Thus, the reduction preserves the type of the expression.

$$
\begin{array}{rll}
 & \mathbf{handle}\langle \eta \rangle\ \mathbf{handle}\langle \eta' \rangle\ \mathbf{do}\langle \eta \rangle\ \lambda\_.\mathbf{do}\langle \eta' \rangle\ \mathtt{1}\ \mathbf{with}\{h_{id}; r_{id}\}\ \mathbf{with}\{h_{id}; r_{id}\} & (5.5)\\
\to & \mathbf{handle}\langle l \rangle\ \mathbf{handle}\langle \eta' \rangle\ \mathbf{do}\langle l \rangle\ \lambda\_.\mathbf{do}\langle \eta' \rangle\ \mathtt{1}\ \mathbf{with}\{h_{id}; r_{id}\}\ \mathbf{with}\{h_{id}; r_{id}\} & (5.6)\\
\to & \mathbf{handle}\langle l \rangle\ \mathbf{handle}\langle l' \rangle\ \mathbf{do}\langle l \rangle\ \lambda\_.\mathbf{do}\langle l' \rangle\ \mathtt{1}\ \mathbf{with}\{h_{id}; r_{id}\}\ \mathbf{with}\{h_{id}; r_{id}\} & (5.7)\\
\to & (r\ x)\{v_c / r\}\{(\lambda\_.\mathbf{do}\langle l' \rangle\ \mathtt{1}) / x\} & (5.8)\\
 & \quad v_c = \lambda z.\mathbf{handle}\langle l \rangle\ \mathbf{handle}\langle l' \rangle\ z\ \mathbf{with}\{h_{id}; r_{id}\}\ \mathbf{with}\{h_{id}; r_{id}\} & (5.9)\\
= & (\lambda z.\mathbf{handle}\langle l \rangle\ \mathbf{handle}\langle l' \rangle\ z\ \mathbf{with}\{h_{id}; r_{id}\}\ \mathbf{with}\{h_{id}; r_{id}\})\ \lambda\_.\mathbf{do}\langle l' \rangle\ \mathtt{1} & (5.10)
\end{array}
$$

We prove type soundness of $\lambda^{l+\eta}_{\mathtt{eff}}$ by showing the progress and preservation theorems. The details of the proof can be found in Appendix F.

Theorem 9 (Preservation of $\lambda^{l+\eta}_{\mathtt{eff}}$ **).**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \iota$ and $\Sigma \vdash e \to e' \dashv \Sigma'$ then $\emptyset \mid \emptyset \mid \Sigma \vdash e' : \tau / \iota$.

Theorem 10 (Progress of $\lambda^{l+\eta}_{\mathtt{eff}}$ **).**
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau / \iota$ then e is a value or there exists e' such that $\Sigma \vdash e \to e' \dashv \Sigma'$.

5.2 $\lambda^{l+\eta}_{\mathtt{del}}$: Delimited Control Operators with Dynamically Generated Prompt Tags

In this section, we define $\lambda^{l+\eta}_{\mathtt{del}}$ which is a calculus that extends $\lambda^{l}_{\mathtt{del}}$ with dynamically generated prompt tags. In the following sections, we define the syntax, the typing rules, and the operational semantics by extending that of $\lambda^{l}_{\mathtt{del}}$ (Sects. 5.2.1-5.2.3). Next, we show type soundness (Sect. 5.2.4).

5.2.1 Syntax

Effect label	ℓ	$::=$	$\dots$	
		$\mid$	η	(dollar label)
Expression	e	$::=$	$\dots$	
		$\mid$	$e[\ell]$	(label application)
		$\mid$	$\langle e \mid x.\ e_r \rangle_\eta$	(labeled dollar)
Value	v	$::=$	$\dots$	
		$\mid$	$\Lambda\eta.e$	(label abstraction)

FIGURE 5.4: Syntax of $\lambda^{l+\eta}_{\mathtt{eff}}$ (extensions)

We define the syntax of effect labels, expressions, and values in Figure 5.4. Effect labels, label abstractions, and label applications are completely identical to $\lambda^{l}_{\texttt{del}}$. A labeled dollar $\langle e \mid x.\, e_r \rangle_\eta$ introduces the effect label η into the body of the dollar operator.

5.2.2 Typing Rules

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma / \rho}$$

$$\frac{\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / \iota}{\Delta \mid \Gamma \mid \Sigma \vdash \Lambda \eta . e : \forall \eta :: \mathbf{L} . \tau / \iota} \; [\textsc{LAbs}]$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall \eta :: \mathbf{L} . \tau / \iota \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Gamma \mid \Sigma \vdash e[\ell] : \tau\{\ell / \eta\} / \iota} \; [\textsc{LApp}]$$

$$\frac{\begin{array}{c} \Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_\eta \, \Delta' .\, \tau / \rho) \cdot \delta(\rho) \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau) / \delta(\rho) \end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta : \delta(\tau) / \delta(\rho)} \; [\textsc{LDollar}]$$

FIGURE 5.5: Typing rules of $\lambda^{l}_{\texttt{del}}$ (extensions)

We define the typing rules of $\lambda^{l+\eta}_{\texttt{del}}$ in Figure 5.5. The LABS and LAPP rules are completely identical to $\lambda^{l+\eta}_{\texttt{eff}}$. The LDOLLAR is almost the same as the DOLLAR rule in Figure 4.5. The only difference from the DOLLAR is that the body of the dollar operator e is typed under the type variable environment $\Delta, \eta :: \mathbf{L}$ so that a new prompt tag η is introduced into type variable environment Δ.

5.2.3 Operational Semantics

Evaluation Context

$$E \ ::= \ \cdots \mid E[\ell]$$

Reduction Rules

$$\boxed{\Sigma \vdash e \mapsto e' \dashv \Sigma'}$$

$$\frac{}{\Sigma \vdash \ (\Lambda\eta.e)[\ell] \mapsto e\{\ell/\eta\} \ \dashv \Sigma} \ [\textsc{ELApp}]$$

$$\frac{p \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta = \{l/\eta\}}{\Sigma \vdash \ \langle e \mid x.\, e_r \rangle_\eta \mapsto \langle \delta(e) \mid x.\, e_r \rangle_\eta \ \dashv \Sigma'} \ [\textsc{EGenLabel}]$$

FIGURE 5.6: Evaluation contexts and Reduction rules of $\lambda^{l}_{\mathtt{del}}$

We define the operational semantics of $\lambda^{l+\eta}_{\mathtt{eff}}$ in Figure 5.3. We use a reduction judgment of the form $\Sigma \vdash e \mapsto e' \dashv \Sigma'$ like $\lambda^{l+\eta}_{\mathtt{eff}}$. The judgment states that under the label environment Σ, we reduce the expression e to the expression e' and generate the new label environment Σ'.

The most interesting rule is EGenLabel. The rule states that the type variable η is replaced by the label l that is dynamically generated at runtime and the label environment is extended with the label l. Note that the substitution for η does not apply to the return clause since it does not include involve η.

5.2.4 Type Soundness

We prove type soundness of $\lambda^{l+\eta}_{\mathtt{del}}$ in a similar way to that of $\lambda^{l+\eta}_{\mathtt{eff}}$. The details of the proof can be found in Appendix E.

Theorem 11 (Preservation of $\lambda^{l+\eta}_{\mathtt{del}}$).
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \iota$ and $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$ then $\emptyset \mid \emptyset \mid \Sigma' \vdash e' : \tau / \iota$.

Theorem 12 (Progress of $\lambda^{l+\eta}_{\mathtt{del}}$).
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau / \iota$ then e is a value or there exists e' such that $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$.

5.3 Translation Between Dynamically Generated Effect Instances and Prompt Tags

In this section, we define $[\![\cdot]\!]^{\mathbf{SPI}}$ and $[\![\cdot]\!]^{\mathbf{SIP}}$, a pair of macro translations between $\lambda^{l+\eta}_{\mathtt{eff}}$ and $\lambda^{l+\eta}_{\mathtt{del}}$. The translations extend the macro translations $[\![\cdot]\!]^{\mathbf{PI}}$ and $[\![\cdot]\!]^{\mathbf{IP}}$ with dynamically generated effect instances and prompt tags. In the following sections, we first define the macro translation from $\lambda^{l+\eta}_{\mathtt{del}}$ to $\lambda^{l+\eta}_{\mathtt{eff}}$ and prove the type and meaning preservation properties (Sect. 5.3.1). Lastly, we define the macro translation in the opposite direction (i.e., from $\lambda^{l+\eta}_{\texttt{eff}}$ to $\lambda^{l+\eta}_{\texttt{del}}$) and prove the type and meaning preservation properties (Sect. 5.3.2). The details of these proofs can be found in Appendix G.

5.3.1 Translation from Dynamically Generated Prompt Tags to Effect Instances

$$\begin{aligned}
\llbracket \Lambda\eta.e \rrbracket^{\mathbf{SPI}} &= \Lambda\eta.\llbracket e \rrbracket^{\mathbf{SPI}} \\
\llbracket e\ [\ell] \rrbracket^{\mathbf{SPI}} &= \llbracket e \rrbracket^{\mathbf{SPI}}\ [\ell] \\
\llbracket \langle e \mid x.\ e_r \rangle_\eta \rrbracket^{\mathbf{SPI}} &= \mathbf{handle}\langle\eta\rangle\ \llbracket e \rrbracket^{\mathbf{SPI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r \rrbracket^{\mathbf{SPI}}\}
\end{aligned}$$

FIGURE 5.7: Macro Translation from $\lambda^{l+\eta}_{\texttt{del}}$ to $\lambda^{l+\eta}_{\texttt{eff}}$

In Figure 5.7, we define $\llbracket\cdot\rrbracket^{\mathbf{SPI}}$ which is a macro translation from $\lambda^{l+\eta}_{\texttt{del}}$ to $\lambda^{l+\eta}_{\texttt{eff}}$. The macro translations for label abstractions and applications are identity. The macro translation for the labeled dollar operator is almost the same as that of unlabeled dollars. The only difference from the macro translation $\llbracket\cdot\rrbracket^{\mathbf{PI}}$ is that a labeled dollar operator is translated to a handler expression that is labeled with the same type variable η.

We prove the type and meaning preservation properties of the macro translation. The first theorem is essentially the same as Theorem 5. The reduction judgement of the second theorem is extended with effect label environment Σ. The details of the proof can be found in Appendix G.

Theorem 13 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{SPI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{SPI}} : \llbracket\tau\rrbracket^{\mathbf{SPI}} / \llbracket\rho\rrbracket^{\mathbf{SPI}}$.

Theorem 14 (Translation preserves meaning)**.**
If $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$ then $\Sigma \vdash \llbracket e\rrbracket^{\mathbf{SPI}} \longrightarrow^{+} \llbracket e'\rrbracket^{\mathbf{SPI}} \dashv \Sigma'$.

5.3.2 Translation from Dynamically Generated Effect Instances to Prompt Tags

$$\begin{aligned}
\llbracket \Lambda\eta.e \rrbracket^{\mathbf{SIP}} &= \Lambda\eta.\llbracket e \rrbracket^{\mathbf{SIP}} \\
\llbracket e\ [\ell] \rrbracket^{\mathbf{SIP}} &= \llbracket e \rrbracket^{\mathbf{SIP}}\ [\ell] \\
\llbracket \mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{SIP}} &= \langle \llbracket e \rrbracket^{\mathbf{SIP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{SIP}} \rangle_\eta\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{SIP}})
\end{aligned}$$

FIGURE 5.8: Macro Translation from $\lambda^{l+\eta}_{\texttt{eff}}$ to $\lambda^{l+\eta}_{\texttt{del}}$

In Figure 5.8, we define $\llbracket\cdot\rrbracket^{\mathbf{SIP}}$, a macro translation from $\lambda^{l+\eta}_{\texttt{eff}}$ to $\lambda^{l+\eta}_{\texttt{del}}$. The macro translations for label abstractions and applications are completely identical to the $\llbracket\cdot\rrbracket^{\mathbf{SPI}}$. The macro translation for the labeled handler is almost the same as the macro translation rule of $\llbracket\cdot\rrbracket^{\mathbf{IP}}$. The only difference from the macro translation $\llbracket\cdot\rrbracket^{\mathbf{IP}}$ is that a labeled handler is translated to a dollar operator that is labeled with the same type variable η.

We prove the type and meaning preservation properties of the macro translation. The first theorem is almost identical to Theorem 7. The reduction judgement of the second theorem is extended with effect label environment Σ as the . The details of the proof can be found in Appendix G.

Theorem 15 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\Delta \mid [\![\Gamma]\!]^{\mathbf{SIP}} \mid \Sigma \vdash [\![e]\!]^{\mathbf{SIP}} : [\![\tau]\!]^{\mathbf{SIP}} / [\![\rho]\!]^{\mathbf{SIP}}$.

Theorem 16 (Translation preserves meaning)**.**
If $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$ then $\Sigma \vdash [\![e]\!]^{\mathbf{SIP}} \longrightarrow_i^+ [\![e']\!]^{\mathbf{SIP}} \dashv \Sigma'$.

Chapter 6

Related Work

6.1 Named Handlers and Effect Instances

Biernacki et al., 2020 propose a calculus with algebraic effect handlers extended with second-class handler names. The calculus has a special syntax for abstraction and application for handler names. It also has lexically scoped, second-class handler names so that handler names never escape their scope. We build our calculus based on Biernacki et al.'s calculus, hence the two calculi share a number of key features.

Xie, Cong, et al., 2022 design a simple and flexible calculus with first-class handler names. Unlike Biernacki et al., their calculus does not have a special syntax and it treats handler names as first-class values. To address the name escaping problem, their calculus employs rank-2 polymorphism, a technique used to provide thread safe interfaces for the `runST` function in Haskell [Launchbury and Jones, 1995]. Thanks to rank-2 polymorphism, the calculus can dynamically generate handler names while maintaining type soundness. This ability can be used to emulate ML-style references, which is impossible in Biernacki et al.'s calculus.

Bauer and Pretnar, 2013 propose the core Eff language, which models an old version of the Eff language. The formalization includes first-class effect instances, which are essentially the same as handler names but cannot be generated dynamically.

6.2 Multi-prompt Delimited Control Operators

Gunter, Rémy, and Riecke, 1995 propose a calculus with multi-prompt delimited control operators. In their calculus, prompt tags are first-class values, but they are not tracked by the type system. Hence, programs may get stuck by executing a control operator that has no corresponding delimiter.

Kiselyov, Shan, and Sabry, 2006 argue that a calculus with dynamic bindings, which they call the *DB* language, can be translated to a calculus with delimited control operators, which they call the *DC* language. The *DC* language features second-class prompt tags, and its type system suffers from the same problem as that of Gunter, Rémy, and Riecke, 1995's. Therefore, in the progress statement of *DC* shown below, they impose a strong assumption that the term M is not a CP-stuck term. The definition of a CP-stuck term is that if M is a CP-stuck term, then there exists some $\texttt{shift}\langle p\rangle$ expression in M that is not surrounded by a $\texttt{reset}\langle p\rangle$ expression.

Theorem (Progress).
If M is a DC term such that $\emptyset \vdash_\Sigma M : \tau$ and M is neither a value nor CP-stuck, then there exists some term M' such that $M \mapsto M'$.

Kobori, Kameyama, and Kiselyov, 2015 define a syntax-directed translation from a source language that has delimited control operators shift/reset with answer-type modification to a target language that has multi-prompt shift/reset without answer-type modification. The translation allows us to implement delimited control operators shift/reset with answer-type modification into the target language that does not have answer-type modification without modifying type system. However, the type system of the target language does not track prompt tags since the formulation of it closely follows the delimcc library [Kiselyov, 2010]. Therefore, the type judgement of the target language does not give us information on whether every shift operator in a well-typed expression is surrounded by a reset operator with the corresponding prompt tags.

6.3 Translation Between Delimited Control Operators and Algebraic Effect Handlers

Forster et al., 2016 propose three calculi with mechanisms for expressing user-defined effects and clarify the relationships between those calculi. Like us, they show the equivalence of expressive power by defining macro translation, but unlike us, they do this in an untyped setting. They then conjecture that extending their translations to a typed setting would require certain forms of polymorphism.

Piróg, Polesiuk, and Sieczkowski, 2019 partially prove Forster et al.'s conjecture by defining typed macro translations between two calculi with algebraic effect handlers and delimited control operators ($\texttt{shift}_0$/$\texttt{dollar}$). They equip the calculi of effect handlers and $\texttt{shift}_0$/$\texttt{dollar}$ with effect polymorphism, which is the key to type preservation.

6.4 Implementation of Algebraic Effect Handlers using Multi-prompt Delimited Control Operators

Kiselyov and Sivaramakrishnan, 2016 present a technique for embedding the Eff language with effect instances into OCaml using the delimcc delimited control operators library [Kiselyov, 2010]. The library has first-class prompt tags and can generate them dynamically. These allow one to program with different instances of the same effects as in our $\lambda^{l}_{\texttt{eff}}$ and $\lambda^{l+\eta}_{\texttt{eff}}$ calculi. However, prompt tags (and hence handler names) can escape their scope, because OCaml does not have an effect system that tracks prompt tags.

Chapter 7

Conclusion and Future Work

In this thesis, we studied the relationship between two mechanisms for expressing computational effects, namely algebraic effect handlers with effect instances and delimited control operators with prompt tags. We first showed the equivalence of expressive power between calculi with static effect instances and prompt tags. To show this, we formalized two calculi $\lambda^{l}_{\texttt{eff}}$ and $\lambda^{l}_{\texttt{del}}$, featuring static effect instances and prompt tags, and defined the macro translations between these calculi. We next did the same for calculi $\lambda^{l+\eta}_{\texttt{eff}}$ and $\lambda^{l+\eta}_{\texttt{del}}$, which can dynamically generate effect instances and prompt tags.

Using the relationship, we can understand one of the two mechanisms via the other mechanism. For theoreticians, they can apply the results about one mechanism to the other. As a theoretical application, we can derive the CPS translation for composing our macro translation and the existing CPS translation for multi-prompt control operators [Downen and Ariola, 2014], similar to Cong and Asai, 2022. For language implementors, they can provide for programmers one of two mechanisms in terms of the other, while preserving types and meaning. In fact, there is an OCaml library of effect instances implemented using multi-prompt control operators [Kiselyov, Shan, and Sabry, 2006], although it uses unsafe type coercion.

As future work, we are considering three directions. First, we would like to extend the results to calculi with first-class effect instances and prompt tags. First-class effect instances are useful for writing flexible programs because we can pass them to a lambda abstraction and pack them into ordinary data structures. Following [Xie, Cong, et al., 2022], we plan to extend our calculus with rank-2 polymorphism to solve the name escaping problem.

Next, we would like to explore the relationship between shallow handlers [Hillerström and Lindley, 2018] with effect instances and $\texttt{control}_0/\texttt{dollar}$ with prompt tags. This mechanisms make it easy to implement mutually recursive functions, for example Unix pipes and copipes [Hillerström and Lindley, 2018]. The relationship between shallow handlers and $\texttt{control}_0/\texttt{dollar}$ has been discussed in a setting without effect instances or prompt tags [Piróg, Polesiuk, and Sieczkowski, 2019]. It is however unclear if the relationship scales straightforwardly to our setting, because Pirog et al. rely on generalization over effect types while we do not allow such generalization.

Lastly, we would like to explore the relationship among monads [Moggi, 1989], algebraic effect handlers and delimited control operators in typed settings. Monads have been a popular tool for representing computational effects. Their expressiveness has been compared to effect handlers and control operators in an untyped setting [Forster et al., 2016], and we intend to investigate how to scale the results to a typed setting.

Bibliography

Bauer, Andrej and Matija Pretnar (2013). "An Effect System for Algebraic Effects and Handlers". In: *Algebra and Coalgebra in Computer Science - 5th International Conference, CALCO 2013, Warsaw, Poland, September 3-6, 2013. Proceedings*. Ed. by Reiko Heckel and Stefan Milius. Vol. 8089. Lecture Notes in Computer Science. Springer, pp. 1–16. DOI: 10.1007/978-3-642-40206-7_1. URL: https://doi.org/10.1007/978-3-642-40206-7%5C_1.

Biernacki, Dariusz et al. (2020). "Binders by day, labels by night: effect instances via lexically scoped handlers". In: *Proc. ACM Program. Lang.* 4.POPL, 48:1–48:29. DOI: 10.1145/3371116. URL: https://doi.org/10.1145/3371116.

Cong, Youyou and Kenichi Asai (2022). "Understanding Algebraic Effect Handlers via Delimited Control Operators". In: *Trends in Functional Programming - 23rd International Symposium, TFP 2022, Virtual Event, March 17-18, 2022, Revised Selected Papers*. Ed. by Wouter Swierstra and Nicolas Wu. Vol. 13401. Lecture Notes in Computer Science. Springer, pp. 59–79. DOI: 10.1007/978-3-031-21314-4_4. URL: https://doi.org/10.1007/978-3-031-21314-4%5C_4.

Convent, Lukas et al. (2020). "Doo bee doo bee doo". In: *J. Funct. Program.* 30, e9. DOI: 10.1017/S0956796820000039. URL: https://doi.org/10.1017/S0956796820000039.

Danvy, Olivier and Andrzej Filinski (1990). "Abstracting Control". In: *Proceedings of the 1990 ACM Conference on LISP and Functional Programming, LFP 1990, Nice, France, 27-29 June 1990*. Ed. by Gilles Kahn. ACM, pp. 151–160. DOI: 10.1145/91556.91622. URL: https://doi.org/10.1145/91556.91622.

Downen, Paul and Zena M. Ariola (2014). "Delimited control and computational effects". In: *J. Funct. Program.* 24.1, pp. 1–55. DOI: 10.1017/S0956796813000312. URL: https://doi.org/10.1017/S0956796813000312.

Felleisen, Matthias (1988). "The Theory and Practice of First-Class Prompts". In: *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, California, USA, January 10-13, 1988*. Ed. by Jeanne Ferrante and Peter Mager. ACM Press, pp. 180–190. DOI: 10.1145/73560.73576. URL: https://doi.org/10.1145/73560.73576.

— (1991). "On the Expressive Power of Programming Languages". In: *Sci. Comput. Program.* 17.1-3, pp. 35–75. DOI: 10.1016/0167-6423(91)90036-W. URL: https://doi.org/10.1016/0167-6423(91)90036-W.

Filinski, Andrzej (1994). "Representing Monads". In: *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '94. Portland, Oregon, USA: Association for Computing Machinery, pp. 446–457. ISBN: 0897916360. DOI: 10.1145/174675.178047. URL: https://doi.org/10.1145/174675.178047.

Forster, Yannick et al. (2016). "On the Expressive Power of User-Defined Effects: Effect Handlers, Monadic Reflection, Delimited Control". In: *CoRR* abs/1610.09161. arXiv: 1610.09161. URL: http://arxiv.org/abs/1610.09161.

Gunter, Carl A., Didier Rémy, and Jon G. Riecke (1995). "A Generalization of Exceptions and Control in ML-like Languages". In: *Proceedings of the seventh international conference on Functional programming languages and computer architecture, FPCA 1995,* *La Jolla, California, USA, June 25-28, 1995*. Ed. by John Williams. ACM, pp. 12–23. DOI: 10.1145/224164.224173. URL: https://doi.org/10.1145/224164.224173.

Hillerström, Daniel and Sam Lindley (2016). "Liberating effects with rows and handlers". In: *Proceedings of the 1st International Workshop on Type-Driven Development, TyDe@ICFP 2016, Nara, Japan, September 18, 2016*. Ed. by James Chapman and Wouter Swierstra. ACM, pp. 15–27. DOI: 10.1145/2976022.2976033. URL: https://doi.org/10.1145/2976022.2976033.

— (2018). "Shallow Effect Handlers". In: *Programming Languages and Systems - 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings*. Ed. by Sukyoung Ryu. Vol. 11275. Lecture Notes in Computer Science. Springer, pp. 415–435. DOI: 10.1007/978-3-030-02768-1_22. URL: https://doi.org/10.1007/978-3-030-02768-1%5C_22.

Kiselyov, Oleg (2010). "Delimited Control in OCaml, Abstractly and Concretely: System Description". In: *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. Proceedings*. Ed. by Matthias Blume, Naoki Kobayashi, and Germán Vidal. Vol. 6009. Lecture Notes in Computer Science. Springer, pp. 304–320. DOI: 10.1007/978-3-642-12251-4_22. URL: https://doi.org/10.1007/978-3-642-12251-4%5C_22.

Kiselyov, Oleg, Chung-chieh Shan, and Amr Sabry (2006). "Delimited dynamic binding". In: *Proceedings of the 11th ACM SIGPLAN International Conference on Functional Programming, ICFP 2006, Portland, Oregon, USA, September 16-21, 2006*. Ed. by John H. Reppy and Julia Lawall. ACM, pp. 26–37. DOI: 10.1145/1159803.1159808. URL: https://doi.org/10.1145/1159803.1159808.

Kiselyov, Oleg and K. C. Sivaramakrishnan (2016). "Eff Directly in OCaml". In: *Proceedings ML Family Workshop / OCaml Users and Developers workshops, ML/OCAML 2016, Nara, Japan, September 22-23, 2016*. Ed. by Kenichi Asai and Mark R. Shinwell. Vol. 285. EPTCS, pp. 23–58. DOI: 10.4204/EPTCS.285.2. URL: https://doi.org/10.4204/EPTCS.285.2.

Kobori, Ikuo, Yukiyoshi Kameyama, and Oleg Kiselyov (2015). "Answer-Type Modification without Tears: Prompt-Passing Style Translation for Typed Delimited-Control Operators". In: *Proceedings of the Workshop on Continuations, WoC 2016, London, UK, April 12th 2015*. Ed. by Olivier Danvy and Ugo de'Liguoro. Vol. 212. EPTCS, pp. 36–52. DOI: 10.4204/EPTCS.212.3. URL: https://doi.org/10.4204/EPTCS.212.3.

Launchbury, John and Simon L. Peyton Jones (1995). "State in Haskell". In: *LISP Symb. Comput.* 8.4, pp. 293–341.

Leijen, Daan (2014). "Koka: Programming with Row Polymorphic Effect Types". In: *Proceedings 5th Workshop on Mathematically Structured Functional Programming, MSFP@ETAPS 2014, Grenoble, France, 12 April 2014*. Ed. by Paul Blain Levy and Neel Krishnaswami. Vol. 153. EPTCS, pp. 100–126. DOI: 10.4204/EPTCS.153.8. URL: https://doi.org/10.4204/EPTCS.153.8.

— (2017). "Type directed compilation of row-typed algebraic effects". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*. Ed. by Giuseppe Castagna and Andrew D. Gordon. ACM, pp. 486–499. DOI: 10.1145/3009837.3009872. URL: https://doi.org/10.1145/3009837.3009872.

Moggi, Eugenio (1989). "Computational Lambda-Calculus and Monads". In: *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*. IEEE Computer Society, pp. 14–23. DOI: 10.1109/LICS.1989.39155. URL: https://doi.org/10.1109/LICS.1989.39155.

Piróg, Maciej, Piotr Polesiuk, and Filip Sieczkowski (2019). "Typed Equivalence of Effect Handlers and Delimited Control". In: *4th International Conference on Formal Structures* *for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. Ed. by Herman Geuvers. Vol. 131. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 30:1–30:16. DOI: 10.4230/LIPIcs.FSCD.2019.30. URL: https://doi.org/10.4230/LIPIcs.FSCD.2019.30.

Plotkin, Gordon D. and Matija Pretnar (2009). "Handlers of Algebraic Effects". In: *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*. Ed. by Giuseppe Castagna. Vol. 5502. Lecture Notes in Computer Science. Springer, pp. 80–94. DOI: 10.1007/978-3-642-00590-9_7. URL: https://doi.org/10.1007/978-3-642-00590-9%5C_7.

Pretnar, Matija (2015). "An Introduction to Algebraic Effects and Handlers. Invited tutorial paper". In: *Electronic Notes in Theoretical Computer Science* 319. The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI)., pp. 19–35. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2015.12.003. URL: http://www.sciencedirect.com/science/article/pii/S1571066115000705.

Wright, Andrew K. and Matthias Felleisen (1994). "A Syntactic Approach to Type Soundness". In: *Inf. Comput.* 115.1, pp. 38–94. DOI: 10.1006/inco.1994.1093. URL: https://doi.org/10.1006/inco.1994.1093.

Xie, Ningning, Jonathan Brachthauser, et al. (2020). "Effect Handlers, Evidently". In: *The 25th ACM SIGPLAN International Conference on Functional Programming (ICFP)*. ACM SIGPLAN. ACM. URL: https://www.microsoft.com/en-us/research/publication/effect-handlers-evidently/.

Xie, Ningning, Youyou Cong, et al. (2022). "First-class names for effect handlers". In: *Proc. ACM Program. Lang.* 6.OOPSLA2, pp. 30–59. DOI: 10.1145/3563289. URL: https://doi.org/10.1145/3563289.

Appendix A

Core Calculus

Syntax of Terms:

$$\begin{array}{llcll}
\text{Expressions} & e & ::= & v & \text{(value)} \\
 & & | & e\ e & \text{(application)} \\
\text{Values} & v & ::= & x & \text{(variable)} \\
 & & | & \lambda x.e & \text{(lambda abstract)}
\end{array}$$

Syntax of Effect Rows:

$$\begin{array}{clll}
\rho & ::= & \iota & \text{(empty effect row)} \\
 & | & \epsilon \cdot \rho & \text{(extended effect row)}
\end{array}$$

Syntax of Kinds and Types:

$$\begin{array}{clll}
\kappa & ::= & \mathbf{T} & \text{(value type)} \\
 & | & \mathbf{E} & \text{(effect type)} \\
 & | & \mathbf{R} & \text{(effect row type)} \\
\sigma, \tau & ::= & \alpha & \text{(type variables)} \\
 & | & \sigma \rightarrow_{\rho} \ \sigma & \text{(function type)} \\
 & | & \forall \alpha :: \kappa . \sigma & \text{(quantified type)}
\end{array}$$

$$\begin{array}{lcl}
\text{Type variable environment} & \Delta & ::= \ \emptyset \mid \Delta, \alpha :: \kappa \\
\text{Type environment} & \Gamma & ::= \ \emptyset \mid \Gamma,\ x : \sigma \\
\text{Label environment} & \Sigma & ::= \ \emptyset \mid \Sigma, l
\end{array}$$

Names:

Type variables $\ni \alpha, \beta, \ldots$ Expression variables $\ni x, y, \ldots$ Labels $\ni l, l_1, \ldots$

FIGURE A.1: Syntax of Core Calculus

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho}$$

$$\frac{x : \tau \in \Gamma}{\Delta \mid \Gamma \mid \Sigma \vdash x : \tau / \iota}\ [\textsc{Var}]$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e_1 : \tau_1 \to_\rho \tau / \rho \qquad \Delta \mid \Gamma \mid \Sigma \vdash e_2 : \tau_1 / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash e_1\ e_2 : \tau / \rho}\ [\textsc{App}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \tau_1 \to_\rho \tau_2 / \iota}\ [\textsc{Abs}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Gamma \mid \Sigma \vdash e : \tau / \iota \qquad \kappa \in \{\mathbf{T}, \mathbf{R}\}}{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall \alpha :: \kappa . \tau / \iota}\ [\textsc{Gen}]$$

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \forall \alpha :: \kappa . \tau / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau\{\sigma / \alpha\} / \rho}\ [\textsc{Inst}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \tau_1 / \rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau_2 / \rho_2}\ [\textsc{Sub}]$$

FIGURE A.2: Typing rules

$$\boxed{\Delta \mid \Sigma \vdash \tau :: \kappa}$$

$$\frac{\alpha :: \kappa \in \Delta}{\Delta \mid \Sigma \vdash \alpha :: \kappa}\ [\textsc{KVar}] \qquad \frac{l \in \Sigma}{\Delta \mid \Sigma \vdash l :: \mathbf{L}}\ [\textsc{KLabel}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Sigma \vdash \rho :: \mathbf{R} \qquad \Delta \mid \Sigma \vdash \tau_2 :: \mathbf{T}}{\Delta \mid \Sigma \vdash \tau_1 \to_\rho \tau_2 :: \mathbf{T}}\ [\textsc{KArrow}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau :: \mathbf{T}}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa . \tau :: \mathbf{T}}\ [\textsc{KGen}] \qquad \frac{}{\Delta \mid \Sigma \vdash \iota :: \mathbf{R}}\ [\textsc{KEmpty}]$$

$$\frac{\Delta \mid \Sigma \vdash \epsilon :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \rho :: \mathbf{R}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho :: \mathbf{R}}\ [\textsc{KRow}]$$

FIGURE A.3: Kinding rules

$$\boxed{\Delta \mid \Sigma \vdash \sigma <: \sigma'}$$

$$\frac{\Delta \mid \Sigma \vdash \sigma_1 \equiv \sigma_2}{\Delta \mid \Sigma \vdash \sigma_1 <: \sigma_2}\ [\textsc{SRefl}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_1^1 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \tau_2^1 <: \tau_2^2}{\Delta \mid \Sigma \vdash \tau_1^1 \rightarrow_{\rho_1} \tau_2^1 <: \tau_1^2 \rightarrow_{\rho_2} \tau_2^2}\ [\textsc{SArrow}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_1 <: \tau_2}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa . \tau_1 <: \forall \alpha :: \kappa . \tau_2}\ [\textsc{SGen}] \qquad \frac{\Delta \mid \Sigma \vdash \rho :: \mathbf{R}}{\Delta \mid \Sigma \vdash \iota <: \rho}\ [\textsc{SEmpty}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 <: \epsilon \cdot \rho_2}\ [\textsc{SRow}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \rho_2 <: \rho_3}{\Delta \mid \Sigma \vdash \rho_1 <: \rho_3}\ [\textsc{STrans}]$$

FIGURE A.4: Subtyping rules

$$\boxed{\Delta \mid \Sigma \vdash \sigma \equiv \sigma'}$$

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa}{\Delta \mid \Sigma \vdash \sigma \equiv \sigma}\ [\text{E}\textsc{Refl}]$$

$$\frac{\Delta \mid \Sigma \vdash \tau_2^1 \equiv \tau_1^1 \qquad \Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \tau_1^2 \equiv \tau_2^2}{\Delta \mid \Sigma \vdash \tau_1^1 \to_{\rho_1} \tau_2^1 \equiv \tau_1^2 \to_{\rho_2} \tau_2^2}\ [\text{E}\textsc{Arrow}]$$

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_1 \equiv \tau_2}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa . \tau_1 \equiv \forall \alpha :: \kappa . \tau_2}\ [\text{E}\textsc{Gen}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \equiv \epsilon \cdot \rho_2}\ [\text{E}\textsc{Row}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon_1 :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \epsilon_2 :: \mathbf{E} \qquad \lceil \epsilon_1 \rceil \neq \lceil \epsilon_2 \rceil}{\Delta \mid \Sigma \vdash \epsilon_1 \cdot \epsilon_2 \cdot \rho_1 \equiv \epsilon_2 \cdot \epsilon_1 \cdot \rho_2}\ [\text{E}\textsc{Swap}]$$

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \rho_2 \equiv \rho_3}{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_3}\ [\text{E}\textsc{Trans}]$$

FIGURE A.5: Equivalence rules

Evaluation Context

Pure Evaluation Context	F	$::=$	$\square \mid e\ F \mid F\ v$
Evaluation Context	E	$::=$	$\square \mid e\ E \mid E\ v$

Reduction Rules

$$\frac{}{(\lambda x.e)\ v \mapsto e\{v/x\}}\ [\textsc{Beta}] \qquad \frac{e \mapsto e'}{E[e] \to E[e']}\ [\textsc{Step}]$$

FIGURE A.6: Evaluation context and Reduction rules

Appendix B

$\lambda^{l}_{\texttt{del}}$: Delimited Control Operators with Static Prompt Tags

B.1 Syntax and Semantics

Syntax of Terms:

$$\begin{array}{llll}
\text{Expressions} & e & ::= & \cdots \mid \mathbf{shift}_0\langle \ell \rangle\ k.\, e & (\text{shift}_0) \\
 & & \mid & \langle e \mid x.\, e \rangle_l & (\text{dollar})
\end{array}$$

Syntax of Effects:

$$\begin{array}{llll}
\text{Effects} & \epsilon & ::= & \exists_\ell\ \Delta'.\ \tau/\rho
\end{array}$$

Evaluation context

$$E \ ::= \ \cdots \mid \langle E \mid x.\, e \rangle_l$$

Prompts Extractor

$$\lceil \Box \rceil \ ::= \ \emptyset \qquad \lceil E\ e \rceil \ ::= \ \lceil E \rceil \qquad \lceil v\ E \rceil \ ::= \ \lceil E \rceil \qquad \lceil \langle E \mid x.\, e \rangle_l \rceil \ ::= \ l, \lceil E \rceil$$

Reduction Rules

$$\frac{}{\langle v \mid x.\, e_r \rangle_l \mapsto e_r\{v/x\}}\ [\text{ERETURN}]$$

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z.\langle E[z] \mid x.\, e_r \rangle_{;}}{\langle E[\mathbf{shift}_0\langle l \rangle\ k.\, e] \mid x.\, e_r \rangle_l \mapsto e\{v_c/x\}}\ [\text{EDOLLAR}]$$

FIGURE B.1: Delimited control operators with static prompt tags

$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma/\rho}$

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \ell :: \mathbf{L} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Delta \mid \Sigma \vdash \tau' :: \mathbf{T} \\ \Delta, \Delta' \mid \Gamma, k : \tau' \to_{\rho} \tau \mid \Sigma \vdash e : \tau/\rho' \qquad \Delta, \Delta' \mid \Sigma \vdash \rho' <: \rho \qquad \Delta \mid \Sigma \vdash (\exists_{\ell}\, \Delta'.\, \tau/\rho) \cdot \rho' :: \mathbf{R}\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\ k.\, e : \tau'/(\exists_{\ell}\, \Delta'.\, \tau/\rho) \cdot \rho'}\ [\textsc{Shift}_0]$$

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau'/(\exists_{l}\, \Delta'.\, \tau/\rho) \cdot \delta(\rho) \quad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_{l} : \delta(\tau)/\delta(\rho)}\ [\textsc{Dollar}]$$

FIGURE B.2: Typing rules

$\boxed{\Delta \mid \Sigma \vdash \tau :: \kappa}$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau :: \mathbf{T} \qquad \Delta, \Delta' \mid \Sigma \vdash \rho :: \mathbf{R} \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_{\ell}\, \Delta'.\, \tau/\rho :: \mathbf{E}}\ [\text{KME}\textsc{ff}]$$

FIGURE B.3: Kinding rules

$\boxed{\Delta \mid \Sigma \vdash \rho \equiv \rho'}$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau \equiv \tau' \qquad \Delta, \Delta' \mid \Sigma \vdash \rho \equiv \rho' \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_{\ell}\, \Delta'.\, \tau/\rho \equiv \exists_{\ell}\, \Delta'.\, \tau'/\rho'}\ [\text{EME}\textsc{ff}]$$

FIGURE B.4: Equivalence

B.2 Proof of Soundness

Definition 4 (Well-formedness for a substitution).
$\Delta \mid \Sigma \vdash \delta :: \Delta' \overset{def}{\Leftrightarrow} \mathsf{dom}(\delta) = \mathsf{dom}(\Delta') \ \wedge \ \forall \alpha \in \mathsf{dom}(\Delta'), \Delta \mid \Sigma \vdash \delta(\alpha) :: \Delta'(\alpha)$

Definition 5 (Substitution composition).
$\delta_2 \circ \delta_1 = \{\overline{\delta_2(\sigma)/\alpha}\}$, where $\delta_1 := \{\overline{\sigma/\alpha}\}$.

Definition 6 (Instantiation relation).

$$\frac{\Delta \mid \Sigma \vdash \tau :: \kappa}{\Delta \mid \Sigma \vdash \tau \rightsquigarrow \tau}\text{[IR}\textsc{efl}\text{]} \qquad \frac{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa.\tau \qquad \Delta \mid \Sigma \vdash \sigma :: \kappa}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa.\tau \rightsquigarrow \tau\{\sigma/\alpha\}}\text{[II}\textsc{nst}\text{]}$$

Definition 7 (Multi step instatiation relation).

$$\frac{\Delta \mid \Sigma \vdash \tau \rightsquigarrow \sigma}{\Delta \mid \Sigma \vdash \tau \rightsquigarrow^* \sigma}\text{[MIR}\textsc{efl}\text{]} \qquad \frac{\Delta \mid \Sigma \vdash \tau_1 \rightsquigarrow^* \tau_2 \qquad \Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_3}{\Delta \mid \Sigma \vdash \tau_1 \rightsquigarrow^* \tau_3}\text{[MII}\textsc{nst}\text{]}$$

Definition 8 (Concat effect rows).

$$\iota * \rho = \rho \qquad\qquad \rho * \iota = \rho \qquad\qquad (\epsilon \cdot \rho) * \rho' = \epsilon \cdot (\rho * \rho')$$

Definition 9 (Prompt extractor).

$$\begin{array}{llll} \lceil F \rceil & ::= \ \cdot & \lceil F[\langle E \mid x.\, e_r \rangle_l] \rceil & ::= \ l, \lceil E \rceil \\ \lceil \iota \rceil & ::= \ \cdot & \lceil \rho' * (\exists_\ell\, \Delta'.\, \tau/\rho) \cdot \iota \rceil & ::= \ \lceil (\exists_\ell\, \Delta'.\, \tau/\rho) \rceil, \lceil \rho' \rceil \\ \lceil (\exists_\ell\, \Delta'.\, \tau/\rho) \rceil & ::= \ \ell & & \end{array}$$

Definition 10 (Unhandled operators extractor).

$$\lfloor v \rfloor \ ::= \ 0 \quad \lfloor F[e] \rfloor \ ::= \ \lfloor e \rfloor \quad \lfloor \mathbf{shift}_0 \langle \ell \rangle\, k.\, e \rfloor \ ::= \ 1 \quad \lfloor \langle e \mid x.\, e \rangle_l \rfloor \ ::= \ 0$$

Lemma 1 (Multi step instantiation relation has some types).
If $\Delta \mid \Sigma \vdash \tau \rightsquigarrow^* \tau'$ then there exists τ_0, Δ' and δ such that $\tau = \forall \Delta'.\tau_0$, $\Delta \mid \Sigma \vdash \delta :: \Delta'$ and $\tau' = \delta(\tau_0)$.

Proof. By induction on a derivation of $\Delta \mid \Sigma \vdash \tau \rightsquigarrow^* \tau'$.

Case MIREFL:
Straightforward.

Case MIINST:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 \rightsquigarrow^* \tau_2 \qquad \Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_3}{\Delta \mid \Sigma \vdash \tau_1 \rightsquigarrow^* \tau_3}\text{[MII}\textsc{nst}\text{]}$$

By I.H and $\Delta \mid \Sigma \vdash \tau_1 \rightsquigarrow^* \tau_2$, we get (1) $\tau_1 = \forall \Delta'.\tau_0^1$, (2) $\Delta \mid \Sigma \vdash \delta_1 :: \Delta'$ and (3) $\tau_2 = \delta_1(\tau_0^1)$.

By I.H and $\Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_3$, we get (4) $\tau_2 = \forall \Delta''.\tau_0^2$, (5) $\Delta \mid \Sigma \vdash \delta_2 :: \Delta''$ and (6) $\tau_3 = \delta_2(\tau_0^2)$.

Let us define $\delta = \{\sigma_1/\alpha_1, \cdots, \sigma_n/\alpha_n, \sigma_1'/\beta_1, \cdots, \sigma_m'/\beta_m\}$, where $\delta_1 = \{\sigma_1/\alpha_1, \cdots, \sigma_n/\alpha_n\}$ and $\delta_2 = \{\sigma_1'/\beta_1, \cdots, \sigma_m'/\beta_m\}$.

By the definition of δ, we get $\mathsf{dom}(\delta) = \alpha_1, \cdots, \alpha_n, \beta_1, \cdots, \beta_m = \mathsf{dom}(\Delta', \Delta'')$.

By the definition of a substitution, (2) and (5), we get (6) $\Delta \mid \Sigma \vdash \delta(\gamma) :: (\Delta', \Delta'')(\gamma)$ for any $\gamma \in \mathsf{dom}(\Delta', \Delta'')$.

By the definition of a substitution and $\delta_1(\tau_0^1) = \forall\Delta''.\tau_0^2$, there exists τ_0 such that $\tau_0^1 = \forall\Delta''.\tau_0$ and $\tau_0^2 = \delta_1(\tau_0)$.

Thus, we get $\tau_1 = \forall\Delta', \Delta''.\tau_0$, $\Delta \mid \Sigma \vdash \delta :: \Delta', \Delta''$ and $\tau_3 = \delta(\tau_0)$.

□

Lemma 2 (Type variable substitution preserves equivalence).
If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau_1 \equiv \tau_2$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau_1) \equiv \delta(\tau_2)$.

Proof. Straightforward. □

Lemma 3 (Type variable substitution preserves subtyping relation).
If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau_1 <: \tau_2$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau_1) <: \delta(\tau_2)$.

Proof. Straightforward. □

Lemma 4 (Inversion lemma: equivalence).

- If $\Delta \mid \Sigma \vdash \iota \equiv \rho$ then $\rho = \iota$.
- If $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \equiv \rho_2$ then there exists ρ_2' $\Delta \mid \Sigma \vdash \rho_2 \equiv \epsilon \cdot \rho_2'$.

Proof. Straightforward. □

Lemma 5 (Inversion lemma: subtyping relation).

- If $\Delta \mid \Sigma \vdash \forall\Delta'.\tau <: \sigma$ then there exists τ_0 such that $\sigma = \forall\Delta'.\tau_0$ and $\Delta, \Delta' \mid \Sigma \vdash \tau <: \tau_0$
- If $\Delta \mid \Sigma \vdash \epsilon_1 \cdot \rho_1 <: \rho$ then there exists ϵ_2 and ρ_2 such that $\rho = \epsilon_2 \cdot \rho_2$, $\epsilon_1 \equiv \epsilon_2$, and $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$.
- If $\Delta \mid \Sigma \vdash \tau_1^1 \to_{\rho_1} \tau_2^1 <: \sigma$ then there exists τ_1^2, τ_2^2 and ρ_2 such that $\sigma = \tau_1^2 \to_{\rho_2} \tau_2^2$, $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_1^1$, $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$ and $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_2^2$.
- If $\Delta \mid \Sigma \vdash \sigma <: \tau_1^2 \to_{\rho_2} \tau_2^2$ then there exists τ_1^1, τ_2^1 and ρ_1 such that $\sigma = \tau_1^1 \to_{\rho_1} \tau_2^1$, $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_1^1$, $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$ and $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_2^2$.

Proof. Straightforward. □

Lemma 6 (Heads of effects are same under subtyping).
If $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\, \tau_1/\rho_1) \cdot \rho_1' <: (\exists_\ell\, \Delta'.\, \tau_2/\rho_2) \cdot \rho_2'$ then $\Delta \mid \Sigma \vdash \tau_1 \equiv \tau_2$, $\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2$ and $\Delta \mid \Sigma \vdash \rho_1' <: \rho_2'$.

Proof. Straightforward. □

Lemma 7 (Swapping effects preserves equivalence).
If $\Delta \mid \Sigma \vdash \rho_1 * \epsilon \cdot \rho_2 :: \mathbf{R}$ and $\lceil \epsilon \rceil \notin \lceil \rho_1 \rceil$ then $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \cdot \rho_2 :: \mathbf{R}$, $\Delta \mid \Sigma \vdash \rho_1 * \epsilon \cdot \rho_2 \equiv \epsilon \cdot \rho_1 * \rho_2$ and $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 * \rho_2 \equiv \rho_1 * \epsilon \cdot \rho_2$.

Proof. By induction on a size of ρ_1.

Case $\mathit{size}(\rho_1) = 0$**:**

We get $\rho_1 = \iota$, because of $\mathit{size}(\rho_1) = 0$. By the definition of $\lceil\cdot\rceil$, we get $\rho_1 * \epsilon \cdot \rho_2 = \iota * \epsilon \cdot \rho_2 = \epsilon \cdot \iota * \rho_2 = \epsilon \cdot \rho_1 * \rho_2$.

Thus, we get $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_2 :: \mathbf{R}$, $\Delta \mid \Sigma \vdash \rho_1 * \epsilon \cdot \rho_2 \equiv \epsilon \cdot \rho_1 * \rho_2$ and $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 * \rho_2 \equiv \rho_1 * \epsilon \cdot \rho_2$ by ERefl.

Case $size(\rho_1) = 1$**:**

Let us define $\rho_1 = \epsilon_0 \cdot \iota$.

By inversion of $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \iota * \epsilon \cdot \rho_2 :: \mathbf{R}$, we get (1) $\Delta \mid \Sigma \vdash \epsilon_0 :: \mathbf{E}$ and (2) $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_2 :: \mathbf{R}$.

By inversion of $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_2 :: \mathbf{R}$, we get (3) $\Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}$ and (4) $\Delta \mid \Sigma \vdash \rho_2 :: \mathbf{R}$.

By $\lceil \epsilon \rceil \notin \lceil \rho_1 \rceil$, we get (5) $\lceil \epsilon \rceil \neq \lceil \epsilon_0 \rceil$.

Thus, we get $\Delta \mid \Sigma \vdash \epsilon \cdot \epsilon_0 \cdot \rho_2 :: \mathbf{R}$, $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \epsilon \cdot \rho_2 \equiv \epsilon \cdot \epsilon_0 \cdot \rho_2$ and $\Delta \mid \Sigma \vdash \epsilon \cdot \epsilon_0 \cdot \rho_2 \equiv \epsilon_0 \cdot \epsilon \cdot \rho_2$ by ESWAP, KROW, (1), (2), (3), (4) and (5).

Case $size(\rho_1) > 1$**:**

Let us define $\rho_1 = \epsilon_0 \cdot \rho_0$.

By inversion of $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \rho_0 * \epsilon \cdot \rho_2 :: \mathbf{R}$, we get (1) $\Delta \mid \Sigma \vdash \epsilon_0 :: \mathbf{E}$ and (2) $\Delta \mid \Sigma \vdash \rho_0 * \epsilon \cdot \rho_2 :: \mathbf{R}$.

By I.H and (2), we get (3) $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_0 * \rho_2 :: \mathbf{R}$, (4) $\Delta \mid \Sigma \vdash \rho_0 * \epsilon \cdot \rho_2 \equiv \epsilon \cdot \rho_0 * \rho_2$ and (5) $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_0 * \rho_2 \equiv \rho_0 * \epsilon \cdot \rho_2$.

By EROW, (1) and (3), we get (6) $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \epsilon \cdot \rho_0 * \rho_2 :: \mathbf{R}$.

By I.H and (6), we get (7) $\Delta \mid \Sigma \vdash \epsilon \cdot \epsilon_0 \cdot \rho_0 * \rho_2 :: \mathbf{R}$, (8) $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \epsilon \cdot \rho_0 * \rho_2 \equiv \epsilon \cdot \epsilon_0 \cdot \rho_0 * \rho_2$ and (9) $\Delta \mid \Sigma \vdash \epsilon \cdot \epsilon_0 \cdot \rho_0 * \rho_2 \equiv \epsilon_0 \cdot \epsilon \cdot \rho_0 * \rho_2$.

By EROW, (1), (2), (4) and (5), we get (10) $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \rho_0 * \epsilon \cdot \rho_2 \equiv \epsilon_0 \cdot \epsilon \cdot \rho_0 * \rho_2$ and (11) $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \epsilon \cdot \rho_0 * \rho_2 \equiv \epsilon_0 \cdot \rho_0 * \epsilon \cdot \rho_2$.

By (7), ETRANS, (8), (10), (9), (11), we get $\Delta \mid \Sigma \vdash \epsilon \cdot \epsilon_0 \cdot \rho_0 * \rho_2 :: \mathbf{R}$, $\Delta \mid \Sigma \vdash \epsilon_0 \cdot \rho_0 * \epsilon \cdot \rho_2 \equiv \epsilon \cdot \epsilon_0 \cdot \rho_0 * \rho_2$ and $\Delta \mid \Sigma \vdash \epsilon \cdot \epsilon_0 \cdot \rho_0 * \rho_2 \equiv \epsilon_0 \cdot \epsilon \cdot \rho_0 * \rho_2$.

□

Lemma 8 (Term substitution lemma).
If $\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/\rho$ and $\Delta \mid \Gamma_1 \mid \Sigma \vdash e' : \sigma/\iota$ then $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau/\rho$

Proof. By induction on a derivation of $\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/\rho$.

Case VAR:

$$\frac{y : \tau \in \Gamma_1, x : \sigma, \Gamma_2}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash y : \sigma/\iota} \text{[VAR]}$$

SubCase $x = y$**:**

We get $\Delta \mid \Gamma_1 \mid \Sigma \vdash e' : \sigma/\iota$ by the assumption of this lemma.

By the definition of the substitution, we get $x\{e'/x\} = e'$.

Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e' : \sigma/\iota$ by weakening.

SubCase $x \neq y$**:**

We get $y \in \Gamma_1, \Gamma_2$ by the premise and $x \neq y$.

By the definition of the substitution, we get $y\{e'/x\} = y$.

Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash y : \sigma/\iota$ by VAR.

Case ABS:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \lambda y.e : \tau_1 \rightarrow_\rho \tau_2 / \iota}\ [\textsc{Abs}]$$

By I.H, we get $\Delta \mid \Gamma_1, \Gamma_2, y : \tau_1 \mid \Sigma \vdash e\{e'/x\} : \tau_2/\rho$.

By the definition of the substitution and α-convention, we get $(\lambda y.e)\{e'/x\} = \lambda y.e\{e'/x\}$.

Thus, we get $\Delta \mid \Gamma \mid \Sigma \vdash \lambda y.e\{e'/x\} : \tau_1 \rightarrow_\rho \tau_2/\iota$ by Abs.

Case App, Gen, Inst and Sub:
The result follows directly from I.H.

Case Shift$_0$:

$$\frac{\begin{array}{c}\Delta \mid \Sigma\ell :: \mathbf{L} \quad \Delta, \Delta' \mid \Sigma \vdash \rho' <: \rho \quad \Delta \mid \Sigma \vdash \tau' :: \mathbf{T} \\ \Delta, \Delta' \mid \Gamma_1, x : \sigma, \Gamma_2, k : \tau' \rightarrow_\rho \tau \mid \Sigma \vdash e : \tau/\rho' \quad \Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\ \tau/\rho) \cdot \rho' :: \mathbf{R}\end{array}}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\ k.\ e : \tau'/(\exists_\ell\, \Delta'.\ \tau/\rho) \cdot \rho'}\ [\textsc{Shift}_0]$$

By I.H, we get $\Delta, \Delta' \mid \Gamma_1, \Gamma_2, k : \tau' \rightarrow_\rho \tau \mid \Sigma \vdash e\{e'/x\} : \tau/\rho'$.

We get $\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\ k.\ e\{e'/x\} : \tau'/(\exists_\ell\, \Delta'.\ \tau/\rho) \cdot \rho'$ by Shift$_0$.

Case Dollar:

$$\frac{\begin{array}{c}\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau'/(\exists_l\, \Delta'.\ \tau/\rho) \cdot \delta(\rho) \\ \Delta \mid \Sigma \vdash \delta :: \Delta' \quad \Delta \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \langle e \mid y.\ e_r \rangle_l : \delta(\tau)/\delta(\rho)}\ [\textsc{Dollar}]$$

By I.H, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau'/(\exists_l\, \Delta'.\ \tau/\rho) \cdot \delta(\rho)$ and $\Delta \mid \Gamma_1, \Gamma_2, y : \tau' \mid \Sigma \vdash e_r\{e'/x\} : \delta(\tau)/\delta(\rho)$.

Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash \langle e\{e'/x\} \mid y.\ e_r\{e'/x\} \rangle_l : \delta(\tau)/\delta(\rho)$ by Dollar.

□

Lemma 9 (Type variable substitution lemma)**.**

1. If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau :: \kappa$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau) :: \kappa$

2. If $\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau)/\delta(\rho)$

Proof.

1. By induction on a derivation of $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau :: \kappa$.

   Case KVar:

$$\frac{\alpha :: \kappa \in \Delta_1, \Delta', \Delta_2}{\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \alpha :: \kappa} \text{ [KVAR]}$$

SubCase $\alpha \in \Delta'$:

Let us define $\Delta' = \beta_1 :: \kappa_1, \cdots, \alpha :: \kappa, \cdots, \beta_n :: \kappa_n$ and $\delta = \{\sigma_1/\beta_1, \cdots, \sigma/\alpha, \cdots, \sigma_n/\beta_n\}$.
By the definition of $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$, we get $\Delta_1 \mid \Sigma \vdash \delta(\alpha) :: \Delta'(\alpha)$.
By $\Delta(\alpha) = \kappa$, we get $\Delta_1 \mid \Sigma \vdash \delta(\alpha) :: \kappa$.
Thus, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\alpha) :: \kappa$ by weakening.

SubCase $\alpha \notin \Delta'$:

By the definition of a substitution and $\alpha \notin \Delta'$, we get $\delta(\alpha) = \alpha$ and $\alpha \in \Delta_1, \Delta_2$.
Thus, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\alpha) :: \kappa$ by VAR.

Case KARROW, KEMPTY, KLABEL and KROW:

Straightforward.

Case KGEN:

$$\frac{\Delta_1, \Delta', \Delta_2, \alpha :: \kappa \mid \Sigma \vdash \tau :: \mathbf{T}}{\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \forall \alpha :: \kappa . \tau :: \mathbf{T}} \text{ [KGEN]}$$

By I.H, we get $\Delta_1, \Delta_2, \alpha :: \kappa \mid \Sigma \vdash \delta(\tau) :: \mathbf{T}$.
Thus, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \forall \alpha :: \kappa . \delta(\tau) :: \mathbf{T}$ by KGEN.

Case KMEFF:

$$\frac{\Delta_1, \Delta', \Delta_2, \Delta'' \mid \Sigma \vdash \tau :: \mathbf{T} \qquad \Delta_1, \Delta', \Delta_2, \Delta'' \mid \Sigma \vdash \rho :: \mathbf{R} \qquad \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \exists_\ell \, \Delta''.\, \tau/\rho :: \mathbf{E}} \text{ [KMEFF]}$$

By I.H, we get $\Delta_1, \Delta_2, \Delta'' \mid \Sigma \vdash \delta(\ell) :: \mathbf{L}$, $\Delta_1, \Delta_2, \Delta'' \mid \Sigma \vdash \delta(\tau) :: \mathbf{T}$ and $\Delta_1, \Delta_2, \Delta'' \mid \Sigma \vdash \delta(\rho) :: \mathbf{R}$.
By the definition of a substitution and $\Delta' \cap \Delta'' = \emptyset$, we get $\delta(\exists_\ell \, \Delta''.\, \tau/\rho) = \exists_{\delta(\ell)} \, \Delta''.\, \delta(\tau)/\delta(\rho)$.
Thus, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\exists_\ell \, \Delta''.\, \tau/\rho) :: \mathbf{E}$ by KMEFF.

2. By induction on a derivation of $\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$.

Case VAR:

$$\frac{x : \tau \in \Gamma}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash x : \tau/\iota} \text{ [VAR]}$$

Let us define $\Gamma = x_1 : \tau_1, \cdots, x : \tau, \cdots, x_n : \tau_n$.
By the definition of a substitution, we get $\delta(\Gamma) = x_1 : \delta(\tau_1), \cdots, x : \delta(\tau), \cdots, x_n : \delta(\tau_n)$.
Thus, we get $\Delta_1, \Delta_2 \mid \Gamma \mid \Sigma \vdash x : \delta(\tau)/\iota$ by VAR.

Case ABS:

$$\frac{\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta_1, \Delta', \Delta_2 \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \lambda x.e : \tau_1 \to_\rho \tau_2 / \iota} \;[\textsc{Abs}]$$

By I.H, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma), x : \delta(\tau_1) \mid \Sigma \vdash e : \delta(\tau_2)/\delta(\rho)$.
By Lemma 9, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau_1) :: \mathbf{T}$.
Thus, we get $\Delta_1, \Delta_2 \mid \Gamma \mid \Sigma \vdash \lambda x.e : \delta(\tau_1 \to_\rho \tau_2)/\iota$ by ABS.

Case GEN:

The result follows directly from I.H.

Case INST:

$$\frac{\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \forall \alpha :: \kappa.\tau / \rho}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau\{\sigma/\alpha\}/\rho} \;[\textsc{Inst}]$$

By I.H, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \forall \alpha :: \kappa.\delta(\tau)/\delta(\rho)$.
By Lemma 9, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\sigma) :: \kappa$.
By INST, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau)\{\delta(\sigma)/\alpha\}/\delta(\rho)$.
By the definition of a substitution and $\alpha \notin \Delta'$, we get $\delta(\tau)\{\delta(\sigma)/\alpha\} = \delta(\tau\{\sigma/\alpha\})$.
Thus, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau\{\sigma/\alpha\})/\delta(\rho)$.

Case SUB:

$$\frac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau_1/\rho_1 \\ \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \rho_1 <: \rho_2\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau_2/\rho_2} \;[\textsc{Sub}]$$

By I.H, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau_1)/\delta(\rho_1)$.
By Lemma 3, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau_1) <: \delta(\tau_2)$ and $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\rho_1) <: \delta(\rho_2)$.
Thus, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau_2)/\delta(\rho_2)$ by SUB.

Case SHIFT$_0$:

The result follows from I.H, Lemma 3 and Lemma 9.

Case DOLLAR:

$$\frac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau'/(\exists_l\, \Delta''.\, \tau/\rho) \cdot \delta_0(\rho) \\ \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \delta_0 :: \Delta'' \qquad \Delta_1, \Delta', \Delta_2 \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta_0(\tau)/\delta_0(\rho)\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_l : \delta_0(\tau)/\delta_0(\rho)} \;[\textsc{Dollar}]$$

First, we prove (1) $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta \circ \delta_0 :: \Delta''$.
By the definition of a substitution composition, we get $\mathsf{dom}(\delta \circ \delta_0) = \mathsf{dom}(\delta_0) = \mathsf{dom}(\Delta'')$.
We get $\Delta_1, \Delta_2 \mid \Sigma \vdash (\delta \circ \delta_0)(\alpha) :: (\Delta_1, \Delta_2)(\alpha)$ for any $\alpha \in \mathsf{dom}(\Delta'')$, because of $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\beta) :: \Delta''(\beta)$ for any $\beta \in \Delta''$.
Second, we prove (2) $(\delta \circ \delta_0)(\delta(\alpha)) = \delta(\delta_0(\alpha))$ for any α.
We proceed by case analysis on α.

Let us define $\Delta' = \alpha_1 :: \kappa_1, \cdots, \alpha :: \kappa, \cdots, \alpha_n :: \kappa_n$, $\Delta'' = \beta_1 :: \kappa'_1, \cdots, \beta :: \kappa', \cdots, \beta_n :: \kappa'_n$, $\delta_0 = \{\sigma_1/\alpha_1, \cdots, \sigma/\alpha, \cdots, \sigma_n/\alpha_n\}$ and $\delta = \{\sigma'_1/\beta_1, \cdots, \sigma'/\beta, \cdots, \sigma'_n/\beta_n\}$.

Case $\alpha \in \Delta'$**:** $(\delta \circ \delta_0)(\delta(\alpha)) = (\delta \circ \delta_0)(\alpha) = \delta(\sigma) = \delta(\delta_0(\alpha))$.

Case $\beta \in \Delta''$ $(\alpha = \beta)$**:** $(\delta \circ \delta_0)(\delta(\beta)) = (\delta \circ \delta_0)(\sigma') = \delta(\sigma') = \sigma' = \delta(\beta) = \delta(\delta_0(\beta))$.

Otherwise: $(\delta \circ \delta_0)(\delta(\alpha)) = \alpha = \delta(\delta_0(\alpha))$.

By I.H, we get (3) $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau')/\delta((\exists_l \Delta''. \tau/\rho) \cdot \delta_0(\rho))$ and (4) $\Delta_1, \Delta_2 \mid \delta(\Gamma), x : \delta(\tau') \mid \Sigma \vdash e_r : \delta(\delta_0(\tau))/\delta(\delta_0(\rho))$.

Then, we get (5) $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau')/\delta((\exists_l \Delta''. \tau/\rho)) \cdot (\delta \circ \delta_0)(\delta(\rho))$ and (6) $\Delta_1, \Delta_2 \mid \delta(\Gamma), x : \delta(\tau') \mid \Sigma \vdash e_r : (\delta \circ \delta_0)(\delta(\tau))/(\delta \circ \delta_0)(\delta(\rho))$ by $(\delta \circ \delta_0)(\delta(\alpha)) = \delta(\delta_0(\alpha))$.

Thus, we get $\Delta_1, \Delta', \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash \langle e \mid x.\, e_r\rangle_l : \delta(\delta_0(\tau))/\delta(\delta_0(\rho))$ by DOLLAR, (1), (2), (5) and (6).

□

Lemma 10 (Value is pure)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\iota$

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\rho$.

Case VAR, ABS, GEN:
Straightforward.

Case APP:
This case is impossible because the term of a conclusion is not a value.

Case INST:

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash v : \forall \alpha :: \kappa.\tau_0/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_0\{\sigma/\alpha\}/\rho}\ [\text{INST}]$$

By I.H, we get $\Delta \mid \Gamma \mid \Sigma \vdash v : \forall \alpha :: \kappa.\tau_0/\iota$.

Thus, we get $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_0\{\sigma/\alpha\}/\iota$ by INST.

Case SUB:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash v : \tau_1/\rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_2/\rho_2}\ [\text{SUB}]$$

By I.H, we get $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_1/\iota$.

Thus, we get $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_2/\iota$ by SUB and SEMPTY.

Case SHIFT$_0$ and DOLLAR:
This case is impossible because the term of a conclusion is not a value.

□

Lemma 11 (Compose/Decompose an evaluation context).
If $\Delta \mid \Gamma \mid \Sigma \vdash E[e] : \tau/\rho$ then there exists σ, ρ' such that

- $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma/\rho' * \rho$
- $\lceil E \rceil = \lceil \rho' \rceil$
- If $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma/\rho' * \rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash E[e'] : \tau/\rho$

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash E[e] : \tau/\rho$. We proceed by case analysis on the E.

Case $E = []$:

Let us define (1) $\sigma := \tau$ and (2) $\rho' := \iota$.

Then, we get $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma/\rho' * \rho$.

We get $\lceil E \rceil = \cdot = \lceil \rho' \rceil$ by the definition of $\lceil \cdot \rceil$.

We also get $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau/\rho$ by (1) to (3), for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma/\rho' * \rho$.

Case $E \neq []$:

SubCase Var:
This case is impossible because the form of a conclusion is $E = []$.

SubCase Abs:
This case is impossible because the form of a conclusion is $E = []$.

SubCase App:
We also proceed by case analysis on the E.

SubSubCase $E = E_0\, e_0$:

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash E_0[e] : \tau_1 \rightarrow_\rho \tau_2/\rho \qquad \Delta \mid \Gamma \mid \Sigma \vdash e_0 : \tau_1/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash E_0[e]\, e_0 : \tau/\rho} \text{[App]}$$

By I.H, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma/\rho' * \rho$, (2) $\lceil E_0 \rceil = \lceil \rho \rceil$ and (3) $\Delta \mid \Gamma \mid \Sigma \vdash E_0[e'] : \tau_1 \rightarrow_\rho \tau_2/\rho$, for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma/\rho' * \rho$.
By the definition of $\lceil \cdot \rceil$, we get $\lceil E \rceil = \lceil E_0\, e_0 \rceil = \lceil E_0 \rceil = \lceil \rho' \rceil$.
We get $\Delta \mid \Gamma \mid \Sigma \vdash E_0[e']\, e_0 : \tau/\rho$ by the premise, App and (3), for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma/\rho' * \rho$.

SubSubCase $E = v_0\, E_0$:
This case is similar to the $E = E_0\, e_0$.

SubCase Gen, Inst, Sub and Shift$_0$:
This case is impossible because the form of a conclusion is $E = []$.

SubCase Dollar:

$$\frac{\begin{array}{c}\Delta \mid \Gamma \mid \Sigma \vdash E_0[e] : \tau'/(\exists_l\, \Delta'.\, \tau/\rho) \cdot \delta(\rho) \\ \Delta \mid \Sigma \vdash \delta :: \Delta' \quad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle E_0[e] \mid x.\, e_r \rangle_l : \delta(\tau)/\delta(\rho)} \text{[Dollar]}$$

By I.H, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau'/\rho' * (\exists_l\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)$, (2) $\lceil E_0 \rceil = \lceil \rho' \rceil$ and (3) $\Delta \mid \Gamma \mid \Sigma \vdash E_0[e'] : \tau'/(\exists_l\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)$, for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau'/\rho' * (\exists_l\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)$.

By the definition of $\lceil \cdot \rceil$ and (2), we get $\lceil E \rceil = \lceil \langle E_0 \mid x.\, e_r \rangle_l \rceil = l, \lceil E_0 \rceil = \lceil \rho' * (\exists_l\, \Delta'.\, \tau/\rho) \rceil$.
We get $\Delta \mid \Gamma \mid \Sigma \vdash \langle E_0[e'] \mid x.\, e_r \rangle_l : \delta(\tau)/\delta(\rho)$ by the premise, DOLLAR and (3), for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau'/\rho' * (\exists_l\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)$.

□

Lemma 12 (Inversion lemma: lambda abstraction)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma/\rho$ then there exists τ_1, τ_2, ρ_1 and Δ' such that $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2/\rho_1$, $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau'$ and $\Delta \mid \Sigma \vdash \tau' <: \sigma$.

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma/\rho$.

Case VAR, APP, SHIFT$_0$ and DOLLAR:
These case cannot actually arise, because the forms of a conclusion aren't lambda abstraction.

Case ABS:
Straightforward.

Case GEN:
The result follows directly from I.H.

Case INST:

$$\frac{\Delta \mid \Sigma \vdash \sigma_0 :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \forall \alpha :: \kappa.\tau/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \tau\{\sigma_0/\alpha\}/\rho}\,[\text{ABS}]$$

By I.H, we get (1) $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2/\rho_1$, (2) $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau'$ and (3) $\Delta \mid \Sigma \vdash \tau' <: \forall \alpha :: \kappa.\tau$.

By Lemma 5 and (3), we get (4) $\tau' = \forall \alpha :: \kappa.\tau_0$ and (5) $\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_0 <: \tau$.

By Lemma 3 and (5), we get (6) $\Delta \mid \Sigma \vdash \tau_0\{\sigma/\alpha\} <: \tau\{\sigma/\alpha\}$.

By MIINST, (2) and (4), we get (7) $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau'(= \forall \alpha :: \kappa.\tau_0) \rightsquigarrow \tau_0\{\sigma/\alpha\}$.

Thus, we get $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2/\rho_1$, $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau_0\{\sigma/\alpha\}$ and $\Delta \mid \Sigma \vdash \tau_0\{\sigma/\alpha\} <: \tau\{\sigma/\alpha\}$ by (1), (6) and (7).

Case SUB:

$$\frac{\Delta \mid \Sigma \vdash \sigma <: \sigma' \qquad \Delta \mid \Sigma \vdash \rho <: \rho' \qquad \Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma'/\rho'}\,[\text{SUB}]$$

By I.H, we get (1) $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2/\rho_1$, (2) $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau'$ and (3) $\Delta \mid \Sigma \vdash \tau' <: \sigma$.

By STRANS and (3), we get (4) $\Delta \mid \Sigma \vdash \tau' <: \sigma'$.

Thus, we get $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2/\rho_1$, $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau'$ and $\Delta \mid \Sigma \vdash \tau' <: \sigma'$ by (1), (2) and (4).

□

Lemma 13 (Unhandled shift$_0$ operators)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ then $\lfloor e \rfloor \leqslant size(\rho)$.

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$.

Case VAR:
Straightforward.

Case ABS, APP, GEN and INST:
The result follows directly from I.H.

Case ABS:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \tau_1 \rightarrow_{\rho} \tau_2/\iota}\ [\textsc{Abs}]$$

By the definition of $\lfloor \cdot \rfloor$, we get $\lfloor \lambda x.e \rfloor = 0 = \mathit{size}(\iota)$.

Thus, we get $\lfloor \lambda x.e \rfloor \leqslant \mathit{size}(\iota)$.

Case APP:

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e_1 : \tau_1 \rightarrow_{\rho} \tau/\rho \qquad \Delta \mid \Gamma \mid \Sigma \vdash e_2 : \tau_1/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash e_1\ e_2 : \tau/\rho}\ [\textsc{App}]$$

By I.H, we get (1) $\lfloor e_1 \rfloor \leqslant \mathit{size}(\rho)$ and (2) $\lfloor e_2 \rfloor \leqslant \mathit{size}(\rho)$.

We proceed by case analysis on the e_1.

SubCase e_1 is value:
Let us define $e_1 = v_1$ and $F = v_1\ []$.
Then, we get $v_1\ e_2 = F[e_2]$.
By the definition of a $\lfloor \cdot \rfloor$, we get (3) $\lfloor v_1\ e_2 \rfloor = \lfloor F[e_2] \rfloor = \lfloor e_2 \rfloor$.
Thus, we get $\lfloor v_1\ e_2 \rfloor \leqslant \mathit{size}(\rho)$ by (2) and (3).

SubCase e_1 is not value:
We get $e_1\ e_2 = F[e_1]$, where $F = []\ e_2$.
By the definition of a $\lfloor \cdot \rfloor$, we get (4) $\lfloor e_1\ e_2 \rfloor = \lfloor F[e_1] \rfloor = \lfloor e_1 \rfloor$.
Thus, we get $\lfloor e_1\ e_2 \rfloor \leqslant \mathit{size}(\rho)$ by (1) and (4).

Case GEN:

$$\frac{\Delta, \alpha :: \kappa \mid \Gamma \mid \Sigma \vdash e : \tau/\iota}{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall \alpha :: \kappa.\tau/\iota}\ [\textsc{Gen}]$$

By I.H, we get $\lfloor e \rfloor \leqslant \mathit{size}(\iota)\ (= 0)$.

Thus, we get $\lfloor e \rfloor \leqslant \mathit{size}(\iota)$.

Case SUB:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \tau_1 / \rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau_2 / \rho_2} \text{[S}\small{\text{UB}}\text{]}$$

By I.H, we get $\lfloor e \rfloor \leqslant \mathit{size}(\rho_1)$.

By the definition of $\mathit{size}(\cdot)$, we get $\mathit{size}(\rho_1) \leqslant \mathit{size}(\rho_2)$.

Thus, we get $\lfloor e \rfloor \leqslant \mathit{size}(\rho_2)$.

Case SHIFT$_0$:

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \ell :: \mathbf{L} \qquad \Delta, \Delta' \mid \Sigma \vdash \rho' <: \rho \qquad \Delta \mid \Sigma \vdash \tau' :: \mathbf{T} \\ \Delta, \Delta' \mid \Gamma, k : \tau' \to_\rho \tau \mid \Sigma \vdash e : \tau / \rho' \qquad \Delta \mid \Sigma \vdash (\exists_\ell \, \Delta'. \, \tau / \rho) \cdot \rho' :: \mathbf{R}\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle \ell \rangle \; k. \, e : \tau' / (\exists_\ell \, \Delta'. \, \tau / \rho) \cdot \rho'} \text{[SHIFT}_0\text{]}$$

By the definition of $\lfloor \cdot \rfloor$, we get $\lfloor \mathbf{shift}_0 \langle \ell \rangle \; k. \, e \rfloor = 1$.

By the definition of $\mathit{size}(\cdot)$, we get $\mathit{size}((\exists_\ell \, \Delta'. \, \tau / \rho) \cdot \rho') = 1 + \mathit{size}(\rho') \geqslant 1$. Thus, we get $\lfloor \mathbf{shift}_0 \langle \ell \rangle \; k. \, e \rfloor \leqslant \mathit{size}((\exists_\ell \, \Delta'. \, \tau / \rho) \cdot \rho')$.

Case DOLLAR:

$$\frac{\begin{array}{c}\Delta \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_l \, \Delta'. \, \tau / \rho) \cdot \delta(\rho) \\ \Delta \mid \Sigma \vdash \delta :: \Delta' \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau) / \delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x. \, e_r \rangle_l : \delta(\tau) / \delta(\rho)} \text{[DOLLAR]}$$

By the definition of $\lfloor \cdot \rfloor$, we get $\lfloor \langle e \mid x. \, e_r \rangle_l \rfloor = 0$.

By the definition of $\mathit{size}(\cdot)$, we get $\mathit{size}(\delta(\rho)) \geqslant 0$.

Thus, we get $\lfloor \langle e \mid x. \, e_r \rangle_l \rfloor \leqslant \mathit{size}(\delta(\rho))$.

□

Lemma 14 (Shift$_0$ operator performs an effect)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle \ell \rangle \; k. \, e : \tau / \rho$ then $\rho = (\exists_\ell \, \Delta'. \, \tau' / \rho') \cdot \rho''$.

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle \ell \rangle \; k. \, e : \tau / \rho$.

Case VAR, ABS, APP and DOLLAR:
These cases cannot actually arise, because the forms of a conclusion aren't a shift expression.

Case GEN:
This case cannot actually arise, because the effect row of a conclusion have to be non-empty row by Lemma 13.

Case Inst:

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle \ell \rangle \; k. \, e : \forall \alpha :: \kappa . \tau / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle \ell \rangle \; k. \, e : \tau\{\sigma / \alpha\} / \rho} \text{[INST]}$$

By I.H, we get $\rho = (\exists_\ell\, \Delta'.\, \tau'/\rho') \cdot \rho''$.

Case Sub:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\; k.\, e : \tau_1/\rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\; k.\, e : \tau_2/\rho_2}\;[\textsc{Sub}]$$

By I.H, we get $\rho_1 = (\exists_\ell\, \Delta'.\, \tau'/\rho') \cdot \rho''$.
Thus, we get $\rho_2 = (\exists_\ell\, \Delta'.\, \tau'/\rho') \cdot \rho_2''$ by Lemma 5 and the premise.

Case Shift$_0$:
Straightforward.

□

Lemma 15 (Inversion lemma: shift$_0$ operator)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\; k.\, e : \sigma/\epsilon \cdot \rho$ then there exists $\tau_1, \tau_2, \tau_3, \rho_1$, and ρ_2 such that

- $\Delta, \Delta' \mid \Gamma, k : \tau_1 \rightarrow_{\rho_1} \tau_2 \mid \Sigma \vdash e : \tau_2/\rho_2$
- $\Delta, \Delta' \mid \Sigma \vdash \rho_2 <: \rho_1$
- $\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T}$
- $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_2/\rho_1) \cdot \rho_2 :: \mathbf{R}$
- $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_2/\rho_1) \cdot \rho_2 <: \epsilon \cdot \rho$
- $\Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_3$
- $\Delta \mid \Sigma \vdash \tau_3 <: \sigma$

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\; k.\, e : \sigma/\epsilon \cdot \rho$.

Case Var, App, Abs and Reset:
These cases cannot actually arise, because the forms of a conclusion aren't a shift expression.

Case Gen:
This case cannot actually arise, because the effect row of a conclusion is an empty row by Lemma 13.

Case Shift:
Straightforward.

Case Inst:

$$\frac{\Delta \mid \Sigma \vdash \sigma_0 :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\; k.\, e : \forall \alpha :: \kappa.\sigma_1/\epsilon \cdot \rho}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\; k.\, e : \sigma_1\{\sigma_0/\alpha\}/\epsilon \cdot \rho}\;[\textsc{Inst}]$$

By I.H, we get (1) $\Delta, \Delta' \mid \Gamma, k : \tau_1 \rightarrow_{\rho_1} \tau_2 \mid \Sigma \vdash e : \tau_2/\rho_2$, (2) $\Delta, \Delta' \mid \Sigma \vdash \rho_2 <: \rho_1$, (3) $\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T}$, (4) $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_2/\rho_1) \cdot \rho_2 :: \mathbf{R}$, (5) $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_2/\rho_1) \cdot \rho_2 <: \epsilon \cdot \rho$, (6) $\Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_3$, and (7) $\Delta \mid \Sigma \vdash \tau_3 <: \forall \alpha :: \kappa.\sigma_1$.

By Lemma 5 and (7), we get (8) $\tau_3 \equiv \forall \alpha :: \kappa.\tau_0$ and (9)$\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_0 <: \sigma_1$.

By IInst, MIInst, (8) and the premise, we get $\Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_0\{\sigma_0/\alpha\}$.

By (7) and Lemma 5, we get (9) $\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_0 <: \sigma_1$.

By Lemma 3, we get $\Delta \mid \Sigma \vdash \tau_0\{\sigma_0/\alpha\} <: \sigma_1\{\sigma_0/\alpha\}$.

Case Sub:

The result follows directly from I.H, STrans and Lemma 14.

□

Lemma 16 (Small step preservation).
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $e \mapsto e'$ then $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau/\rho$

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$.

Case Var and Shift:

These case cannot actually arise, since we assumed $e \mapsto e'$ and there are no reduction rules for variables and shifts .

Case Gen, Inst and Sub:

The result follows directly from I.H.

Case App:

The forms of e and e' are $e = (\lambda x.e_1)\ v$ and $e' = e_1\{v/x\}$ respectively, because of the $(\mapsto)$ relation.

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e_1 : \tau' \rightarrow_\rho \tau/\rho \qquad \Delta \mid \Gamma \mid \Sigma \vdash v : \tau'/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash (\lambda x.e_1)\ v : \tau/\rho}\ [\text{App}]$$

By Lemma 12, there exists τ_1, τ_2, ρ_1 and Δ' such that (1) $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e_1 : \tau_2/\rho_1$, (2) $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau_0$ and (3) $\Delta \mid \Sigma \vdash \tau_0 <: \tau' \rightarrow_\rho \tau$.

By Lemma 5 and (3), we get (4) $\tau_0 = \tau_1' \rightarrow_{\rho'} \tau_2'$, (5) $\Delta \mid \Sigma \vdash \tau' <: \tau_1'$, (6) $\Delta \mid \Sigma \vdash \rho' <: \rho$ and (7) $\Delta \mid \Sigma \vdash \tau_2 <: \tau$.

By Lemma 1, (1) and (4), we get (8) $\Delta \mid \Sigma \vdash \delta :: \Delta'$ and (9) $\delta(\tau_1 \rightarrow_{\rho_1} \tau_2) = \tau_1' \rightarrow_{\rho'} \tau_2'(= \tau_0)$.

By Lemma 9, (1) and (8), we get (10) $\Delta \mid \Gamma, x : \delta(\tau_1) \mid \Sigma \vdash e_1 : \delta(\tau_2)/\delta(\rho_1)$.

By the definition of a substitution, (9) and (10), we get (11) $\Delta \mid \Gamma, x : \tau_1' \mid \Sigma \vdash e_1 : \tau_2'/\rho'$.

By Sub, (6) and (7), we get (12) $\Delta \mid \Gamma, x : \tau_1' \mid \Sigma \vdash e_1 : \tau/\rho$.

By Sub, (5) and the premise, we get (13) $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_1'/\rho$.

By Lemma 10 and (13), we get (14) $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau_1'/\iota$.

Thus, we get $\Delta \mid \Gamma \mid \Sigma \vdash e_1\{v/x\} : \tau/\rho$ by Lemma 8, (12) and (14).

Case Reset: By the definition of the reduction rules, there are two subcases.

SubCase EReturn:

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta \mid \Gamma \mid \Sigma \vdash v : \tau' / (\exists_l\, \Delta'.\, \tau_0 / \rho_0) \cdot \delta(\rho_0)\end{array} \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau_0) / \delta(\rho_0)}{\Delta \mid \Gamma \mid \Sigma \vdash \langle v \mid x.\, e_r \rangle_l : \delta(\tau_0) / \delta(\rho_0)} \;[\textsc{Reset}]$$

By Lemma 10, we get $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau' / \iota$.
By Lemma 8, we get $\Delta \mid \Gamma \mid \Sigma \vdash e_r\{v/x\} : \delta(\tau_0) / \delta(\rho_0)$.

SubCase EReset:

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z. \langle E[z] \mid x.\, e_r \rangle_l}{\langle E[\mathbf{shift}_0 \langle l \rangle\; k.\, e] \mid x.\, e_r \rangle_l \mapsto e\{v_c / x\}} \;[\textsc{EReset}]$$

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau_0) / \delta(\rho_0) \\ \Delta \mid \Gamma \mid \Sigma \vdash E[\mathbf{shift}_0 \langle l \rangle\; k.\, e] : \tau' / (\exists_l\, \Delta'.\, \tau_0 / \rho_0) \cdot \delta(\rho_0)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle E[\mathbf{shift}_0 \langle l \rangle\; k.\, e] \mid x.\, e_r \rangle_l : \delta(\tau_0) / \delta(\rho_0)} \;[\textsc{Reset}]$$

By Lemma 11 and the premise, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0 \langle l \rangle\; k.\, e : \sigma / \rho'_0 * (\exists_l\, \Delta'.\, \tau_0 / \rho_0) \cdot \rho_0$.
By Lemma 15 and (1), we get (2) $\Delta, \Delta' \mid \Gamma, k : \tau'_1 \to_{\rho_1} \tau_1 \mid \Sigma \vdash e : \tau_1 / \rho'_1$, (3) $\Delta, \Delta' \mid \Sigma \vdash \rho'_1 <: \rho_1$, (4) $\Delta \mid \Sigma \vdash \tau'_1 :: \mathbf{T}$, (5) $\Delta \mid \Sigma \vdash (\exists_l\, \Delta'.\, \tau_1 / \rho_1) \cdot \rho'_1 :: \mathbf{R}$, (6) $\Delta \mid \Sigma \vdash (\exists_l\, \Delta'.\, \tau_1 / \rho_1) \cdot \rho'_1 <: (\exists_l\, \Delta'.\, \tau_0 / \rho_0) \cdot \rho'_0 * \delta(\rho_0)$, (7) $\Delta \mid \Sigma \vdash \tau'_1 \rightsquigarrow^* \tau''_1$ and (8) $\Delta \mid \Sigma \vdash \tau''_1 <: \sigma$.
By Lemma 1 and (7), we get (9) $\tau'_1 \equiv \forall \Delta''. \tau_2$, (10) $\Delta \mid \Sigma \vdash \delta_1 :: \Delta''$ and (11) $\tau''_1 \equiv \delta_1(\tau_2)$.
By (6) and Lemma 6, we get (12) $\Delta \mid \Sigma \vdash \tau_1 \equiv \tau_0$ and $\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_0$.
By (2), (3), (12) and Sub, we get (13) $\Delta, \Delta' \mid \Gamma, k : \tau'_1 \to_{\rho_0} \tau_0 \mid \Sigma \vdash e : \tau_0 / \rho_0$.
By Lemma 9, (4) and (13), we get (14) $\Delta \mid \Gamma, k : \tau'_1 \to_{\delta(\rho_0)} \delta(\tau_0) \mid \Sigma \vdash e : \delta(\tau_0) / \delta(\rho_0)$.
By Var, we get (15) $\Delta \mid \Gamma, z : \tau'_1 \mid \Sigma \vdash z : \tau'_1 / \iota$.
By Inst, (9), (10) and (11), we get (16) $\Delta \mid \Gamma, z : \tau'_1 \mid \Sigma \vdash z : \tau''_1 / \iota$.
By Sub, (8) and (16), we get (17) $\Delta \mid \Gamma, z : \tau'_1 \mid \Sigma \vdash z : \sigma / \rho'_0 * (\exists_l\, \Delta'.\, \tau_0 / \rho_0) \cdot \delta(\rho_0)$.
By Lemma 11 and (17), we get (18) $\Delta \mid \Gamma, z : \tau'_1 \mid \Sigma \vdash E[z] : \tau' / (\exists_l\, \Delta'.\, \tau_0 / \rho_0) \cdot \delta(\rho_0)$.
By Reset and the premises and (18), we get (19) $\Delta \mid \Gamma, z : \tau'_1 \mid \Sigma \vdash \langle E[z] \mid x.\, e_r \rangle_l : \delta(\tau_0) / \delta(\rho_0)$.
By Abs and (19), we get (20) $\Delta \mid \Gamma \mid \Sigma \vdash \lambda z. \langle E[z] \mid x.\, e_r \rangle_l : \tau'_1 \to_{\delta(\rho_0)} \delta(\tau_0) / \iota$.
Thus, we get $\Delta \mid \Gamma \mid \Sigma \vdash e\{v_c / x\} : \delta(\tau_0) / \delta(\rho_0)$ by Lemma 8, (14) and (20).

□

Lemma 17.
If $\Delta \mid \Sigma \vdash \rho <: \rho'$ and $\ell \in \lceil \rho \rceil$ then $\ell \in \lceil \rho' \rceil$.

Proof. Straightforward. □

Lemma 18 (Prompt tags are captured).
If $\Delta \mid \Gamma \mid \Sigma \vdash E[\mathbf{shift}_0 \langle \ell \rangle\; k.\, e] : \tau / \rho$ then $\ell \in \lceil E \rceil$ or $\ell \in \lceil \rho \rceil$

Proof. By Lemma 11, there exists σ and ρ' such that (1) $\Delta \mid \Gamma \mid \Sigma \vdash \textbf{shift}_0\langle \ell \rangle\, k.\, e : \sigma / \rho' * \rho$, where $\lceil \rho' \rceil = \lceil E \rceil$.

By (1) and Lemma 15, we get (2) $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\, \tau_0 / \rho_0) \cdot \rho'_0 < \rho' * \rho$.

By (2), $\ell \in \lceil (\exists_\ell\, \Delta'.\, \tau_0 / \rho_0) \cdot \rho'_0 \rceil$ and Lemma 17, we get $\ell \in \lceil \rho' * \rho \rceil$.

Thus, we get $\ell \in \lceil \rho' \rceil = \lceil E \rceil$ or $\ell \in \lceil \rho \rceil$. □

Lemma 19 (Progress with effects)**.**
If $\Delta \mid \emptyset \mid \Sigma \vdash e : \tau / \rho$ then e is value, $\exists\, e'$ s.t. $e \rightarrow e'$, or $e = E[\textbf{shift}_0\langle \ell \rangle\, k.\, e_0]$, where $\ell \notin \lceil E \rceil$

Proof. By induction on a derivation of $\Delta \mid \emptyset \mid \Sigma \vdash e : \tau / \rho$.

Case Var**:**

This case is impossible because the variable context Γ is empty.

Case Abs**:**

Straightforward.

Case App**:**

$$\dfrac{\Delta \mid \emptyset \mid \Sigma \vdash e_1 : \tau_1 \rightarrow_\rho \tau / \rho \qquad \Delta \mid \emptyset \mid \Sigma \vdash e_2 : \tau_1 / \rho}{\Delta \mid \emptyset \mid \Sigma \vdash e_1\, e_2 : \tau / \rho}\ [\textsc{App}]$$

,where $e = e_1\, e_2$.

We proceed by case analysis on the e_1.

SubCase $e_1 \rightarrow e'_1$**:**

$$\dfrac{e_0 \mapsto e'_0}{E[e_0] \rightarrow E[e'_0]}\ [\textsc{Step}]$$

,where $e_1 = E[e_0]$ and $e'_1 = E[e'_0]$.
Thus, we get a following derivation by Step.

$$\dfrac{e_0 \mapsto e'_0}{E[e_0]\, e_2 \rightarrow E[e'_0]\, e_2}\ [\textsc{Step}]$$

SubCase $e_1 = E_0[\textbf{shift}_0\langle \ell \rangle\, k.\, e_0]$**, where** $\ell \notin \lceil E_0 \rceil$**:**
We get $\ell \notin \lceil E_0\, e_2 \rceil$ because of $\lceil E_0\, e_2 \rceil = \lceil E_0 \rceil$.
Let us define E as $E_0\, e_2$.
Thus, we get $e = E[\textbf{shift}_0\langle \ell \rangle\, k.\, e_0]$, where $\ell \notin \lceil E \rceil$.

SubCase $e_1 = v_1$**:**
We also proceed by case analysis on the e_2.

SubSubCase $e_2 \rightarrow e'_2$**:**
This case is similar to the $e_1 \rightarrow e'_1$ case.

SubSubCase $e_2 = E_0[\textbf{shift}_0\langle \ell \rangle\, k.\, e_0]$**, where** $\ell \notin \lceil E_0 \rceil$**:**
This case is similar to the $e_1 = E_0[\textbf{shift}_0\langle \ell \rangle\, k.\, e_0]$.

SubSubCase $e_2 = v_2$**:**

As Γ is an empty context and v_1 is well-typed, v_1 is not a variable.
So, v_1 is a lambda abstraction.
Let us define v_1 as $\lambda x.e$.
We get $(\lambda x.e)\ v_2 \mapsto e\{v_2/x\}$ by applying the BETA.
Thus, we get $(\lambda x.e)\ v_2 \to e\{v_2/x\}$ by applying the STEP.

Case GEN, INST and SUB:

We can directly get the conclusion by I.H.

Case SHIFT$_0$:

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \ell :: \mathbf{L} \quad \Delta, \Delta' \mid \Sigma \vdash \rho' <: \rho_0 \quad \Delta \mid \Sigma \vdash \tau' :: \mathbf{T} \\ \Delta, \Delta' \mid k : \tau' \to_{\rho_0} \tau_0 \mid \Sigma \vdash e_0 : \tau_0/\rho' \quad \Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\ \tau_0/\rho_0) \cdot \rho' :: \mathbf{R}\end{array}}{\Delta \mid \varnothing \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\ k.\ e_0 : \tau'/(\exists_\ell\, \Delta'.\ \tau/\rho_0) \cdot \rho'}\ [\textsc{Shift}_0]$$

Let us define E as $[]$.

Thus, we get $e = E[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$, where $\ell \notin \lceil E \rceil$.

Case DOLLAR:

$$\frac{\begin{array}{c}\Delta \mid \varnothing \mid \Sigma \vdash e_1 : \tau'/(\exists_l\, \Delta'.\ \tau_1/\rho_1) \cdot \delta(\rho_1) \\ \Delta \mid \Sigma \vdash \delta :: \Delta' \quad \Delta \mid x : \tau' \mid \Sigma \vdash e_r : \delta(\tau_1)/\delta(\rho_1)\end{array}}{\Delta \mid \varnothing \mid \Sigma \vdash \langle e_1 \mid x.\ e_r \rangle_l : \delta(\tau_1)/\delta(\rho_1)}\ [\textsc{Dollar}]$$

By I.H, e_1 is a value, $e_1 \to e_1'$ or $e_1 = E_0[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$, where $\ell \notin \lceil E_0 \rceil$.

We proceed by case analysis on the e_1.

SubCase $e_1 = v$**:**

We get $\langle e_1 \mid x.\ e_r \rangle_l \mapsto e_r\{v/x\}$ by applying the RETURN.
Thus, we get $\langle e_1 \mid x.\ e_r \rangle_l \to e_r\{v/x\}$

SubCase $e_1 \to e_1'$**:**

We directly get $\langle e_1 \mid x.\ e_r \rangle_l \to \langle e_1 \mid x.\ e_r \rangle_l$ in a similar way to APP.

SubCase $e_1 = E_0[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$**, where** $\ell \notin \lceil E_0 \rceil$**:**

SubSubCase $\ell \neq l$**:**

Let us define E as $\langle E_0 \mid x.\ e_r \rangle_l$.
We get $\ell \notin \lceil E \rceil$ because of $\lceil E) \rceil = \lceil \langle E_0 \mid x.\ e_r \rangle_l \rceil = l, \lceil E_0 \rceil$. Thus, we get $e = E[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$, where $\ell \notin \lceil E \rceil$.

SubSubCase $\ell = l$**:**

We get $\langle E_0[\mathbf{shift}_0\langle l \rangle\ k.\ e_0] \mid x.\ e_r \rangle_l \to e_0\{v_c/k\}$ by applying the EDOLLAR and STEP, where $v_c = \lambda z.\langle E_0[z] \mid x.\ e_r \rangle_l$.

□

Theorem 1 (Preservation)**.**
If $\varnothing \mid \varnothing \mid \Sigma \vdash e : \tau/\rho$ and $e \to e'$ then $\varnothing \mid \varnothing \mid \Sigma \vdash e' : \tau/\rho$

Proof.

$$\frac{e_0 \mapsto e_0'}{E_0[e_0] \rightarrow E_0[e_0']}\ [\textsc{Step}]$$

,where $e = E_0[e_0]$ and $e' = E_0[e_0']$.

By Lemma 11, we get τ_0, ρ_0 and Δ' such that $\Delta' \mid \emptyset \mid \Sigma \vdash e_0 : \tau_0/\rho_0 * \rho$, where $\rho_0 = \lceil E_0 \rceil$.

By Lemma 16, we get $\Delta' \mid \emptyset \mid \Sigma \vdash e_0' : \tau_0/\rho_0 * \rho$.

Thus, we get $\emptyset \mid \emptyset \mid \Sigma \vdash e' : \tau/\rho$ by Lemma 11.

□

Theorem 2 (Progress)**.**
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau/\iota$ then e is value or $\exists\, e'$ s.t. $e \rightarrow e'$

Proof. By Lemma 19, e is a value, there exists e' such that $e \rightarrow e'$, or $e = E_0[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$, where $\ell \notin \lceil E_0 \rceil$.

Case e is value:
Straightforward.

Case $e \rightarrow e'$:
Straightforward.

Case $e = E_0[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$, where $\ell \notin \lceil E_0 \rceil$: This case is impossible, so we prove it by contradiction.
We assume that $e = E_0[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0]$, where $\ell \notin \lceil E_0 \rceil$. By Lemma 18 and $\emptyset \mid \emptyset \mid \Sigma \vdash E_0[\mathbf{shift}_0\langle \ell \rangle\ k.\ e_0] : \tau/\iota$, we can get following relationships.

$$\ell \in \lceil E_0 \rceil \text{ or } \ell \in \lceil \iota \rceil$$

ℓ is not an element of $\lceil \iota \rceil$, because of $\lceil \iota \rceil = \cdot$. Thus, we can get $\ell \in \lceil E_0 \rceil$. However, it is a contradiction to an assumption $\ell \notin \lceil E_0 \rceil$.

□

Appendix C

$\lambda^{l}_{\texttt{eff}}$: Algebraic Effect Handlers with Static Effect Instances

C.1 Syntax and Semantics

Syntax of Terms:

$$\begin{array}{llll} \text{Expressions} & e & ::= & \cdots \mid \textbf{do}\langle \ell \rangle\ v & \text{(do)} \\ & & \mid & \textbf{handle}\langle l \rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} & \text{(handle)} \end{array}$$

Syntax of Effects:

$$\text{Effects} \quad \epsilon \quad ::= \quad \exists_{\ell}\, \Delta'.\tau_1 \Rightarrow \tau_2$$

Evaluation context

$$E \ ::= \ \cdots \mid \textbf{handle}\langle l \rangle\ E\ \textbf{with}\ \{x, r.e_h; x.e_r\}$$

Count prompts

$$\lceil \Box \rceil \ ::= \ \emptyset \quad \lceil E\ e \rceil \ ::= \ \lceil E \rceil \quad \lceil v\ E \rceil \ ::= \ \lceil E \rceil \quad \lceil \textbf{handle}\langle l \rangle\ E\ \textbf{with}\ \{x, r.e_h; x.e_r\} \rceil \ ::= \ l, \lceil E \rceil$$

Reduction rules

$$\frac{}{\textbf{handle}\langle l \rangle\ v\ \textbf{with}\ \{x, r.e_h; x.e_r\} \mapsto e_r\{v/x\}}\ [\text{ERETURN}]$$

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z.\textbf{handle}\langle p \rangle\ E[z]\ \textbf{with}\ \{x, r.e_h; x.e_r\}}{\textbf{handle}\langle l \rangle\ E[\textbf{do}\langle l \rangle\ v]\ \textbf{with}\ \{x, r.e_h; x.e_r\} \mapsto e_h\{v/x, v_c/r\}}\ [\text{EHANDLE}]$$

FIGURE C.1: Algebraic effect handlers with effect instances

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho}$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1)/\iota \qquad \Delta \mid \Sigma \vdash \delta :: \Delta' \qquad \Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \delta(\tau_2)/(\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota}\ [\textsc{Do}]$$

$$\frac{\begin{array}{c}\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho \quad \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau/(\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \quad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r/\rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r/\rho}\ [\textsc{Handle}]$$

FIGURE C.2: Typing rules

$$\boxed{\Delta \mid \Sigma \vdash \tau :: \kappa}$$

$$\frac{\Delta, \Delta' \mid \Sigma \mid \tau_1 :: \mathbf{T} \qquad \Delta, \Delta' \mid \Sigma \mid \tau_2 :: \mathbf{T} \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}\ [\textsc{kLEff}]$$

FIGURE C.3: kinding rules

$$\boxed{\Delta \mid \Sigma \vdash \rho \equiv \rho'}$$

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau_1 \equiv \tau_1' \qquad \Delta, \Delta' \mid \Sigma \vdash \tau_2 \equiv \tau_2' \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 \equiv \exists_\ell\, \Delta'.\tau_1' \Rightarrow \tau_2'}\ [\textsc{EIEff}]$$

FIGURE C.4: Equivalence

C.2 Proof of Soundness

Definition 11 (Prompt extractor).

$$\begin{array}{lllllll} \lceil F \rceil & ::= & \cdot & \quad & \lceil F[\textbf{handle}\langle \ell \rangle\ E\ \textbf{with}\ \{x, r.e_h; x.e_r\}] \rceil & ::= & \ell, \lceil E \rceil \\ \lceil \iota \rceil & ::= & \cdot & \quad & \lceil \rho' * (\exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota \rceil & ::= & \ell, \lceil \rho' \rceil \end{array}$$

Definition 12 (Unhandled operations extractor).

$$\lfloor v \rfloor \ ::= \ 0 \quad \lfloor F[e] \rfloor \ ::= \ \lfloor e \rfloor \quad \lfloor \textbf{do}\langle \ell \rangle\ v \rfloor \ ::= \ 1 \quad \lfloor \textbf{handle}\langle \ell \rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} \rfloor \ ::= \ 0$$

Lemma 20 (Type variable substitution preserves equivalence).
If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau_1 \equiv \tau_2$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau_1) \equiv \delta(\tau_2)$.

Proof. Straightforward. □

Lemma 21 (Type variable substitution preserves subtyping relation).
If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau_1 <: \tau_2$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau_1) <: \delta(\tau_2)$.

Proof. Straightforward. □

Lemma 22 (Inversion lemma: equivalence).

- If $\Delta \mid \Sigma \vdash \iota \equiv \rho$ then $\rho = \iota$.
- If $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \equiv \rho_2$ then there exists ρ_2' $\Delta \mid \Sigma \vdash \rho_2 \equiv \epsilon \cdot \rho_2'$.

Proof. Straightforward. □

Lemma 23 (Inversion lemma: subtyping relation).

- If $\Delta \mid \Sigma \vdash \forall \Delta'.\tau <: \sigma$ then there exists τ_0 such that $\sigma = \forall \Delta'.\tau_0$ and $\Delta, \Delta' \mid \Sigma \vdash \tau <: \tau_0$
- If $\Delta \mid \Sigma \vdash \epsilon_1 \cdot \rho_1 <: \rho$ then there exist ϵ_2 and ρ_2 such that $\rho = \epsilon_2 \cdot \rho_2$, $\epsilon_1 \equiv \epsilon_2$, and $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$.
- If $\Delta \mid \Sigma \vdash \tau_1^1 \to_{\rho_1} \tau_2^1 <: \sigma$ then there exists τ_1^2, τ_2^2 and ρ_2 such that $\sigma = \tau_1^2 \to_{\rho_2} \tau_2^2$, $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_1^1$, $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$ and $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_2^2$.
- If $\Delta \mid \Sigma \vdash \sigma <: \tau_1^2 \to_{\rho_2} \tau_2^2$ then there exists τ_1^1, τ_2^1 and ρ_1 such that $\sigma = \tau_1^1 \to_{\rho_1} \tau_2^1$, $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_1^1$, $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$ and $\Delta \mid \Sigma \vdash \tau_1^2 <: \tau_2^2$.

Proof. Straightforward. □

Lemma 24 (Heads of effects are same under subtyping).
If $\Delta \mid \Sigma \vdash (\exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho_1 <: (\exists_\ell\ \Delta'.\tau_3 \Rightarrow \tau_4) \cdot \rho_2$ then $\Delta \mid \Sigma \vdash \tau_1 \equiv \tau_3$, $\Delta \mid \Sigma \vdash \tau_2 \equiv \tau_4$ and $\Delta \mid \Sigma \vdash \rho_1 <: \rho_2$.

Proof. Straightforward. □

Lemma 25 (Swapping effects preserves equivalence).
If $\Delta \mid \Sigma \vdash \rho_1 * \epsilon \cdot \rho_2 :: \mathbf{R}$ and $\lceil \epsilon \rceil \notin \lceil \rho_1 \rceil$ then $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \cdot \rho_2 :: \mathbf{R}$, $\Delta \mid \Sigma \vdash \rho_1 * \epsilon \cdot \rho_2 \equiv \epsilon \cdot \rho_1 * \rho_2$ and $\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 * \rho_2 \equiv \rho_1 * \epsilon \cdot \rho_2$.

Proof. Proof of this lemma is the same as Lemma 7.

□

Lemma 26 (Term substitution lemma)**.**
If $\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/\rho$ and $\Delta \mid \Gamma_1 \mid \Sigma \vdash e' : \sigma/\iota$ then $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau/\rho$.

Proof. We prove the difference cases only from Lemma 8. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 8 and prove DO and HANDLE cases.

Case DO:

$$\frac{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash v : \delta(\tau_1)/\iota \qquad \Delta \mid \Sigma \vdash \delta :: \Delta' \qquad \Delta \mid \Sigma \vdash \exists_{\ell}\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \delta(\tau_2)/(\exists_{\ell}\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota}\ [\textsc{Do}]$$

By I.H, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash v\{e'/x\} : \delta(\tau_1)/\iota$.

Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v\{e'/x\} : \delta(\tau_2)/(\exists_{\ell}\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota$ by DO.

Case HANDLE:

$$\frac{\begin{array}{c}\Delta, \Delta' \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau_1, r : \tau_2 \rightarrow_{\rho} \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho \qquad \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/(\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \qquad \Delta \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau \mid \Sigma \vdash e_r : \tau_r/\rho\end{array}}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; y.e_r\} : \tau_r/\rho}\ [\textsc{Handle}]$$

By I.H, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau/(\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$, $\Delta, \Delta' \mid \Gamma_1, \Gamma_2, y : \tau_1, r : \tau_2 \rightarrow_{\rho} \tau_r \mid \Sigma \vdash e_h\{e'/x\} : \tau_r/\rho$ and $\Delta \mid \Gamma_1, \Gamma_2, y : \tau \mid \Sigma \vdash e_r\{e'/x\} : \tau_r/\rho$.

Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ e\{e'/x\}\ \mathbf{with}\ \{y, r.e_h\{e'/x\}; r.e_r\{e'/x\}\} : \tau_r/\rho$ by HANDLE.

□

Lemma 27 (Type variable substitution lemma)**.**

1. If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau :: \kappa$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau) :: \kappa$
2. If $\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau)/\delta(\rho)$

Proof. We prove the difference cases only from Lemma 9.

1. We remove the KMEFF case from Lemma 9 and prove a KLEFF case.

   Case KLEFF:

$$\frac{\Delta_1, \Delta', \Delta_2, \Delta'' \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta_1, \Delta', \Delta_2, \Delta'' \mid \Sigma \vdash \tau_2 :: \mathbf{T} \qquad \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \exists_{\ell}\, \Delta''.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}\ [\textsc{KlEff}]$$

By I.H, we get $\Delta_1, \Delta_2, \Delta'' \mid \Sigma \vdash \delta(\tau_1) :: \mathbf{T}$, $\Delta_1, \Delta_2, \Delta'' \mid \Sigma \vdash \delta(\tau_2) :: \mathbf{T}$, and $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\ell) :: \mathbf{L}$.
Thus, we get $\Delta_1, \Delta_2 \mid \Sigma \vdash \exists_{\delta(\ell)}\, \Delta''.\delta(\tau_1) \Rightarrow \delta(\tau_2) :: \mathbf{E}$ by KiEff.

2. We remove the Shift$_0$ and Dollar cases from Lemma 9 and prove Do and Handle cases.

Case Do:

$$\dfrac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \delta_0 :: \Delta'' \\ \Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash v : \delta_0(\tau_1) / \iota \quad \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \delta_0(\tau_2) / (\exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \iota}\ [\textsc{Do}]$$

First, we prove (1) $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta \circ \delta_0 :: \Delta''$.
By the definition of a substitution composition, we get $\mathsf{dom}(\delta \circ \delta_0) = \mathsf{dom}(\delta_0) = \mathsf{dom}(\Delta'')$.
We get $\Delta_1, \Delta_2 \mid \Sigma \vdash (\delta \circ \delta_0)(\alpha) :: \Delta''(\alpha)$ for any $\alpha \in \mathsf{dom}(\Delta'')$, because of $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\beta) :: \Delta''(\beta)$ for any $\beta \in \mathsf{dom}(\Delta')$.
Second, we prove (2) $(\delta \circ \delta_0)(\delta(\alpha)) = \delta(\delta_0(\alpha)$ for any α.
We proceed by case analysis on the α.
Let us define $\Delta' = \alpha_1 :: \kappa_1, \cdots, \alpha :: \kappa, \cdots, \alpha_n :: \kappa_n$, $\Delta'' = \beta_1 :: \kappa'_1, \cdots, \beta :: \kappa', \cdots, \beta_n :: \kappa'_n, \delta_0 = \{\sigma_1/\alpha_1, \cdots, \sigma/\alpha, \cdots, \sigma_n/\alpha_n\}$ and $\delta = \{\sigma'_1/\beta_1, \cdots, \sigma'/\beta, \cdots, \sigma'_n/\beta_n\}$.

Case $\alpha \in \Delta'$**:** $(\delta \circ \delta_0)(\delta(\alpha)) = (\delta \circ \delta_0)(\alpha) = \delta(\sigma) = \delta(\delta_0(\alpha))$.

Case $\beta \in \Delta''$ $(\alpha = \beta)$**:** $(\delta \circ \delta_0)(\delta(\beta)) = (\delta \circ \delta_0)(\sigma') = \delta(\sigma') = \sigma' = \delta(\beta) = \delta(\delta_0(\beta))$.

Otherwise: $(\delta \circ \delta_0)(\delta(\alpha)) = \alpha = \delta(\delta_0(\alpha))$.

By I.H, we get (3) $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash v : \delta(\delta_0(\tau_1))/\iota$.
By (2) and (3), we get (4) $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash v : (\delta \circ \delta_0)(\delta((\tau_1))/\iota$.
By Lemma 27, we get (5) $\Delta_1, \Delta_2 \mid \Sigma \vdash \exists_{\delta(\ell)}\, \Delta''.\delta(\tau_1) \Rightarrow \delta(\tau_2) :: \mathbf{E}$.
Thus, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \delta(\delta_0(\tau_2))/\delta(\exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \iota$ by (1), (2), (4), (5) and Do.

Case Handle:

$$\dfrac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau/(\exists_l\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \rho \quad \Delta_1, \Delta', \Delta_2 \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r/\rho \\ \Delta_1, \Delta', \Delta_2, \Delta'' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho \quad \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash l :: \mathbf{L}\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r/\rho}\ [\textsc{Handle}]$$

By I.H, we get (1) $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau)/\delta((\exists_l\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \rho)$, (2) $\Delta_1, \Delta_2, \Delta'' \mid \delta(\Gamma), x : \delta(\tau_1), r : \delta(\tau_2) \to_{\delta(\rho)} \delta(\tau_r) \mid \Sigma \vdash e_h : \delta(\tau_r)/\delta(\rho)$, (3) $\Delta_1, \Delta_2 \mid \delta(\Gamma), x : \delta(\tau) \mid \Sigma \vdash e_r : \delta(\tau_r)/\delta(\rho)$, and (4) $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(l) :: \mathbf{L}$.
Thus, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.\delta(e_h); x.\delta(e_r)\} : \delta(\tau_r)/\delta(\rho)$ by (1), (2), (3), (4) and Handle.

□

Lemma 28 (Value is pure).
If $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\iota$.

Proof. We prove the difference cases only from Lemma 10. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 10 and prove DO and HANDLE cases.

Case DO and HANDLE:

This case is impossible because a form of the conclusion is not a value.

□

Lemma 29 (Compose/Decompose an evaluation context).
If $\Delta \mid \Gamma \mid \Sigma \vdash E[e] : \tau / \rho$ then there exists σ, ρ' such that

- $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma / \rho' * \rho$
- $\lceil E \rceil = \lceil \rho' \rceil$
- If $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma / \rho' * \rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash E[e'] : \tau / \rho$

Proof. We prove the difference cases only from Lemma 11. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 11 and prove DO and HANDLE cases.

Case $E \neq []$:

Case DO:

This case is impossible because the form of a conclusion is $E = []$.

Case HANDLE:

$$\frac{\begin{array}{cc}\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_{\rho} \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho & \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta \mid \Gamma \mid \Sigma \vdash E_0[e] : \tau / (\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho & \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ E_0[e]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{Handle}]$$

By I.H, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma / \rho' * (\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$, (2) $\lceil E_0 \rceil = \lceil \rho' \rceil$ and (3) $\Delta \mid \Gamma \mid \Sigma \vdash E_0[e'] : \tau / \rho$ for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma / \rho' * (\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$.
By the definition of $\lceil \cdot \rceil$, we get $\lceil E \rceil = \lceil \mathbf{handle}\langle l \rangle\ E_0\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rceil = l, \lceil E_0 \rceil = \lceil \rho' * (\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota \rceil$.
We get $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ E_0[e']\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho$ by the premise, HANDLE and (3), for any $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma / \rho' * (\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$.

□

Lemma 30 (Inversion lemma: lambda abstraction).
If $\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma / \rho$ then there exists τ_1, τ_2, ρ_1 and Δ' such that $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho_1$, $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \to_{\rho_1} \tau_2 \rightsquigarrow^{*} \tau'$ and $\Delta \mid \Sigma \vdash \tau' <: \tau$.

Proof. We prove the difference cases only from Lemma 12. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 12 and prove DO and HANDLE cases. Both of them cannot actually arise, because the form of a conclusion aren't lambda abstraction. □

Lemma 31 (Unhandled shift$_0$ operators).
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\lfloor e \rfloor \leqslant size(\rho)$.

Proof. We prove the difference cases only from Lemma 13. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 8 and prove DO and HANDLE cases.

Case Do:

$$\frac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash v : \delta_0(\tau_1)/\iota \\ \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \delta_0 :: \Delta'' \qquad \Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \delta_0(\tau_2)/(\exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \iota}\ [\textsc{Do}]$$

By the definition of $\lfloor\cdot\rfloor$, we get $\lfloor \mathbf{do}\langle\ell\rangle\ v \rfloor = 1$. By the definition of $size(\cdot)$, we get $size((\exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \iota) = 1$. Thus, we get $\lfloor e \rfloor \leqslant size((\exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \iota)$.

Case Handle:

$$\frac{\begin{array}{cc}\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho & \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau/(\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho & \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r/\rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r/\rho}\ [\textsc{Handle}]$$

By the definition of $\lfloor\cdot\rfloor$, we get $\lfloor \mathbf{handle}\langle l\rangle\ E_0[e]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rfloor = 0$.

By the definition of $size(\cdot)$, we get $size(\rho) \geqslant 0$.

Thus, we get $\lfloor \mathbf{handle}\langle l\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rfloor \leqslant size(\rho)$.

□

Lemma 32 (Operation performs an effect)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \tau/\rho$ then $\rho = (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho'$.

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle p\rangle\ v : \tau/\rho$.

Case Var, Abs, App and Handle:
These cases cannot actually arise, because the form of a conclusion aren't a do-operation.

Case Gen:
This case cannot actually arise, because the effect row of a conclusion have to be non-empty row.

Case Inst:

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \forall\alpha :: \kappa.\tau/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \tau\{\sigma/\alpha\}/\rho}\ [\textsc{Inst}]$$

By I.H, we get $\rho = (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho'$.

Case Sub:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \tau_1/\rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \tau_2/\rho_2}\ [\textsc{Sub}]$$

By I.H, we get $\rho_1 = (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho'$.

Thus, we get $\rho_2 = (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho''$ by Lemma 23 and the premise.

Case Do:

Straightforward.

□

Lemma 33 (Inversion lemma: do operation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \textbf{do}\langle \ell \rangle\ v : \tau / \epsilon \cdot \rho$ then there exist $\tau_1, \tau_2, \tau'', \Delta'$ such that

- $\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1) / \iota$
- $\Delta \mid \Sigma \vdash \delta :: \Delta'$
- $\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}$
- $\Delta \mid \Sigma \vdash \delta(\tau_2) \rightsquigarrow^* \tau''$
- $\Delta \mid \Sigma \vdash \tau'' <: \tau$
- $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota <: \epsilon \cdot \rho$.

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \textbf{do}\langle \ell \rangle\ v : \tau / \epsilon \cdot \rho$.

Case Abs, Handle, Var and App:

These cases cannot actually arise, because the form of a conclusion isn't a do-operation.

Case Gen:

This case cannot actually arise, because the effect row of a conclusion is an empty row.

Case Inst:

$$\frac{\Delta \mid \Sigma \vdash \sigma_0 :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash \textbf{do}\langle \ell \rangle\ v : \forall \alpha :: \kappa.\tau / \epsilon \cdot \rho}{\Delta \mid \Gamma \mid \Sigma \vdash \textbf{do}\langle \ell \rangle\ v : \tau\{\sigma_0/\alpha\} / \epsilon \cdot \rho}\ [\text{Inst}]$$

By I.H, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1)/\iota$, (2) $\Delta \mid \Sigma \vdash \delta :: \Delta'$, (3) $\Delta \mid \Sigma \vdash \exists_p\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}$, (4) $\Delta \mid \Sigma \vdash \delta(\tau_2) \rightsquigarrow^* \tau''$, (5) $\Delta \mid \Sigma \vdash \tau'' <: \forall \alpha :: \kappa.\tau$ and (6) $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota <: \epsilon \cdot \rho$.

By Lemma 23 and (5), we get (7) $\tau'' = \forall \alpha :: \kappa.\tau_0$ and (8) $\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau_0 <: \tau$.

By Lemma MIInst, (7), (8) and the premise, we get (9) $\Delta \mid \Sigma \vdash \delta(\tau_2) \rightsquigarrow^* \tau_0\{\sigma_0/\alpha\}$.

By Lemma 21 and (8), we get (10) $\Delta \mid \Sigma \vdash \tau_0\{\sigma_0/\alpha\} <: \tau\{\sigma_0/\alpha\}$.

Case Sub:

The result follows direcly from I.H, STrans and Lemma 32.

□

Lemma 34 (Small step preservation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $e \mapsto e'$ then $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau/\rho$

Proof. We prove the difference cases only from Lemma 16. We remove the Shift$_0$ and Dollar cases from Lemma 8 and prove Do and Handle cases.

Case Do:

This case cannot actually arise, since there are no reduction rules for an operation.

Case Handle:

We proceed by case analysis on a reduction rule.

SubCase EReturn:

$$\frac{\begin{array}{cc} \Delta \mid \Gamma \mid \Sigma \vdash v : \tau / (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho & \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho & \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho \end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ v\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{Handle}]$$

By Lemma 28, we get $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau / \iota$. By th premise and Lemma 26, we get $\Delta \mid \Gamma \mid \Sigma \vdash e_r\{v/x\} : \tau_r / \rho$.

SubCase EHandle:

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z.\mathbf{handle}\langle l \rangle\ E[z]\ \mathbf{with}\ \{x, r.e_h; x.e_r\}}{\mathbf{handle}\langle l \rangle\ E[\mathbf{do}\langle l \rangle\ v]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \mapsto e_h\{v/x, v_c/r\}}\ [\text{EH}\textsc{andle}]$$

$$\frac{\begin{array}{cc} \Delta \mid \Gamma \mid \Sigma \vdash E[\mathbf{do}\langle l \rangle\ v] : \tau / (\exists_p\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho & \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho & \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho \end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ E[\mathbf{do}\langle l \rangle\ v]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{Handle}]$$

By Lemma 29, we get (1) $\Delta \mid \Sigma \vdash \mathbf{do}\langle l \rangle\ v : \sigma / \rho' * (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$, and (2) $\lceil E \rceil = \lceil \rho' \rceil$.

By Lemma 33 and (1), we get (3) $\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1') / \iota$, (4) $\Delta \mid \Gamma \mid \Sigma \vdash \delta :: \Delta'$, (5) $\Delta \mid \Sigma \vdash \exists_l\ \Delta'.\tau_1' \Rightarrow \tau_2' :: \mathbf{E}$, (6) $\Delta \mid \Sigma \vdash \delta(\tau_2') \leadsto^* \tau''$, (7) $\Delta \mid \Sigma \vdash \tau'' <: \sigma$ and (8) $\Delta \mid \Sigma \vdash (\exists_l\ \Delta'.\tau_1' \Rightarrow \tau_2') \cdot \iota <: (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho' * \rho$.

By Lemma 1 and (6), we get (9) $\delta(\tau_2) = \forall \Delta''.\tau_0$, (10) $\Delta \mid \Sigma \vdash \delta_1 :: \Delta''$ and (11) $\tau'' = \delta_1(\tau_0)$.

By (8) and Lemma 5, we get (12) $\Delta, \Delta' \mid \Sigma \vdash \tau_1 \equiv \tau_1'$ and (13) $\Delta, \Delta' \mid \Sigma \vdash \tau_2 \equiv \tau_2'$.

By Var and Sub, we get (14) $\Delta \mid \Gamma, z : \delta(\tau_2) \mid \Sigma \vdash z : \delta(\tau_2) / \rho' * (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$.

By Inst, (9), (10), (11) and (14), we get (15) $\Delta \mid \Gamma, z : \delta(\tau_2) \mid \Sigma \vdash z : \tau'' / \rho' * (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$.

By Sub, (7) and (15), we get (16) $\Delta \mid \Gamma, z : \delta(\tau_2) \mid \Sigma \vdash z : \sigma / \rho' * (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$.

By Lemma 29 and (16), we get (17) $\Delta \mid \Gamma, z : \delta(\tau_2) \mid \Sigma \vdash E[z] : \tau / (\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$.

By Weakening, the premises and (17), we get (18) $\Delta, \Delta' \mid \Gamma, z : \delta(\tau_2), x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho$ and (19) $\Delta \mid \Gamma, z : \delta(\tau_2), x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho$.

By Handle, (17), (18), (19) and the premise, we get (20) $\Delta \mid \Gamma, z : \delta(\tau_2) \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ E[z]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho$.

By ABS and (20), we get (21) $\Delta \mid \Gamma \mid \Sigma \vdash \lambda z.\mathbf{handle}\langle l\rangle\ E[z]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \delta(\tau_2) \rightarrow_\rho \tau_r/\iota$.

By Lemma 27, (4), the premise and (21), we get (22) $\Delta \mid \Gamma, x : \delta(\tau_1), r : \delta(\tau_2) \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho$.

Thus, we get $\Delta \mid \Gamma, x : \delta(\tau_1), r : \delta(\tau_2) \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h\{v/x, v_c/r\} : \tau_r/\rho$ by Lemma 26, (3), (21) and (22).

□

Lemma 35.
If $\Delta \mid \Sigma \vdash \rho <: \rho'$ and $\ell \in \lceil\rho\rceil$ then $p\ell \in \lceil\rho'\rceil$.

Proof. Straightforward. □

Lemma 36 (Effect instances are captured).
If $\Delta \mid \Gamma \mid \Sigma \vdash E[\mathbf{do}\langle\ell\rangle\ v] : \tau/\rho$ then $\ell \in \lceil E\rceil$ or $\ell \in \lceil\rho\rceil$.

Proof. By Lemma 29, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle p\rangle\ v : \sigma/\rho' * \rho$, where $\lceil\rho'\rceil = \lceil E\rceil$.
By (1) and Lemma 33, we get (2) $\Delta \mid \Sigma \vdash (\exists_\ell \Delta'.\tau_1 \Rightarrow \tau_2)\cdot\iota <: \rho'\cdot\rho$.
By (2), $p \in \lceil(\exists_\ell \Delta'.\tau_1 \Rightarrow \tau_2)\cdot\iota\rceil$ and Lemma 35, we get (3) $p \in \lceil\rho'\cdot\rho\rceil$.
Thus, we get $\ell \in \lceil\rho'\rceil$ or $\ell \in \lceil\rho\rceil$. □

Lemma 37 (Progress with effects).
If $\Delta \mid \varnothing \mid \Sigma \vdash e : \tau/\rho$ then e is value, $\exists\, e'$ s.t. $e \rightarrow e'$, or $e = E[\mathbf{do}\langle\ell\rangle\ v]$, where $\ell \notin \lceil E\rceil$

Proof. We prove the difference cases only from Lemma 19. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 19 and prove DO and HANDLE cases.

Case DO:

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash v : \delta_0(\tau_1)/\iota \qquad \Delta \mid \Sigma \vdash \delta_0 :: \Delta'' \qquad \Delta \mid \Sigma \vdash \exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\ v : \delta_0(\tau_2)/(\exists_\ell\, \Delta''.\tau_1 \Rightarrow \tau_2)\cdot\iota}\ [\textsc{Do}]$$

Let us define E as $[]$.

Thus, we get $e = E[\mathbf{do}\langle\ell\rangle\ v]$, where $\ell \notin \lceil E\rceil$.

Case HANDLE:

$$\frac{\begin{array}{cc} \Delta \mid \Gamma \mid \Sigma \vdash e_1 : \tau/(\exists_l\, \Delta'.\tau_1 \Rightarrow \tau_2)\cdot\rho & \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho & \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r/\rho \end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l\rangle\ e_1\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r/\rho}\ [\textsc{Handle}]$$

By I.H, we get (1) e_1 is a value, (2) $e_1 \rightarrow e_1'$, or (3) $e_1 = E_0[\mathbf{do}\langle\ell\rangle\ v]$, where $\ell \notin \lceil E_0\rceil$.

We proceed by case analysis on e_1.

SubCase $e_1 = v$:
We get $\mathbf{handle}\langle l\rangle\ v\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \mapsto e_r\{v/x\}$ by ERETURN.
Thus, we get $\mathbf{handle}\langle l\rangle\ v\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rightarrow e_r\{v/x\}$.

SubCase $e_1 \rightarrow e_1'$:
We get $\mathbf{handle}\langle l\rangle\ e_1\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rightarrow \mathbf{handle}\langle l\rangle\ e_1'\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$ in a similar way to APP.

SubCase $e_1 = E_0[\mathbf{do}\langle p\rangle\ v]$, where $\ell \notin \lceil E_0 \rceil$:

SubSubCase $\ell \neq l$:

Let us define E as $\mathbf{handle}\langle l\rangle\ E_0\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$.
We get $\ell \notin \lceil E \rceil$ because of $\lceil E \rceil = \lceil \mathbf{handle}\langle l\rangle\ E_0\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rceil = \ell, \lceil E_0 \rceil$.
Thus, we get $e = E[\mathbf{do}\langle l\rangle\ v]$, where $\ell \notin \lceil E \rceil$.

SubSubCase $\ell = l$:

We get $\mathbf{handle}\langle l\rangle\ E_0[\mathbf{do}\langle l\rangle\ v]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \longrightarrow e_h\{v_c/r\}\{v/x\}$ by EHANDLE, where $v_c = \lambda z.\mathbf{handle}\langle l\rangle\ E_0[z]\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$.

□

Theorem 3 (Preservation).
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau / \rho$ and $e \longrightarrow e'$ then $\emptyset \mid \emptyset \mid \Sigma \vdash e' : \tau / \rho$

Proof. We can prove this lemma by a similar way to Lemma 1 using Lemma 29 and Lemma 34 instead of Lemma 11 and Lemma 16. □

Theorem 4 (Progress).
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau / \rho$ then e is value or $\exists\, e'$ s.t. $e \longrightarrow e'$

Proof. We prove the difference cases only from Lemma 1. We remove the "$e = E_0[\mathbf{shift}_0\langle \ell\rangle\ k.\ e_0]$, where $\ell \notin \lceil E_0 \rceil$" case from Lemma 1 and prove "$e = E_0[\mathbf{shift}_0\langle \ell\rangle\ k.\ e_0]$, where $\ell \notin \lceil E_0 \rceil$" case.

Case $e = E_0[\mathbf{do}\langle \ell\rangle\ v]$, where $\ell \notin \lceil E_0 \rceil$:

This case is impossible, so we prove it by contradiction.

We assume that $e = E_0[\mathbf{do}\langle \ell\rangle\ v]$, where $\ell \notin \lceil E_0 \rceil$.

By Lemma 36 and $\emptyset \mid \emptyset \mid \Sigma \vdash E_0[\mathbf{do}\langle \ell\rangle\ v] : \tau / \iota$, we can get following relationships.

$$\ell \in \lceil E_0 \rceil \text{ or } \ell \in \lceil \iota \rceil$$

ℓ is not an element of $\lceil \iota \rceil$, because of $\lceil \iota \rceil = \cdot$.

Thus, we can get $\ell \in \lceil E_0 \rceil$.

However, it is a contradiction to an assumption $\ell \notin \lceil E_0 \rceil$.

□

Appendix D

Macro Translations Between $\lambda^{l}_{\texttt{del}}$ and $\lambda^{l}_{\texttt{eff}}$

D.1 Macro Translation from $\lambda^{l}_{\texttt{del}}$ to $\lambda^{l}_{\texttt{eff}}$

$$\begin{aligned}
\llbracket x \rrbracket^{\mathbf{PI}} &= x \\
\llbracket \lambda x.e \rrbracket^{\mathbf{PI}} &= \lambda x.\llbracket e \rrbracket^{\mathbf{PI}} \\
\llbracket e\ e \rrbracket^{\mathbf{PI}} &= \llbracket e \rrbracket^{\mathbf{PI}}\ \llbracket e \rrbracket^{\mathbf{PI}} \\
\llbracket \mathbf{shift}_0\langle \ell \rangle\ k.\ e \rrbracket^{\mathbf{PI}} &= \mathbf{do}\langle \ell \rangle\ (\lambda k.\llbracket e \rrbracket^{\mathbf{PI}}) \\
\llbracket \langle e \mid x.\ e_r \rangle_l \rrbracket^{\mathbf{PI}} &= \mathbf{handle}\langle l \rangle\ \llbracket e \rrbracket^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r \rrbracket^{\mathbf{PI}}\}
\end{aligned}$$

FIGURE D.1: Macro translation of terms

$$\begin{aligned}
\llbracket \alpha \rrbracket^{\mathbf{PI}} &= \alpha \\
\llbracket \tau \to_{\rho} \tau \rrbracket^{\mathbf{PI}} &= \llbracket \tau \rrbracket^{\mathbf{PI}} \to_{\llbracket \rho \rrbracket^{\mathbf{PI}}} \llbracket \tau \rrbracket^{\mathbf{PI}} \\
\llbracket \forall \alpha :: \kappa.\tau \rrbracket^{\mathbf{PI}} &= \forall \alpha :: \kappa.\llbracket \tau \rrbracket^{\mathbf{PI}} \\
\llbracket \epsilon \cdot \rho \rrbracket^{\mathbf{PI}} &= \llbracket \epsilon \rrbracket^{\mathbf{PI}} \cdot \llbracket \rho \rrbracket^{\mathbf{PI}} \\
\llbracket \exists_{\ell}\ \Delta'.\ \tau / \rho \rrbracket^{\mathbf{PI}} &= \exists_{\ell}\ \alpha :: \mathbf{T}.(\forall \Delta'.(\alpha \to_{\llbracket \rho \rrbracket^{\mathbf{PI}}} \llbracket \tau \rrbracket^{\mathbf{PI}}) \to_{\llbracket \rho \rrbracket^{\mathbf{PI}}} \llbracket \tau \rrbracket^{\mathbf{PI}}) \Rightarrow \alpha
\end{aligned}$$

FIGURE D.2: Macro translation of types

$$\begin{aligned}
\llbracket \varnothing \rrbracket^{\mathbf{PI}} &= \varnothing \\
\llbracket \Gamma, x : \tau \rrbracket^{\mathbf{PI}} &= \llbracket \Gamma \rrbracket^{\mathbf{PI}}, x : \llbracket \tau \rrbracket^{\mathbf{PI}}
\end{aligned}$$

FIGURE D.3: Macro translation of contexts

$$[\![\{\overline{\tau/\alpha}\}]\!]^{\mathbf{PI}} \quad = \quad \{\overline{[\![\tau]\!]^{\mathbf{PI}}/\alpha}\}$$

FIGURE D.4: Macro translation of substitutions

$$\begin{array}{lcl}
[\![\square]\!]^{\mathbf{PI}} & = & \square \\
[\![E\ e]\!]^{\mathbf{PI}} & = & [\![E]\!]^{\mathbf{PI}}\ [\![e]\!]^{\mathbf{PI}} \\
[\![v\ E]\!]^{\mathbf{PI}} & = & [\![v]\!]^{\mathbf{PI}}\ [\![E]\!]^{\mathbf{PI}} \\
[\![\langle E \mid x.\ e_r\rangle_l]\!]^{\mathbf{PI}} & = & \mathbf{handle}\langle l\rangle\ [\![E]\!]^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.[\![e_r]\!]^{\mathbf{PI}}\}
\end{array}$$

FIGURE D.5: Macro translation of evaluation contexts

$$\frac{}{e \rightarrow^{*} e}\ [\text{MSREFL}] \qquad \frac{e_1 \rightarrow e_2}{e_1 \rightarrow^{*} e_2}\ [\text{MSRED}] \qquad \frac{e_1 \rightarrow^{*} e_2 \qquad e_2 \rightarrow^{*} e_3}{e_1 \rightarrow^{*} e_3}\ [\text{MSTRANS}]$$

FIGURE D.6: Multi-step reductions

$$\frac{e_1 \rightarrow e_2 \qquad e_2 \rightarrow^{*} e_3}{e_1 \rightarrow^{+} e_3}\ [\text{S-MSRED}]$$

FIGURE D.7: Strict multi-step reductions

D.2 Proof of Meaning and Typability Preservation Properties

Lemma 38 (Translation preserves equivalence)**.**
If $\Delta \mid \Sigma \vdash \tau \equiv \tau'$ then $\Delta \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{PI}} \equiv [\![\tau']\!]^{\mathbf{PI}}$.

Proof. By induction on a derivation of $\Delta \mid \Sigma \vdash \tau \equiv \tau'$.

Case $\textbf{E}\textsc{Refl}$**:**

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa}{\Delta \mid \Sigma \vdash \sigma \equiv \sigma}\ [\text{E}\textsc{Refl}]$$

By Lemma 40, we get $\Delta \mid \Sigma \vdash [\![\sigma]\!]^{\mathbf{PI}} :: \kappa$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\sigma]\!]^{\mathbf{PI}} \equiv [\![\sigma]\!]^{\mathbf{PI}}$.

Case $\textbf{E}\textsc{Arrow}$**,** $\textbf{E}\textsc{Gen}$**,** $\textbf{E}\textsc{Trans}$**:**
The result follows directly from I.H.

Case $\textbf{E}\textsc{Row}$**:**

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 \equiv \epsilon \cdot \rho_2}\ [\text{E}\textsc{Row}]$$

By I.H, we get (1) $\Delta \mid \Sigma \vdash [\![\rho_1]\!]^{\mathbf{PI}} \equiv [\![\rho_2]\!]^{\mathbf{PI}}$.

By Lemma 40, we get (2) $\Delta \mid \Sigma \vdash [\![\epsilon]\!]^{\mathbf{PI}} :: \mathbf{E}$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\epsilon \cdot \rho_1]\!]^{\mathbf{PI}} \equiv [\![\epsilon \cdot \rho_2]\!]^{\mathbf{PI}}$.

Case $\textbf{E}\textsc{Swap}$**:**

$$\frac{\Delta \mid \Sigma \vdash \rho_1 \equiv \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon_1 :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \epsilon_2 :: \mathbf{E} \qquad \lceil \epsilon_1 \rceil \neq \lceil \epsilon_2 \rceil}{\Delta \mid \Sigma \vdash \epsilon_1 \cdot \epsilon_2 \cdot \rho_1 \equiv \epsilon_2 \cdot \epsilon_1 \cdot \rho_2}\ [\text{E}\textsc{Swap}]$$

By I.H, we get (1) $\Delta \mid \Sigma \vdash [\![\rho_1]\!]^{\mathbf{PI}} \equiv [\![\rho_2]\!]^{\mathbf{PI}}$.

By Lemma 40, we get (2) $\Delta \mid \Sigma \vdash [\![\epsilon_1]\!]^{\mathbf{PI}} :: \mathbf{E}$ and (3) $\Delta \mid \Sigma \vdash [\![\epsilon_2]\!]^{\mathbf{PI}} :: \mathbf{E}$.

By the definition of a $[\![\cdot]\!]^{\mathbf{PI}}$, we get (4) $\lceil [\![\epsilon_1]\!]^{\mathbf{PI}} \rceil \neq \lceil [\![\epsilon_2]\!]^{\mathbf{PI}} \rceil$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\epsilon_1 \cdot \epsilon_2 \cdot \rho_1]\!]^{\mathbf{PI}} \equiv [\![\epsilon_2 \cdot \epsilon_1 \cdot \rho_2]\!]^{\mathbf{PI}}$ by E$\textsc{Swap}$.

Case $\textbf{EME}\textsc{ff}$**:**

$$\frac{\Delta, \Delta' \mid \Sigma \vdash \tau_1 \equiv \tau_1' \qquad \Delta, \Delta' \mid \Sigma \vdash \tau_2 \equiv \tau_2'}{\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 \equiv \exists_\ell\, \Delta'.\tau_1' \Rightarrow \tau_2'}\ [\text{EME}\textsc{ff}]$$

□

Lemma 39 (Translation preserves subtyping relations)**.**
If $\Delta \mid \Sigma \vdash \tau <: \tau'$ then $\Delta \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{PI}} <: [\![\tau']\!]^{\mathbf{PI}}$.

Proof. By induction on a derivation of $\Delta \mid \Sigma \vdash \tau <: \tau'$.

Case SREFL:

$$\frac{\Delta \mid \Sigma \vdash \sigma_1 \equiv \sigma_2}{\Delta \mid \Sigma \vdash \sigma_1 <: \sigma_2}\ [\textsc{SRefl}]$$

By Lemma 38, we get $\Delta \mid \Sigma \vdash [\![\sigma_1]\!]^{\mathbf{PI}} \equiv [\![\sigma_2]\!]^{\mathbf{PI}}$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\sigma_1]\!]^{\mathbf{PI}} <: [\![\sigma_2]\!]^{\mathbf{PI}}$ by SREFL.

Case SARROW, SGEN and STRANS:
The result follows directly from I.H.

Case SEMPTY:

$$\frac{\Delta \mid \Sigma \vdash \rho :: \mathbf{R}}{\Delta \mid \Sigma \vdash \iota <: \rho}\ [\textsc{SEmpty}]$$

By Lemma 40, we get $\Delta \mid \Sigma \vdash [\![\rho]\!]^{\mathbf{PI}} :: \mathbf{R}$.

By the definition of a $[\![\cdot]\!]^{\mathbf{PI}}$, we get $[\![\iota]\!]^{\mathbf{PI}} = \iota$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\iota]\!]^{\mathbf{PI}} <: [\![\rho]\!]^{\mathbf{PI}}$ by SEMPTY.

Case SROW:

$$\frac{\Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Sigma \vdash \epsilon :: \mathbf{E}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho_1 <: \epsilon \cdot \rho_2}\ [\textsc{SRow}]$$

By I.H, we get $\Delta \mid \Sigma \vdash [\![\rho_1]\!]^{\mathbf{PI}} <: [\![\rho_2]\!]^{\mathbf{PI}}$.

By Lemma 40, we get $\Delta \mid \Sigma \vdash [\![\epsilon]\!]^{\mathbf{PI}} :: \mathbf{E}$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\epsilon \cdot \rho_1]\!]^{\mathbf{PI}} <: [\![\epsilon \cdot \rho_2]\!]^{\mathbf{PI}}$ by SROW.

□

Lemma 40 (Translation preserves kindings).
If $\Delta \mid \Sigma \vdash \tau :: \kappa$ then $\Delta \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{PI}} :: \kappa$.

Proof. By induction of a derivation of $\Delta \mid \Sigma \vdash \tau :: \kappa$.

Case KVAR:

$$\frac{\alpha :: \kappa \in \Delta}{\Delta \mid \Sigma \vdash \alpha :: \kappa}\ [\textsc{KVar}]$$

By the definition of $[\![\cdot]\!]^{\mathbf{PI}}$, we get $[\![\alpha]\!]^{\mathbf{PI}} = \alpha$.

Thus, we get $\Delta \mid \Sigma \vdash [\![\alpha]\!]^{\mathbf{PI}} :: \kappa$.

Case KARROW:

$$\frac{\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T} \qquad \Delta \mid \Sigma \vdash \rho :: \mathbf{R} \qquad \Delta \mid \Sigma \vdash \tau_2 :: \mathbf{T}}{\Delta \mid \Sigma \vdash \tau_1 \to_\rho \tau_2 :: \mathbf{T}}\ [\text{KArrow}]$$

By I.H, we get $\Delta \mid \Sigma \vdash [\![\tau_1]\!]^{\mathbf{PI}} :: \mathbf{T}$, $\Delta \mid \Sigma \vdash [\![\rho]\!]^{\mathbf{PI}} :: \mathbf{R}$ and $\Delta \mid \Sigma \vdash [\![\tau_2]\!]^{\mathbf{PI}} :: \mathbf{T}$.
Thus, we get $\Delta \mid \Sigma \vdash [\![\tau_1 \to_\rho \tau_2]\!]^{\mathbf{PI}} :: \mathbf{T}$ by KArrow.

Case KGen:

$$\frac{\Delta, \alpha :: \kappa \mid \Sigma \vdash \tau :: \mathbf{T}}{\Delta \mid \Sigma \vdash \forall \alpha :: \kappa.\tau :: \mathbf{T}}\ [\text{KGen}]$$

By I.H, we get $\Delta, \alpha :: \kappa \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{PI}} :: \mathbf{T}$.
Thus, we get $\Delta \mid \Sigma \vdash \forall [\![\alpha :: \kappa.\tau]\!]^{\mathbf{PI}} :: \mathbf{T}$ by KGen.

Case KEmpty:
Straightforward.

Case KRow:

$$\frac{\Delta \mid \Sigma \vdash \epsilon :: \mathbf{E} \qquad \Delta \mid \Sigma \vdash \rho :: \mathbf{R}}{\Delta \mid \Sigma \vdash \epsilon \cdot \rho :: \mathbf{R}}\ [\text{KRow}]$$

By I.H, we get (1) $\Delta \mid \Sigma \vdash [\![\epsilon]\!]^{\mathbf{PI}} :: \mathbf{E}$ (2) $\Delta \mid \Sigma \vdash [\![\rho]\!]^{\mathbf{PI}} :: \mathbf{R}$.
Thus, we get $\Delta \mid \Sigma \vdash [\![\epsilon \cdot \rho]\!]^{\mathbf{PI}} :: \mathbf{R}$.

Case KMEff:
Straightforward.

□

Lemma 41 (Translation preserves well-formedness of substitutions)**.**
If $\Delta \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta \mid \Sigma \vdash [\![\delta]\!]^{\mathbf{PI}} :: \Delta'$.

Proof.
Let us define $\delta = \{\sigma_1/\alpha_1, \cdots, \sigma_n/\alpha_n\}$ and $\Delta' = \sigma_1 :: \kappa_1, \ldots, \sigma_n :: \kappa_n$. By the definition of $\Delta \mid \Sigma \vdash \delta :: \Delta'$, we get $\mathsf{dom}(\delta) = \mathsf{dom}(\Delta') = \alpha_1, \ldots, \alpha_n$. $\Delta \mid \Sigma \vdash \delta(\alpha_i) :: \Delta(\alpha_i)$, for any $\alpha_i \in \mathsf{dom}(\delta)$, (i.e., $\Delta \mid \Sigma \vdash \sigma_i :: \kappa_i$, for any i). By the definition of $[\![\cdot]\!]^{\mathbf{PI}}$, we get $[\![\delta]\!]^{\mathbf{PI}} = \{[\![\sigma_1]\!]^{\mathbf{PI}}/\alpha_1, \cdots, [\![\sigma_n]\!]^{\mathbf{PI}}/\alpha_n\}$. Then, we get $\mathsf{dom}([\![\delta]\!]^{\mathbf{PI}}) = \alpha_1, \ldots, \alpha_n = \Delta'$. By Lemma 40, we get $\Delta \mid \Sigma \vdash [\![\sigma_i]\!]^{\mathbf{PI}} :: \kappa_i$, for any i (i.e., $\Delta \mid \Sigma \vdash [\![\delta(\alpha_i)]\!]^{\mathbf{PI}} :: \Delta'(\alpha_i)$, for any $\alpha_i \in \mathsf{dom}(\Delta')$). Thus, we get $\Delta \mid \Sigma \vdash [\![\delta]\!]^{\mathbf{PI}} :: \Delta'$. □

Lemma 42 (Translation of types is commutative)**.**
$[\![\delta(\tau)]\!]^{\mathbf{PI}} = [\![\delta]\!]^{\mathbf{PI}}([\![\tau]\!]^{\mathbf{PI}})$, for any type τ.

Proof. By induction on the structure of τ.
Let us define $\delta = \{\sigma_1/\alpha_1, \cdots, \sigma_n/\alpha_n\}$.

Case $\tau = \alpha$: We proceed by case analysis on a α.

SubCase $\alpha\ (= \alpha_i) \in \mathsf{dom}(\delta)$: $[\![\delta]\!]^{\mathbf{PI}}([\![\alpha_i]\!]^{\mathbf{PI}}) = [\![\delta]\!]^{\mathbf{PI}}(\alpha_i) = [\![\sigma_i]\!]^{\mathbf{PI}} = [\![\delta(\alpha_i)]\!]^{\mathbf{PI}}$.

SubCase $\alpha \notin \mathsf{dom}(\delta)$**:** $\llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\alpha\rrbracket^{\mathbf{PI}}) = \llbracket\delta\rrbracket^{\mathbf{PI}}(\alpha) = \alpha = \llbracket\alpha\rrbracket^{\mathbf{PI}} = \llbracket\delta(\alpha)\rrbracket^{\mathbf{PI}}$.

Case otherwise:
The result follows directly from I.H.

□

Theorem 5 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ then $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau\rrbracket^{\mathbf{PI}}/\llbracket\rho\rrbracket^{\mathbf{PI}}$.

Proof. By induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$.

Case **Var:**

$$\frac{x : \tau \in \Gamma}{\Delta \mid \Gamma \mid \Sigma \vdash x : \tau/\iota}\ [\textsc{Var}]$$

By the definition of $\llbracket\cdot\rrbracket^{\mathbf{PI}}$, we get $x : \llbracket\tau\rrbracket^{\mathbf{PI}} \in \llbracket\Gamma\rrbracket^{\mathbf{PI}}$. Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash x : \llbracket\tau\rrbracket^{\mathbf{PI}}/\llbracket\iota\rrbracket^{\mathbf{PI}}$.

Case **Abs, App, Gen:**
The result follows directly from I.H.

Case **Inst:**

$$\frac{\Delta \mid \Sigma \vdash \sigma :: \kappa \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \forall\alpha :: \kappa.\tau/\rho}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau\{\sigma/\alpha\}/\rho}\ [\textsc{Inst}]$$

By Lemma 40, we get $\Delta \mid \Sigma \vdash \llbracket\sigma\rrbracket^{\mathbf{PI}} :: \kappa$

By I.H, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \forall\alpha :: \kappa.\llbracket\tau\rrbracket^{\mathbf{PI}}/\llbracket\rho\rrbracket^{\mathbf{PI}}$.

By Inst, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau\rrbracket^{\mathbf{PI}}\{\llbracket\sigma\rrbracket^{\mathbf{PI}}/\alpha\}/\llbracket\rho\rrbracket^{\mathbf{PI}}$.

Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau\{\sigma/\alpha\}\rrbracket^{\mathbf{PI}}/\llbracket\rho\rrbracket^{\mathbf{PI}}$ by Lemma 42.

Case **Sub:**

$$\frac{\Delta \mid \Sigma \vdash \tau_1 <: \tau_2 \qquad \Delta \mid \Sigma \vdash \rho_1 <: \rho_2 \qquad \Delta \mid \Gamma \mid \Sigma \vdash e : \tau_1/\rho_1}{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau_2/\rho_2}\ [\textsc{Sub}]$$

By I.H, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau_1\rrbracket^{\mathbf{PI}}/\llbracket\rho_1\rrbracket^{\mathbf{PI}}$

By Lemma 39, we get $\Delta \mid \Sigma \vdash \llbracket\tau_1\rrbracket^{\mathbf{PI}} <: \llbracket\tau_2\rrbracket^{\mathbf{PI}}$ and $\Delta \mid \Sigma \vdash \llbracket\rho_1\rrbracket^{\mathbf{PI}} <: \llbracket\rho_2\rrbracket^{\mathbf{PI}}$.

Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau_2\rrbracket^{\mathbf{PI}}/\llbracket\rho_2\rrbracket^{\mathbf{PI}}$.

Case **Shift$_0$:**

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \ell :: \mathbf{L} \qquad \Delta, \Delta' \mid \Sigma \vdash \rho' <: \rho \qquad\qquad \Delta \mid \Sigma \vdash \tau' :: \mathbf{T} \\ \Delta, \Delta' \mid \Gamma, k : \tau' \rightarrow_{\rho} \tau \mid \Sigma \vdash e : \tau/\rho' \qquad \Delta \mid \Sigma \vdash (\exists_p\, \Delta'.\, \tau/\rho) \cdot \rho' :: \mathbf{R}\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle\ell\rangle\, k.\, e : \tau'/(\exists_\ell\, \Delta'.\, \tau/\rho) \cdot \rho'}\ [\textsc{Shift}_0]$$

By I.H, we get (1) $\Delta, \Delta' \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}}, k : \llbracket\tau' \to_\rho \tau\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau\rrbracket^{\mathbf{PI}} / \llbracket\rho'\rrbracket^{\mathbf{PI}}$.

By Lemma 39, we get (2) $\Delta, \Delta' \mid \Sigma \vdash \llbracket\rho'\rrbracket^{\mathbf{PI}} <: \llbracket\rho\rrbracket^{\mathbf{PI}}$.

By Lemma 40, we get (3) $\Delta \mid \Sigma \vdash \llbracket\tau'\rrbracket^{\mathbf{PI}} :: \mathbf{T}$ and (4) $\Delta \mid \Sigma \vdash \llbracket(\exists_p\, \Delta'.\, \tau/\rho) \cdot \rho'\rrbracket^{\mathbf{PI}} :: \mathbf{R}$.

By the definition of $\llbracket\cdot\rrbracket^{\mathbf{PI}}$, we get (5) $\llbracket(\exists_\ell\, \Delta'.\, \tau/\rho)\rrbracket^{\mathbf{PI}} = \exists_\ell\, \alpha :: \mathbf{T}.(\forall\Delta'.(\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \Rightarrow \alpha$ and (6) $\llbracket\mathbf{shift}_0\langle\ell\rangle\; k.\, e\rrbracket^{\mathbf{PI}} = \mathbf{do}\langle\ell\rangle\; (\lambda k.\llbracket e\rrbracket^{\mathbf{PI}})$.

By SUB, (1) and (2), we get (7) $\Delta, \Delta' \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}}, k : \llbracket\tau' \to_\rho \tau\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau\rrbracket^{\mathbf{PI}} / \llbracket\rho\rrbracket^{\mathbf{PI}}$.

By ABS and (7), we get (8) $\Delta, \Delta' \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \lambda k.\llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau' \to_\rho \tau\rrbracket^{\mathbf{PI}} \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}} / \iota$.

By GEN and (8), we get (9) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \lambda k.\llbracket e\rrbracket^{\mathbf{PI}} : \forall\Delta'.\llbracket\tau' \to_\rho \tau\rrbracket^{\mathbf{PI}} \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}} / \iota$.
Let us define $\delta = \{\llbracket\tau'\rrbracket^{\mathbf{PI}}/\alpha\}$.

By (3), we get (10) $\Delta \mid \Sigma \vdash \delta :: (\alpha :: \mathbf{T})$. By (5) and (10), we get (11) $\delta(\forall\Delta'.(\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) = \forall\Delta'.\llbracket\tau' \to_\rho \tau\rrbracket^{\mathbf{PI}} \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}$.

By (9), (11) and (10) and DO, we get (12) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\; \lambda k.\llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau'\rrbracket^{\mathbf{PI}} / \exists_\ell\, \alpha :: \mathbf{T}.(\forall\Delta'.(\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \Rightarrow \alpha$.

Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \mathbf{do}\langle\ell\rangle\; \lambda k.\llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau'\rrbracket^{\mathbf{PI}} / (\exists_\ell\, \alpha :: \mathbf{T}.(\forall\Delta'.(\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \Rightarrow \alpha) \cdot \llbracket\rho\rrbracket^{\mathbf{PI}}$ by SUB, (11) and (12).

Case DOLLAR:

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_l\, \Delta'.\, \tau/\rho) \cdot \delta(\rho) \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r\rangle_l : \delta(\tau)/\delta(\rho)}\;[\textsc{Dollar}]$$

By I.H, we get (1) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{PI}} : \llbracket\tau'\rrbracket^{\mathbf{PI}} / \llbracket(\exists_p\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)\rrbracket^{\mathbf{PI}}$ and (2) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}}, x : \llbracket\tau'\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \llbracket e_r\rrbracket^{\mathbf{PI}} : \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} / \llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}$.

By Lemma 41, we get (3) $\Delta \mid \Sigma \vdash \llbracket\delta\rrbracket^{\mathbf{PI}} :: \Delta'$.

By the definition of $\llbracket\cdot\rrbracket^{\mathbf{PI}}$, we get (4) $\llbracket(\exists_l\, \Delta'.\, \tau/\rho)\rrbracket^{\mathbf{PI}} = \exists_l\, \alpha :: \mathbf{T}.(\forall\Delta'.(\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \Rightarrow \alpha$.

By VAR and SUB, we get (5) $\Delta, \alpha :: \mathbf{T}, \Delta' \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}}, x : (\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}, r : \alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash x : (\alpha \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}}) \to_{\llbracket\rho\rrbracket^{\mathbf{PI}}} \llbracket\tau\rrbracket^{\mathbf{PI}} / \llbracket\rho\rrbracket^{\mathbf{PI}}$.

By Lemma 27, Lemma 42, (3) and (5), we get (6) $\Delta, \alpha :: \mathbf{T} \mid \llbracket\delta(\Gamma)\rrbracket^{\mathbf{PI}}, x : (\alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}) \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}, r : \alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash x : (\alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}) \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} / \llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}$.

Moreover, we get (7) $\Delta, \alpha :: \mathbf{T} \mid \llbracket\delta(\Gamma)\rrbracket^{\mathbf{PI}}, x : (\alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}) \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}, r : \alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash r : \alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} / \llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}$.

By APP, (6) and (7), we get (8) $\Delta, \alpha :: \mathbf{T} \mid \llbracket\delta(\Gamma)\rrbracket^{\mathbf{PI}}, x : (\alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}) \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}}, r : \alpha \to_{\llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}} \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash x\; r : \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} / \llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}$.

Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash \mathbf{handle}\langle l\rangle\; \llbracket e\rrbracket^{\mathbf{PI}}\; \mathbf{with}\; \{x, r.x\; r; x.\llbracket e_r\rrbracket^{\mathbf{PI}}\} : \llbracket\delta(\tau)\rrbracket^{\mathbf{PI}} / \llbracket\delta(\rho)\rrbracket^{\mathbf{PI}}$ by HANDLE, (1), (2) and (8).

□

Lemma 43 (Translated value is also a value)**.**
If v is a value then $\llbracket v\rrbracket^{\mathbf{PI}}$ is also a value.

Proof. Straightforward by the definition of $[\![\cdot]\!]^{\mathbf{PI}}$. □

Lemma 44 (Translation of expressions is commutative)**.**
$[\![e\{v/x\}]\!]^{\mathbf{PI}} = [\![e]\!]^{\mathbf{PI}}\{[\![v]\!]^{\mathbf{PI}}/x\}$, for any expression e.

Proof. By induction on structure of e.

Case $e = y$**:** We proceed by case analysis on y.

SubCase $x = y$**:** $[\![x]\!]^{\mathbf{PI}}\{[\![v]\!]^{\mathbf{PI}}/x\} = x\{[\![v]\!]^{\mathbf{PI}}/x\} = [\![v]\!]^{\mathbf{PI}} = [\![x\{v/x\}]\!]^{\mathbf{PI}}$.

SubCase $x \neq y$**:** $[\![y]\!]^{\mathbf{PI}}\{[\![v]\!]^{\mathbf{PI}}/x\} = y\{[\![v]\!]^{\mathbf{PI}}/x\} = y = [\![y\{v/x\}]\!]^{\mathbf{PI}}$.

Case $e = \lambda y.e_0$ **and** $e = e_1\ e_2$**:**
The result follows directly from I.H.

Case $e = \mathbf{shift}_0\langle\ell\rangle\ k.\ e_0$**:**
By the definition of $[\![\cdot]\!]^{\mathbf{PI}}$ and $\{v/x\}$,
we get $[\![(\mathbf{shift}_0\langle\ell\rangle\ k.\ e_0)\{v/x\}]\!]^{\mathbf{PI}} = [\![\mathbf{shift}_0\langle\ell\rangle\ k.\ e_0\{v/x\}]\!]^{\mathbf{PI}} = \mathbf{do}\langle p\rangle\ (\lambda k.[\![e_0\{v/x\}]\!]^{\mathbf{PI}}$.

By I.H, we get $\mathbf{do}\langle\ell\rangle\ (\lambda k.[\![e_0\{v/x\}]\!]^{\mathbf{PI}} = \mathbf{do}\langle\ell\rangle\ (\lambda k.[\![e_0]\!]^{\mathbf{PI}}[\![\{v/x\}]\!]^{\mathbf{PI}}$.

By the definition of $[\![\cdot]\!]^{\mathbf{PI}}$ and $\{v/x\}$,
we get $[\![(\mathbf{shift}_0\langle\ell\rangle\ k.\ e_0)]\!]^{\mathbf{PI}}[\![\{v/x\}]\!]^{\mathbf{PI}} = [\![\mathbf{shift}_0\langle\ell\rangle\ k.\ e_0]\!]^{\mathbf{PI}}[\![\{v/x\}]\!]^{\mathbf{PI}} = \mathbf{do}\langle\ell\rangle\ (\lambda k.[\![e_0]\!]^{\mathbf{PI}})[\![\{v/x\}]\!]^{\mathbf{PI}} = \mathbf{do}\langle\ell\rangle\ (\lambda k.[\![e_0]\!]^{\mathbf{PI}}[\![\{v/x\}]\!]^{\mathbf{PI}})$.

Thus, we get $[\![(\mathbf{shift}_0\langle\ell\rangle\ k.\ e_0)\{v/x\}]\!]^{\mathbf{PI}} = [\![\mathbf{shift}_0\langle\ell\rangle\ k.\ e_0]\!]^{\mathbf{PI}}[\![\{v/x\}]\!]^{\mathbf{PI}}$.

Case $e = \langle e_0 \mid x.\ e_h\rangle_l$**:**
Straightforward.

□

Lemma 45 (Translation of evaluation contexts is commutative)**.**
$[\![E[e]]\!]^{\mathbf{PI}} = [\![E]\!]^{\mathbf{PI}}[[\![e]\!]^{\mathbf{PI}}]$, for any evaluation context E.

Proof. By induction on structure of E.

Case $E = \square$**:**
Straightforward.

Case $E = E_0\ e$ **and** $E = v\ E_0$**:**
The result follow directly from I.H.

Case $E = \langle E_0 \mid x.\ e_r\rangle_l$**:**

$$\begin{array}{rcll}
[\![E[e]]\!]^{\mathbf{PI}} & = & [\![\langle E_0[e] \mid x.\ e_r\rangle_l]\!]^{\mathbf{PI}} & \\
& = & \mathbf{handle}\langle l\rangle\ [\![E_0[e]]\!]^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.e_r\} & (\text{the definition of } [\![\cdot]\!]^{\mathbf{PI}}) \\
& = & \mathbf{handle}\langle l\rangle\ [\![E_0]\!]^{\mathbf{PI}}[[\![e]\!]^{\mathbf{PI}}]\ \mathbf{with}\ \{x, r.x\ r; x.e_r\} & (\mathit{I.H.}) \\
& = & [\![\langle E_0 \mid x.\ e_r\rangle_l]\!]^{\mathbf{PI}} & (\text{the definition of } [\![\cdot]\!]^{\mathbf{PI}}) \\
& = & [\![E]\!]^{\mathbf{PI}}[[\![e]\!]^{\mathbf{PI}}] &
\end{array}$$

□

Lemma 46 (Translation preserves a prompt extractor)**.**
If $\ell \notin \lceil E \rceil$ then $\ell \notin \lceil [\![E]\!]^{\mathbf{PI}} \rceil$.

Proof. Straightforward. by the definition of $[\![\cdot]\!]^{\mathbf{PI}}$. □

Lemma 47 (Multi-step reductions)**.**
If $E[e]$ and $e \rightarrow^* e'$ then $E[e] \rightarrow^* E[e']$.

Proof. By indcution on a derivation of $e_0 \rightarrow^* e'_0$.

Case MREFL:
Straightforward.

Case MRED:

$$\frac{e \rightarrow e'}{e \rightarrow^* e'}\ [\text{MR}\textsc{ed}]$$

By the definition of $(\rightarrow)$, we get the following derivation:

$$\frac{e_0 \mapsto e'_0}{e = E_0[e_0] \rightarrow E_0[e'_0] = e'}\ [\textsc{Step}]$$

By STEP, we get $E[E_0[e_0]] \rightarrow E[E_0[e'_0]]$.

Thus, we get $E[E_0[e_0]] \rightarrow^* E[E_0[e'_0]]$ by MRED.

Case MTRANS:

$$\frac{e_1 \rightarrow^* e_2 \qquad e_3 \rightarrow^* e_3}{e_1 \rightarrow^* e_3}\ [\text{MT}\textsc{rans}]$$

By I.H, we get $E[e_1] \rightarrow^* E[e_2]$ and $E[e_2] \rightarrow^* E[e_3]$.

Thus, we get $E[e_1] \rightarrow^* E[e_3]$ by MTRABS.

□

Lemma 48 (Strict Multi-step reductions)**.**
If $E[e_1]$ and $e_1 \rightarrow^+ e_3$ then $E[e_1] \rightarrow^+ E[e_3]$.

Proof.

$$\frac{e_1 \rightarrow e_2 \qquad e_2 \rightarrow^* e_3}{e_1 \rightarrow^+ e_3}\ [\text{S-MSR}\textsc{ed}]$$

By the definition of $(e_1 \rightarrow e_2)$, we get

$$\frac{e'_1 \mapsto e'_2}{e_1 = E_0[e'_1] \rightarrow E_0[e'_2] = e_2}\ [\textsc{Step}]$$

By STEP, we get $E[E_0[e'_1]] \rightarrow E[E_0[e'_2]]$.
By Lemma 47, we get $E[e_2] \rightarrow^* E[e_3]$.
Thus, we get $E[e_1] \rightarrow^+ E[e_3]$ by S-MSRED. □

Lemma 49 (Translation preserves reductions)**.**
If $e \mapsto e'$ then $\llbracket e \rrbracket^{\mathbf{PI}} \rightarrow^+ \llbracket e' \rrbracket^{\mathbf{PI}}$.

Proof. **Case BETA:**

$$\frac{}{e = (\lambda x.e_0)\ v \mapsto e_0\{v/x\} = e'}\ [\textsc{Beta}]$$

$$\begin{aligned}
\llbracket(\lambda x.e_0)\ v\rrbracket^{\mathbf{PI}} &= (\lambda x.\llbracket e_0\rrbracket^{\mathbf{PI}})\ \llbracket v\rrbracket^{\mathbf{PI}} && (\text{definition of } \llbracket\cdot\rrbracket^{\mathbf{PI}})\\
&\to \llbracket e_0\rrbracket^{\mathbf{PI}}\{\llbracket v\rrbracket^{\mathbf{PI}}/x\} && (\textsc{Beta} \text{ and Lemma } 43)\\
&= \llbracket e_0\{v/x\}\rrbracket^{\mathbf{PI}} && (\text{Lemma } 44)
\end{aligned}$$

Thus, we get $\llbracket e\rrbracket^{\mathbf{PI}} \to^{+} \llbracket e'\rrbracket^{\mathbf{PI}}$.

Case EReturn:

$$\frac{}{\langle v \mid x.\, e_r\rangle_l \mapsto e_r\{v/x\}}\ [\textsc{EReturn}]$$

$$\begin{aligned}
\llbracket\langle v \mid x.\, e_r\rangle_l\rrbracket^{\mathbf{PI}} &= \mathbf{handle}\langle l\rangle\ \llbracket v\rrbracket^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r\rrbracket^{\mathbf{PI}}\} && (\text{definition of } \llbracket\cdot\rrbracket^{\mathbf{PI}})\\
&\to \llbracket e_r\rrbracket^{\mathbf{PI}}\{\llbracket v\rrbracket^{\mathbf{PI}}/x\} && (\textsc{EReturn} \text{ and Lemma } 43)\\
&= \llbracket e_r\{v/x\}\rrbracket^{\mathbf{PI}} && (\text{Lemma } 44)
\end{aligned}$$

Thus, we get $\llbracket e\rrbracket^{\mathbf{PI}} \to^{+} \llbracket e'\rrbracket^{\mathbf{PI}}$.

Case EDollar:

$$\frac{p \notin \lceil E\rceil \qquad v_c = \lambda z.\langle E[z] \mid x.\, e_r\rangle_l}{\langle E[\mathbf{shift}_0\langle l\rangle\ k.\, e] \mid x.\, e_r\rangle_l \mapsto e\{v_c/k\}}\ [\textsc{EDollar}]$$

By the definition of $\llbracket\cdot\rrbracket^{\mathbf{PI}}$, we get
(1) $\llbracket v_c\rrbracket^{\mathbf{PI}} = \llbracket\lambda z.\langle E[z] \mid x.\, e_r\rangle_l\rrbracket^{\mathbf{PI}} = \lambda z.\mathbf{handle}\langle l\rangle\ \llbracket E[z]\rrbracket^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r\rrbracket^{\mathbf{PI}}\}$

$$\begin{aligned}
&\llbracket\langle E[\mathbf{shift}_0\langle l\rangle\ k.\, e] \mid x.\, e_r\rangle_p\rrbracket^{\mathbf{PI}}\\
&= \mathbf{handle}\langle l\rangle\ \llbracket E[\mathbf{shift}_0\langle l\rangle\ k.\, e_0]\rrbracket^{\mathbf{PI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r\rrbracket^{\mathbf{PI}}\} && (\text{definition of } \llbracket\cdot\rrbracket^{\mathbf{PI}})\\
&= \mathbf{handle}\langle l\rangle\ \llbracket E\rrbracket^{\mathbf{PI}}[\llbracket\mathbf{shift}_0\langle l\rangle\ k.\, e_0\rrbracket^{\mathbf{PI}}]\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r\rrbracket^{\mathbf{PI}}\} && (\text{Lemma } 45)\\
&= \mathbf{handle}\langle l\rangle\ \llbracket E\rrbracket^{\mathbf{PI}}[\mathbf{do}\langle l\rangle\ (\lambda k.\llbracket e_0\rrbracket^{\mathbf{PI}})]\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r\rrbracket^{\mathbf{PI}}\} && (\text{definition of } \llbracket\cdot\rrbracket^{\mathbf{PI}})\\
&\to (x\ r)\{(\lambda k.\llbracket e_0\rrbracket^{\mathbf{PI}})/x\}\{\llbracket v_c\rrbracket^{\mathbf{PI}}/r\} && (\textsc{EHandle}, \text{ Lemma } 46 \text{ and } (1))\\
&\to ((\lambda k.\llbracket e_0\rrbracket^{\mathbf{PI}})\ r)\{\llbracket v_c\rrbracket^{\mathbf{PI}}/r\} && (\textsc{Beta})\\
&= ((\lambda k.\llbracket e_0\rrbracket^{\mathbf{PI}})\ \llbracket v_c\rrbracket^{\mathbf{PI}}) && (\text{substitution})\\
&\to \llbracket e_0\rrbracket^{\mathbf{PI}}\{\llbracket v_c\rrbracket^{\mathbf{PI}}/k\} && (\textsc{Beta})\\
&= \llbracket e_0\{v_c/k\}\rrbracket^{\mathbf{PI}} && (\text{Lemma } 44)
\end{aligned}$$

Thus, we get $\llbracket e\rrbracket^{\mathbf{PI}} \to^{+} \llbracket e'\rrbracket^{\mathbf{PI}}$.

□

Theorem 6 (Translation preserves meaning)**.**
If $e \to e'$ then $\llbracket e\rrbracket^{\mathbf{PI}} \to^{+} \llbracket e'\rrbracket^{\mathbf{PI}}$.

Proof.

$$\frac{e_0 \mapsto e_0'}{e = E[e_0] \to E[e_0'] = e'}\ [\textsc{Step}]$$

By Lemma 49, we get $\llbracket e_0\rrbracket^{\mathbf{PI}} \to^{+} \llbracket e_0'\rrbracket^{\mathbf{PI}}$.
Thus, we get $\llbracket e\rrbracket^{\mathbf{PI}} = \llbracket E[e_0]\rrbracket^{\mathbf{PI}} = \llbracket E\rrbracket^{\mathbf{PI}}[\llbracket e_0\rrbracket^{\mathbf{PI}}] \to^{+} \llbracket E\rrbracket^{\mathbf{PI}}[\llbracket e_0\rrbracket^{\mathbf{PI}}] = \llbracket E[\llbracket e_0'\rrbracket^{\mathbf{PI}}]\rrbracket^{\mathbf{PI}} = \llbracket e'\rrbracket^{\mathbf{PI}}$
by Lemma 45 and 48. □

D.3 Macro Translation from $\lambda^{l}_{\texttt{eff}}$ to $\lambda^{l}_{\texttt{del}}$

$$
\begin{array}{lcl}
\llbracket x \rrbracket^{\mathbf{IP}} & = & x \\
\llbracket \lambda x.e \rrbracket^{\mathbf{IP}} & = & \lambda x.\llbracket e \rrbracket^{\mathbf{IP}} \\
\llbracket e\ e \rrbracket^{\mathbf{IP}} & = & \llbracket e \rrbracket^{\mathbf{IP}}\ \llbracket e \rrbracket^{\mathbf{IP}} \\
\llbracket \mathbf{do}\langle \ell \rangle\ v \rrbracket^{\mathbf{IP}} & = & \mathbf{shift}_0\langle \ell \rangle\ k.\ \lambda h.h\ \llbracket v \rrbracket^{\mathbf{IP}}\ (\lambda x.k\ x\ h) \\
\llbracket \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{IP}} & = & \langle \llbracket e \rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}} \rangle_l\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}})
\end{array}
$$

FIGURE D.8: Macro translation of expressions

$$
\begin{array}{lcl}
\llbracket \alpha \rrbracket^{\mathbf{IP}} & = & \alpha \\
\llbracket \tau \to_{\rho} \tau \rrbracket^{\mathbf{IP}} & = & \llbracket \tau \rrbracket^{\mathbf{IP}} \to_{\llbracket \rho \rrbracket^{\mathbf{IP}}} \llbracket \tau \rrbracket^{\mathbf{IP}} \\
\llbracket \forall \alpha :: \kappa.\tau \rrbracket^{\mathbf{IP}} & = & \forall \alpha :: \kappa.\llbracket \tau \rrbracket^{\mathbf{IP}} \\
\llbracket \epsilon \cdot \rho \rrbracket^{\mathbf{IP}} & = & \llbracket \epsilon \rrbracket^{\mathbf{IP}} \cdot \llbracket \rho \rrbracket^{\mathbf{IP}} \\
\llbracket \exists_{\ell}\ \Delta'.\tau_1 \Rightarrow \tau_2 \rrbracket^{\mathbf{IP}} & = & \exists_{\ell}\ \alpha :: \mathbf{T}, \beta :: \mathbf{R}.\ ((\forall \Delta'.\llbracket \tau_1 \rrbracket^{\mathbf{IP}} \to_{\iota} (\llbracket \tau_2 \rrbracket^{\mathbf{IP}} \to_{\beta} \alpha) \to_{\beta} \alpha) \to_{\beta} \alpha)/\beta
\end{array}
$$

FIGURE D.9: Macro translation of types

$$
\begin{array}{lcl}
\llbracket \emptyset \rrbracket^{\mathbf{IP}} & = & \emptyset \\
\llbracket \Gamma, x : \tau \rrbracket^{\mathbf{IP}} & = & \llbracket \Gamma \rrbracket^{\mathbf{IP}}, x : \llbracket \tau \rrbracket^{\mathbf{IP}}
\end{array}
$$

FIGURE D.10: Macro translation of contexts

$$
\llbracket \{\overline{\tau/\alpha}\} \rrbracket^{\mathbf{IP}} \quad = \quad \{\overline{\llbracket \tau \rrbracket^{\mathbf{IP}}/\alpha}\}
$$

FIGURE D.11: Macro translation of substitutions

$$
\begin{array}{lcl}
\llbracket \square \rrbracket^{\mathbf{IP}} & = & \square \\
\llbracket E\ e \rrbracket^{\mathbf{IP}} & = & \llbracket E \rrbracket^{\mathbf{IP}}\ \llbracket e \rrbracket^{\mathbf{IP}} \\
\llbracket v\ E \rrbracket^{\mathbf{IP}} & = & \llbracket v \rrbracket^{\mathbf{IP}}\ \llbracket E \rrbracket^{\mathbf{IP}} \\
\llbracket \mathbf{handle}\langle l \rangle\ E\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{IP}} & = & \langle \llbracket E \rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}} \rangle_l\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}})
\end{array}
$$

FIGURE D.12: Macro translation of terms

$$\text{General context} \quad C \quad ::= \quad \square \mid C\ e \mid e\ C \mid \lambda x.C \mid \langle C \mid x.\ e_r \rangle_l \mid \langle e \mid x.\ C \rangle_l \mid \mathbf{shift}_0 \langle \ell \rangle\ k.\ C$$

FIGURE D.13: General contexts

$$\frac{e_1 \mapsto e_2}{C[e_1] \rightarrow_i C[e_2]}\ [\textsc{GStep}]$$

FIGURE D.14: General step

$$\frac{}{e \rightarrow_i^* e}\ [\textsc{MSGRefl}] \qquad \frac{e_1 \rightarrow_i e_2}{e_1 \rightarrow_i^* e_2}\ [\textsc{MSGRed}]$$

$$\frac{e_1 \rightarrow_i^* e_2 \qquad e_2 \rightarrow_i^* e_3}{e_1 \rightarrow_i^* e_3}\ [\textsc{MSGTrans}]$$

FIGURE D.15: Multi-step general reductions

$$\frac{e_1 \rightarrow_i e_2 \qquad e_2 \rightarrow_i^* e_3}{e_1 \rightarrow_i^+ e_3}\ [\textsc{S-MSGRed}]$$

FIGURE D.16: Strict multi-step general reductions

D.4 Proof of Meaning and Typability Preservation Properties

Lemma 50 (Translation preserves equivalence)**.**
If $\Delta \mid \Sigma \vdash \tau \equiv \tau'$ then $\Delta \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{IP}} \equiv [\![\tau']\!]^{\mathbf{IP}}$.

Proof. Proof of this lemma is the same as Lemma 38.

□

Lemma 51 (Translation preserves subtyping relations)**.**
If $\Delta \mid \Sigma \vdash \tau <: \tau'$ then $\Delta \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{IP}} <: [\![\tau']\!]^{\mathbf{IP}}$.

Proof. Proof of this lemma is the same as Lemma 39. □

Lemma 52 (Translation preserves kindings)**.**
If $\Delta \mid \Sigma \vdash \tau :: \kappa$ then $\Delta \mid \Sigma \vdash [\![\tau]\!]^{\mathbf{IP}} :: \kappa$.

Proof. We prove the difference cases only from Lemma 40. We remove the KMEFF case from Lemma 40 and prove the KIEFF case.

Case KIEFF:

$$\frac{\Delta, \Delta' \mid \Sigma \mid \tau_1 :: \mathbf{T} \qquad \Delta, \Delta' \mid \Sigma \mid \tau_2 :: \mathbf{T} \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}\ [\text{KIEFF}]$$

By I.H, we get (1) $\Delta, \Delta' \mid \Sigma \mid [\![\tau_1]\!]^{\mathbf{IP}} :: \mathbf{T}$ and (2) $\Delta, \Delta' \mid \Sigma \mid [\![\tau_2]\!]^{\mathbf{IP}} :: \mathbf{T}$.
Thus, we get $\Delta \mid \Sigma \vdash [\![\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2]\!]^{\mathbf{IP}} :: \mathbf{E}$ by KIEFF.

□

Lemma 53 (Translation preserves well-formedness of substitutions)**.**
If $\Delta \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta \mid \Sigma \vdash [\![\delta]\!]^{\mathbf{IP}} :: \Delta'$.

Proof. Proof of this lemma is the same as Lemma 41. □

Lemma 54 (Translation of types is commutative)**.**
$[\![\delta(\tau)]\!]^{\mathbf{IP}} = [\![\delta]\!]^{\mathbf{IP}}([\![\tau]\!]^{\mathbf{IP}})$, for any type τ.

Proof. Proof of this lemma is the same as Lemma 42. □

Theorem 7 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\Delta \mid [\![\Gamma]\!]^{\mathbf{IP}} \mid \Sigma \vdash [\![e]\!]^{\mathbf{IP}} : [\![\tau]\!]^{\mathbf{IP}} / [\![\rho]\!]^{\mathbf{IP}}$.

Proof. We prove the difference cases only from Theorem 5. We remove the SHIFT$_0$ and DOLLAR cases from Lemma 5 and prove the DO and HANDLE cases.

Case DO:

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1)/\iota \qquad \Delta \mid \Sigma \vdash \delta :: \Delta' \qquad \Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \delta(\tau_2) / (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota}\ [\text{DO}]$$

By I.H, we get (1) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket v\rrbracket^{\mathbf{IP}} : \llbracket\delta(\tau_1)\rrbracket^{\mathbf{IP}} / \iota$.

By Lemma 53, we get (2) $\Delta \mid \Sigma \vdash \llbracket\delta\rrbracket^{\mathbf{IP}} :: \Delta'$.

By Lemma 52, we get (3) $\Delta \mid \Sigma \vdash \llbracket\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2\rrbracket^{\mathbf{IP}} :: \mathbf{E}$.

By the deifnition of $\llbracket\cdot\rrbracket^{\mathbf{IP}}$, we get (4) $\llbracket\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2\rrbracket^{\mathbf{IP}} = \exists_\ell\, \alpha :: \mathbf{T}, \beta :: \mathbf{R}.\ ((\forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \rightarrow_\iota (\llbracket\tau_2\rrbracket^{\mathbf{IP}} \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha)/\beta$ and (5) $\llbracket\mathbf{do}\langle\ell\rangle\, v\rrbracket^{\mathbf{IP}} = \mathbf{shift}_0\langle\ell\rangle\, k.\, \lambda h.h\, \llbracket v\rrbracket^{\mathbf{IP}}\, (\lambda x.k\, x\, h)$.

By Var, we get (7) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}/\iota$.

By Inst and (2), we get (8) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash h : \llbracket\delta\rrbracket^{\mathbf{IP}}(\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}})/\iota$.

By Weakening and (1), we get (9) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket v\rrbracket^{\mathbf{IP}} : \llbracket\delta(\tau_1)\rrbracket^{\mathbf{IP}}/\iota$.

By Lemma 54, App, (8) and (9), we get (10) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash h\, \llbracket v\rrbracket^{\mathbf{IP}} : \llbracket\delta\rrbracket^{\mathbf{IP}}(\llbracket(\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}})/\iota$.

By Var, we get (11) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}, x : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta (\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha)\rrbracket^{\mathbf{PI}})/\beta$ and (12) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}, x : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash x : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}}/\beta$.

By App, (11) and (12), we get (13) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}, x : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash k\, x : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket(\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}/\beta$.

By Weakening, Sub and (7), we get (14) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}, x : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}/\beta$.

By App, (13) and (14), we get (15) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}, x : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}} \mid \Sigma \vdash k\, x\, h : \alpha/\beta$.

By Abs and (15), we get (16) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \lambda x.k\, x\, h : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{PI}} \rightarrow_\beta \alpha/\iota$.

By Weakening and (10), we get (17) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash h\, \llbracket v\rrbracket^{\mathbf{IP}} : \llbracket\delta\rrbracket^{\mathbf{IP}}(\llbracket(\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}})/\iota$.

By App, Sub, (16) and (17), we get (18) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}), h : \forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash h\, \llbracket v\rrbracket^{\mathbf{IP}}\, (\lambda x.k\, x\, h) : \alpha/\beta$.

By Abs and (18), we get (19) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, k : \llbracket\delta\rrbracket^{\mathbf{PI}}(\llbracket\tau_2 \rightarrow_\beta ((\forall\Delta'.\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{PI}}) \mid \Sigma \vdash \lambda h.h\, \llbracket v\rrbracket^{\mathbf{IP}}\, (\lambda x.k\, x\, h) : (\forall\Delta'.\llbracket\tau_1 \rightarrow_\iota (\tau_2 \rightarrow_\beta \alpha) \rightarrow_\beta \alpha\rrbracket^{\mathbf{IP}}) \rightarrow_\beta \alpha/\iota$.

By the premise, we get (20) $\Delta, \Delta' \mid \Sigma \vdash \tau_2 :: \mathbf{T}$.

By Lemma 54, Lemma 52, (2) and (20), we get (21) $\Delta \mid \Sigma \vdash \llbracket\delta(\tau_2)\rrbracket^{\mathbf{IP}} :: \mathbf{T}$.

By KRow and (3), we get (22) $\Delta \mid \Sigma \vdash \llbracket\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2\rrbracket^{\mathbf{IP}} \cdot \iota :: \mathbf{R}$.

By SEmpty, we get (23) $\Delta, \alpha :: \mathbf{T}, \beta :: \mathbf{R} \mid \Sigma \vdash \iota <: \beta$.

Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \mathbf{shift}_0\langle\ell\rangle\ k.\ \lambda h.h\ \llbracket v\rrbracket^{\mathbf{IP}}\ (\lambda x.k\ x\ h) : \llbracket\delta(\tau_2)\rrbracket^{\mathbf{IP}} / \llbracket\exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2\rrbracket^{\mathbf{IP}} \cdot \iota$ by SHIFT$_0$, (19), (21), (22), (23).

Case HANDLE:

$$\frac{\begin{array}{c}\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \qquad \Delta \mid \Sigma \vdash l :: \mathbf{L} \\ \Delta \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_p\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \qquad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{Handle}]$$

By I.H, we get (1) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{IP}} : \llbracket\tau\rrbracket^{\mathbf{IP}} / \llbracket(\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho\rrbracket^{\mathbf{IP}}$, (2) $\Delta, \Delta' \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, x : \llbracket\tau_1\rrbracket^{\mathbf{IP}}, r : \llbracket\tau_2 \to_\rho \tau_r\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket e_h\rrbracket^{\mathbf{IP}} : \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \llbracket\rho\rrbracket^{\mathbf{IP}}$ and (3) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, x : \llbracket\tau\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket e_r\rrbracket^{\mathbf{IP}} : \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \llbracket\rho\rrbracket^{\mathbf{IP}}$.

By the definition of $\llbracket\cdot\rrbracket^{\mathbf{IP}}$, we get (4) $\llbracket(\exists_l\ \Delta'.\tau_1 \Rightarrow \tau_2)\rrbracket^{\mathbf{IP}} = \exists_l\ \alpha :: \mathbf{T}, \beta :: \mathbf{R}.\ ((\forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota (\llbracket\tau_2\rrbracket^{\mathbf{IP}} \to_\beta \alpha) \to_\beta \alpha) \to_\beta \alpha) / \beta$ and (5) $\llbracket\mathbf{handle}\langle l\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}\rrbracket^{\mathbf{IP}} = \langle\llbracket e\rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r\rrbracket^{\mathbf{IP}}\rangle_l\ (\lambda x.\lambda r.\llbracket e_h\rrbracket^{\mathbf{IP}})$.

By ABS and (2), we get (6) $\Delta, \Delta' \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \lambda x.\lambda r.\llbracket e_h\rrbracket^{\mathbf{IP}} : \llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota (\llbracket\tau_2 \to_\rho \tau_r\rrbracket^{\mathbf{IP}}) \to_{\llbracket\rho\rrbracket^{\mathbf{IP}}} \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \iota$.

By GEN and (6), we get (7) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \lambda x.\lambda r.\llbracket e_h\rrbracket^{\mathbf{IP}} : \forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota (\llbracket\tau_2 \to_\rho \tau_r\rrbracket^{\mathbf{IP}}) \to_{\llbracket\rho\rrbracket^{\mathbf{IP}}} \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \iota$.

Let us define $\delta = \{\llbracket\tau_r\rrbracket^{\mathbf{IP}} / \alpha, \llbracket\rho\rrbracket^{\mathbf{IP}} / \beta\}$.

We get (8) $\Delta \mid \Sigma \vdash \delta :: (\alpha :: \mathbf{T}, \beta :: \mathbf{R})$.

By the definition of δ, (4) and (7), we get (9) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket e\rrbracket^{\mathbf{IP}} : \llbracket\tau\rrbracket^{\mathbf{IP}} / (\exists_p\ \alpha :: \mathbf{T}, \beta :: \mathbf{R}.\ ((\forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota (\llbracket\tau_2\rrbracket^{\mathbf{IP}} \to_\beta \alpha) \to_\beta \alpha) \to_\beta \alpha) / \beta) \cdot \delta(\beta)$.

By Weakening and (3), we get (10) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, x : \llbracket\tau\rrbracket^{\mathbf{IP}}, h : \forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota ((\llbracket\tau_2 \to_\rho \tau_r\rrbracket^{\mathbf{IP}}) \to_{\llbracket\rho\rrbracket^{\mathbf{IP}}} \llbracket\tau_r\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \llbracket e_r\rrbracket^{\mathbf{IP}} : \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \llbracket\rho\rrbracket^{\mathbf{IP}}$.

By ABS, SUB and (10), we get (11) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, x : \llbracket\tau\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \lambda h.\llbracket e_r\rrbracket^{\mathbf{IP}} : (\forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota ((\llbracket\tau_2 \to_\rho \tau_r\rrbracket^{\mathbf{IP}}) \to_{\llbracket\rho\rrbracket^{\mathbf{IP}}} \llbracket\tau_r\rrbracket^{\mathbf{IP}}) \to_{\llbracket\rho\rrbracket^{\mathbf{IP}}} \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \llbracket\rho\rrbracket^{\mathbf{IP}}$.

By the definition of δ and (11), we get (12) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}}, x : \llbracket\tau\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \lambda h.\llbracket e_r\rrbracket^{\mathbf{IP}} : \delta((\forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota ((\llbracket\tau_2\rrbracket^{\mathbf{IP}} \to_\beta \alpha) \to_\beta \alpha) \to_\beta \alpha) / \delta(\beta)$.

By DOLLAR, (8), (9) and (12), we get (13) $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \langle\llbracket e\rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r\rrbracket^{\mathbf{IP}}\rangle_l : \delta((\forall\Delta'.\llbracket\tau_1\rrbracket^{\mathbf{IP}} \to_\iota ((\llbracket\tau_2\rrbracket^{\mathbf{IP}} \to_\beta \alpha) \to_\beta \alpha) \to_\beta \alpha) / \delta(\beta)$.

Thus, we get $\Delta \mid \llbracket\Gamma\rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \langle\llbracket e\rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r\rrbracket^{\mathbf{IP}}\rangle_l\ (\lambda x.\lambda r.\llbracket e_h\rrbracket^{\mathbf{IP}}) : \llbracket\tau_r\rrbracket^{\mathbf{IP}} / \llbracket\rho\rrbracket^{\mathbf{IP}}$ by APP, SUB, (7) and (13).

□

Lemma 55 (Translated value is also a value)**.**
If v is a value then $\llbracket v\rrbracket^{\mathbf{IP}}$ is also a value.

Proof. Straightforward. by the definition of $\llbracket\cdot\rrbracket^{\mathbf{PI}}$. □

Lemma 56 (Translation of expressions is commutative)**.**
$\llbracket e\{v/x\}\rrbracket^{\mathbf{IP}} = \llbracket e\rrbracket^{\mathbf{IP}}\{\llbracket v\rrbracket^{\mathbf{IP}}/x\}$, for any expression e.

Proof. Proof of this lemma is the same as Lemma 44. □

Lemma 57 (Translation of evaluation contexts is commutative)**.**
$\llbracket E[e] \rrbracket^{\mathbf{IP}} = \llbracket E \rrbracket^{\mathbf{IP}}[\llbracket e \rrbracket^{\mathbf{IP}}]$, for any evaluation context E.

Proof. We prove the difference cases only from Lemma 45. We remove the $E = \langle E_0 \mid x.\, e_r \rangle_l$ case from Lemma 45 and prove the $E = \mathbf{handle}\langle l \rangle\ E_0\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$ case.

Case $E = \mathbf{handle}\langle l \rangle\ E_0\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$**:**

$$\begin{aligned}
&\llbracket \mathbf{handle}\langle l \rangle\ E_0[e]\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{IP}} \\
&= \langle \llbracket E_0[e] \rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}} \rangle_l\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}}) && (\text{definition of } \llbracket \cdot \rrbracket^{\mathbf{IP}}) \\
&= \llbracket E_0 \rrbracket^{\mathbf{IP}}[\llbracket e \rrbracket^{\mathbf{IP}}] \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}}\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}}) && (I.H) \\
&= \llbracket \mathbf{handle}\langle l \rangle\ E_0\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{IP}}[\llbracket e \rrbracket^{\mathbf{IP}}] && (\text{definition of } \llbracket \cdot \rrbracket^{\mathbf{IP}})
\end{aligned}$$

□

Lemma 58 (Translation preserves a prompt extractor)**.**
If $\ell \notin \lceil E \rceil$ then $\ell \notin \lceil \llbracket E \rrbracket^{\mathbf{IP}} \rceil$.

Proof. Straightforward. by the definition of $\llbracket \cdot \rrbracket^{\mathbf{PI}}$. □

Lemma 59 (Multi-step general reductions)**.**
If $e = E[e_0]$ and $e_0 \rightarrow_i^{*} e_0'$ then $E[e_0] \rightarrow_i^{*} E[e_0']$.

Proof. We can prove this lemma by the same way to Lemma 47 using GSTEP instead of STEP. □

Lemma 60 (Strict multi-step general reductions)**.**
If $e = E[e_0]$ and $e_0 \rightarrow_i^{+} e_0'$ then $E[e_0] \rightarrow_i^{+} E[e_0']$.

Proof. We can prove this lemma by the same way to Lemma 48 using Lemma 59 and GSTEP instead of Lemma 47 and STEP. □

Lemma 61 (Translation preserves reductions)**.**
If $e \mapsto e'$ then $\llbracket e \rrbracket^{\mathbf{PI}} \rightarrow_i^{+} \llbracket e' \rrbracket^{\mathbf{PI}}$.

Proof. We prove the difference cases only from Lemma 49. We remove the ERETURN and EDOLLAR cases from Lemma 49 and prove the ERETURN and EHANDLE cases.

Case ERETURN:

$$\frac{}{\mathbf{handle}\langle l \rangle\ v\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \mapsto e_r\{v/x\}}\ [\text{ERETURN}]$$

$$\begin{aligned}
&\llbracket \mathbf{handle}\langle l \rangle\ v\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{IP}} \\
&= \langle \llbracket v \rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}} \rangle_l\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}}) && (\text{definition of } \llbracket \cdot \rrbracket^{\mathbf{IP}}) \\
&\rightarrow (\lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}})\{\llbracket v \rrbracket^{\mathbf{IP}}/x\}\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}}) && (\text{ERETURN and Lemma } 55) \\
&\rightarrow \llbracket e_r \rrbracket^{\mathbf{IP}}\{\llbracket v \rrbracket^{\mathbf{IP}}/x\}\{(\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}})/h\} && (\text{BETA}) \\
&= \llbracket e_r \rrbracket^{\mathbf{IP}}\{\llbracket v \rrbracket^{\mathbf{IP}}/x\} && (h \text{ is not contained in } e_r) \\
&= \llbracket e_r\{v/x\} \rrbracket^{\mathbf{IP}} && (\text{definition of } \llbracket \cdot \rrbracket^{\mathbf{IP}})
\end{aligned}$$

Thus, we get $\llbracket e \rrbracket^{\mathbf{IP}} \rightarrow_i^{+} \llbracket e' \rrbracket^{\mathbf{IP}}$.

Case EHANDLE:

$$\frac{l \notin \lceil E \rceil \qquad v_c = \lambda z.\textbf{handle}\langle l \rangle\ E[z]\ \textbf{with}\ \{x, r.e_h; x.e_r\}}{\textbf{handle}\langle l \rangle\ E[\textbf{do}\langle p \rangle\ v]\ \textbf{with}\ \{x, r.e_h; x.e_r\} \mapsto e_h\{v/x\}\{v_c/r\}}\ [\text{EHANDLE}]$$

Let us define $v' = \lambda y.\langle [\![E]\!]^{\mathbf{IP}}[y] \mid x'.\ \lambda h.[\![e_r]\!]^{\mathbf{IP}} \rangle_l$.

Theorem 8 (Translation preserves meaning)**.**
If $e \longrightarrow e'$ then $[\![e]\!]^{\mathbf{PI}} \longrightarrow^{+} [\![e']\!]^{\mathbf{PI}}$.

Proof.
We can prove this lemma by the same way to Theorem 6 using Lemma 59, 57 and 60 instead of Lemma 49, 45 and 48. □

Appendix E

$\lambda^{l+\eta}_{\texttt{del}}$: Delimited Control Operators with Dynamically Generated Prompt Tags

E.1 Syntax and Semantics

Syntax of Terms:

$$
\begin{array}{llcll}
\text{Expression} & e & ::= & \ldots & \\
 & & \mid & e[\ell] & \text{(label application)} \\
 & & \mid & \langle e \mid x.\, e_r \rangle_\eta & \text{(labeled dollar)} \\
\text{Value} & v & ::= & \ldots & \\
 & & \mid & \Lambda\eta.e & \text{(label abstraction)}
\end{array}
$$

Syntax of Effects:

$$
\begin{array}{llll}
\text{Effects} & \epsilon & ::= & \exists_\ell\, \Delta'.\, \tau/\rho
\end{array}
$$

Syntax of Labels:

$$
\begin{array}{llll}
\text{Labels} & \ell & ::= & l \mid \eta
\end{array}
$$

Names:

$$\text{Labels} \ni \eta, \eta_1, \eta_2, \ldots$$

Evaluation Context

$$E \;::=\; \cdots \mid E[\ell]$$

Count Label

$$\begin{aligned}
\lceil \square \rceil &::= \emptyset \\
\lceil E\; e \rceil &::= \lceil E \rceil \\
\lceil v\; E \rceil &::= \lceil E \rceil \\
\lceil E[\ell] \rceil &::= \lceil E \rceil \\
\lceil \langle E \mid x.\, e_r \rangle_l \rceil &::= l, \lceil E \rceil
\end{aligned}$$

Reduction Rules

$\boxed{\Sigma \vdash e \mapsto e' \dashv \Sigma'}$

$$\frac{}{\Sigma \vdash (\Lambda\eta.e)[\ell] \mapsto e\{\ell/\eta\} \dashv \Sigma}\;[\text{EIA}\textsc{pp}]$$

$$\frac{p \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta = \{l/\eta\}}{\Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta \mapsto \langle \delta(e) \mid x.\, e_r \rangle_\eta \dashv \Sigma'}\;[\text{EG}\textsc{en}\text{L}\textsc{abel}]$$

FIGURE E.1: Delimited control operators with dynamically generated prompt tags

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho}$$

$$\frac{\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / \iota}{\Delta \mid \Gamma \mid \Sigma \vdash \Lambda \eta . e : \forall \eta :: \mathbf{L} . \tau / \iota} \text{[LABS]}$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall \eta :: \mathbf{L} . \tau / \iota \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Gamma \mid \Sigma \vdash e[\ell] : \tau\{\ell / \eta\} / \iota} \text{[LAPP]}$$

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_\eta \, \Delta' . \, \tau / \rho) \cdot \delta(\rho) \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau) / \delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x . \, e_r \rangle_\eta : \delta(\tau) / \delta(\rho)} \text{[LDOLLAR]}$$

FIGURE E.2: Typing rules

E.2 Proof of Soundness

Lemma 62 (Term substitution lemma)**.**
If $\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/\rho$ and $\Delta \mid \Gamma_1 \mid \Sigma \vdash e' : \sigma/\iota$ then $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau/\rho$

Proof. We prove the difference cases only from Lemma 8.

Case LAbs and LApp:
Straightforward.

Case LDollar:

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho) \\ \Delta, \eta :: \mathbf{L} \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau'/(\exists_\eta\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid y.\, e_r \rangle_\eta : \delta(\tau)/\delta(\rho)}\ [\text{LDollar}]$$

By I.H, we get (1) $\Delta, \eta :: \mathbf{L} \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau'/(\exists_\eta\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)$ and (2) $\Delta \mid \Gamma_1, \Gamma_2, x : \tau' \mid \Sigma \vdash e_r\{e'/x\} : \delta(\tau)/\delta(\rho)$.
Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash \langle e \mid y.\, e_r \rangle_\eta : \delta(\tau)/\delta(\rho)$ by LDollar, (1), (2), and the premise.

□

Lemma 63 (Type variable substitution lemma)**.**

1. If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau :: \kappa$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau) :: \kappa$
2. If $\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash e : \delta(\tau)/\delta(\rho)$

Proof.

1. The proof of this lemma is same as Lemma 9
2. We prove the difference cases only from Lemma 9.

   Case LAbs and LApp:
   Straightforward.

   Case LDollar:

$$\frac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \delta_0 :: \Delta' \\ \Delta_1, \Delta', \Delta_2, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau'/(\exists_\eta\, \Delta'.\, \tau/\rho) \cdot \delta_0(\rho) \\ \Delta_1, \Delta', \Delta_2 \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta_0(\tau)/\delta_0(\rho)\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta : \delta_0(\tau)/\delta_0(\rho)}\ [\text{LDollar}]$$

   By the same way of Dollar, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash \langle \delta(e) \mid x.\, \delta(e_r) \rangle_\eta : \delta(\delta_0(\tau))/\delta(\delta_0(\rho))$.

□

Lemma 64 (Value is pure)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau/\iota$

Proof. We prove the difference cases only from Lemma 10.

Case LAbs:

Straightforward.

Case LApp and LDollar:

These cases are impossible because the term of the conclusion is not a value.

□

Lemma 65 (Compose/Decompose an evaluation context).
If $\Delta \mid \Gamma \mid \Sigma \vdash E[e] : \tau / \rho$ then there exists σ, ρ' such that

- $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma / \rho' * \rho$
- $\lceil E \rceil = \lceil \rho' \rceil$
- If $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma / \rho' * \rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash E[e'] : \tau / \rho$

Proof. We prove the difference cases only from Lemma 10.

Case $E \neq []$:

SubCase LAbs:

This case is impossible because the form of the conclusion is $E = []$.

SubCase LApp:

The result follows directly from I.H.

□

Lemma 66 (Inversion lemma: lambda abstraction).
If $\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma / \rho$ then there exists τ_1, τ_2, ρ_1 and Δ' such that $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho_1$, $\Delta \mid \Sigma \vdash \forall \Delta'. \tau_1 \to_{\rho_1} \tau_2 \rightsquigarrow^* \tau'$ and $\Delta \mid \Sigma \vdash \tau' <: \sigma$.

Proof. We prove the difference cases only from Lemma 12.

Case LAbs, LApp, and LDollar:

These case cannot actually arise, because the forms of the conclusion aren't lambda abstraction.

□

Lemma 67 (Unhandled shift$_0$ operators).
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\lfloor e \rfloor \leqslant size(\rho)$.

Proof. We prove the difference cases only from Lemma 13.

Case LAbs and LApp:

The result follows directly from I.H.

Case LDollar:

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_\eta \, \Delta'. \, \tau / \rho) \cdot \delta(\rho) \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau) / \delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x. \, e_r \rangle_\eta : \delta(\tau) / \delta(\rho)} \text{[LDollar]}$$

By the definition of $\lfloor\cdot\rfloor$, we get $\lfloor\langle e \mid x.\, e_r\rangle_\eta\rfloor = 0$. By the definition of $size(\cdot)$, we get $size(\delta(\rho)) \geqslant 0$. Thus, we get $\lfloor\langle e \mid x.\, e_r\rangle_\eta\rfloor \leqslant size(\delta(\rho))$.

□

Lemma 68 (Shift$_0$ operator performs an effect)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle\ell\rangle\, k.\, e : \tau/\rho$ then $\rho = (\exists_\ell\, \Delta'.\, \tau'/\rho') \cdot \rho''$.

Proof. We prove the difference cases only from Lemma 14.

Case LABS, LAPP, and LDOLLAR:
These cases cannot actually arise, because the forms of the conclusion aren't a shift expression.

□

Lemma 69 (Inversion lemma: shift$_0$ operator)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle\ell\rangle\, k.\, e : \sigma/\epsilon \cdot \rho$ then

- $\Delta, \Delta' \mid \Gamma, k : \tau_1 \to_{\rho_1} \tau_2 \mid \Sigma \vdash e : \tau_2/\rho_2$
- $\Delta, \Delta' \mid \Sigma \vdash \rho_2 <: \rho_1$
- $\Delta \mid \Sigma \vdash \tau_1 :: \mathbf{T}$
- $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_2/\rho_1) \cdot \rho_2 :: \mathbf{R}$
- $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_2/\rho_1) \cdot \rho_2 <: \epsilon \cdot \rho$
- $\Delta \mid \Sigma \vdash \tau_2 \rightsquigarrow^* \tau_3$
- $\Delta \mid \Sigma \vdash \tau_3 <: \sigma$

Proof. We prove the difference cases only from Lemma 15.

Case LAPP, LABS and LDOLLAR:
These cases cannot actually arise, because the forms of the conclusion aren't a shift expression.

□

Lemma 70 (Inversion lemma: label abstraction)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \Lambda\eta.e : \sigma/\rho$ then there exists τ_1, ρ_1, and Δ' such that $\Delta, \Delta', \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau_1/\iota$, $\Delta \mid \Sigma \vdash \forall(\Delta', \eta :: \mathbf{L}).\tau_1 \rightsquigarrow \tau'$, and $\Delta \mid \Sigma \vdash \tau' <: \sigma$.

Proof. Straightforward by induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \Lambda\eta.e : \sigma/\rho$. □

Lemma 71 (Small step preservation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $\Sigma \vdash e \mapsto e' \dashv \Sigma'$ then $\Delta \mid \Gamma \mid \Sigma' \vdash e' : \tau/\rho$

Proof. We prove the difference cases only from Lemma 16.

Case LABS:
These case cannot actually arise, since we assumed $\Sigma \vdash e \mapsto e' \dashv \Sigma'$ and there are no reduction rules for label abstractions.

Case LAPP:
Straightforward.

Case LDOLLAR**:**

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_\eta\, \Delta'.\, \tau/\rho) \cdot \delta(\rho) \quad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta : \delta(\tau)/\delta(\rho)} \text{[LDOLLAR]}$$

$$\frac{p \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta' = \{l/\eta\}}{\Sigma \vdash\ \langle e \mid x.\, e_r \rangle_\eta \mapsto \langle \delta'(e) \mid x.\, e_r \rangle_l\ \dashv \Sigma'} \text{[EGENLABEL]}$$

By weakining, we get (1) $\Delta \mid \Sigma' \vdash \delta :: \Delta'$, (2) $\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma' \vdash e : \tau'/(\exists_\eta\, \Delta'.\, \tau/\rho) \cdot \delta(\rho)$, and (3) $\Delta \mid \Gamma, x : \tau' \mid \Sigma' \vdash e_r : \delta(\tau)/\delta(\rho)$.
By Lemma 63, we get (4) $\Delta \mid \Sigma' \vdash \delta' \circ \delta :: \Delta'$, (5) $\Delta \mid \Gamma \mid \Sigma' \vdash \delta'(e) : \tau'/(\exists_\eta\, \Delta'.\, \delta'(\tau)/\delta'(\rho)) \cdot \delta'(\delta(\rho))$, and (6) $\Delta \mid \Gamma, x : \delta'(\tau') \mid \Sigma' \vdash \delta'(e_r) : \delta'(\delta(\tau))/\delta'(\delta(\rho))$.
By $\eta \notin \mathbf{ftv}(e_r) \cup \mathbf{ftv}(\delta(\tau)) \cup \mathbf{ftv}(\delta(\rho))$, we get (6) $\delta'(e_r) = e_r$, (7) $\delta'(\delta(\tau)) = \delta(\tau)$, and (8) $\delta'(\delta(\rho)) = \delta(\rho)$.
Thus, we get $\Delta \mid \Gamma \mid \Sigma' \vdash \langle \delta'(e) \mid x.\, e_r \rangle_l : \delta(\tau)/\delta(\rho)$ by LDOLLAR.

□

Lemma 72 (Prompt tags are captured)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash E[\mathbf{shift}_0\langle \ell \rangle\, k.\, e] : \tau/\rho$ then $\ell \in \lceil E \rceil$ or $\ell \in \lceil \rho \rceil$

Proof. By Lemma 65, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{shift}_0\langle \ell \rangle\, k.\, e : \sigma/\rho' * \rho$, where $\lceil \rho' \rceil = \lceil E \rceil$.
By (1) and Lemma 69, we get (2) $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\, \tau_0/\rho_0) \cdot \rho_0' < \rho' * \rho$.
By (2), $\ell \in \lceil (\exists_\ell\, \Delta'.\, \tau_0/\rho_0) \cdot \rho_0' \rceil$ and Lemma 17, we get $\ell \in \lceil \rho' * \rho \rceil$.
Thus, we get $\ell \in \lceil \rho' \rceil = \lceil E \rceil$ or $\ell \in \lceil \rho \rceil$. □

Lemma 73 (Progress with effects)**.**
If $\Delta \mid \varnothing \mid \Sigma \vdash e : \tau/\rho$ then e is value, $\exists\, e'$ s.t. $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$, or $e = E[\mathbf{shift}_0\langle \ell \rangle\, k.\, e_0]$, where $\ell \notin \lceil E \rceil$

Proof. We prove the difference cases only from Lemma 19.

Case LABS**:**
Straightforward.

Case LAPP**:**
We can directly get the conclusion by I.H.

Case LDOLLAR**:**

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_\eta\, \Delta'.\, \tau/\rho) \cdot \delta(\rho) \quad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta : \delta(\tau)/\delta(\rho)} \text{[LDOLLAR]}$$

We get $\Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta \mapsto \langle \delta(e) \mid x.\, e_r \rangle_l \vdash \Sigma'$, where $\delta = \{l/\eta\}$ and $\Sigma' = \Sigma, l$ by EGENLABEL.

□

Theorem 9 (Preservation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $e \longrightarrow e'$ then $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau/\rho$

Proof.

$$\frac{e_0 \mapsto e'_0}{E_0[e_0] \longrightarrow E_0[e'_0]}\ [\textsc{Step}]$$

,where $e = E_0[e_0]$ and $e' = E_0[e'_0]$.
By Lemma 65, there exists τ_0, ρ_0 and Δ' such that $\Delta' \mid \emptyset \mid \Sigma \vdash e_0 : \tau_0/\rho_0 * \rho$, where $\rho_0 = \lceil E_0 \rceil$. By Lemma 71, we get $\Delta' \mid \emptyset \mid \Sigma \vdash e'_0 : \tau_0/\rho_0 * \rho$. Thus, we get $\emptyset \mid \emptyset \mid \Sigma \vdash e' : \tau/\rho$ by Lemma 65.

□

Theorem 10 (Progress)**.**
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau/\iota$ then e is value or $\exists\, e'$ s.t. $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$

Proof. We can prove this lemma by a similar way to Lemma 2 using Lemma 72 instead of Lemma 18. □

Appendix F

$\lambda^{l+\eta}_{\texttt{eff}}$: Algebraic Effect Handlers with Dynamically Generated Effect Instances

F.1 Syntax and Semantics

Syntax of Terms:

$$\begin{array}{llcll}
\text{Expression} & e & ::= & \ldots & \\
 & & \mid & e[\ell] & \text{(label application)} \\
 & & \mid & \textbf{handle}\langle\eta\rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} & \text{(labeled handler)} \\
\text{Value} & v & ::= & \ldots & \\
 & & \mid & \Lambda\eta.e & \text{(label abstraction)}
\end{array}$$

Syntax of Effects:

$$\begin{array}{llll}
\text{Effects} & \epsilon & ::= & \exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2
\end{array}$$

Syntax of Effect Instances:

$$\begin{array}{llll}
\text{Labels} & \ell & ::= & l \mid \eta
\end{array}$$

Names:

$$\text{Labels} \ni \eta, \eta_1, \eta_2, \ldots$$

Evaluation Context

$$E \quad ::= \quad \cdots \mid E[\ell]$$

Count Label

$$\begin{array}{lcl}
\lceil \Box \rceil & ::= & \emptyset \\
\lceil E\ e \rceil & ::= & \lceil E \rceil \\
\lceil v\ E \rceil & ::= & \lceil E \rceil \\
\lceil E[\ell] \rceil & ::= & \lceil E \rceil \\
\lceil \textbf{handle}\langle l \rangle\ E\ \textbf{with}\ \{x, r.e_h; x.e_r\} \rceil & ::= & l, \lceil E \rceil
\end{array}$$

Reduction Rules

$$\boxed{\Sigma \vdash\ e \mapsto e'\ \dashv \Sigma'}$$

$$\frac{}{\Sigma \vdash\ (\Lambda\eta.e)[\ell] \mapsto e\{\ell/\eta\}\ \dashv \Sigma}\ [\textsc{EIApp}]$$

$$\frac{l \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta = \{l/\eta\}}{\Sigma \vdash\ \textbf{handle}\langle \eta \rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} \mapsto \textbf{handle}\langle l \rangle\ \delta(e)\ \textbf{with}\ \{x, r.e_h; x.e_r\}\ \dashv \Sigma'}\ [\textsc{EGenLabel}]$$

FIGURE F.1: Algebraic effect handlers with effect instances

$$\boxed{\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho}$$

$$\frac{\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / \iota}{\Delta \mid \Gamma \mid \Sigma \vdash \Lambda \eta . e : \forall \eta :: \mathbf{L} . \tau / \iota} \text{ [LA}\textsc{bs}\text{]}$$

$$\frac{\Delta \mid \Gamma \mid \Sigma \vdash e : \forall \eta :: \mathbf{L} . \tau / \iota \qquad \Delta \mid \Sigma \vdash \ell :: \mathbf{L}}{\Delta \mid \Gamma \mid \Sigma \vdash e[\ell] : \tau\{\ell / \eta\} / \iota} \text{ [LA}\textsc{pp}\text{]}$$

$$\frac{\begin{array}{c}\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_\eta \, \Delta' . \tau_1 \Rightarrow \tau_2) \cdot \rho \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \quad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle \eta \rangle \; e \; \mathbf{with} \; \{x, r . e_h ; x . e_r\} : \tau_r / \rho} \text{ [LH}\textsc{andle}\text{]}$$

FIGURE F.2: Typing rules

F.2 Proof of Soundness

Lemma 74 (Term substitution lemma)**.**
If $\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/\rho$ and $\Delta \mid \Gamma_1 \mid \Sigma \vdash e' : \sigma/\iota$ then $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau/\rho$.

Proof. We prove the difference cases only from Lemma 26.

Case LAbs **and** LApp**:**
The result follows directly from I.H.

Case LHandle**:**

$$\dfrac{\begin{array}{c}\Delta \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau \mid \Sigma \vdash e_r : \tau_r/\rho \\ \Delta, \eta :: \mathbf{L} \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash e : \tau/(\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \\ \Delta, \Delta' \mid \Gamma_1, x : \sigma, \Gamma_2, y : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho\end{array}}{\Delta \mid \Gamma_1, x : \sigma, \Gamma_2 \mid \Sigma \vdash \mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{y, r.e_h; y.e_r\} : \tau_r/\rho}\ [\text{LHandle}]$$

By I.H, we get $\Delta, \eta :: \mathbf{L} \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash e\{e'/x\} : \tau/(\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$, $\Delta, \Delta' \mid \Gamma_1, \Gamma_2, y : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h\{e'/x\} : \tau_r/\rho$ and $\Delta \mid \Gamma_1, \Gamma_2, y : \tau \mid \Sigma \vdash e_r\{e'/x\} : \tau_r/\rho$.
Thus, we get $\Delta \mid \Gamma_1, \Gamma_2 \mid \Sigma \vdash \mathbf{handle}\langle\eta\rangle\ e\{e'/x\}\ \mathbf{with}\ \{y, r.e_h\{e'/x\}; r.e_r\{e'/x\}\} : \tau_r/\rho$ by LHandle.

□

Lemma 75 (Type variable substitution lemma)**.**

1. If $\Delta_1, \Delta', \Delta_2 \mid \Sigma \vdash \tau :: \kappa$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \Sigma \vdash \delta(\tau) :: \kappa$
2. If $\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash e : \tau/\rho$ and $\Delta_1 \mid \Sigma \vdash \delta :: \Delta'$ then $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash \delta(e) : \delta(\tau)/\delta(\rho)$

Proof. We prove the difference cases only from Lemma 27.

1. The proof of this case is same as Lemma 27.
2. **Case** LAbs **and** LApp**:**
   The result follows directly from I.H.
   Case LHandle**:**

$$\dfrac{\begin{array}{c}\Delta_1, \Delta', \Delta_2 \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r/\rho \\ \Delta_1, \Delta', \Delta_2, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau/(\exists_\eta\ \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \rho \\ \Delta_1, \Delta', \Delta_2, \Delta'' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r/\rho\end{array}}{\Delta_1, \Delta', \Delta_2 \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r/\rho}\ [\text{LHandle}]$$

   By I.H, we get (1) $\Delta_1, \Delta_2, \eta :: \mathbf{L} \mid \delta(\Gamma) \mid \Sigma \vdash \delta(e) : \delta(\tau)/\delta((\exists_\eta\ \Delta''.\tau_1 \Rightarrow \tau_2) \cdot \rho)$, (2) $\Delta_1, \Delta_2, \Delta'' \mid \delta(\Gamma), x : \delta(\tau_1), r : \delta(\tau_2) \rightarrow_{\delta(\rho)} \delta(\tau_r) \mid \Sigma \vdash \delta(e_h) : \delta(\tau_r)/\delta(\rho)$ and (3) $\Delta_1, \Delta_2 \mid \delta(\Gamma), x : \delta(\tau) \mid \Sigma \vdash \delta(e_r) : \delta(\tau_r)/\delta(\rho)$.
   Thus, we get $\Delta_1, \Delta_2 \mid \delta(\Gamma) \mid \Sigma \vdash \mathbf{handle}\langle\eta\rangle\ \delta(e)\ \mathbf{with}\ \{x, r.\delta(e_h); x.\delta(e_r)\} : \delta(\tau_r)/\delta(\rho)$ by (1), (2), (3) and LHandle.

□

Lemma 76 (Value is pure)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau / \rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash v : \tau / \iota$.

Proof. We prove the difference cases only from Lemma 28.

Case LAbs:
Straightforward.

Case LApp and LHandle:
These cases are impossible because the form of the conclusion is not a value.

□

Lemma 77 (Compose/Decompose an evaluation context)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash E[e] : \tau / \rho$ then there exists σ, ρ' such that

- $\Delta \mid \Gamma \mid \Sigma \vdash e : \sigma / \rho' * \rho$
- $\lceil E \rceil = \lceil \rho' \rceil$
- If $\Delta \mid \Gamma \mid \Sigma \vdash e' : \sigma / \rho' * \rho$ then $\Delta \mid \Gamma \mid \Sigma \vdash E[e'] : \tau / \rho$

Proof. We prove the difference cases only from Lemma 29.

Case $E \neq []$**:**

SubCase LAbs:
This case is impossible because the form of the conclusion is $E = []$.

SubCase LApp:
The result follows directly from I.H.

□

Lemma 78 (Inversion lemma: lambda abstraction)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \lambda x.e : \sigma / \rho$ then there exists τ_1, τ_2, ρ_1 and Δ' such that $\Delta, \Delta' \mid \Gamma, x : \tau_1 \mid \Sigma \vdash e : \tau_2 / \rho_1$, $\Delta \mid \Sigma \vdash \forall \Delta'.\tau_1 \rightarrow_{\rho_1} \tau_2 \rightsquigarrow^* \tau'$ and $\Delta \mid \Sigma \vdash \tau' <: \sigma$.

Proof. We prove the difference cases only from Lemma 30.

Case LAbs, LApp, and LHandle:
These cases cannot actually arise, because the forms of the conclusion aren't lambda abstraction.

□

Lemma 79 (Unhandled shift$_0$ operators)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\lfloor e \rfloor \leqslant size(\rho)$.

Proof. We prove the difference cases only from Lemma 13.

Case LAbs and LApp:
The result follows directly from I.H.

Case LHandle:

$$\frac{\begin{array}{l}\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \\ \quad \Delta \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_\eta\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \qquad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \textbf{handle}\langle \eta \rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\text{LHANDLE}]$$

By the definition of $\lfloor \cdot \rfloor$, we get $\lfloor \textbf{handle}\langle \eta \rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} \rfloor = 0$. By the definition of $size(\cdot)$, we get $size(\rho) \geqslant 0$. Thus, we get $\lfloor \textbf{handle}\langle \eta \rangle\ e\ \textbf{with}\ \{x, r.e_h; x.e_r\} \rfloor \leqslant size(\rho)$.

□

Lemma 80 (Operation performs an effect)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \textbf{do}\langle \ell \rangle\ v : \tau / \rho$ then $\rho = (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho'$.

Proof. We prove the difference cases only from Lemma 32.

Case LABS, LAPP and LHANDLE:
These cases cannot actually arise, because the forms of the conclusion aren't a do-operation.

□

Lemma 81 (Inversion lemma: do operation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \textbf{do}\langle \ell \rangle\ v : \tau / \epsilon \cdot \rho$ then there exist $\tau_1, \tau_2, \tau'', \Delta'$ such that

- $\Delta \mid \Gamma \mid \Sigma \vdash v : \delta(\tau_1) / \iota$
- $\Delta \mid \Sigma \vdash \delta :: \Delta'$
- $\Delta \mid \Sigma \vdash \exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2 :: \mathbf{E}$
- $\Delta \mid \Sigma \vdash \delta(\tau_2) \rightsquigarrow^* \tau''$
- $\Delta \mid \Sigma \vdash \tau'' <: \tau$
- $\Delta \mid \Sigma \vdash (\exists_\ell\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota <: \epsilon \cdot \rho$.

Proof. We prove the difference cases only from Lemma 33.

Case LABS, LAPP, and LHANDLE:
These cases cannot actually arise, because the form of a conclusion isn't a do-operation.

□

Lemma 82 (Inversion lemma: label abstraction)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash \Lambda\eta.e : \sigma / \rho$ then there exists τ_1, ρ_1, and Δ' such that $\Delta, \Delta', \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau_1 / \iota$, $\Delta \mid \Sigma \vdash \forall(\Delta', \eta :: \mathbf{L}).\tau_1 \rightsquigarrow \tau'$, and $\Delta \mid \Sigma \vdash \tau' <: \sigma$.

Proof. Straightforward by induction on a derivation of $\Delta \mid \Gamma \mid \Sigma \vdash \Lambda\eta.e : \sigma / \rho$. □

Lemma 83 (Small step preservation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ and $\Sigma \vdash e \mapsto e' \dashv \Sigma'$ then $\Delta \mid \Gamma \mid \Sigma' \vdash e' : \tau / \rho$

Proof. We prove the difference cases only from Lemma 34.

Case LAbs:
These case cannot actually arise, since we assumed $\Sigma \vdash e \mapsto e' \dashv \Sigma'$ and there are no reduction rules for variables and shifts

Case LApp:
Straightforward.

Case Do:
This case cannot actually arise, since there are no reduction rules for an operation.

Case LHandle:

$$\frac{\begin{array}{c}\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \quad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\text{LHandle}]$$

$$\frac{l \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta' = \{l/\eta\}}{\Sigma \vdash\ \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \mapsto \mathbf{handle}\langle l \rangle\ \delta'(e)\ \mathbf{with}\ \{x, r.e_h; x.e_r\}\ \dashv \Sigma'}\ [\text{EGenLabel}$$

By weakning, we get (1) $\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma' \vdash e : \tau / (\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho$, (2) $\Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \rightarrow_\rho \tau_r \mid \Sigma' \vdash e_h : \tau_r / \rho$, and (3) $\Delta \mid \Gamma, x : \tau \mid \Sigma' \vdash e_r : \tau_r / \rho$.
By Lemma 75, we get (4) $\Delta \mid \Gamma \mid \Sigma' \vdash \delta'(e) : \delta'(\tau) / \delta'((\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho)$, (5) $\Delta, \Delta' \mid \Gamma, x : \delta'(\tau_1), \delta'(r : \tau_2 \rightarrow_\rho \tau_r) \mid \Sigma' \vdash \delta'(e_h) : \delta'(\tau_r) / \delta'(\rho)$, and (6) $\Delta \mid \Gamma, x : \tau \mid \Sigma' \vdash \delta'(e_r) : \delta'(\tau_r) / \delta'(\rho)$.
By $\eta \notin \mathbf{ftv}(e_r) \cup \mathbf{ftv}(e_h) \cup \mathbf{ftv}(\tau_r) \cup \mathbf{ftv}(\rho)$, we get (7) $\delta'(e_r) = e_r$, (8) $\delta'(e_h) = e_h$, (9) $\delta'(\tau_r) = \tau_r$, and (10) $\delta'(\rho) = \rho$.
Thus, we get $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle l \rangle\ \delta(e)\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho$ by LHandle.

□

Lemma 84 (Effect instances are captured).
If $\Delta \mid \Gamma \mid \Sigma \vdash E[\mathbf{do}\langle \ell \rangle\ v] : \tau / \rho$ then $\ell \in \lceil E \rceil$ or $\ell \in \lceil \rho \rceil$.

Proof. By Lemma 77, we get (1) $\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{do}\langle \ell \rangle\ v : \sigma / \rho' * \rho$, where $\lceil \rho' \rceil = \lceil E \rceil$. By (1) and Lemma 81, we get (2) $\Delta \mid \Sigma \vdash (\exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota <: \rho' \cdot \rho$. By (2), $\ell \in \lceil (\exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \iota \rceil$ and Lemma 35, we get (3) $\ell \in \lceil \rho' \cdot \rho \rceil$. Thus, we get $\ell \in \lceil \rho' \rceil$ or $\ell \in \lceil \rho \rceil$. □

Lemma 85 (Progress with effects).
If $\Delta \mid \varnothing \mid \Sigma \vdash e : \tau / \rho$ then e is value, $\exists\, e'$ s.t. $\Sigma \vdash e \rightarrow e' \dashv \Sigma'$, or $e = E[\mathbf{do}\langle \ell \rangle\ v]$, where $\ell \notin \lceil E \rceil$

Proof. We prove the difference cases only from Lemma 19.

Case LAbs:
Straightforward.

Case LApp:
We can directly get the conclusion by I.H.

Case LHandle**:**

$$\frac{\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_{\eta}\, \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \qquad \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_{\rho} \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \quad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{LHandle}]$$

By EGenLabel, we get $\Sigma \vdash \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \mapsto \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$ Σ', where $\delta = \{l/\eta\}$ and $\Sigma' = \Sigma$.

□

Theorem 11 (Preservation)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ and $e \to e'$ then $\Delta \mid \Gamma \mid \Sigma \vdash e' : \tau / \rho$

Proof. We can prove this lemma by a similar way to Lemma 3 using Lemma 77 and Lemma 83 instead of Lemma 29 and Lemma 34. □

Theorem 12 (Progress)**.**
If $\emptyset \mid \emptyset \mid \Sigma \vdash e : \tau / \rho$ then e is value or $\exists\, e'$ s.t. $e \to e'$

Proof. We can prove this lemma by a similar way to Lemma 4 using Lemma 84 instead of Lemma 36. □

Appendix G

Macro Translations Between $\lambda^{l+\eta}_{\texttt{del}}$ and $\lambda^{l+\eta}_{\texttt{eff}}$

G.1 Macro Translation from $\lambda^{l+\eta}_{\texttt{del}}$ to $\lambda^{l+\eta}_{\texttt{eff}}$

$$
\begin{array}{lcl}
\llbracket x \rrbracket^{\mathbf{SPI}} & = & x \\
\llbracket \lambda x.e \rrbracket^{\mathbf{SPI}} & = & \lambda x.\llbracket e \rrbracket^{\mathbf{SPI}} \\
\llbracket e\ e \rrbracket^{\mathbf{SPI}} & = & \llbracket e \rrbracket^{\mathbf{SPI}}\ \llbracket e \rrbracket^{\mathbf{SPI}} \\
\llbracket \mathbf{shift}_0 \langle \ell \rangle\ k.\ e \rrbracket^{\mathbf{SPI}} & = & \mathbf{do}\langle \ell \rangle\ (\lambda k.\llbracket e \rrbracket^{\mathbf{SPI}}) \\
\llbracket \langle e \mid x.\ e_r \rangle_l \rrbracket^{\mathbf{SPI}} & = & \mathbf{handle}\langle l \rangle\ \llbracket e \rrbracket^{\mathbf{SPI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r \rrbracket^{\mathbf{SPI}}\} \\
\llbracket \Lambda \eta.e \rrbracket^{\mathbf{SPI}} & = & \Lambda \eta.\llbracket e \rrbracket^{\mathbf{SPI}} \\
\llbracket e\ [\ell] \rrbracket^{\mathbf{SPI}} & = & \llbracket e \rrbracket^{\mathbf{SPI}}\ [\ell] \\
\llbracket \langle e \mid x.\ e_r \rangle_\eta \rrbracket^{\mathbf{SPI}} & = & \mathbf{handle}\langle \eta \rangle\ \llbracket e \rrbracket^{\mathbf{SPI}}\ \mathbf{with}\ \{x, r.x\ r; x.\llbracket e_r \rrbracket^{\mathbf{SPI}}\}
\end{array}
$$

FIGURE G.1: Macro translation of expressions

$$
\begin{array}{lcl}
\llbracket \alpha \rrbracket^{\mathbf{SPI}} & = & \alpha \\
\llbracket \tau \to_\rho \tau \rrbracket^{\mathbf{SPI}} & = & \llbracket \tau \rrbracket^{\mathbf{SPI}} \to_{\llbracket \rho \rrbracket^{\mathbf{SPI}}} \llbracket \tau \rrbracket^{\mathbf{SPI}} \\
\llbracket \forall \alpha :: \kappa.\tau \rrbracket^{\mathbf{SPI}} & = & \forall \alpha :: \kappa.\llbracket \tau \rrbracket^{\mathbf{SPI}} \\
\llbracket \epsilon \cdot \rho \rrbracket^{\mathbf{SPI}} & = & \llbracket \epsilon \rrbracket^{\mathbf{SPI}} \cdot \llbracket \rho \rrbracket^{\mathbf{SPI}} \\
\llbracket \exists_\ell\ \Delta'.\ \tau/\rho \rrbracket^{\mathbf{SPI}} & = & \exists_\ell\ \alpha :: \mathbf{T}.(\forall \Delta'.(\alpha \to_{\llbracket \rho \rrbracket^{\mathbf{SPI}}} \llbracket \tau \rrbracket^{\mathbf{SPI}}) \to_{\llbracket \rho \rrbracket^{\mathbf{SPI}}} \llbracket \tau \rrbracket^{\mathbf{SPI}}) \Rightarrow \alpha
\end{array}
$$

FIGURE G.2: Macro translation of types

$$
\begin{array}{lcl}
[\![\emptyset]\!]^{\mathbf{SPI}} & = & \emptyset \\
[\![\Gamma, x : \tau]\!]^{\mathbf{SPI}} & = & [\![\Gamma]\!]^{\mathbf{SPI}}, x : [\![\tau]\!]^{\mathbf{SPI}}
\end{array}
$$

FIGURE G.3: Macro Translation of contexts

$$
[\![\{\overline{\tau/\alpha}\}]\!]^{\mathbf{SPI}} \quad = \quad \{\overline{[\![\tau]\!]^{\mathbf{SPI}}/\alpha}\}
$$

FIGURE G.4: Macro translation of substitutions

$$
\begin{array}{lcl}
[\![\square]\!]^{\mathbf{SPI}} & = & \square \\
[\![E\ e]\!]^{\mathbf{SPI}} & = & [\![E]\!]^{\mathbf{SPI}}\ [\![e]\!]^{\mathbf{SPI}} \\
[\![v\ E]\!]^{\mathbf{SPI}} & = & [\![v]\!]^{\mathbf{SPI}}\ [\![E]\!]^{\mathbf{SPI}} \\
[\![\langle E \mid x.\ e_r\rangle_l]\!]^{\mathbf{SPI}} & = & \mathbf{handle}\langle l\rangle\ [\![E]\!]^{\mathbf{SPI}}\ \mathbf{with}\ \{x, r.x\ r; x.[\![e_r]\!]^{\mathbf{SPI}}\}
\end{array}
$$

FIGURE G.5: Macro translation of evaluation contexts

$$
\frac{}{e \longrightarrow^* e}\ [\text{MSR}\textsc{efl}] \qquad \frac{e_1 \longrightarrow e_2}{e_1 \longrightarrow^* e_2}\ [\text{MSR}\textsc{ed}] \qquad \frac{e_1 \longrightarrow^* e_2 \qquad e_2 \longrightarrow^* e_3}{e_1 \longrightarrow^* e_3}\ [\text{MST}\textsc{rans}]
$$

FIGURE G.6: Multi-step reductions

$$
\frac{e_1 \longrightarrow e_2 \qquad e_2 \longrightarrow^* e_3}{e_1 \longrightarrow^+ e_3}\ [\text{S-MSR}\textsc{ed}]
$$

FIGURE G.7: Strict multi-step reductions

G.2 Proof of Soundness

Theorem 13 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\Delta \mid [\![\Gamma]\!]^{\mathbf{SPI}} \mid \Sigma \vdash [\![e]\!]^{\mathbf{SPI}} : [\![\tau]\!]^{\mathbf{SPI}} / [\![\rho]\!]^{\mathbf{SPI}}$.

Proof. We only prove the difference cases from Theorem 5.

Case L**A**BS **and** L**A**PP**:**
The result follws from I.H directly.

Case LD**OLLAR:**

$$\frac{\begin{array}{c}\Delta \mid \Sigma \vdash \delta :: \Delta' \\ \Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau' / (\exists_\eta \, \Delta'.\, \tau/\rho) \cdot \delta(\rho) \qquad \Delta \mid \Gamma, x : \tau' \mid \Sigma \vdash e_r : \delta(\tau)/\delta(\rho)\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \langle e \mid x.\, e_r \rangle_\eta : \delta(\tau)/\delta(\rho)} \text{ [LDOLLAR]}$$

By I.H, we get (1) $\Delta, \eta :: \mathbf{L} \mid [\![\Gamma]\!]^{\mathbf{SPI}} \mid \Sigma \vdash [\![e]\!]^{\mathbf{SPI}} : [\![\tau']\!]^{\mathbf{SPI}} / [\![(\exists_\eta \, \Delta'.\, \tau/\rho) \cdot \delta(\rho)]\!]^{\mathbf{SPI}}$ and (2) $\Delta \mid [\![\Gamma]\!]^{\mathbf{SPI}}, x : [\![\tau']\!]^{\mathbf{SPI}} \mid \Sigma \vdash [\![e_r]\!]^{\mathbf{SPI}} : [\![\delta(\tau)]\!]^{\mathbf{SPI}} / [\![\delta(\rho)]\!]^{\mathbf{SPI}}$.
By the same way of DOLLAR, we get (3) $\Delta, \alpha :: \mathbf{T} \mid [\![\delta(\Gamma)]\!]^{\mathbf{SPI}}, x : (\alpha \rightarrow_{[\![\delta(\rho)]\!]^{\mathbf{SPI}}} [\![\delta(\tau)]\!]^{\mathbf{SPI}}) \rightarrow_{[\![\delta(\rho)]\!]^{\mathbf{SPI}}} [\![\delta(\tau)]\!]^{\mathbf{SPI}}), r : \alpha \rightarrow_{[\![\delta(\rho)]\!]^{\mathbf{SPI}}} [\![\delta(\tau)]\!]^{\mathbf{SPI}} \mid \Sigma \vdash x\, r : [\![\delta(\tau)]\!]^{\mathbf{SPI}} / [\![\delta(\rho)]\!]^{\mathbf{SPI}}$.
Thus, we get $\Delta \mid [\![\Gamma]\!]^{\mathbf{SPI}} \mid \Sigma \vdash \mathbf{handle}\langle \ell \rangle \; [\![e]\!]^{\mathbf{SPI}} \; \mathbf{with} \; \{x, r. x\; r; x. [\![e_r]\!]^{\mathbf{SPI}}\} : [\![\delta(\tau)]\!]^{\mathbf{SPI}} / [\![\delta(\rho)]\!]^{\mathbf{SPI}}$ by LHANDLE, (1), (2) and (3).

□

Lemma 86 (Translation of expressions is commutative)**.**
$[\![e\{v/x\}]\!]^{\mathbf{SPI}} = [\![e]\!]^{\mathbf{SPI}}\{[\![v]\!]^{\mathbf{SPI}}/x\}$, for any expression e.

Proof. We only prove the difference cases from Lemma 44.

Case $e = \Lambda\eta.e_0$ **and** $e = e_0[\ell]$**:**
The result follows directly from I.H.

Case $e = \langle e \mid x.\, e_r \rangle_\eta$**:**
Straightforward.

□

Lemma 87 (Translation of evaluation contexts is commutative)**.**
$[\![E[e]]\!]^{\mathbf{SPI}} = [\![E]\!]^{\mathbf{SPI}}[[\![e]\!]^{\mathbf{SPI}}]$, for any evaluation context E.

Proof. We only prove the difference cases from Lemma 45.

Case $E = E_0 \; [\ell]$**:**
The result follows directly from I.H.

□

Lemma 88 (Translation preserves reductions)**.**
If $e \longmapsto e'$ then $\Sigma \vdash [\![e]\!]^{\mathbf{SPI}} \rightarrow^{+} [\![e']\!]^{\mathbf{SPI}} \dashv \Sigma'$.

Proof. We only prove the difference cases from Lemma 49.

Case ELApp:

Straightforward.

Case EGenLabel:

$$\frac{l \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta = \{l/\eta\}}{\Sigma \vdash \ \langle e \mid x.\, e_r \rangle_\eta \mapsto \langle \delta(e) \mid x.\, e_r \rangle_\eta \ \dashv \Sigma'} \text{[EGenLabel]}$$

$$\begin{aligned}
&\Sigma \vdash [\![\langle e \mid x.\, e_r \rangle_\eta]\!]^{\mathbf{SPI}} \\
&= \mathbf{handle}\langle \eta \rangle\ [\![e]\!]^{\mathbf{SPI}}\ \mathbf{with}\ \{x, r.x\ r; x.[\![e_r]\!]^{\mathbf{SPI}}\} && (\text{definition of } [\![\cdot]\!]^{\mathbf{SPI}}) \\
&\to \mathbf{handle}\langle l \rangle\ \delta([\![e]\!]^{\mathbf{SPI}})\ \mathbf{with}\ \{x, r.x\ r; x.[\![e_r]\!]^{\mathbf{SPI}}\} && (\text{EGebLabel}) \\
&= \mathbf{handle}\langle l \rangle\ [\![\delta(e)]\!]^{\mathbf{SPI}}\ \mathbf{with}\ \{x, r.x\ r; x.[\![e_r]\!]^{\mathbf{SPI}}\} && (\text{definition of } [\![\cdot]\!]^{\mathbf{SPI}} \text{ and } \delta) \dashv \Sigma' \\
&= [\![\langle \delta(e) \mid x.\, e_r \rangle_l]\!]^{\mathbf{SPI}} && (\text{definition of } [\![\cdot]\!]^{\mathbf{SPI}})
\end{aligned}$$

Thus, we get $\Sigma \vdash [\![e]\!]^{\mathbf{SPI}} \to^+ [\![e']\!]^{\mathbf{SPI}} \dashv \Sigma'$.

□

Theorem 14 (Translation preserves meaning)**.**
If $e \to e'$ then $\Sigma \vdash [\![e]\!]^{\mathbf{SPI}} \to^+ [\![e']\!]^{\mathbf{SPI}} \dashv \Sigma'$.

Proof.

$$\frac{e_0 \mapsto e'_0}{\Sigma \vdash e = E[e_0] \to E[e'_0] = e' \dashv \Sigma'} \text{[Step]}$$

By Lemma 88, we get $\Sigma \vdash [\![e_0]\!]^{\mathbf{SPI}} \to^+ [\![e'_0]\!]^{\mathbf{SPI}} \dashv \Sigma'$. Thus, we get $\Sigma \vdash [\![e]\!]^{\mathbf{SPI}} = [\![E[e_0]]\!]^{\mathbf{SPI}} = [\![E]\!]^{\mathbf{SPI}}[[\![e_0]\!]^{\mathbf{SPI}}] \to^+ [\![E]\!]^{\mathbf{SPI}}[[\![e_0]\!]^{\mathbf{SPI}}] = [\![E[[\![e'_0]\!]^{\mathbf{SPI}}]]\!]^{\mathbf{SPI}} = [\![e']\!]^{\mathbf{SPI}} \dashv \Sigma'$ by Lemma 87 and 48. □

G.3 Macro Translation from $\lambda^{l+\eta}_{\texttt{eff}}$ to $\lambda^{l+\eta}_{\texttt{del}}$

$$\begin{array}{lcl}
\llbracket x \rrbracket^{\mathbf{SIP}} & = & x \\
\llbracket \lambda x.e \rrbracket^{\mathbf{SIP}} & = & \lambda x.\llbracket e \rrbracket^{\mathbf{SIP}} \\
\llbracket e\ e \rrbracket^{\mathbf{SIP}} & = & \llbracket e \rrbracket^{\mathbf{SIP}}\ \llbracket e \rrbracket^{\mathbf{SIP}} \\
\llbracket \mathbf{do}\langle \ell \rangle\ v \rrbracket^{\mathbf{SIP}} & = & \mathbf{shift}_0\langle \ell \rangle\ k.\ \lambda h.h\ \llbracket v \rrbracket^{\mathbf{SIP}}\ (\lambda x.k\ x\ h) \\
\llbracket \mathbf{handle}\langle l \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{SIP}} & = & \langle \llbracket e \rrbracket^{\mathbf{SIP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{SIP}} \rangle_l\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{SIP}}) \\
\llbracket \Lambda \eta.e \rrbracket^{\mathbf{SIP}} & = & \Lambda \eta.\llbracket e \rrbracket^{\mathbf{SIP}} \\
\llbracket e\ [\ell] \rrbracket^{\mathbf{SIP}} & = & \llbracket e \rrbracket^{\mathbf{SIP}}\ [\ell] \\
\llbracket \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \rrbracket^{\mathbf{SIP}} & = & \langle \llbracket e \rrbracket^{\mathbf{SIP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{SIP}} \rangle_\eta\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{SIP}})
\end{array}$$

FIGURE G.8: Macro translation of expressions

$$\begin{array}{lcl}
\llbracket \alpha \rrbracket^{\mathbf{SIP}} & = & \alpha \\
\llbracket \tau \rightarrow_\rho \tau \rrbracket^{\mathbf{SIP}} & = & \llbracket \tau \rrbracket^{\mathbf{SIP}} \rightarrow_{\llbracket \rho \rrbracket^{\mathbf{SIP}}} \llbracket \tau \rrbracket^{\mathbf{SIP}} \\
\llbracket \forall \alpha :: \kappa.\tau \rrbracket^{\mathbf{SIP}} & = & \forall \alpha :: \kappa.\llbracket \tau \rrbracket^{\mathbf{SIP}} \\
\llbracket \epsilon \cdot \rho \rrbracket^{\mathbf{SIP}} & = & \llbracket \epsilon \rrbracket^{\mathbf{SIP}} \cdot \llbracket \rho \rrbracket^{\mathbf{SIP}} \\
\llbracket \exists_\ell\ \Delta'.\tau_1 \Rightarrow \tau_2 \rrbracket^{\mathbf{SIP}} & = & \exists_\ell\ \alpha :: \mathbf{T}, \beta :: \mathbf{R}.\ ((\forall \Delta'.\llbracket \tau_1 \rrbracket^{\mathbf{IP}} \rightarrow_\iota (\llbracket \tau_2 \rrbracket^{\mathbf{IP}} \rightarrow_\beta \alpha) \rightarrow_\beta \alpha) \rightarrow_\beta \alpha)/\beta
\end{array}$$

FIGURE G.9: Macro translation of types

$$\begin{array}{lcl}
\llbracket \varnothing \rrbracket^{\mathbf{SIP}} & = & \varnothing \\
\llbracket \Gamma, x : \tau \rrbracket^{\mathbf{SIP}} & = & \llbracket \Gamma \rrbracket^{\mathbf{SIP}}, x : \llbracket \tau \rrbracket^{\mathbf{SIP}}
\end{array}$$

FIGURE G.10: Macro translation of contexts

$$\llbracket \{\overline{\tau/\alpha}\} \rrbracket^{\mathbf{SIP}} \quad = \quad \{\overline{\llbracket \tau \rrbracket^{\mathbf{SIP}}/\alpha}\}$$

FIGURE G.11: Macro translation of substitutions

$$\begin{array}{lcl}
[\![\square]\!]^{\mathbf{SIP}} & = & \square \\
[\![E\ e]\!]^{\mathbf{SIP}} & = & [\![E]\!]^{\mathbf{SIP}}\ [\![e]\!]^{\mathbf{SIP}} \\
[\![v\ E]\!]^{\mathbf{SIP}} & = & [\![v]\!]^{\mathbf{SIP}}\ [\![E]\!]^{\mathbf{SIP}} \\
[\![\mathbf{handle}\langle l\rangle\ E\ \mathbf{with}\ \{x, r.e_h; x.e_r\}]\!]^{\mathbf{SIP}} & = & \langle [\![E]\!]^{\mathbf{SIP}} \mid x.\ \lambda h.[\![e_r]\!]^{\mathbf{SIP}} \rangle_l\ (\lambda x.\lambda r.[\![e_h]\!]^{\mathbf{SIP}})
\end{array}$$

FIGURE G.12: Macro translation of evaluation contexts

$$\begin{array}{llcl}
\text{General Context} & C & ::= & \square \mid C\ e \mid e\ C \mid \lambda x.C \mid \langle C \mid x.\ e_r\rangle_l \mid \langle e \mid x.\ C\rangle_l \mid \mathbf{shift}_0\langle \ell\rangle\ k.\ C \\
 & & \mid & \Lambda\eta.C \mid C[\ell] \mid \langle C \mid x.\ e_r\rangle_\eta
\end{array}$$

FIGURE G.13: General contexts

$$\frac{e_1 \mapsto e_2}{C[e_1] \rightarrow_i C[e_2]}\ [\textsc{GStep}]$$

FIGURE G.14: General step

$$\frac{}{e \rightarrow_i^* e}\ [\textsc{MSGRefl}] \qquad \frac{e_1 \rightarrow_i e_2}{e_1 \rightarrow_i^* e_2}\ [\textsc{MSGRed}]$$

$$\frac{e_1 \rightarrow_i^* e_2 \qquad e_2 \rightarrow_i^* e_3}{e_1 \rightarrow_i^* e_3}\ [\textsc{MSGTrans}]$$

FIGURE G.15: Multi-step general reductions

$$\frac{e_1 \rightarrow_i e_2 \qquad e_2 \rightarrow_i^* e_3}{e_1 \rightarrow_i^+ e_3}\ [\textsc{S-MSGRed}]$$

FIGURE G.16: Strict multi-step general reductions

G.4 Proof of Soundness

Theorem 15 (Translation preserves typability)**.**
If $\Delta \mid \Gamma \mid \Sigma \vdash e : \tau / \rho$ then $\Delta \mid \llbracket \Gamma \rrbracket^{\mathbf{SIP}} \mid \Sigma \vdash \llbracket e \rrbracket^{\mathbf{SIP}} : \llbracket \tau \rrbracket^{\mathbf{SIP}} / \llbracket \rho \rrbracket^{\mathbf{SIP}}$.

Proof. We prove the difference cases only from Theorem 7.

Case $\textsc{LAbs}$ **and** $\textsc{LApp}$**:**
The result follows from I.H directly.

Case $\textsc{LHandle}$**:**

$$\frac{\begin{array}{c}\Delta, \eta :: \mathbf{L} \mid \Gamma \mid \Sigma \vdash e : \tau / (\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \\ \Delta, \Delta' \mid \Gamma, x : \tau_1, r : \tau_2 \to_\rho \tau_r \mid \Sigma \vdash e_h : \tau_r / \rho \qquad \Delta \mid \Gamma, x : \tau \mid \Sigma \vdash e_r : \tau_r / \rho\end{array}}{\Delta \mid \Gamma \mid \Sigma \vdash \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} : \tau_r / \rho}\ [\textsc{LHandle}]$$

By I.H, we get (1) $\Delta, \eta :: \mathbf{L} \mid \llbracket \Gamma \rrbracket^{\mathbf{SIP}} \mid \Sigma \vdash \llbracket e \rrbracket^{\mathbf{SIP}} : \llbracket \tau \rrbracket^{\mathbf{SIP}} / \llbracket (\exists_\eta\ \Delta'.\tau_1 \Rightarrow \tau_2) \cdot \rho \rrbracket^{\mathbf{SIP}}$, (2) $\Delta, \Delta' \mid \llbracket \Gamma \rrbracket^{\mathbf{SIP}}, x : \llbracket \tau_1 \rrbracket^{\mathbf{SIP}}, r : \llbracket \tau_2 \to_\rho \tau_r \rrbracket^{\mathbf{SIP}} \mid \Sigma \vdash \llbracket e_h \rrbracket^{\mathbf{SIP}} : \llbracket \tau_r \rrbracket^{\mathbf{SIP}} / \llbracket \rho \rrbracket^{\mathbf{SIP}}$ and (3) $\Delta \mid \llbracket \Gamma \rrbracket^{\mathbf{SIP}}, x : \llbracket \tau \rrbracket^{\mathbf{SIP}} \mid \Sigma \vdash \llbracket e_r \rrbracket^{\mathbf{SIP}} : \llbracket \tau_r \rrbracket^{\mathbf{SIP}} / \llbracket \rho \rrbracket^{\mathbf{SIP}}$.
By the same way of $\textsc{Handle}$ and $\delta = \{\llbracket \tau_r \rrbracket^{\mathbf{IP}} / \alpha, \llbracket \rho \rrbracket^{\mathbf{IP}} / \beta\}$, we get (4) $\Delta \mid \llbracket \Gamma \rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{IP}} : \forall \Delta'.\llbracket \tau_1 \rrbracket^{\mathbf{IP}} \to_\iota ((\llbracket \tau_2 \to_\rho \tau_r \rrbracket^{\mathbf{IP}}) \to_{\llbracket \rho \rrbracket^{\mathbf{IP}}} \llbracket \tau_r \rrbracket^{\mathbf{IP}} / \iota$. (5) $\Delta \mid \llbracket \Gamma \rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \langle \llbracket e \rrbracket^{\mathbf{IP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{IP}} \rangle_\eta : \delta((\forall \Delta'.\llbracket \tau_1 \rrbracket^{\mathbf{IP}} \to_\iota ((\llbracket \tau_2 \rrbracket^{\mathbf{IP}} \to_\beta \alpha) \to_\beta \alpha) \to_\beta \alpha) / \delta(\beta)$.
Thus, we get $\Delta \mid \llbracket \Gamma \rrbracket^{\mathbf{IP}} \mid \Sigma \vdash \langle \llbracket e \rrbracket^{\mathbf{SIP}} \mid x.\ \lambda h.\llbracket e_r \rrbracket^{\mathbf{SIP}} \rangle_\eta\ (\lambda x.\lambda r.\llbracket e_h \rrbracket^{\mathbf{SIP}}) : \llbracket \tau_r \rrbracket^{\mathbf{SIP}} / \llbracket \rho \rrbracket^{\mathbf{SIP}}$ by $\textsc{App}$, $\textsc{Sub}$, (4) and (5).

□

Lemma 89 (Translated value is also a value)**.**
If v is a value then $\llbracket v \rrbracket^{\mathbf{SIP}}$ is also a value.

Proof. Straightforward. by the definition of $\llbracket \cdot \rrbracket^{\mathbf{PI}}$. □

Lemma 90 (Translation of expressions is commutative)**.**
$\llbracket e\{v/x\} \rrbracket^{\mathbf{SIP}} = \llbracket e \rrbracket^{\mathbf{SIP}}\{\llbracket v \rrbracket^{\mathbf{SIP}}/x\}$, for any expression e.

Proof. We prove the difference cases only from Lemma 56.

Case $e = \Lambda\eta.e_0$, $e = e_0[\ell]$ **and** $e = \mathbf{handle}\langle \eta \rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}$**:**
The result follows directly from I.H.

□

Lemma 91 (Translation of evaluation contexts is commutative)**.**
$\llbracket E[e] \rrbracket^{\mathbf{SIP}} = \llbracket E \rrbracket^{\mathbf{SIP}}[\llbracket e \rrbracket^{\mathbf{SIP}}]$, for any evaluation context E.

Proof. We prove the difference cases only from Lemma 57.

Case $E = E_0\ [\ell]$**:**
The result follow directly from I.H.

□

Lemma 92 (Translation preserves reductions)**.**
If $\Sigma \vdash e \mapsto e' \dashv \Sigma'$ then $\Sigma \vdash [\![e]\!]^{\mathbf{PI}} \rightarrow_i^+ [\![e']\!]^{\mathbf{PI}} \dashv \Sigma'$.

Proof. We prove the difference cases only from Lemma 49.

Case ELAPP:
Straightforward.

Case EGENLABEL:

$$\frac{l \text{ is fresh} \qquad \Sigma' = \Sigma, l \qquad \delta = \{l/\eta\}}{\Sigma \vdash\ \mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\} \mapsto \mathbf{handle}\langle l\rangle\ \delta(e)\ \mathbf{with}\ \{x, r.e_h; x.e_r\}\ \dashv \Sigma'}\ [\text{EGENLABEL}$$

$$\begin{array}{ll}
\Sigma \dashv [\![\mathbf{handle}\langle\eta\rangle\ e\ \mathbf{with}\ \{x, r.e_h; x.e_r\}]\!]^{\mathbf{SIP}} & \\
= \langle [\![e]\!]^{\mathbf{SIP}} \mid x.\ \lambda h.[\![e_r]\!]^{\mathbf{SIP}} \rangle_\eta\ (\lambda x.\lambda r.[\![e_h]\!]^{\mathbf{SIP}}) & (\text{definition of } [\![\cdot]\!]^{\mathbf{SIP}}) \\
\rightarrow \langle \delta([\![e]\!]^{\mathbf{SIP}}) \mid x.\ \lambda h.[\![e_r]\!]^{\mathbf{SIP}} \rangle_l\ (\lambda x.\lambda r.[\![e_h]\!]^{\mathbf{SIP}}) & (\text{EGENLABEL and } \delta = \{l/\eta\}) \\
= \langle [\![\delta(e)]\!]^{\mathbf{SIP}} \mid x.\ \lambda h.[\![e_r]\!]^{\mathbf{SIP}} \rangle_l\ (\lambda x.\lambda r.[\![e_h]\!]^{\mathbf{SIP}}) & (\text{definition of } [\![\cdot]\!]^{\mathbf{SIP}}) \\
= [\![\mathbf{handle}\langle l\rangle\ \delta(e)\ \mathbf{with}\ \{x, r.e_h; x.e_r\}]\!]^{\mathbf{SIP}} \dashv \Sigma' & (\text{definition of } [\![\cdot]\!]^{\mathbf{SIP}})
\end{array}$$

Thus, we get $\Sigma \dashv [\![e]\!]^{\mathbf{SIP}} \rightarrow_i^+ [\![e']\!]^{\mathbf{SIP}} \dashv \Sigma'$.

□

Theorem 16 (Translation preserves meaning)**.**
If $\Sigma \vdash e \longrightarrow e' \dashv \Sigma'$ then $\Sigma \vdash [\![e]\!]^{\mathbf{SPI}} \rightarrow_i^+ [\![e']\!]^{\mathbf{SPI}} \dashv \Sigma'$.

Proof.
We can prove this lemma by the same way to Theorem 8 using Lemma 59, 91, and 60. □