Resource Usage Analysis for a Functional Language with Exceptions

Futoshi Iwama
Tohoku University
iwama@kb.ecei.tohoku.ac.jp

Atsushi Igarashi
Kyoto University †
igarashi@kuis.kyoto-u.ac.jp

Naoki Kobayashi
Tohoku University †
koba@kb.ecei.tohoku.ac.jp

December 19, 2006

Abstract

Igarashi and Kobayashi have proposed a general type system for checking whether resources such as files and memory are accessed in a valid manner. Their type system is, however, for call-by-value λ-calculus with resource primitives, and does not deal with non-functional primitives such as exceptions and pointers. We extend their type system to deal with exception primitives and prove soundness of the type system. Dealing with exception primitives is especially important in practice, since many resource access primitives may raise exceptions. The extension is non-trivial: While Igarashi and Kobayashi's type system is based on linear types, our new type system is a combination of linear types and effect systems. We also report on a prototype analyzer based on the new type system.

1 Introduction

Background There has recently been growing interest in methods for verifying that resources (such as files, memories or network channels) used in a program are accessed in a valid manner [1, 3, 8, 10, 23, 25]. For example, Tofte et al.'s type system [23] can ensure that certain memories are not accessed after their deallocation and Freund and Mitchell's type system [8] can ensure that objects are used only after their initialization is finished. Igarashi and Kobayashi [10] have coined the term "resource usage analysis" for such general verification problems and proposed a type-based method for resource usage analysis for call-by-value λ-calculus with resource primitives.

The idea of their type system is to augment types of resources with information about in which order resources are accessed. For example, the type of read-only files is given by $(\mathbf{File}, R^*; C)$ and the type of read-write files is given by $(\mathbf{File}, (R + W)^*; C)$, where the order of operations on files is represented by regular expressions (the concatenation is given by ';') with R, W and C as labels for read, write and close operations, respectively (in their type system, actually, a more expressive language called *usage expressions* is used to represent the access order).

Typing rules are designed by taking the evaluation order into account. For example, the usual typing rule for **let** expressions:

$$\frac{\Gamma \vdash M : \sigma_1 \qquad \Gamma, x : \sigma_1 \vdash N : \sigma_2}{\Gamma \vdash \mathbf{let}\ x = M\ \mathbf{in}\ N : \sigma_2}$$

is replaced by:

$$\frac{\Gamma \vdash M : \sigma_1 \qquad \Delta, x : \sigma_1 \vdash N : \sigma_2}{\Gamma; \Delta \vdash \mathbf{let}\ x = M\ \mathbf{in}\ N : \sigma_2}$$

Here, $\Gamma;\Delta$ denotes the type environment obtained by composing the usage expression of Γ and Δ by ';'. For example, if $\Gamma = x : (\mathbf{File}, R^*)$, which intuitively means the resource x is read several times during the evaluation of M, and if $\Delta = x : (\mathbf{File}, C)$, which means x is closed (once) during the evaluation of N, then $\Gamma;\Delta = x : (\mathbf{File}, R^*; C)$, which means x is read several times *and then* closed. The new typing rule reflects the fact that M is evaluated first and then N is in **let** $x = M$ **in** N.

In this way, their type system can keep track of the order of accesses to resources. However, the target language is pure call-by-value λ-calculus only with resource primitives. So, it is not clear that the method can be extended to practical programming languages, which are usually equipped with non-functional primitives such as exceptions and references.

Our Contributions In this paper, we extend their type system to deal with exception primitives. This extension is very important in practice because access primitives in practical programming languages may raise exceptions such as *End_of_File*. Our technical contributions are summarized as follows:

- Design and formalization of a type system for resource usage analysis for a functional language with an exception handling mechanism;
- Proof of soundness of the type system;
- Development of a type reconstruction algorithm; and
- Implementation of a prototype resource usage analyzer.

Although the exception mechanism here is much simpler than that of ML or Java (there is only one exception constructor, which does not carry values), the extension is already non-trivial. While the type system of Igarashi and Kobayashi [10] is based on linear types, our new type system is based on a combination of linear type and effect systems [21, 22]. In fact, even for the problem of just checking that certain values are used only once (which can be considered an instance of the resource usage analysis problem), previous linear type systems [10, 13, 16, 24] are insufficient; For example, they cannot judge that x is used once in the following program (suppose that `use` is a function that uses its argument just once):

```
let   f () = use x
in    try (if b then f() else raise E)
      with E -> use x
```

Key Ideas in the Type System As mentioned above, we extend Igarashi and Kobayashi's type system to deal with exception primitives, in particular, we add the new terms **raise** and **try** M_1 **with** M_2. Here, **raise** raise an exception and **try** M_1 **with** M_2 evaluates a term M_1 and then evaluates M_2 if an exception is raised during the evaluation of M_1. So, we must analyze in which order each resource is accessed even in situations that an exception is raised and uncaught or the exception is caught by an exception handler and the evaluation of the hander code proceeds.

First, we keep track of information on exceptions, as well as that on access sequences, by usage expressions. For this purpose, we extend usage expressions by introducing the special label E and the constructor $;_E$. E means that a resource is not accessed any more because an exception is raised; for example, the usage $R; E$ means a resource is first read and then an exception is raised. The usage $U_1;_E U_2$, which corresponds to exception handling, means that a resource is used according to U_1 and if an exception is raised, then it is used according to U_2. Thus, for example, the usage $(R; E);_E C$ is equivalent to $R; C$.

In order to deal with new exception primitives, it may seem enough to add the following simple typing rules using the new usage expressions to Igarashi and Kobayashi's type system:

$$\frac{Use(\tau_i) = E \ \ for\ each\ i \in \{1, \cdots, n\}}{x_1 : \tau_1, \ldots, x_n : \tau_n \vdash \mathbf{raise} : \tau} \qquad \text{(T-Raise-Simple)}$$

$$\frac{\Gamma \vdash M_1 : \tau \qquad \Delta \vdash M_2 : \tau \qquad dom(\Gamma) = dom(\Delta)}{\Gamma ;_E \Delta \vdash \mathbf{try}\ M_1\ \mathbf{with}\ M_2 : \tau} \qquad \text{(T-Try-Simple)}$$

Here, the premis part of the first rule states that if τ_i is resouce type, then the usage of type τ_i is E, which means an exception is raised and $\Gamma ;_E \Delta$ in the conclution part of the second rule denotes the type environment obtained by composing the usage expression of Γ and Δ by '$;_E$'. The first rule says that, during the evaluation of the term **raise**, any resouce is never accessed but an exception is raised and the secont rule says that, during evaluation of **try** M_1 **with** M_2, each resource is accessed according to Γ during the evaluation of M_1 and, if an exception is raised, is accessed according to Δ during the evaluation of M_2.

Now, for example, `try (read(x);raise) with E -> close(x)` is typed as follows:

$$\frac{\Pi = \dfrac{\begin{array}{c} x : (\mathbf{File}, R) \vdash \texttt{read(x)} : \mathbf{bool} \\ x : (\mathbf{File}, E) \vdash \texttt{raise} : \mathbf{bool} \end{array}}{x : (\mathbf{File}, R; E) \vdash \texttt{read(x)};\texttt{raise} : \mathbf{bool}} \qquad \Pi \quad x : (\mathbf{File}, C) \vdash \mathbf{close}(x) : \mathbf{bool}}{x : (\mathbf{File}, (R; E);_E C) \vdash \begin{array}{ll} \texttt{try} & \texttt{read(x);raise} \\ \texttt{with} & \texttt{E -> close(x)} \end{array} : \mathbf{bool}.}$$

Notice that $;_E$ corresponds to the use of `try`. The type judgment above tells us that, during the evaluation of M, the file x is first read and then closed. (Here, we assume access primitives themselves do not raise exceptions; we can model access primitives that may raise exceptions by combining them with conditional expressions.)

Unfortunately, using only usage expressions is not sufficient to obtain the accuracy required for practical programs using exceptions (so, the typing rules (T-Raise-Simple), (T-Try-Simple) is not sufficient). The problem stems from the fact that Igarashi and Kobayashi's type system does not give very accurate usage information when resources are put in function closures. For example, let us concider the followings:

$$\begin{array}{lcl} M_{rc} & \triangleq & \texttt{let f} = \lambda\texttt{y.close(x)in (read(x);f())} \\ M_{rec} & \triangleq & \texttt{let f} = \lambda\texttt{y.raise in (try read(x);f() with close(x))}. \end{array}$$

Here, the typing rules for lambda-abstraction terms in Igarashi and Kobayashi's type system is given as follows:

$$\frac{\Gamma, x : \tau_1 \vdash M : \tau_2}{\Diamond\Gamma \vdash \lambda x.M : \tau_1 \to \tau_2.} \qquad \text{(T-Abs-IK)}$$

The intuitive meanigs of $\Diamond\Gamma$ in the conclusion part of the rule (T-Abs-IK) is that accesses according to Γ are encapsulated in the body of a function and may be delayed (until the function is called).

So, using this typing rule (T-Abs-IK), the usage of resource x in the term M_{rc} is given as $\Diamond C; R$. As mentioned above , the usage $\Diamond C$ roughly means that the close operation represented by C may be delayed. As a result the usage $(\Diamond C); R$ can mean either $R; C$ or $C; R$, which means the resource x may be read after closed.

Similarly, using this typing rules (T-Abs-IK), (T-Raise-Simple) and (T-Try-Simple), the term M_{rec} is typed under the type environment $x : (\mathbf{R}, (\Diamond E); (R;_E C))$, which can mean that the resouce x is accessed according to either $E; (R;_E C)$ (equivalent to E), $(E; R);_E C$ (equivalent to C), $(R; E);_E C$

(equivalent to $R;C$) or $(R;_E C);E$ (equivalent to $R;E$). The first usage expression says that the resouce is never accessed, the second $\sim$ forth says that the resouce is just closed, read and then closed, just read respectively. We therefore cannot know almost in what order and by what kind of operations the resource x is accessed in term M_{rec}.

As is shown in the above example M_{rc}, the inaccurate usage problem goes for only the exception but also for other operations to access resources [14]. However the timing of exception raising have a big influence on the control of evaluation and on the usage of *all* resources in the context as shown in above example M_{rec}. Note that we get the inaccurate information about usage of resouce x even though the function $\lambda y.\mathbf{raise}$ does not include x as a free variable. So, this problem of inaccuracy due to an exception will be very serious.

To solve this problem, we exploit the idea of effect systems [21, 22] to keep track of more accurate information on *when* exceptions are raised. As a result, a type judgment becomes of the form:

$$\Gamma \parallel \varphi \vdash M : \sigma,$$

which means that, during the evaluation of M, each resource is accessed according to (usages in) Γ and exceptions are raised according to effect φ. Under the type judement, the new typing rule for functions is given by:

$$\frac{\Gamma, x : \sigma_1 \,||\, \varphi \vdash M : \sigma_2}{\Diamond(\Gamma \backslash E) \,||\, \mathbf{0} \vdash \lambda x.M : (\sigma_1 \backslash E) \xrightarrow{\varphi} \sigma_2}$$

In the above rule, $(\Gamma \backslash E), (\sigma_1 \backslash E)$ denotes the type environment, type obtained by deleting the information about exception raising (namely, usage expression E) form Γ, σ. This rule states that if an exception is raised according to φ during the evaluation of function body M, then the effect φ is expressed as the latent effect of the function $\lambda x.M$ and that the information about exception raising is deleted from type environment Γ and type of function's argument σ_1 because the information is expressed as latent effect in the type of functions. For example, using the new typing rule (and others introduced in this papar) the usage of resource x in the above term M_{rec} is given as $(R;E);_E C$ as expected.

Rest of This Paper In Section 2, we introduce our target language formally. In Section 3, we present our type system for resource usage analysis. Section 4 shows the correctness of the type system. Section 5 describes a type inference algorithm. Section 6 reports on implementation and experiments. We discuss other possible methods for dealing with the exception mechanism in Section 7. After discussing related work in Section 8, we conclude in Section 9.

2 Target Language $\lambda^{\mathcal{R}}_{E}$

This section introduces the target language $\lambda^{\mathcal{R}}_{E}$ of our analysis. $\lambda^{\mathcal{R}}_{E}$ is a call-by-value λ-calculus extended with resources and exceptions.

We assume a finite set $\mathcal{A}$ of *access-labels*, ranged over by a. An access label denotes the kind of access to a resource; We shall use access labels I, R, W and C for initialization, read, write, and close operations respectively.

A *trace* is a sequence consisting of access labels and the special label $\downarrow$. Formally, the set $\mathcal{A}^{*,\downarrow}$ of traces is defined by:

$$\mathcal{A}^{*,\downarrow} = \mathcal{A}^* \cup \{s \downarrow \,|\, s \in \mathcal{A}^*\}$$

Here, $\mathcal{A}^*$ is the set of finite sequences of elements of $\mathcal{A}$. The additional symbol $\downarrow$ expresses that the evaluation terminates normally or abruptly with an exception.

A trace expresses how a resource has been accessed at a certain point of the execution of a program. A trace $a_1 a_2 \ldots a_k$ means that the resource is accessed by each operation a_i in the order $a_1, a_2, \ldots, a_k$.

A trace $a_1 a_2 \ldots a_k \downarrow$ means that the evaluation has terminated after the resource is accessed in the order $a_1, a_2, \ldots, a_k$. For example, a trace $RRC\downarrow$ means that the resource has been read twice, closed, and then the evaluation has terminated.

A *trace set* is a set of traces that is closed under the prefix operation. We write $S^{\#}$ for the set of prefixes of elements of S. A set S of traces is called a *trace set* if $S^{\#} = S$. We use the metavariable Φ for a trace set.

Consider a regular expression $(RW\downarrow)$, then $(RW\downarrow)^{\#} = \{\epsilon, R, RW, RW\downarrow\}$ is a trace set. We can consider a trace set Φ as a specification of each resource, which requires the resource is accessed only according to a trace in the set Φ.

Example 2.1 Let us consider the trace set $\Phi = (IR^*C\downarrow)^{\#}$ and program: $\mathbf{init}(x); \mathbf{read}(x); \mathbf{close}(x)$. The program initializes, reads and closes the resource x. This program satisfies the specification Φ. On the other hand, neither $\mathbf{read}(x); \mathbf{close}(x)$ nor $\mathbf{init}(x); \mathbf{read}(x)$ satisfies Φ, since the resource x can be accessed according to $RC\downarrow$ or $IR\downarrow$ but neither trace is contained in Φ. The meaning of $\downarrow$ is a little tricky; Let us consider the program:
$\mathbf{init}(x); \mathbf{read}(x);$ *loop_infinitely*. It initializes and reads the resource x, and then goes into an infinite loop. This program *does* satisfy the specification Φ. Actually, the resource x is accessed according to a trace in the set $\{IR\}^{\#} \subset \Phi$. □

Definition 2.2 (Terms, Values)

$$\begin{array}{lcl} M \text{ (terms)} & ::= & v \mid M_1 M_2 \mid \mathbf{let}\ x = M_1\ \mathbf{in}\ M_2 \mid \mathbf{if}\ M_1\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3 \\ & \mid & \mathbf{new}^{\Phi}() \mid \mathbf{acc}^{a}(M) \mid M^{\{x\}} \mid \mathbf{try}\ M_1\ \mathbf{with}\ M_2 \mid \mathbf{raise} \\ v \text{ (values)} & ::= & \mathbf{true} \mid \mathbf{false} \mid x \mid \mathbf{fun}(f, x, M) \end{array}$$

The first line shows standard constructs for the λ-calculus. $\mathbf{fun}(f, x, M)$ is a recursive function (which is often defined by $f(x) = M$). We write $\lambda x.M$ when f is not free in M. The primitives for resources are the same as those of $\lambda^{\mathcal{R}}$ [10]: $\mathbf{new}^{\Phi}()$ is the primitive for creating a resource. The trace set Φ specifies how the resource should be accessed afterwards. In this paper, we often use a regular expression to specify a trace set. $\mathbf{acc}^{a}(M)$ is the primitive for accessing the resource M with an operation specified by a. We assume that the resource access primitive returns a boolean. $M^{\{x\}}$ is the same as M, except that the evaluation gets stuck if the resource bound to x escapes from M. The escape information is used to refine the accuracy of the analysis. A separate escape analysis is assumed, which checks that x does not escape from M in $M^{\{x\}}$. The term $\mathbf{try}\ M_1\ \mathbf{with}\ M_2$ first evaluates M_1. If an exception is raised, then M_2 is evaluated; Otherwise, the value of M_1 is returned. The term **raise** raises an exception. For the sake of simplicity, we consider a single kind of exception, so **raise** takes no argument.

We write $M_1; M_2$ for $\mathbf{let}\ x = M_1\ \mathbf{in}\ M_2$ if variable x is not free in M_2.

Example 2.3 Consider the following terms:

$$\begin{array}{l} M_1 \triangleq \mathbf{if}\ \mathbf{init}(x)^{\{x\}}\ \mathbf{then}\ (\mathbf{write}(x)^{\{x\}}; \mathbf{close}(x)^{\{x\}})\ \mathbf{else}\ \mathbf{raise} \\ M_2 \triangleq \mathbf{try}\ M_1\ \mathbf{with}\ \mathbf{close}(x)^{\{x\}} \\ M \triangleq \mathbf{let}\ x = \mathbf{new}^{(I(W)^*C\downarrow)^{\#}}()\ \mathbf{in}\ M_2 \end{array}$$

M_1 first initializes x. If the initialization returns **true**, then it writes and closes x; otherwise an exception is raised. The term M_2 closes x when an exception is raised by M_1. Therefore, the resource x is closed no matter whether $\mathbf{init}(x)$ returns **true** or **false**. The term M creates a resource x and evaluates M_2. The trace set $(I(W)^*C\downarrow)^{\#}$ specifies that x should be first initialized, that it can be written an arbitrary number of times after that, and that it must be closed once before the program terminates. The term M_2 obeys that specification, so that M should be accepted as a good program. □

Example 2.4 The following is a fragment of a typical OCaml program accessing files.

```
try  while (true)
      do  write_char(y,read_char(x))  done
with  End_of_File -> close(x);close(y)
```

It copies a character from x to y until the end-of-file exception is raised, when the exception handler is executed and x and y are closed.

The above program can be modeled in our language as the following term M.

$$M \triangleq \mathbf{try}\ \mathbf{fun}(f, z, M_1)\mathbf{true}\ \mathbf{with}\ (\mathbf{close}(x); \mathbf{close}(y))$$

Here, M_1 is

$$(\mathbf{if}\ \mathbf{read}(x)\ \mathbf{then\ true\ else\ raise}); \mathbf{write}(y); f\ \mathbf{true}$$

Note that the library function call `read_char(x)`, which may raise an exception, has been modeled by **if read**(x) **then true else raise**. □

2.1 Operational semantics of $\lambda_E^{\mathcal{R}}$

An operational semantics of $\lambda_E^{\mathcal{R}}$ is given by reduction of pairs of a term and a *heap*, used to record the states of resources.

Definition 2.1 (Heap) *A* heap H *is a finite mapping from variables to trace sets.*

We write $\{x_1 \mapsto \Phi_1, \ldots, x_n \mapsto \Phi_n\}$ $(n \geq 0)$ for the heap H such that $dom(H) = \{x_1, \ldots, x_n\}$ and $H(x_i) = \Phi_i$. It expresses that each variable x_i refers to a resource that should be used according to one of the traces of Φ_i. When $dom(H_1) \cap dom(H_2) = \emptyset$, we write $H_1 \uplus H_2$ for the heap H such that $dom(H) = dom(H_1) \cup dom(H_2)$ and $H(x) = H_i(x)$ for x in $dom(H_i)$.

The reduction relation is given using evaluation contexts, defined below.

Definition 2.2 (Evaluation Contexts) Evaluation contexts*, ranged over by* $\mathcal{E}$*, and* evaluation contexts without **try***, ranged over by* $\mathcal{E}^{\overline{try}}$*, are defined by the following syntax:*

$$\begin{array}{rcl} \mathcal{E} & ::= & [\,] \mid \mathbf{if}\ \mathcal{E}\ \mathbf{then}\ M_1\ \mathbf{else}\ M_2 \mid \mathcal{E}\ M \mid v\ \mathcal{E} \\ & \mid & \mathbf{let}\ x = \mathcal{E}\ \mathbf{in}\ M \mid \mathcal{E}^{\{x\}} \mid \mathbf{acc}^a(\mathcal{E}) \mid \mathbf{try}\ \mathcal{E}\ \mathbf{with}\ M \\ \\ \mathcal{E}^{\overline{try}} & ::= & [\,] \mid \mathbf{if}\ \mathcal{E}^{\overline{try}}\ \mathbf{then}\ M_1\ \mathbf{else}\ M_2 \mid \mathcal{E}^{\overline{try}}\ M \mid v\ \mathcal{E}^{\overline{try}} \\ & \mid & \mathbf{let}\ x = \mathcal{E}^{\overline{try}}\ \mathbf{in}\ M \mid \mathcal{E}^{\overline{try}\{x\}} \mid \mathbf{acc}^a(\mathcal{E}^{\overline{try}}) \end{array}$$

$\mathcal{E}$ is an ordinary call-by-value evaluation context; $\mathcal{E}^{\overline{try}}$ is one without exception handlers and used to identify an innermost exception handler. We write $\mathcal{E}[M]$ and $\mathcal{E}^{\overline{try}}[M]$ for the expressions obtained by replacing $[\,]$ in $\mathcal{E}$ and $\mathcal{E}^{\overline{try}}$ respectively with M.

We write $[M_1/x_1, \ldots, M_n/x_n]$ for the capture-avoiding simultaneous substitution of M_i for x_i and Φ^{-a} for $\{s \mid as \in \Phi\}$. $\mathbf{FV}(M)$ denotes the set of free variables in M.

Definition 2.3 (Reduction Relation) *The relation* $(H, M) \rightsquigarrow P$*, where* P *is either a pair* (H', M') *or* **Error***, is defined as the least relation closed under the rules in Figure 1. We write* $\rightsquigarrow^*$ *for the reflexive and transitive closure of* $\rightsquigarrow$*.*

The rule R-New is for fresh resource allocation. The rules R-Acc and R-AccErr express an access to a resource. The rule R-Acc is for successful resource access: the trace set Φ^{-a} after the access is obtained by removing the label a at the head of a trace (if the trace begins with a; the traces not beginning with a are discarded). If no trace begins with a (i.e., $\Phi^{-a} = \emptyset$), then the resource access results in an **Error**. We do not care about the result of resource access here, it is left unspecified which boolean values are returned in R-Acc, so reduction $\rightsquigarrow$ is nondeterministic. The rule R-TryRai is for exception handling. A term of the form **try** $\mathcal{E}^{\overline{try}}[\mathbf{raise}]$ **with** M represents the execution state in which an exception is being raised and the innermost handler is M, and so reduces to M.

$$\frac{z \text{ fresh}}{(H, \mathcal{E}[\mathbf{new}^{\Phi}()]) \rightsquigarrow (H \uplus \{z \mapsto \Phi\}, \mathcal{E}[z])} \quad \text{(R-New)}$$

$$\frac{b = \mathbf{true} \text{ or } \mathbf{false} \qquad \Phi^{-a} \neq \emptyset}{(H \uplus \{x \mapsto \Phi\}, \mathcal{E}[\mathbf{acc}^{a}(x)]) \rightsquigarrow (H \uplus \{x \mapsto \Phi^{-a}\}, \mathcal{E}[b])} \quad \text{(R-Acc)}$$

$$\frac{\Phi^{-a} = \emptyset}{(H \uplus \{x \mapsto \Phi\}, \mathcal{E}[\mathbf{acc}^{a}(x)]) \rightsquigarrow \mathbf{Error}} \quad \text{(R-AccErr)}$$

$$(H, \mathcal{E}[\mathbf{fun}(f, x, M)\ v]) \rightsquigarrow (H, \mathcal{E}[[\mathbf{fun}(f, x, M)/f, v/x]M]) \quad \text{(R-App)}$$

$$(H, \mathcal{E}[\mathbf{if\ true\ then}\ M_1\ \mathbf{else}\ M_2]) \rightsquigarrow (H, \mathcal{E}[M_1]) \quad \text{(R-IfT)}$$

$$(H, \mathcal{E}[\mathbf{if\ false\ then}\ M_1\ \mathbf{else}\ M_2]) \rightsquigarrow (H, \mathcal{E}[M_2]) \quad \text{(R-IfF)}$$

$$\frac{x \notin \mathbf{FV}(v)}{(H, \mathcal{E}[v^{\{x\}}]) \rightsquigarrow (H, \mathcal{E}[v])} \quad \text{(R-ECheck)}$$

$$(H, \mathcal{E}[\mathbf{try}\ v\ \mathbf{with}\ M]) \rightsquigarrow (H, \mathcal{E}[v]) \quad \text{(R-Try)}$$

$$(H, \mathcal{E}[\mathbf{try}\ \mathcal{E}^{\overline{try}}[\mathbf{raise}]\ \mathbf{with}\ M]) \rightsquigarrow (H, \mathcal{E}[M]) \quad \text{(R-TryRai)}$$

Figure 1: Operational semantics of $\lambda_{E}^{\mathcal{R}}$

3 Type System

In this section, we present a type system to guarantee that all accesses to resources in a well-typed term obey the specification (given by the trace set Φ attached to each resource creation $\mathbf{new}^{\Phi}()$).

3.1 Usages

Usage expressions (in short, usages) describe in which order and by which operations a resource can be accessed. As mentioned above, we express information about exceptions also with usages. We therefore add new constructors E and $U_1 ;_E U_2$ to the usages given in [10].

Syntax of Usages.

Let the set $\mathcal{L}$ of *labels*, ranged over by l, be $\mathcal{A} \cup \{1, \tau, E\}$. The label 1 is a special label used to count the number of function applications; the labels τ and E denote exception handling (which is an unobservable, internal action, hence τ) and a raised exception, respectively.

Definition 3.1 (Usages) *The set $\mathcal{U}$ of* usages*, ranged over by U, is defined by:*

$$U \quad ::= \quad \mathbf{0} \mid l \mid \alpha \mid U_1; U_2 \mid U_1 \otimes U_2 \mid U_1 \& U_2 \mid \Diamond U \mid \blacklozenge U \mid \mu\alpha.U \mid U_1 ;_E U_2$$

We assume that the unary usage constructors $\Diamond$ and $\blacklozenge$ bind tighter than the binary constructors ($\&, ;, \otimes$ and $;_E$).

We briefly explain informal meaning of usage constructors; see also Igarashi and Kobayashi [10]. The usage $\mathbf{0}$ means that a resource cannot be accessed at all. The usage $l \in \mathcal{L}$ means that a resource is accessed by an access primitive labeled with l (if $l \in \mathcal{A}$), or that an event corresponding to l occurs: especially, $E \in \mathcal{L}$ means that a resource is not accessed later due to a raise of an exception. α is the usage variable, bound in the form of $\mu\alpha.U$, which denotes a recursive usage that satisfies $\alpha = U$. $U_1; U_2$ means that a resource is first accessed according to U_1 and then according to U_2. $U_1 \otimes U_2$ means that a resource is accessed according to a sequence obtained by interleaving U_1 and U_2. $U_1 \& U_2$ means that a resource is accessed according to either U_1 or U_2. $U_1;_E U_2$ means a resource is accessed according to U_1 and, if an exception is raised during the execution, then the resource is accessed according to U_2. For example, $((R\&E); W);_E C$ is equivalent to $(R; W)\&C$. $\Diamond U$ means that some of the resource access expressed by U may be delayed. For example, $(\Diamond l_1); l_2$ expresses access order either $l_1; l_2$ or $l_2; l_1$. $\blacklozenge U$, which cancels $\Diamond$, means that the access represented by U must occur *now*. So, $(\blacklozenge\Diamond U_1); U_2$ is equivalent to $U_1; U_2$.

Example 3.1 The accesses to x in M_2 of Example 2.3 is expressed by the usage $I; ((W; C)\&E);_E C$. Similarly, the accesses to x in M of Example 2.4 is expressed by the usage $\mu\alpha.(R; (\mathbf{0}\&E); \alpha);_E C$.

In what follows, we write $U \setminus E$ for $U;_E \mathbf{0}$, which cancels exceptions in U. For example, $((l_1; l_2) \otimes E) \setminus E$ is equivalent to $\mathbf{0}\&l_1\&(l_1; l_2)$.

Semantics of Usages and Subusage Relation.

We give the formal semantics of usages via a labeled transition system. Then, we define the subusage relation $U_1 \leq U_2$, which means that the access order U_1 is more general than U_2.

We define the transition relation of the form $U \xrightarrow{l} U'$, which means that a resource of usage U can be first accessed by l and then accessed according to U'. In this paper, the transition relation is defined in a manner similar to the one for process calculi.

Definition 3.2 *Let $\mathcal{C}^{\mathcal{T}}$ be evaluation usage contexts defined by the following syntax:*

$$\mathcal{C}^{\mathcal{T}} ::= [\,] \mid \Diamond\mathcal{C}^{\mathcal{T}} \mid \blacklozenge\mathcal{C}^{\mathcal{T}} \mid \mathcal{C}^{\mathcal{T}} \otimes U \mid \mathcal{C}^{\mathcal{T}}; U \mid \mathcal{C}^{\mathcal{T}};_E U.$$

The binary relation $U \preceq U'$ on usages are the least preorder that satisfies the rules in Figure 2 and preserved by all evaluation usage contexts, where $U \equiv U'$ means $U \preceq U'$ and $U' \preceq U$.

For example, $(\Diamond U_1\&U_2); U_3 \preceq \Diamond U_1 \otimes U_3$ holds.

$$\Diamond\mathbf{0} \equiv \mathbf{0} \quad \blacklozenge\mathbf{0} \equiv \mathbf{0}$$
$$\mathbf{0} \otimes U \equiv U \quad \mathbf{0}; U \equiv U \quad U; \mathbf{0} \equiv U \quad \mathbf{0};_E U \equiv \mathbf{0}$$
$$U_1 \otimes U_2 \equiv U_2 \otimes U_1 \quad U_1\&U_2 \equiv U_2\&U_1$$
$$\Diamond U_1 \otimes \Diamond U_2 \equiv \Diamond(U_1 \otimes U_2) \quad \Diamond U_1;_E U_2 \preceq \Diamond(U_1;_E U_2)$$
$$\Diamond U_1; U_2 \equiv \Diamond U_1 \otimes U_2$$
$$U_1\&U_2 \preceq U_1 \quad \mu\alpha.U \equiv [\mu\alpha.U/\alpha]U$$

Figure 2: Structual preorder rules on usages

Now, we give the transition rules on usages.

Definition 3.3 (Transition Relation on Usages) *The transition relation $U \xrightarrow{l} U'$ on usages is the least relation closed under the rules in Figure 3. The multi-step transition relation $U \xrightarrow{t} U'$, where*

$t \in (\mathcal{A} \cup \{1, \tau\})^*$, *is defined inductively as follows.*

$$\xrightarrow{t} \stackrel{def}{=} \begin{cases} \preceq & \text{if } t = \epsilon \\ \xrightarrow{l} \xrightarrow{t'} & \text{if } t = lt' \end{cases}$$

$$l \xrightarrow{l} \mathbf{0} \text{ (UR-Lab)} \quad \frac{U \xrightarrow{l} U'}{\Diamond U \xrightarrow{l} \Diamond U'} \text{ (UR-Box)} \quad \frac{U \xrightarrow{l} U'}{\blacklozenge U \xrightarrow{l} \blacklozenge U'} \text{ (UR-Ubox)}$$

$$\frac{U_1 \xrightarrow{l} U_1'}{U_1 \otimes U_2 \xrightarrow{l} U_1' \otimes U_2} \text{ (UR-Par)} \quad \frac{U_1 \xrightarrow{l} U_1'}{U_1; U_2 \xrightarrow{l} U_1'; U_2} \text{ (UR-Seq)}$$

$$\frac{U_1 \xrightarrow{l} U_1' \quad l \neq E}{U_1 ;_E U_2 \xrightarrow{l} U_1' ;_E U_2} \text{ (UR-SeqE)} \quad \frac{U_1 \xrightarrow{E} U'}{U_1 ;_E U_2 \xrightarrow{\tau} U_2} \text{ (UR-Hdlr)}$$

$$\frac{U_1 \preceq U_1' \quad U_1' \xrightarrow{l} U_2' \quad U_2' \preceq U_2}{U_1 \xrightarrow{l} U_2} \text{ (UR-Pcon)}$$

Figure 3: Usage transition rules

Example 3.2 Usage $R; (C \& E);_E C$ has the following transition sequences:

$$\begin{array}{llll} (R; (C \& E));_E C & \xrightarrow{R} & (C \& E);_E C & \xrightarrow{C} \mathbf{0} \\ (R; (C \& E));_E C & \xrightarrow{R} & (C \& E);_E C & \xrightarrow{\tau} C \xrightarrow{C} \mathbf{0} \end{array}$$

By using the labeled transition system, we define the trace set expressed by a usage.

Definition 3.4 *Let U be a usage. The trace set $[\![U]\!]$ is defined by*

$$\begin{array}{rcl} [\![U]\!] & = & \{\hat{t} \mid \exists U'.(U \xrightarrow{t} U')\} \\ & & \cup \{\hat{t}\downarrow \mid U \xrightarrow{t} \mathbf{0}\} \cup \{\hat{t}\downarrow \mid \exists U'.(U \xrightarrow{t} \xrightarrow{E} U')\} \end{array}$$

Here, $\hat{t} \in \mathcal{A}^*$ is the label sequence obtained by removing all the occurrences of τ from t.

Example 3.3 $[\![\mu\alpha.\alpha]\!] = \{\epsilon\}$, $[\![\mathbf{0}]\!] = \{\epsilon, \downarrow\}$, $[\![\Diamond l_1; l_2]\!] = \{l_1 l_2\downarrow, l_2 l_1\downarrow\}^{\#}$, $[\![\blacklozenge\Diamond l_1; l_2]\!] = \{l_1 l_2\downarrow\}^{\#}$, $[\![(R \& E); C]\!] = \{RC\downarrow\}^{\#}$, and $[\![((R \& E); C);_E C]\!] = \{RC\downarrow, C\downarrow\}^{\#}$.

We write $U_1 \Longrightarrow U_2$ when $U_1 \xrightarrow{\tau \cdots \tau} U_2$ (where $\tau \cdots \tau$ is a possibly empty sequence of τ). We also write $\stackrel{l}{\Longrightarrow}$ for $\Longrightarrow \xrightarrow{l} \Longrightarrow$.

The subusage relation $U_1 \leq U_2$ is defined as a weak simulation relation closed under usage contexts. A *usage context*, written $\mathcal{C}$, is an expression obtained from a usage by replacing one occurrence of a free usage variable with $[\,]$. Suppose that the set of free usage variables in U are disjoint from the set of bound usage variables in $\mathcal{C}$. We write $\mathcal{C}[U]$ for the usage obtained by replacing $[\,]$ with U. For example, if $\mathcal{C} = \mu\alpha.([\,]\,; \alpha)$, then $\mathcal{C}[U] = \mu\alpha.(U\,; \alpha)$.

Definition 3.5 (Subusage relation) *$U_1 \leq U_2$ is the largest relation that satisfies the following conditions:*

(1) $\mathcal{C}[U_1] \leq \mathcal{C}[U_2]$ *for any usage context* $\mathcal{C}$*,*

(2) If $U_2 \xrightarrow{l} U_2'$ *and* $l \in \mathcal{A} \cup \{1\}$*, then* $U_1 \stackrel{l}{\Longrightarrow} U_1'$ *and* $U_1' \leq U_2'$ *for some* U_1'*,*

(3) If $U_2 \xrightarrow{\tau} U_2'$*, then* $U_1 \Longrightarrow U_1'$ *and* $U_1' \leq U_2'$ *for some* U_1'*,*

(4) If $U_2 \xrightarrow{\epsilon} \mathbf{0}$*, then* $U_1 \Longrightarrow \mathbf{0}$*, and*

(5) If $U_2 \xrightarrow{E} U_2'$*, then* $U_1 \stackrel{E}{\Longrightarrow} U_1'$ *for some* U_1'*.*

We write $U_1 \cong U_2$ if $U_1 \leq U_2$ and $U_2 \leq U_1$.

Example 3.4 $E;_E C \cong C$ and $(C\&E)\setminus E \cong C\&\mathbf{0}$ hold.

Note that, if $U_1 \leq U_2$, then $[\![\mathcal{C}[U_1]]\!] \supseteq [\![\mathcal{C}[U_2]]\!]$ for any $\mathcal{C}$—in particular, $[\![U_1]\!] \supseteq [\![U_2]\!]$. Moreover, $U \leq \mu\alpha.\alpha$ and $\Diamond U \leq U$ hold for any usage U. $U_1 \preceq U_2$ implies $U_1 \leq U_2$ and $U_1 \equiv U_2$ implies $U_1 \cong U_2$.

3.2 Effects and Types

We proceed to the definitions of effects and types.

An *effect* expresses the termination behavior of an evaluation. Intuitively, the effect $\mathbf{0}$ means that evaluation can terminate normally; E that evaluation can abort with an exception; $E^?$ that evaluation can terminate normally or abort; and, finally, $\top$ that evaluation cannot terminate.

Definition 3.6 (Effects and Subeffect Relation) *The set of* effects*, ranged over by* φ*, is* $\{E^?, \mathbf{0}, E, \top\}$*. The subeffect relation* $\sqsubseteq$ *is the partial order given by* $E^? \sqsubseteq \mathbf{0} \sqsubseteq \top$ *and* $E^? \sqsubseteq E \sqsubseteq \top$*.*

Note that non-termination is denoted by $\top$, which is the greatest element of the order $\sqsubseteq$, as opposed to the common practice in denotational semantics, where non-termination denotes the least element. Viewing effects as the set of *termination capabilities* that a program can exercise, we define the order so that lower elements have more capabilities, similarly to the subusage relation.

Effects can be considered usages that do not include access labels. We write $(\varphi)^{use}$ for the usage corresponding to φ, defined by:

$$(E^?)^{use} = \mathbf{0}\&E \quad (\mathbf{0})^{use} = \mathbf{0} \quad (E)^{use} = E \quad (\top)^{use} = \mu\alpha.\alpha.$$

We define some operations on effects, which correspond to usage constructors of the same symbols.

Definition 3.7 (Operations on effects) *The operations on effects* $\varphi_1 \mathbf{op} \varphi_2$ *are defined (where* **op** *is either* $;$*,* $\&$*,*$\otimes$ *or* $;_E$*) by the following tables (the leftmost columns correspond to* φ_1 *and the topmost rows* φ_2*) :*

$;$	$E^?$	$\mathbf{0}$	E	$\top$
$E^?$	$E^?$	$E^?$	E	E
$\mathbf{0}$	$E^?$	$\mathbf{0}$	E	$\top$
E	E	E	E	E
$\top$	$\top$	$\top$	$\top$	$\top$

$\&$	$E^?$	$\mathbf{0}$	E	$\top$
$E^?$	$E^?$	$E^?$	$E^?$	$E^?$
$\mathbf{0}$	$E^?$	$\mathbf{0}$	$E^?$	$\mathbf{0}$
E	$E^?$	$E^?$	E	E
$\top$	$E^?$	$\mathbf{0}$	E	$\top$

$\otimes$	$E^?$	$\mathbf{0}$	E	$\top$
$E^?$	$E^?$	$E^?$	E	E
$\mathbf{0}$	$E^?$	$\mathbf{0}$	E	$\top$
E	E	E	E	E
$\top$	E	$\top$	E	$\top$

$;_E$	$E^?$	$\mathbf{0}$	E	$\top$
$E^?$	$E^?$	$\mathbf{0}$	$E^?$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
E	$E^?$	$\mathbf{0}$	E	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

Example 3.5 $E;\mathbf{0} = E$ and $E\&\mathbf{0} = E^?$ and $E^?;_E \mathbf{0} = \mathbf{0}$.

Definition 3.8 (Types) *The set of* types*, ranged over by σ, is given by the following syntax:*

$$\begin{array}{rcl} \sigma & ::= & \mathbf{bool} \mid (\delta_1 \xrightarrow{\varphi} \delta_2, U) \mid (\mathbf{R}, U) \\ \delta & ::= & \mathbf{bool} \mid (\delta_1 \xrightarrow{\varphi} \delta_2, U_0) \mid (\mathbf{R}, U_0) \end{array}$$

Here, U_0 ranges over the set of usages that satisfy $U_0 \cong U_0 \setminus E$.

bool is the type of boolean values. $(\delta_1 \xrightarrow{\varphi} \delta_2, U)$ is the type of functions that take a value of type δ_1 as an argument and return a value of type δ_2 and that, during the execution of the body, may raise an exception according to φ. U describes how a function is accessed (i.e., called). $(\mathbf{R}, U)$ is the type of resources that are accessed according to U.

For example, a function of the type $(\mathbf{R}, ((R\&\mathbf{0}); C)) \xrightarrow{E^?} \mathbf{bool}, 1; 1)$ takes a resource as an argument, closes the resource after a possible read, and may raise an exception during the function call. Moreover, the usage $1; 1$ states that the function is called twice.

We write the (outermost) usage of σ under effect φ by: $Use_\varphi(\mathbf{bool}) = (\varphi)^{use}$, $Use_\varphi((\sigma_1 \xrightarrow{\varphi'} \sigma_2, U)) = U$ and
$Use_\varphi((\mathbf{R}, U)) = U$.

The subusage relation defined in Section 3.1 is extended to the subtype relation $\sigma_1 \leq \sigma_2$ below. It means that a value of type σ_1 may be used as a value of type σ_2.

Definition 3.9 (Subtype relation) *$\sigma_1 \leq \sigma_2$ is the least relation closed under the following rules:*

$$\mathbf{bool} \leq \mathbf{bool} \qquad \frac{U \leq U' \qquad \varphi' \sqsubseteq \varphi}{(\sigma_1 \xrightarrow{\varphi} \sigma_2, U) \leq (\sigma_1 \xrightarrow{\varphi'} \sigma_2, U')} \qquad \frac{U \leq U'}{(\mathbf{R}, U) \leq (\mathbf{R}, U')}$$

3.3 Type Judgment

A type judgment is of the form $\Gamma \parallel \varphi \vdash M : \delta$, read "term M is given type δ under type environment Γ and effect φ" where Γ is a finite mapping from variables to types. The intended meaning of $\Gamma \parallel \varphi \vdash M : \delta$ is that (1) the term M is evaluated to a value of type δ, if the evaluation terminates, and (2) during the evaluation, each free variable x in M are used according to type $\Gamma(x)$ and an exception may be raised according to effect φ. The meaning of the judgment is tricky when $\diamond$ appears in Γ [10]: If a usage in $\Gamma(x)$ is guarded by $\diamond$, the access represented by the usage may be postponed until the value of M is used; otherwise the access cannot be postponed. For example, $x\!:\!(\mathbf{R}, R; C) \parallel \mathbf{0} \vdash \mathbf{read}(x); \mathbf{close}(x) : \mathbf{bool}$ and $x\!:\!(\mathbf{R}, R; \diamond C) \parallel \mathbf{0} \vdash \mathbf{read}(x); x : (\mathbf{R}, C)$ are valid judgments, while $x\!:\!(\mathbf{R}, R; C) \parallel \mathbf{0} \vdash \mathbf{read}(x); x : (\mathbf{R}, C)$ is invalid. (Actually, $\mathbf{read}(x)$ and $\mathbf{close}(x)$ must be annotated with $\cdot^{\{x\}}$ in our type system.)

We write $\emptyset$ for the empty type environment, and when $x \notin dom(\Gamma)$, we write $\Gamma, x : \sigma$ for the type environment Δ such that $dom(\Delta) = dom(\Gamma) \cup \{x\}$, $\Delta(x) = \sigma$ and $\Delta(y) = \Gamma(y)$ for $y \in dom(\Gamma)$.

The type judgment relation will be defined by using typing rules. We first give a few auxiliary definitions used in the typing rules.

Definition 3.10 *Let $\mathcal{C}$ be a usage context. Suppose that the set of free usage variables in σ or Γ is disjoint from the set of bound usage variables in $\mathcal{C}$. We define $\mathcal{C}[\sigma]$ and $\mathcal{C}[\Gamma]$ by:*

$$\mathcal{C}[\mathbf{bool}] = \mathbf{bool} \quad \mathcal{C}[(\mathbf{R}, U)] = (\mathbf{R}, \mathcal{C}[U])$$
$$\mathcal{C}[(\sigma_1 \xrightarrow{\varphi} \sigma_2, U)] = (\sigma_1 \xrightarrow{\varphi} \sigma_2, \mathcal{C}[U])$$

$$dom(\mathcal{C}[\Gamma]) = dom(\Gamma) \quad \mathcal{C}[\Gamma](x) = \mathcal{C}[\Gamma(x)]$$

Definition 3.11 *Let* **op** *be a binary usage constructor ';' , '&' or ';$_E$'. $\sigma_1 \mathbf{op} \sigma_2$ is defined as follows:*

$$\begin{array}{lcll} \mathbf{bool} & \mathbf{op} & \mathbf{bool} & = & \mathbf{bool} \\ (\sigma_1 \xrightarrow{\varphi} \sigma_2, U_1) & \mathbf{op} & (\sigma_1 \xrightarrow{\varphi} \sigma_2, U_2) & = & (\sigma_1 \xrightarrow{\varphi} \sigma_2, U_1 \mathbf{op} U_2) \\ (\mathbf{R}, U_1) & \mathbf{op} & (\mathbf{R}, U_2) & = & (\mathbf{R}, U_1 \mathbf{op} U_2) \end{array}$$

Let Γ_1 and Γ_2 be type environments with the same domain ($dom(\Gamma_1) = dom(\Gamma_2)$). Then, $\Gamma_1 \mathbf{op} \Gamma_2$ is defined as follows:

$$dom(\Gamma_1 \mathbf{op} \Gamma_2) = dom(\Gamma_1) (= dom(\Gamma_2))$$
$$(\Gamma_1 \mathbf{op} \Gamma_2)(x) = \Gamma_1(x) \mathbf{op} \Gamma_2(x)$$

For example, the type environment $\Gamma_1; \Gamma_2$ states that the value stored in each variable $x \in dom(\Gamma_1) (= dom(\Gamma_2))$ should be first used according to $\Gamma_1(x)$ and then should be used according to $\Gamma_2(x)$.

We also define the type environment $\blacklozenge_x \Gamma$ as follows:

$$\blacklozenge_x \Gamma = \begin{cases} \Gamma & \text{if } x \notin dom(\Gamma) \\ \Gamma', x : (\mathbf{R}, \blacklozenge U) & \text{if } \Gamma = \Gamma', x : (\mathbf{R}, U). \end{cases}$$

Note that, if $\Gamma(x) = \mathbf{bool}$ or $\Gamma(x) = (\sigma_1 \xrightarrow{\varphi} \sigma_2, U)$, then $\blacklozenge_x \Gamma$ is not defined.

Definition 3.12 (Type Judgment) *The type judgment relation $\Gamma \parallel \varphi \vdash M : \delta$ is the least relation closed under the rules in Figure 4.*

Note that when $\Gamma_1; \Gamma_2$ or $\Gamma_1;_E \Gamma_2$ appears in the conclusion of a rule, the rule can be applied only when the operation is well-defined; In particular, it must be the case that $dom(\Gamma_1) = dom(\Gamma_2)$.

Now we explain the key typing rules of T-Raise, T-Try, T-Fun, T-App, and T-Weak below. The others are essentially the same as those in the original type system [10] (except for the effect part in type judgments).

Rule T-Raise is the easiest: Since the term **raise** immediately raises an exception without accessing any resources, it is typed under the empty type environment with effect E.

Rule T-Try is explained as follows. Usages in Γ_1 and Γ_2 record how each resource is accessed before and after, respectively, an exception is raised. So, the total usage for **try** M_1 **with** M_2 is expressed by $\Gamma_1;_E \Gamma_2$, obtained by applying $;_E$ to usages in those type environments.

Rule T-Fun is defined according to the following intuitions. First, the premise says: *Each time* the function $\mathbf{fun}(f, x, M)$ is called, its body M causes effect φ, accessing the function's free variables according to Γ. In addition, M recursively calls f according to usage U_1. M also uses the argument x according to type σ_1 and yields a value of type δ_2. Therefore, the function is given a type $(\delta_1 \xrightarrow{\varphi} \delta_2, U)$, where $\delta_1 \leq \sigma_1 \backslash E$. Here, E is removed by $\backslash E$ from σ_1 since any possible exception that may be raised in M is already considered in the latent effect φ. The type environment for the function is obtained by *multiplying* $\diamond(\Gamma \backslash E)$ (which expresses how the function's free variables are accessed *each time* the function is called) according to U (which expresses how often the function is called from the outside) and $U_1 \backslash E$ (which expresses how often the function is called recursively). We safely approximate this multiplication by considering only the following three simple cases: the function is never called, it is called exactly once, or it is called an arbitrary number of times. In the first case, the free variables are never accessed. In the second case, the free variables are accessed exactly according to $\diamond(\Gamma \backslash E)$. In the last case, $\diamond(\Gamma \backslash E)$ is arbitrarily replicated by ! where !U is defined by $\mu\alpha.\mathbf{0} \& (U \otimes \alpha)$ and !Γ is its pointwise extension. Thus, the approximated multiplication $\Delta^{\mathbf{fun}}_{(U, U_1, \Gamma)}$ is defined as follows:

$$\Delta^{\mathbf{fun}}_{(U, U_1, \Gamma)} = \begin{cases} \emptyset & \text{if } 1 \notin [\![U]\!] \\ \Gamma & \text{if } (1 \in [\![U]\!] \subseteq \{\epsilon, 1, 1\downarrow\}) \wedge (1 \notin [\![U_1]\!]) \\ !\Gamma & \text{otherwise .} \end{cases}$$

Specially, the typing rule for non-recursive functions $\lambda x.M$ is given as follows:

$$\frac{\Gamma, x:\sigma_1 \parallel \varphi \vdash M : \delta_2 \quad \sigma_1' \leq (\tau_1 \setminus E)}{\Delta^{\mathbf{fun}}_{(U,\mathbf{0},\diamond(\Gamma\setminus E))} \parallel \mathbf{0} \vdash \lambda x.M : (\sigma_1' \xrightarrow{\varphi} \sigma_2, U).} \quad \text{(T-Abs)}$$

Rule T-App is explained as follows. When the term $M_1 M_2$ is evaluated, the term M_1 is first evaluated to a function and M_2 is then evaluated and finally the function is called. The type environment $\Gamma_1; \Gamma_2; (\varphi_3)^{use}$ reflects this order, where φ_3 comes from the latent effect of the type of the function.

Rule T-Weak deals with weakening and subsumption (on types and effects). Here, $\Gamma \leq_\varphi \Gamma'$ is defined by

$$\Gamma \leq_\varphi \Gamma' \quad \Leftrightarrow \quad \begin{cases} dom(\Gamma) \supseteq dom(\Gamma') & \\ \Gamma(x) \leq \Gamma'(x) & \text{for each } x \in dom(\Gamma') \\ Use_\varphi(\Gamma'(x)) \leq (\varphi)^{use} & \text{for each } x \in dom(\Gamma) \setminus dom(\Gamma') \end{cases}$$

It means that if we add $x : \sigma$ to the domain of Γ', σ must respect the effect φ of the term. For example, from $\Gamma' \parallel E \vdash M : \delta$, we can derive $\Gamma', x : (\mathbf{R}, E) \parallel E \vdash M : \delta$ (where $x \notin dom(\Gamma')$) but not $\Gamma', x : (\mathbf{R}, \mathbf{0}) \parallel E \vdash M : \delta$. In the latter, the usage of x contradicts with the effect E.

Example 3.6 The following type judgments can be derived for the terms M_1, M_2 and M of Example 2.3:

$$x : (\mathbf{R}, I; ((W; C)\&E)) \parallel E^? \vdash M_1 : \mathbf{bool}$$
$$x : (\mathbf{R}, I; ((W; C)\&E));_E C \parallel \mathbf{0} \vdash M_2 : \mathbf{bool}$$
$$\emptyset \parallel \mathbf{0} \vdash M : \mathbf{bool}.$$

Example 3.7 Let us consider the term

$$M_y^{fun} \triangleq \lambda x.(\mathbf{if}\ \mathbf{read}(x)^{\{x\}}\ \mathbf{then}\ \mathbf{write}(y)^{\{y\}}\ \mathbf{else}\ \mathbf{raise})$$
$$M_{xy}^{body} \triangleq \mathbf{if}\ \mathbf{read}(x)^{\{x\}}\ \mathbf{then}\ \mathbf{write}(y)^{\{y\}}\ \mathbf{else}\ \mathbf{raise}$$

This term M_{fy} is tyed as follows:

$$\Pi_1 = \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : (\mathbf{R}, \diamond R) \parallel \mathbf{0} \vdash x : (\mathbf{R}, R)}\ \text{(T-Var)}}{x : (\mathbf{R}, \diamond R) \parallel \mathbf{0} \vdash \mathbf{read}(x) : \mathbf{bool}}\ \text{(T-Acc)}}{x : (\mathbf{R}, \blacklozenge\diamond R) \parallel \mathbf{0} \vdash \mathbf{read}(x)^{\{x\}} \vdash \mathbf{bool}}\ \text{(T-Now)}}{x : (\mathbf{R}, R) \parallel \mathbf{0} \vdash \mathbf{read}(x)^{\{x\}} \vdash \mathbf{bool}}\ \text{(T-Weak)}}{x : (\mathbf{R}, R), y(\mathbf{R}, \mathbf{0}) \parallel \mathbf{0} \vdash \mathbf{read}(x)^{\{x\}} \vdash \mathbf{bool}}\ \text{(T-Weak)}$$

$$\Pi_2 = \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : (\mathbf{R}, \diamond W) \parallel \mathbf{0} \vdash x : (\mathbf{R}, W)}\ \text{(T-Var)}}{x : (\mathbf{R}, \diamond W) \parallel \mathbf{0} \vdash \mathbf{write}(y) : \mathbf{bool}}\ \text{(T-Acc)}}{y : (\mathbf{R}, \blacklozenge\diamond W) \parallel \mathbf{0} \vdash \mathbf{read}(y)^{\{y\}} \vdash \mathbf{bool}}\ \text{(T-Now)}}{y : (\mathbf{R}, W) \parallel \mathbf{0} \vdash \mathbf{write}(y)^{\{y\}} \vdash \mathbf{bool}}\ \text{(T-Weak)}}{x : (\mathbf{R}, \mathbf{0}), y : (\mathbf{R}, W) \parallel \mathbf{0} \vdash \mathbf{write}(y)^{\{y\}} \vdash \mathbf{bool}}\ \text{(T-Weak)}}{x : (\mathbf{R}, \mathbf{0}\&E), y : (\mathbf{R}, W\&E) \parallel \mathbf{0} \vdash \mathbf{write}(y)^{\{y\}} \vdash \mathbf{bool}}\ \text{(T-Weak)}$$

$$\Pi_3 = \dfrac{\dfrac{\dfrac{}{\emptyset \parallel E \vdash \mathbf{raise} : \mathbf{bool}}\ \text{(T-Const)}}{x : (\mathbf{R}, E), y : (\mathbf{R}, E) \parallel E^? \vdash \mathbf{raise} : \mathbf{bool}}\ \text{(T-Weak)}}{x : (\mathbf{R}, \mathbf{0}\&E), y : (\mathbf{R}, W\&E) \parallel E^? \vdash \mathbf{raise} : \mathbf{bool}}\ \text{(T-Weak)}$$

and

$$\dfrac{\dfrac{\dfrac{\Pi_1 \quad \Pi_2 \quad \Pi_3}{x:(\mathbf{R},R;(\mathbf{0}\&E)),y:(\mathbf{R},\mathbf{0};(W\&E)) \parallel E^? \vdash M_{xy}^{body}:\mathbf{bool}}\ (\text{T-If})}{\Delta^{\mathbf{fun}}_{(U,\mathbf{0},(y:(\mathbf{R},\diamond((\mathbf{0};(W\&E))\setminus E))))} \parallel \mathbf{0} \vdash M_y^{fun}:((\mathbf{R},(R;(\mathbf{0}\&E))\setminus E) \xrightarrow{E^?} \mathbf{bool},U)}\ (\text{T-Abs})}{\Delta^{\mathbf{fun}}_{(U,\mathbf{0},(y:(\mathbf{R},\diamond(W\&\mathbf{0}))))} \parallel \mathbf{0} \vdash M_y^{fun}:((\mathbf{R},R) \xrightarrow{E^?} \mathbf{bool},U).}\ (\text{T-Weak})$$

Here, we use subtype relations $(R;(\mathbf{0}\&E))\setminus E \cong R$ and $\diamond((\mathbf{0};(W\&E))\setminus E) \cong \diamond(W\&\mathbf{0})$.

So, for example, when the function M is called once (that is $U = 1$), we have

$$y:\diamond(W\&\mathbf{0}) \parallel \mathbf{0} \vdash M:((\mathbf{R},R) \xrightarrow{E^?} \mathbf{bool},1).$$

Example 3.8 By annotating every access to a resource x by $(\cdot)^{\{x\}}$—for example, $\mathbf{close}(x)$ becomes $\mathbf{close}(x)^{\{x\}}$— the term M of Example 2.4 is typed as follows (the type derivation is shown in Appendix A).

$$x:(\mathbf{R},(!R);C),y:(\mathbf{R},(!W);C) \parallel \mathbf{0} \vdash M:\mathbf{bool}$$

Here, we have $[\![(!R);C]\!] = (R^*C\downarrow)^{\#}$. Similarly, $[\![(!W);C]\!] = (W^*C\downarrow)^{\#}$. Therefore, we can conclude the resources x and y are closed when the evaluation of M terminates.

4 Type Soundness

Our type system is sound in the sense that if a closed well-typed term of type τ where $Use(\tau) = \mathbf{0}$ is evaluated, any resource is accessed according to the specification (declared by the resource creation primitive $\mathbf{new}^{\Phi}()$).

We say that M is *well-annotated* if all the annotations on escape information $\cdot^{\{x\}}$ are sound, i.e., if $(\{\}, M)$ is never reduced to a configuration $(H, \mathcal{E}[v^{\{x\}}])$ such that $x \in \mathbf{FV}(v)$. The soundness of our type system is stated formally as follows:

Theorem 4.1 (Type Soundness) *Suppose M is well-annotated. If $\emptyset \parallel \varphi \vdash M:\delta$ and $Use_{\mathbf{0}}(\delta) \leq \mathbf{0}$, then all the following properties hold:*

(1) $(\{\}, M) \not\rightsquigarrow^*$ **Error**.

(2) If $(\{\}, M) \rightsquigarrow^* (H, M') \not\rightsquigarrow$, *then* $\forall x \in dom(H).\downarrow \in H(x)$.

The condition $Use_{\mathbf{0}}(\tau) \leq \mathbf{0}$ states that even if the term M is evaluated to a resource, the resource may not be accessed after the evaluation. Property (1) means that M never performs an illegal resource access. Property (2) means that all the resources are used up when the evaluation terminates (normally or abruptly). Note that property (1) holds even if $Use_{\mathbf{0}}(\delta) \not\leq \mathbf{0}$.

We give an outline of the proof of Theorem 4.1 below. The full proof is shown in Appendix B.

We first define a type judgment relation $\varphi \vdash (H, M) : \delta$, which means that the state (H, M) is well-typed under the effect φ.

Definition 4.1

$$\dfrac{\begin{array}{c}x_1:(\mathbf{R},U_1),\ldots,x_n:(\mathbf{R},U_n) \parallel \varphi \ \vdash M:\delta \\ dom(H) = \{x_1,\ldots,x_n\} \\ [\![U_1]\!] \subseteq H(x_1),\ldots,[\![U_n]\!] \subseteq H(x_n)\end{array}}{\varphi \vdash (H,M):\delta}$$

$$\frac{c = \textbf{true} \text{ or } \textbf{false}}{\emptyset \parallel \mathbf{0} \vdash c : \textbf{bool}} \quad \text{(T-Const)}$$

$$x : \Diamond\delta \parallel \mathbf{0} \vdash x : \delta \quad \text{(T-Var)}$$

$$\frac{[\![U]\!] \subseteq \Phi}{\emptyset \parallel \mathbf{0} \vdash \textbf{new}^{\Phi}() : (\mathbf{R}, U)} \quad \text{(T-New)}$$

$$\frac{\Gamma \parallel \varphi \vdash M : (\mathbf{R}, a)}{\Gamma \parallel \varphi \vdash \textbf{acc}^{a}(M) : \textbf{bool}} \quad \text{(T-Acc)}$$

$$\frac{\Gamma_1 \parallel \varphi_1 \vdash M_1 : \textbf{bool} \quad \Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta \quad \Gamma_2 \parallel \varphi_2 \vdash M_3 : \delta}{\Gamma_1;\Gamma_2 \parallel \varphi_1;\varphi_2 \vdash \textbf{if } M_1 \textbf{ then } M_2 \textbf{ else } M_3 : \delta} \quad \text{(T-If)}$$

$$\frac{\Gamma_1 \parallel \varphi_1 \vdash M_1 : \sigma_1 \backslash E \quad \Gamma_2, x : \sigma_1 \parallel \varphi_2 \vdash M_2 : \delta_2}{\Gamma_1;\Gamma_2 \parallel \varphi_1;\varphi_2 \vdash \textbf{let } x = M_1 \textbf{ in } M_2 : \delta_2} \quad \text{(T-Let)}$$

$$\frac{\Gamma_1 \parallel \varphi_1 \vdash M_1 : (\delta_1 \xrightarrow{\varphi_3} \delta_2, 1) \quad \Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta_1}{\Gamma_1;\Gamma_2;(\varphi_3)^{use} \parallel \varphi_1;\varphi_2;\varphi_3 \vdash M_1 M_2 : \delta_2} \quad \text{(T-App)}$$

$$\frac{\Gamma, f : (\delta_1 \xrightarrow{\varphi} \delta_2, U_1), x : \sigma_1 \parallel \varphi \vdash M : \delta_2 \quad \delta_1 \leq \sigma_1 \backslash E}{\Delta^{\textbf{fun}}_{(U, U_1 \backslash E, \Diamond(\Gamma \backslash E))} \parallel \mathbf{0} \vdash \textbf{fun}(f, x, M) : (\delta_1 \xrightarrow{\varphi} \delta_2, U)} \quad \text{(T-Fun)}$$

$$\frac{\Gamma \parallel \varphi \vdash M : \delta}{\blacklozenge_x \Gamma \parallel \varphi \vdash M^{\{x\}} : \delta} \quad \text{(T-Now)}$$

$$\emptyset \parallel E \vdash \textbf{raise} : \delta \quad \text{(T-Raise)}$$

$$\frac{\Gamma_1 \parallel \varphi_1 \vdash M_1 : \delta \quad \Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta}{\Gamma_1 ;_E \Gamma_2 \parallel \varphi_1 ;_E \varphi_2 \vdash \textbf{try } M_1 \textbf{ with } M_2 : \delta} \quad \text{(T-Try)}$$

$$\frac{\varphi \sqsubseteq \varphi' \quad \Gamma \leq_{\varphi'} \Gamma' \quad \Gamma' \parallel \varphi' \vdash M : \delta' \quad \delta' \leq \delta}{\Gamma \parallel \varphi \vdash M : \delta} \quad \text{(T-Weak)}$$

Figure 4: Typing Rules

The first premise means that M uses the resources $x_1, \ldots, x_n$ according to $U_1, \ldots, U_n$. The other premises mean that the current heap indeed allows such resource usage.

We list main lemmas below. Lemma 4.4 states that typing is preserved by reductions. Lemma 4.1 states that an invalid resource access cannot happen immediately in a well-typed state. Lemma 4.3 states that evaluation may terminate only when the expression becomes a value or raises an uncaught exception. Lemma 4.2 states that every heap element contains $\downarrow$ in a well-typed, final state. (See Appendix B for proofs.)

Lemma 4.1 (Safety I) *If $\varphi \vdash (H, M) : \delta$, then $(H, M) \not\rightsquigarrow$* **Error**.

Proof Suppose $\varphi \vdash (H, M) : \delta$ and $(H, M) \rightsquigarrow$ **Error**. $(H, M) \rightsquigarrow$ **Error** must have been derived from (R-AccErr). Hence, there exist $\mathcal{E}$, x and a such that $M = \mathcal{E}[\mathbf{acc}^a(x)]$ and $H(x)^{-a} = \emptyset$. From $\varphi \vdash (H, M) : \delta$, it follows that there exist Γ' and U_x such that $\Gamma', x : (\mathbf{R}, U_x) \parallel \varphi \vdash \mathcal{E}[\mathbf{acc}^a(x)] : \delta$ and $[\![U_x]\!] \subseteq H(x)$. We can easily show that $\Gamma, x : (\mathbf{R}, U_x) \parallel \varphi \vdash \mathcal{E}[\mathbf{acc}^a(x)] : \delta$ implies $a \in [\![U_x]\!]$ (by induction on the structure of $\mathcal{E}$). Therefore, we have $a \in [\![U_x]\!] (\subseteq H(x))$, which implies $\epsilon \in H(x)^{-a} \neq \emptyset$. This is a contradiction. □

Lemma 4.2 (Safety II)

(1) If $\varphi \vdash (H, v) : \delta$ and $Use_{\mathbf{0}}(\delta) \leq \mathbf{0}$, then $\forall x \in dom(H). \downarrow \in H(x)$.

(2) If $\varphi \vdash (H, \mathcal{E}^{\overline{try}}[\mathbf{raise}]) : \delta$, then $\forall x \in dom(H). \downarrow \in H(x)$.

Lemma 4.3 (Progress) *Suppose M is well-annotated. If $\varphi \vdash (H, M) : \delta$, then either*

(1) $(H, M) \rightsquigarrow (H', M')$ for some H' and M' or

(2) M is either a value v or of the form $\mathcal{E}^{\overline{try}}[\mathbf{raise}]$

Lemma 4.4 (Type Preservation) *If $\varphi \vdash (H, M) : \sigma$ and $(H, M) \rightsquigarrow (H', M')$, then $\varphi \vdash (H', M') : \sigma$.*

Theorem 4.1 is an immediate corollary of the above lemmas.

Proof of Theorem 4.1**.**

- Property (1): Let (H_1, M_1) be $(\{\}, M)$ and assume $(H_1, M_1) \rightsquigarrow \cdots \rightsquigarrow (H_n, M_n)$ and $(H_n, M_n) \rightsquigarrow$ **Error**. By Theorem 4.4 and $\varphi \vdash (\{\}, M) : \tau$, we have $\varphi \vdash (H_n, M_n) : \tau$. Hence, the assumption $(H_n, M_n) \rightsquigarrow$ **Error** contradicts with Theorem 4.1.
- Property (2): Let (H_1, M_1) be $(\{\}, M)$ and assume that $(H_1, M_1) \rightsquigarrow \cdots \rightsquigarrow (H_n, M_n) \not\rightsquigarrow$. By Lemma 4.4, we have $\varphi \vdash (H_n, M_n) : \delta$. By Lemma 4.3, M_n is either a value v or of the form $\mathcal{E}^{\overline{try}}[\mathbf{raise}]$. So, $\forall x \in dom(H_n). \downarrow \in H_n(x)$ follows from Lemma 4.2. □

5 Type Inference

By the soundness of the type system, a sufficient condition for a closed term M to access resources in a valid manner is that there exists an effect φ and δ such that $\emptyset \parallel \varphi \vdash M : \delta$ and $Use_{\mathbf{0}}(\delta) \leq \mathbf{0}$. (Actually, it is sufficient to give an algorithm to check whether $\emptyset \parallel \varphi \vdash M :$ **bool**, since if M does not have type **bool**, we can check the term $(\lambda x.\mathbf{true})M$ instead.) We sketch an algorithm for checking the sufficient condition in this section.

The overall structure of the algorithm is the same as the constraint-based type inference algorithm for Igarashi and Kobayashi's type system [10]. Based on the typing rules, we can construct an algorithm which, given a closed term M, generates constraints on variables expressing unknown usages, effects, and types as a sufficient and necessary condition for $\emptyset \parallel \varphi \vdash M : \delta$. By reducing the constraints on type variables (using the standard unification algorithm), we can obtain constraints of the following form:

$$\left\{ \begin{array}{l} \xi_1 \leq \varphi_1, \ldots, \xi_m \leq \varphi_m, \\ \alpha_1 \leq U_1, \ldots, \alpha_n \leq U_n, \quad [\![U'_1]\!] \subseteq \Phi_1, \ldots, [\![U'_k]\!] \subseteq \Phi_k \end{array} \right\}$$

At this point, $U_1, \ldots, U_n$ may contain effect variables (in the form of $(\xi)^{use}$) and expressions of the form $\Delta^{\mathbf{fun}}_{(U_1,U_2,U_3)}$ (which is defined in the same way as $\Delta^{\mathbf{fun}}_{(U_1,U_2,\Gamma)}$), where U_1 and U_2 are the usages of

functions. To remove them, we first solve constraints on effects and function usages by using a standard method for solving constraints over a finite lattice [18].

Then, the greatest solution for a subusage constraint of the form $\alpha \leq U$ (where U no longer contains an effect or $\Delta^{\mathbf{fun}}_{(U_1,U_2,U_3)}$) can be represented by $\mu\alpha.U$. Thus, the above constraints can be further reduced to constraints of the form: $\{[\![U_1'']\!] \subseteq \Phi_1, \ldots, [\![U_k'']\!] \subseteq \Phi_k\}$.

Like in our previous type system [10], the relation $[\![U]\!] \subseteq \Phi$ is generally undecidable (think of the case where Φ is a context-free language). We, however, believe that typical resource usage specifications can be expressed in regular languages. As hinted in [10], in such cases, it is not difficult to develop an algorithm (which may be incomplete but sound at least) to verify the condition $[\![U]\!] \subseteq \Phi$. In fact, we have already implemented such an algorithm for the case where $\Phi = (R^* C\downarrow)^{\#}$: see Section 6.

Example 5.1 Consider the term

$$\mathbf{let}\ x = \mathbf{new}^{(R^* C\downarrow)^{\#}}()\ \mathbf{in}\ \mathbf{let}\ y = \mathbf{new}^{(W^* C\downarrow)^{\#}}()\ \mathbf{in}\ M^a$$

Here, M^a is a term obtained by annotating every access to a resource x by $(\cdot)^{\{x\}}$ —for example, $\mathbf{close}(x)$ becomes $\mathbf{close}(x)^{\{x\}}$— in the term M of Example 2.4. Then, as shown in Appendix A, we finally gain the following constraints after extracting and solving subusage and subeffect constrains:

$$[\![(!(\Diamond((R; (\mathbf{0}\&E); E)\backslash E)); E);_E C]\!] \subseteq (R^* C\downarrow)^{\#}$$
$$[\![(!(\Diamond((W; (\mathbf{0}\&E); E)\backslash E)); E);_E C]\!] \subseteq (W^* C\downarrow)^{\#}$$

Since these constraints are satisfied, we can conclude that the above term is well-typed. □

Example 5.2 Consider the term

$$M \triangleq \mathbf{let}\ x = \mathbf{new}^{(R^* C\downarrow)^{\#}}()\ \mathbf{in}\ M_2,$$

where $M_2 \triangleq \mathbf{try}\ (\mathbf{read}(x)^{\{x\}}; \mathbf{raise})\ \mathbf{with}\ \mathbf{close}(x)$. The following constraints are extracted.

$$\begin{array}{ll}
\Gamma_{\mathbf{read}(x)} = x : (\mathbf{R}, \alpha_1) & \Gamma_{\mathbf{read}(x)^{\{x\}};\mathbf{raise}} = x : (\mathbf{R}, \alpha_5) \\
\Gamma_{\mathbf{raise}} = x : (\mathbf{R}, \alpha_2) & \Gamma_{M_2} = x : (\mathbf{R}, \alpha_6) \\
\Gamma_{\mathbf{close}(x)} = x : (\mathbf{R}, \alpha_3) & \Gamma_{\mathbf{new}^{(R^* C\downarrow)^{\#}}()} = \emptyset \\
\Gamma_{\mathbf{read}(x)^{\{x\}}} = x : (\mathbf{R}, \alpha_4) & \Gamma_M = \emptyset
\end{array}$$

$$\begin{array}{c}
\varphi_{\mathbf{read}(x)} = \xi_1 \quad \varphi_{\mathbf{raise}} = \xi_2 \quad \varphi_{\mathbf{close}(x)} = \xi_3 \\
\varphi_{\mathbf{read}(x)^{\{x\}}} = \xi_4 \quad \varphi_{\mathbf{read}(x)^{\{x\}};\mathbf{raise}} = \xi_5 \\
\varphi_{M_2} = \xi_6 \quad \varphi_{\mathbf{new}^{(R^* C\downarrow)^{\#}}()} = \xi_7 \quad \varphi_M = \xi_8
\end{array}$$

$$\begin{array}{c}
\delta_{\mathbf{read}(x)} = \mathbf{bool} \quad \delta_{\mathbf{raise}} = \mathbf{bool} \quad \delta_{\mathbf{close}(x)} = \mathbf{bool} \\
\delta_{\mathbf{read}(x)^{\{x\}}} = \mathbf{bool} \quad \delta_{\mathbf{read}(x)^{\{x\}};\mathbf{raise}} = \mathbf{bool} \\
\delta_{M_2} = \mathbf{bool} \quad \delta_{\mathbf{new}^{(R^* C\downarrow)^{\#}}()} = (\mathbf{R}, \alpha_7) \quad \delta_M = \mathbf{bool}
\end{array}$$

$$\begin{array}{llll}
[\![\alpha_7]\!] \subseteq (R^* C\downarrow)^{\#} & \alpha_4 \leq \blacklozenge\alpha_1 & \xi_1 \sqsubseteq \mathbf{0} & \xi_5 \sqsubseteq \xi_4; \xi_2 \\
\alpha_1 \leq \Diamond R & \alpha_5 \leq \alpha_4; \alpha_2 & \xi_2 \sqsubseteq E & \xi_6 \sqsubseteq \xi_5;_E \xi_3 \\
\alpha_2 \leq E & \alpha_6 \leq \alpha_5;_E \alpha_3 & \xi_3 \sqsubseteq \mathbf{0} & \xi_7 \sqsubseteq \mathbf{0} \\
\alpha_3 \leq \Diamond C & \alpha_7 \leq \alpha_6 & \xi_4 \sqsubseteq \xi_3 & \xi_8 \sqsubseteq \xi_7; \xi_6
\end{array}$$

Here, Γ_N, ξ_N, and δ_N are respectively the type environment, effect, and type of a subterm N. By solving the constraints on effects and usages, we obtain $\alpha_7 = (\blacklozenge\Diamond R; E);_E \Diamond C$ and $\varphi_8 = \mathbf{0}$. Since $[\![\alpha_7]\!] \subseteq (R^* C\downarrow)^{\#}$ holds, we can conclude that M is well-typed. □

Example 5.3 Consider the term M defined by:

$$\begin{array}{lcl} M & \triangleq & \textbf{let } f = M_f \textbf{ in let } x = \textbf{new}^{(R^*C\downarrow)^\#}() \textbf{ in } M_{body} \\ M_f & \triangleq & \lambda x.(\textbf{if read}(x)^{\{x\}} \textbf{ then write}(y)^{\{y\}} \textbf{ else raise}) \\ M_{body} & \triangleq & \textbf{try } f(x); \textbf{close}(x)^{\{x\}} \textbf{ with close}(x)^{\{x\}}. \end{array}$$

This term first defines function f that reads from a given resource and, if an certain condition holds (**read**(x) return **true**), write to a globally-defined resource y , otherwise raise an exception, and then creates resource x and then apply the function f to the resource x.

We will show a sketch of the type inference for the term M. For simplicity, we assume that the raw type (the part of type obtained by removing usage expressions) of every term have been already obtained (by usual type inference). Moreover, we also assume that the type environments, effects and types of $\textbf{new}^{(R^*C\downarrow)^\#}()$ and M_f have been inferred as follows:

$$\begin{array}{ll} \Gamma_{M_f} = y : (\mathbf{R}, \Delta^{\textbf{fun}}_{(\alpha_f,1,\diamond(W\&\mathbf{0})}), b : \textbf{bool} & \Gamma_{\textbf{new}^{(R^*C\downarrow)^\#}()} = \emptyset \\ \varphi_{M_f} = \mathbf{0} & \varphi_{\textbf{new}^{(R^*C\downarrow)^\#}()} = \mathbf{0} \\ \delta_{M_f} = ((\mathbf{R}, R) \xrightarrow{E^?} \textbf{bool}, \alpha_{f1}) & \delta_{\textbf{new}^{(R^*C\downarrow)^\#}()} = (\mathbf{R}, \alpha_0) \end{array}$$

Here, Γ_N, ξ_N, and δ_N are respectively the type environment, effect, and type of a term N. Then, the following constraints are extracted for the terms M_{body} and $M_1 \triangleq \textbf{let } x = \textbf{new}^{(R^*C\downarrow)^\#}() \textbf{ in } M_{body}$ (we reduce several constraints for simplicity).

$$\begin{array}{ll} \Gamma_{M_{body}} = & f : ((\mathbf{R}, \alpha_1) \xrightarrow{\xi_f} \textbf{bool}, \alpha_{f2}), \\ & x : (\mathbf{R}, \alpha_2), y : (\mathbf{R}, \alpha_3), b : \textbf{bool} \\ \varphi_{M_{body}} = \xi_1 & \\ \delta_{M_{body}} = \textbf{bool} & \end{array} \qquad \begin{array}{ll} \Gamma_{M_1} = & f : ((\mathbf{R}, \alpha_1) \xrightarrow{\xi_f} \textbf{bool}, \alpha_{f3}), \\ & x : (\mathbf{R}, \alpha_4), y : (\mathbf{R}, \alpha_5), b : \textbf{bool} \\ \varphi_{M_1} = \xi_2 & \\ \delta_{M_1} = \textbf{bool} & \end{array}$$

$$\begin{array}{lllll} [\![\alpha_0]\!] \subseteq (R^*C\downarrow)^\# & \alpha_2 \le (\alpha_1; (\xi_f)^{use}; C);_E C & \alpha_4 \le \mathbf{0}; \alpha_2 & \alpha_{f2} \le 1 & \xi_1 \sqsubseteq (E^?; \mathbf{0});_E \mathbf{0} \\ \alpha_0 \le \alpha_4 & \alpha_3 \le \mathbf{0} & \alpha_5 \le \mathbf{0}; \alpha_3 & \alpha_{f3} \le \mathbf{0}; \alpha_{f2} & \xi_2 \sqsubseteq \mathbf{0}; \xi_1 \end{array}$$

Moreover, for the term M, the following constraints are generated.

$$\begin{array}{lll} \Gamma_M = y : (\mathbf{R}, \alpha_6), b : \textbf{bool} & \alpha_1 \le R & \xi_3 \le 0; \xi_2 \\ \varphi_M = \xi_3 & \alpha_6 \le \Delta^{\textbf{fun}}_{(\alpha_{f1},1,\diamond(W\&\mathbf{0}))}; \alpha_3 & \xi_f \le E^? \\ \delta_M = \textbf{bool} & \alpha_{f1} \le \alpha_{f3} & \end{array}$$

Note that, constraints $\alpha_1 \le R$ and $\alpha_{f1} \le \alpha_{f3}$ are generated from from the constraint $\delta_{M_f} \le \Gamma_{M_1}(f)$ (this is generated based on the typing-rules (T-Let) and (T-Weak)).

As mentioned above, we first solve function usage constrains α_{f1}, α_{f2} and α_{f3} and gain $\alpha_{f1} = \alpha_{f2} = \alpha_{f3} = 1$. So we have $\Delta^{\textbf{fun}}_{(\alpha_{f1},1,\diamond(W\&\mathbf{0}))} = \diamond(W\&\mathbf{0})$. Now, other usage and effect constraints are solved as follows:

$$\begin{array}{lcllcllcl} \alpha_0 & \le & (R; (\mathbf{0}\&E); C);_E C & \alpha_4 & \le & (R; (\mathbf{0}\&E); C);_E C & \xi_1 & \sqsubseteq & \mathbf{0} \\ \alpha_1 & \le & R\&\mathbf{0} & \alpha_5 & \le & \mathbf{0} & \xi_2 & \sqsubseteq & \mathbf{0} \\ \alpha_2 & \le & (R; (\mathbf{0}\&E); C);_E C & \alpha_6 & \le & \diamond(W\&\mathbf{0}) & \xi_3 & \sqsubseteq & \mathbf{0} \\ \alpha_3 & \le & \mathbf{0} & & & & \xi_f & \sqsubseteq & E^? \end{array}$$

Here, since $[\![R; (\mathbf{0}\&E); C);_E C]\!] \subseteq (R^*C\downarrow)^\#$ holds the constraint $[\![\alpha_0]\!] \subseteq (R^*C\downarrow)^\#$ satisfies. So, all constrains are satisfied and the term M has type **bool** under the type environment $\Gamma_M = y : (\mathbf{R}, \diamond(W\&\mathbf{0})), b : \textbf{bool}$ and effect $\mathbf{0}$.

6 Experiments

Based on our type system, we have implemented a prototype resource usage analyzer. The implementation will be made available at `http://www.kb.ecei.tohoku.ac.jp/~iwama/rue/`. The analyzer inputs a program written in $\lambda^{\mathcal{R}}_{E}$, without annotations $(\cdot^{\{x\}})$ on escape information. The analyzer first performs the standard type inference and annotate terms of non-function types with escape information. It then performs the usage analysis as described in the previous section. In the final phase, constraints of the form $[\![U]\!] \subseteq \Phi$ are checked. Currently, the analyzer accepts only the specification $\Phi = (R^* C \downarrow)^{\#}$, and uses a sound but incomplete algorithm for checking $[\![U]\!] \subseteq \Phi$. The algorithm works as sketched in Section 6.6 of our previous paper [10]. The basic observation behind the algorithm is as follows. Although the language of usage expressions is very expressive (for example, it can express any context-free languages as well as some context-sensitive languages), we can approximate usages by using a finite set of *abstract usages* as long as the specification Φ is regular; For example, we need not distinguish between usages $R; C$ and $R; R; C$ when the specification is $(R^* C \downarrow)^{\#}$. We have designed an abstract usage domain that is sufficient for checking the inclusion with respect to the specification $\Phi = (R^* C \downarrow)^{\#}$, so that the constraint $[\![U]\!] \subseteq \Phi$ can be replaced by a decidable, sufficient condition $[\![\alpha(U)]\!] \subseteq \Phi$ (where α is the abstraction function). The formalization of an algorithm that can deal with more general specifications Φ is left for future work.

Experiments We have tested several programs including the examples given in this paper (where **init**(x) is replaced by **read**(x) since the current system can handle only the specification $(R^* C \downarrow)^{\#}$). We confirmed that the analyzer gives correct answers. The tested programs include the following tricky one.

```
let create =
  fun(f,x,let y=new[read*;close]() in y) in
let repeat =
  fun(g,x, let z = create x in
    try
      if acc[read](z) then raise
      else (g x; acc[close](z))
    with acc[close](z)) in
repeat true;;
```

The above program repeatedly creates a new resource and closes it. Note that arbitrarily many resources may be created, and also that arbitrarily many exception handlers can be nested.

We have also inspected source programs of O'Caml compiler (3.08.4), manually translated some fragments of the programs accessing input files, and run our analyzer. Of 46 fragments of the code we have inspected, 42 of them can be categorized into the access patterns (expressed in our target language) summarized in Figure 5. We have confirmed that all of the four patterns can be analyzed by our prototype system. For example, the following is an example of the 4th pattern:

```
let exclude filename =
let ic = open_in filename in
try
  while true do
    let s = input_line ic in
    primitives := StringSet.remove s !primitives
  done
with End_of_file -> close_in ic
   | x -> close_in ic; raise x
```

The body of the above function is expressed in our language:

Normal pattern: 16 places ... (**let** $z = \mathbf{new}^{(R^*C)^\#}()$ **in** (**read**(z); ..; **close**(z))) ...
TryWith pattern: 18 places **let** $z = \mathbf{new}^{(R^*C)^\#}()$ **in** **try** **read**(z); .. ; **close**(z) **with** **close**(z)
TryClose pattern: 3 places **let** $z = \mathbf{new}^{(R^*C)^\#}()$ **in** (**try** **read**(z); .. **with** ..); **close**(z)
WhileTrue pattern: 5 places **let** $f = \lambda x.(..;$ **read**$(x); ..)$ **in** **let** $z = \mathbf{new}^{(R^*C)^\#}()$ **in** **try** $while(true)\{..;\ (f\ z);\ ..\}$ **with** **close**(z)

Figure 5: Typical file access patterns in O'Caml program

```
let input_line = lambda x.
        if acc[read](x) then true else raise in
let ic = new[read*;close]()
in  try  fun(g,x, input_line ic;g x) true
    with acc[close](ic);;
```

Our prototype analyzer accepts the above program, while it rejects the slightly modified program obtained by replacing `acc[close](ic)` with `false`.

Additionally, we have found that there are two fragments that seem to forget to close a file (in `asmcomp/asmlink.ml` and `debugger/source.ml`).

The 4 fragments that our analyzer cannot deal with use pointers (reference cells) or records to store file pointers. The following is the most interesting one:

```
let ic = open_in_bin Sys.executable_name in
Bytesections.read_toc ic;
{ read_string = Bytesections.read_section_string ic;
  read_struct = Bytesections.read_section_struct ic;
  close_reader = fun () -> close_in ic }
```

It opens a file, and then creates a record consisting of closures for reading and closing files. To properly handle this, we need to refine the type system to control the order between the uses of record elements.

7 Discussion

Alternative approach for dealing with exceptions An alternative, more straightforward approach for dealing with exceptions would be to encode exception primitives into $\lambda^{\mathcal{R}}$ (e.g., by using the continuation-passing style) [10] or the extension of π-calculus with resources [15], and then apply previous type systems [10, 15]. The resulting analysis is not, however, accurate enough to deal with the examples given in this paper.

To encode the exception handling primitive **try** M_1 **with** M_2 into a function, we just have to express the exception handler M_2 as a function and give it to a function denoting M_1 as a function parameter. For example, let consider the following program:

$$\mathbf{let}\ f = \lambda y.\mathbf{raise}\ \mathbf{in}\ (\mathbf{try}\ f()\ \mathbf{with}\ \mathbf{write}(x)); \mathbf{close}(x)$$

This program defines the function f, which may raise an exception, and calls the function under exception handler $\mathbf{write}(x)$ and then evaluates $\mathbf{close}(x)$. This program is encoded into the following function:

$$\begin{array}{l}\mathbf{let}\ f = \lambda y.\lambda h.\lambda c.h()\ \mathbf{in}\\ \qquad \mathbf{let}\ c = \lambda z.\mathbf{close}(x)\ \mathbf{in}\\ \qquad \mathbf{let}\ h = \lambda z.(\mathbf{write}(x); c())\ \mathbf{in}\ f()(h)(c)\end{array}$$

Note that the exception handler $\mathbf{write}(x)$ (and the ordinary continuation) is given to the function f as an argument. However, with this straightforward technique, the accuracy of inferred usages are insufficiency as mentiond above. For example, with this technique, we infer usage $\diamond C; \diamond W$ of resource x for the above term. By the inferred usage, we do not know which operation either W or C is done first to the resource, while our type system infers usage $(E;_E W); C$ for the resource, that says that a write operation is first performed and then close operation is performed.

Another approach would be to put information about both resource usage and exceptions into effects to make the type system simpler. For example, a function

$$\lambda x.(\mathbf{write}(x); \mathbf{close}(x); \mathbf{raise})$$

can be given a type $(\mathbf{R}, \rho) \xrightarrow{\rho^W;\rho^C;E} \mathbf{bool}$, where ρ is a abstract resource (called a region). The effect $\rho^W; \rho^C; E$ means that a resource belonging to the region ρ is written and closed, and then an exception is raised. As discussed elsewhere [10, 14], however, this approach does not work well when different resources are aliased to the same region.

Extensions for multiple exceptions and exception arguments Unlike the simple language studied in this paper, real languages like ML allow multiple exceptions and exception arguments. We can extend our type system to deal with multiple exceptions, by introducing distinct usage constructors E_i and $;_{E_i}$ for each kind of exception. As for exception arguments, there are two main issues: (1) how to deal with an exception having a resource as an argument (for example, consider the case where an exception carries a file that must be closed), and (2) how to deal with pattern matching on arguments, like "`try ... with E 1 -> ...`". We can deal with both issues by combining our type system with analyses of uncaught exceptions [17, 28]. For the first issue, we can impose a restriction that the usage U of a resource passed as an argument of an uncaught exception must be a subusage of $\mathbf{0}$. For the second issue, we can extend usage constructors E_i and $;_{E_i}$ by annotating them with information (like "rows" [17]) about exception arguments.

8 Related Work

A number of type systems have been proposed for statically checking whether a certain kind of resource is accessed in a valid manner [1, 3, 8, 10, 23, 25]. Only a few of them, however, deal with exception primitives. Type systems for JVM lock primitives [3, 12] support exceptions. In those type systems, the handler for each exception is statically known, so that exceptions can be treated in the same manner as *if*-statements.

It seems easier to extend effect-based type systems [4, 7, 19] for dealing with exceptions than to extend Igarashi and Kobayashi's type system. The effect-based approach, however, suffers from the problem mentioned in Section 7.

Another approach to the analysis of resource usage in the presence of exceptions would be to extend work on typestates [5, 6, 20, 26, 27]. In this approach, each type have several states (typestates) and a typestate of each resource may be changed by resource accesses or procedure calls. Each access to a resource is permitted only if the resource is in a valid typestate, so, inferring the typestate of each resource at any program points, we can verify valid resource usage. Indeed, the original work on typestates [20] does deal with exceptions. However, unlike ours, their method (1) requires explicit

annotations on procedures (with a specification: *pre-typestates*, *post-typestates*, *exception-typestates* meaning that typestates of argument values before resource-access/procedure-call after the resource-access/procedure-call has been performed normally and abruptly) and (2) cannot deal with aliasing, that makes their verification be unsound; (3) cannot deal with higher-order functions.

The succeeding works on typestates [5, 6, 26, 27] extends the original work to lift the restriction on aliasing and deal with data structures (or objects) and pointers (although exceptions are no longer explicitly discussed). However, these requires every procedure in their language to be annotated with at least alias information about how argument values and a return value are aliased to keep track of alias relation of variables. In order to extend their techniques for dealing with exception handling, we must find enough alias information that is invariant at each entry point of exception hander for any exception-flow. It seem to be possible but not trivial. The paper [6] handles similar problems in this paper (e.g. whether a resources is accessed according to $read^*; close$ in a program) and deals with alias precisely on the assumption that all paths in the program are feasible. However, their target language is relatively theoretical one i.e. that does not include procedure calls and exceptions.

On the other hand, the general advantage for the typestate techniques is that (1) the analysis is accurate due to the annotations about alias information, (it also could be that an extra alias analysis or annotations make our analysis more accurate). (2) properties described in reglar language are verifiable, while we have not identified how large class of properties is verified with our type system (i.e. we does not identified the class of Φ such that $[\![U]\!] \subseteq \Phi$ is decidable). Moreover , the paper [5, 26, 27] handle pointers and date-structures (objects) that our target language do not deal with. The papers [6] give verification algorithms and it's computational complexities. These issues for our work are future works for us.

Kobayashi [14] has proposed another combination of linear types and effect systems. His type system is, however, so complicated that no reasonable type inference algorithm has been developed.

In parallel to the present work, we have recently studied type-based resource usage analysis for concurrent programs [15]. It would be interesting future work to integrate the type system in this paper with that type system.

Model checking technologies [2, 9] have recently been applied to verification of temporal properties of programs. Advantages of our type-based approach are that our analysis is modular, and that our analysis can be deal with programs creating infinitely many resources (recall the tricky example shown in Section 6).

9 Conclusion

We have extended Igarashi and Kobayashi's type-based resource usage analysis to deal with exceptions, proved the soundness of the extended analysis, and implemented a prototype analyzer.

Future work includes extension of the type system to deal with a larger set of language constructs (e.g., multiple exceptions, pointers, concurrency primitives) and development of an algorithm for checking $[\![U]\!] \subseteq \Phi$ for a certain class of languages Φ.

References

[1] Alexander Aiken, Manuel Fähndrich, and Raph Levien. Improving region-based analysis of higher-order languages. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 174–185, 1995.

[2] Thomas Ball and Sriram K. Rajamani. The SLAM project: Debugging system software via static analysis. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, pages 1–3, 2002.

[3] Gaetano Bigliardi and Cosimo Laneve. A type system for JVM threads. In *Proceedings of 3rd ACM SIGPLAN Workshop on Types in Compilation (TIC2000)*, Montreal, Canada, 2000.

[4] Robert DeLine and Manuel Fähndrich. Adoption and focus: Practical linear types for imperative programming. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2002.

[5] Robert DeLine and Manuel Fähndrich. Typestates for objects. In Martin Odersky, editor, *ECOOP*, volume 3086 of *Lecture Notes in Computer Science*, pages 465–490. Springer, 2004.

[6] John Field, Deepak Goyal, G. Ramalingam, and Eran Yahav. Typestate verification: Abstraction techniques and complexity results. In Radhia Cousot, editor, *SAS*, volume 2694 of *Lecture Notes in Computer Science*, pages 439–462. Springer, 2003.

[7] Jeffrey S. Foster, Tachio Terauchi, and Alex Aiken. Flow-sensitive type qualifiers. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2002.

[8] Stephen N. Freund and John C. Mitchell. The type system for object initialization in the Java bytecode language. *ACM Transactions on Programming Languages and Systems*, 21(6):1196–1250, 1999.

[9] Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Gregoire Sutre. Lazy abstraction. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, pages 58–70, 2002.

[10] Atsushi Igarashi and Naoki Kobayashi. Resource usage analysis. *ACM Transactions on Programming Languages and Systems*, 27(2):264–313, March 2005.

[11] Futoshi Iwama, Atsushi Igarashi, and Naoki Kobayashi. Resource usage analysis for a functional language with exceptions, 2005. Full version. Available from `http://www.kb.ecei.tohoku.ac.jp/~iwama/rue/res-use-exce-full.pdf`.

[12] Futoshi Iwama and Naoki Kobayashi. A new type system for JVM lock primitives. In *Proceedings of ASIA-PEPM'02*, pages 156–168. ACM Press, 2002.

[13] Naoki Kobayashi. Quasi-linear types. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, pages 29–42, 1999.

[14] Naoki Kobayashi. Time regions and effects for resource usage analysis. In *Proceedings of ACM SIGPLAN International Workshop on Types in Languages Design and Implementation (TLDI'03)*, pages 50–61, 2003.

[15] Naoki Kobayashi, Kohei Suenaga, and Lucian Wischik. Resource usage analysis for π-calculus. In *Proceedings of VMCAI'06*, 2006.

[16] Ian Mackie. Lilac : A functional programming language based on linear logic. *Journal of Functional Programming*, 4(4):1–39, October 1994.

[17] F. Pessaux and X. Leroy. Type-based analysis of uncaught exceptions. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, pages 276–290, 1999.

[18] Jakob Rehof and Torben Mogensen. Tractable constraints in finite semilattices. *Science of Computer Programming*, 35(2):191–221, 1999.

[19] Christian Skalka and Scott Smith. History effects and verification. In *Proceedings of APLAS 2004*, volume 3302 of *Lecture Notes in Computer Science*, pages 107–128. Springer-Verlag, 2004.

[20] Robert. E. Strom and Shaula Yemini. Typestate: A programming language concept for enhancing software reliability. *IEEE, Transactions on Software Engineering*, 12(1):157–171, January 1986.

[21] Jean-Pierre Talpin and Pierre Jouvelot. Polymorphic type, region and effect inference. *Journal of Functional Programming*, 2(3):245–271, 1992.

[22] Jean-Pierre Talpin and Pierre Jouvelot. The type and effect discipline. In *Proceedings of IEEE Symposium on Logic in Computer Science*, pages 162–173, 1992.

[23] Mads Tofte and Jean-Pierre Talpin. Region-based memory management. *Information and Computation*, 132(2):109–176, 1997.

[24] David N. Turner, Philip Wadler, and Christian Mossin. Once upon a type. In *Proceedings of Functional Programming Languages and Computer Architecture*, pages 1–11, 1995.

[25] David Walker, Karl Crary, and J. Gregory Morrisett. Typed memory management via static capabilities. *ACM Transactions on Programming Languages and Systems*, 22(4):701–771, 2000.

[26] Zhichen Xu, Barton P. Miller, and Thomas Reps. Safety checking of machine code. In *PLDI '00: Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation*, pages 70–82, New York, NY, USA, 2000. ACM Press.

[27] Zhichen Xu, Thomas W. Reps, and Barton P. Miller. Typestate checking of machine code. In *ESOP '01: Proceedings of the 10th European Symposium on Programming Languages and Systems*, pages 335–351, London, UK, 2001. Springer-Verlag.

[28] Kwangkeun Yi. Compile-time detection of uncaught exceptions in Standard ML programs. In *Proceedings of SAS'94*, volume 864 of *Lecture Notes in Computer Science*, pages 238–254, 1994.

A Type Derivation for the Term from Example 3.8

We show how the term M from Example 3.8 (or Example 2.4). below is typed.

$$M \triangleq \mathbf{try}\ \mathbf{fun}(f, z, M_1)\mathbf{true}\ \mathbf{with}\ \mathbf{close}(x)^{\{x\}}; \mathbf{close}(y)^{\{y\}}$$
$$M_1 \triangleq (\mathbf{if}\ \mathbf{read}(x)^{\{x\}}\ \mathbf{then}\ \mathbf{true}\ \mathbf{else}\ \mathbf{raise}); \mathbf{write}(y)^{\{y\}}; f\ \mathbf{true}$$

As mentioned before, each resource access is annotated with $(\cdot)^{\{x\}}$ to get more accurate usage information. First, the type derivation for $\mathbf{read}(x)^{\{x\}}$ is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : (\mathbf{R}, \Diamond R) \parallel \mathbf{0} \vdash x : (\mathbf{R}, R)}\ (\text{T-Var})}{x : (\mathbf{R}, \Diamond R) \parallel \mathbf{0} \vdash \mathbf{read}(x) : \mathbf{bool}}\ (\text{T-Acc})}{x : (\mathbf{R}, \blacklozenge\Diamond R) \parallel \mathbf{0} \vdash \mathbf{read}(x)^{\{x\}} \vdash \mathbf{bool}}\ (\text{T-Now})}{x : (\mathbf{R}, R) \parallel \mathbf{0} \vdash \mathbf{read}(x)^{\{x\}} \vdash \mathbf{bool}}\ (\text{T-Weak})$$

Terms $\mathbf{write}(y)^{\{y\}}$, $\mathbf{close}(x)^{\{x\}}$, and $\mathbf{close}(y)^{\{y\}}$ are typed similarly. Then, the type judgments $x : (\mathbf{R}, \mathbf{0}) \parallel E^? \vdash \mathbf{true} : \mathbf{bool}$ and $x : (\mathbf{R}, E) \parallel E^? \vdash \mathbf{raise} : \mathbf{bool}$ are derived as follows:

$$\dfrac{\dfrac{}{\emptyset \parallel \mathbf{0} \vdash \mathbf{true} : \mathbf{bool}}\ (\text{T-Raise})}{x : (\mathbf{R}, \mathbf{0}) \parallel E^? \vdash \mathbf{true} : \mathbf{bool}}\ (\text{T-Weak}) \qquad \dfrac{\dfrac{}{\emptyset \parallel E \vdash \mathbf{raise} : \mathbf{bool}}\ (\text{T-Const})}{x : (\mathbf{R}, E) \parallel E^? \vdash \mathbf{raise} : \mathbf{bool}}\ (\text{T-Weak})$$

Then, $M_2 \triangleq \textbf{if } \textbf{read}(x)^{\{x\}} \textbf{ then true else raise}$ is typed as follows:

$$\Pi_1 = \dfrac{\dfrac{x : (\mathbf{R}, R) \parallel \mathbf{0} \vdash \textbf{read}(x)^{\{x\}} : \textbf{bool} \quad \begin{array}{c} x : (\mathbf{R}, \mathbf{0}) \parallel E^? \vdash \textbf{true} : \textbf{bool} \\ x : (\mathbf{R}, E) \parallel E^? \vdash \textbf{raise} : \textbf{bool} \end{array}}{x : (\mathbf{R}, R; (\mathbf{0}\&E)) \parallel E^? \vdash M_2 : \textbf{bool}}\ (\text{T-If})}{x : (\mathbf{R}, R; (\mathbf{0}\&E)), y : (\mathbf{R}, \mathbf{0}\&E) \parallel E^? \vdash M_2 : \textbf{bool}}\ (\text{T-Weak})$$

(As the domains of type environments in premises must be the same, we need to apply T-Weak appropriately; We will often omit applications of T-Weak in what follows.) The function $\textbf{fun}(f, z, M_1)$ is typed as follows:

$$\Pi_2 = \dfrac{\dfrac{}{f : (\textbf{bool} \xrightarrow{E} \textbf{bool}, \Diamond 1) \parallel \mathbf{0} \vdash f : (\textbf{bool} \xrightarrow{E} \textbf{bool}, 1)}\ (\text{T-Var}) \quad \vdots}{f : (\textbf{bool} \xrightarrow{E} \textbf{bool}, \Diamond 1; E) \parallel E \vdash f\ \textbf{true} : \textbf{bool}}\ (\text{T-App})$$

$$\Pi_3 = \dfrac{\dfrac{\dfrac{\Pi_1 \quad x : (\mathbf{R}, \mathbf{0}), y : (\mathbf{R}, W) \parallel \mathbf{0} \vdash \textbf{write}(y)^{\{y\}} : \textbf{bool}}{\Gamma_1 \parallel E^? \vdash M_2; \textbf{write}(y)^{\{y\}} : \textbf{bool}}\ (\text{T-Let}) \quad \Pi_2}{\Gamma_2 \parallel E \vdash M_1 : \textbf{bool}}\ (\text{T-Let})}{\Gamma_3 \parallel \mathbf{0} \vdash \textbf{fun}(f, z, M_1) : (\textbf{bool} \xrightarrow{E} \textbf{bool}, 1)}\ (\text{T-Fun})$$

where

$$\begin{array}{l} \Gamma_1 = x : (\mathbf{R}, R; (\mathbf{0}\&E)), y : (\mathbf{R}, (\mathbf{0}\&E); W) \\ U_x = R; (\mathbf{0}\&E); E \qquad U_y = (\mathbf{0}\&E); W; E \\ \Gamma_2 = x : (\mathbf{R}, U_x), y : (\mathbf{R}, U_y), f : (\textbf{bool} \xrightarrow{E} \textbf{bool}, \Diamond 1; E) \\ \Gamma_3 = x : (\mathbf{R}, !\Diamond(U_x \setminus E)), y : (\mathbf{R}, !\Diamond(U_y \setminus E)). \end{array}$$

Finally, we complete the derivation of the type judgment of the term M as follows:

$$\Pi_4 = \dfrac{\Pi_3 \quad x : (\mathbf{R}, \mathbf{0}), y : (\mathbf{R}, \mathbf{0}) \parallel \mathbf{0} \vdash \textbf{true} : \textbf{bool}}{\Gamma_4 \parallel E \vdash \textbf{fun}(f, z, M_1)\ \textbf{true} : \textbf{bool}}\ (\text{T-App})$$

where $\Gamma_4 = x : (\mathbf{R}, !\Diamond(U_x \setminus E); E), y : (\mathbf{R}, !\Diamond(U_y \setminus E); E)$ and

$$\dfrac{\dfrac{\Pi_4 \quad \begin{array}{c} x : (\mathbf{R}, C), y : (\mathbf{R}, \mathbf{0}) \parallel \mathbf{0} \vdash \textbf{close}(x)^{\{x\}} : \textbf{bool} \\ x : (\mathbf{R}, \mathbf{0}), y : (\mathbf{R}, C) \parallel \mathbf{0} \vdash \textbf{close}(y)^{\{y\}} : \textbf{bool} \end{array}}{x : (\mathbf{R}, C), y : (\mathbf{R}, C) \parallel \mathbf{0} \vdash \textbf{close}(x)^{\{x\}}; \textbf{close}(y)^{\{y\}} : \textbf{bool}}\ (\text{T-Let})}{x : (\mathbf{R}, (!\Diamond(U_x \setminus E); E);_E C), y : (\mathbf{R}, (!\Diamond(U_y \setminus E); E);_E C) \parallel \mathbf{0} \vdash M : \textbf{bool}}\ (\text{T-Try})}{x : (\mathbf{R}, (!R); C), y : (\mathbf{R}, (!W); C) \parallel \mathbf{0} \vdash M : \textbf{bool}}\ (\text{T-Weak})$$

B Proofs of the Main Lemmas

In this appendix, we will prove Lemmas 4.2 – 4.4. First, in Section B.1, we show several properties of usages. In Section B.2, we show properties of type judgments and type environments, which are used in later proofs. In Section B.3, we prove Lemmas 4.2 and 4.3. Section B.4 presents a substitution lemma, which is necessary for proving Lemma 4.4. Finally, in Section B.5, we prove Lemma 4.4 (type preservation).

B.1 Basic properties of usages

Lemma B.1 *The relation $\leq$ satisfies the following properties:*

1. *if $U_1 \preceq U_2$, then $U_1 \leq U_2$,*

2. $U \leq \mu\alpha.\alpha$,

3. $U_1 \otimes U_2 \leq U_1 \,;\, U_2$,

4. $(U_1 \,;\, U_2) \otimes (U_3 \,;\, U_4) \leq (U_1 \otimes U_3) \,;\, (U_2 \otimes U_4)$,

5. $(U_1 \otimes U_3) \,\&\, (U_2 \otimes U_3) \cong (U_1 \,\&\, U_2) \otimes U_3$,

6. $\Diamond\Diamond U \cong \Diamond U \leq U$,

7. $!(\Diamond U) \cong \Diamond(!U)$,

8. $U \leq \blacklozenge U$,

9. $\blacklozenge U_1 \otimes \blacklozenge U_2 \leq \blacklozenge(U_1 \otimes U_2)$,

10. $E; U \cong E$,

11. $(U \backslash E) \backslash E \cong U \backslash E$,

12. $!(U \backslash E) \backslash E \cong !(U \backslash E)$,

13. $(U_1 ;_E U_2) ;_E U_3 \leq U_1 ;_E (U_2 ;_E U_3)$,

14. $\blacklozenge U_1 ;_E U_2 \leq \blacklozenge(U_1 ;_E U_2)$,

15. *if* $U \Longrightarrow U'$ *then* $U \leq U'$,

16. *if* $U \leq C[U]$, *then* $U \leq \mu\alpha.C[\alpha]$,

17. $((\varphi)^{use} \backslash E) \otimes (\varphi)^{use} \leq (\varphi)^{use}$,

18. $(\varphi)^{use} \cong \blacklozenge(\varphi)^{use}$,

19. *If* $U_1 \backslash E \otimes (\varphi_1)^{use} \leq U_1$ *and* $U_2 \backslash E \otimes (\varphi_2)^{use} \leq U_2$, *then the followings hold:*
    (1) $(U_1; U_2) \backslash E \otimes (\varphi_1; \varphi_2)^{use} \leq U_1; U_2$ *and*
    (2) $(U_1 ;_E U_2) \backslash E \otimes (\varphi_1 ;_E \varphi_2)^{use} \leq U_1 ;_E U_2$,

20. $[\![U]\!] = [\![\Diamond U]\!]$ *and*

21. *if* $U_1 \leq U_2$, *then* $[\![U_2]\!] \subseteq [\![U_1]\!]$.

In order to prove the above properties, we first introduce an "up-to" technique for proving the subusage relation.

Definition B.1 $\mathcal{S}$ *is a weak usage simulation up to* $\preceq$, *if the following conditions hold for any* $U_1 \mathcal{S} U_2$.

(C1) $(\mathcal{C}[U_1], \mathcal{C}[U_2]) \in \mathcal{S}$ *for any usage context* $\mathcal{C}$ *with a hole..*

(C2) If $U_2 \xrightarrow{l} U_2'$ *and* $l \in \mathcal{A} \cup \{1\}$, *then* $U_1 \stackrel{l}{\Longrightarrow} U_1'$ *and* $U_1' \preceq \mathcal{S} \preceq U_2'$ *for some* U_1'.

(C3) If $U_2 \xrightarrow{\tau} U_2'$, *then* $U_1 \Longrightarrow U_1'$ *and* $U_1' \preceq \mathcal{S} \preceq U_2'$ *for some* U_1'.

(C4) If $U_2 \xrightarrow{\epsilon} \mathbf{0}$, *then* $U_1 \Longrightarrow \mathbf{0}$.

(C5) If $U_2 \xrightarrow{E} U_2'$, *then* $U_1 \stackrel{E}{\Longrightarrow} U_1'$ *for some* U_1'.

Lemma B.2 *If* $\mathcal{S}$ *is a weak usage simulation up to* $\preceq$, *then* $\mathcal{S} \subseteq \leq$.

Proof It suffieces to show that $\preceq \mathcal{S} \preceq$ satisfies the five conditions (C1),...,(C5) in Definition B.1 for any $U_1 \preceq \mathcal{S} \preceq U_2$. So, we must show $\mathcal{C}[U_1] \preceq \mathcal{S} \preceq \mathcal{C}[U_2]$ for any usage context $\mathcal{C}$ (this is trivial) and must find U_1' which completes each of the following diagrams, given the top row and the right transition:

$$\begin{array}{ccccc} U_1 & \preceq \mathcal{S} \preceq & U_2 & \quad & U_1 & \preceq \mathcal{S} \preceq & U_2 \\ \Downarrow l & & \downarrow l & & \Downarrow & & \downarrow \tau \\ U_1' & \preceq \mathcal{S} \preceq & U_2' & & U_1' & \preceq \mathcal{S} \preceq & U_2' \end{array} \qquad \begin{array}{ccc} U_1 & \preceq \mathcal{S} \preceq & U_2 \\ \Downarrow & & \downarrow \epsilon \\ \mathbf{0} & & \mathbf{0} \end{array} \qquad \begin{array}{ccc} U_1 & \preceq \mathcal{S} \preceq & U_2 \\ \Downarrow E & & \downarrow E \\ U_1' & & U_2' \end{array}$$

We treat only the first diagram; the other diagrams are treated in the same way. By $U_1 \preceq \mathcal{S} \preceq U_2$, we have U_{11}, U_{22} such that $U_1 \preceq U_{11}\ \mathcal{S}\ U_{22} \preceq U_2$. Here note that, by usage transition rules, $U_0 \preceq U$ and $U \xrightarrow{l} U'$ imply $U_0 \xrightarrow{l} U'$. Therefore, by the fact that $\mathcal{S}$ is a weak usage simulation up to $\preceq$ and the right-most transition $U_2 \xrightarrow{l} U_2'$, there exists U_{22}', U_{11}' and U_1' that satisfy the following diagrams.

$$\begin{array}{ccccc} U_1 & \preceq & U_{11} & \quad & U_{11} \\ \Downarrow l & & \Downarrow l & & \Downarrow l \\ U_1' & = & U_{11}' & & U_{11}' \end{array} \qquad \begin{array}{ccc} & \mathcal{S} & \\ & & \\ \preceq & \mathcal{S} & \preceq \end{array} \qquad \begin{array}{ccccc} U_{22} & \quad & U_{22} & \preceq & U_2 \\ \downarrow l & & \downarrow l & & \downarrow l \\ U_{22}' & & U_{22}' & = & U_2' \end{array}$$

Since $\preceq$ is transitive, we have $U_1' \preceq \mathcal{S} \preceq U_2'$ and $U_1 \stackrel{l}{\Longrightarrow} U_1'$ as required. □

Lammas B.1(20,21) follows from the definition of $[\![\ ]\!]$. For the other sub-usage properties, we give only proofs of Lammas B.1(3,15). The others $(1, 2, 4, \ldots, 13, 15, \ldots, 19)$ are shown in a similar way. Firstly, we give a proof of Lemma B.1(3) and then give a proof of Lemma B.1(15).

Let $\mathcal{S}_3$ be the following binary relation on usages.

$$\mathcal{S}_3 = \left\{ \begin{array}{l} (\mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}], \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]) \\ \quad \mid V_1, \ldots V_n \in \mathcal{U}, \mathcal{C} \text{ is a usage context with } n\ (n \geq 0) \text{ holes.} \end{array} \right\}$$

The required property $U_1 \otimes U_2 \leq U_1; U_2$ follows if we show $\mathcal{S}_3 \subseteq \leq$. By Lemma B.2, it suffices to show that $\mathcal{S}_3$ is a weak usage simulation up to $\preceq$. In order to show this, we first prove several propositions.

Lemma B.3

(1) If $(U_1, \mathbf{0}) \in \mathcal{S}_3$ then $U_1 = \mathbf{0}$.

(2) If $(U_1, \diamond U_b) \in \mathcal{S}_3$ then $U_1 = \diamond U_b'$ for some U_b' such that $(U_b', U_b) \in \mathcal{S}_3$

Proof By the definition of $\mathcal{S}_3$.

Lemma B.4 *If $(U_1, U_2) \in \mathcal{S}_3$ and $U_2 \preceq U_2'$, then $U_1 \preceq U_1'$ and $(U_1', U_2') \in \mathcal{S}_3$ for a usage U_1'.*

Proof If $(U_1, U_2) \in \mathcal{S}_3$ then there exist $\mathcal{C}$ and $V_{11}, V_{12} \ldots, V_{n1}, V_{n2}$ such that $U_1 = \mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ and $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]$. By induction on derivation of $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] \preceq U_2'$ with case analysis on the last rule used, we show that there is a usage U_1' that satisfies $U_1 \preceq U_1'$ and $(U_1', U_2') \in \mathcal{S}_3$.

- Case $\diamond\mathbf{0} \preceq \mathbf{0}$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \diamond\mathbf{0} \preceq \mathbf{0} = U_2'$. So the following case holds:

  (a) $\mathcal{C} = \diamond\mathcal{C}_1$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \mathbf{0}$

  In the case (a), by Lemma B.3(1), we have $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] = \mathbf{0}$. So, $U_1 = \diamond\mathbf{0} \preceq \mathbf{0}$ holds. Thus, $U_1' = \mathbf{0}$ finishes the case.

- Case $\mathbf{0} \preceq \Diamond\mathbf{0}$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \mathbf{0} \preceq \Diamond\mathbf{0} = U_2'$. By Lemma B.3(1), $\mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] = \mathbf{0}$. So, we have $U_1 = \mathbf{0} \preceq \Diamond\mathbf{0}$ and this satisfies the required conditions.
- Case $\blacklozenge\mathbf{0} \preceq \mathbf{0}$ and $\mathbf{0} \preceq \blacklozenge\mathbf{0}$. Similar to the cases $\Diamond\mathbf{0} \preceq \mathbf{0}$ and $\mathbf{0} \preceq \Diamond\mathbf{0}$.
- Case $\mathbf{0}; U \preceq U$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \mathbf{0}; U \preceq U = U_2'$. So either of the following case holds:
  (a) $\mathcal{C} = \mathcal{C}_1; \mathcal{C}_2$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = \mathbf{0}$ and $\mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = U$
  (b) $\mathcal{C} = [\,]$ and $V_{11}; V_{12} = \mathbf{0}; U$

  In the case (a), by Lemma B.3(1), we have $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] = \mathbf{0}$. So, $U_1 = \mathbf{0}; \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] \preceq \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$. Hence $U_1' = \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ satisfies the required conditions.

  In the case (b), it follows from $V_{11} = \mathbf{0}, V_{12} = U$ that $U_1 = V_{11} \otimes V_{12} = \mathbf{0} \otimes U \preceq U$. So, $U_1' = U$ satisfies the required conditions.
- Case $U \preceq \mathbf{0}; U$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = U \preceq \mathbf{0}; U = U_2'$. For the usage context $\mathbf{0}; \mathcal{C}$, $(\mathbf{0}; \mathcal{C})[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = U_2'$ holds. So, since $U_1 = \mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] \preceq \mathbf{0}; \mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] = (\mathbf{0}; \mathcal{C})[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ holds, $U_1' = (\mathbf{0}; \mathcal{C})[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ satisfies the required conditions.
- Cases $\mathbf{0} \otimes U \equiv U$, $U; \mathbf{0} \equiv U$ and $\mathbf{0};_E U \equiv U$. Similar to the cases $\mathbf{0}; U \preceq U$ and $U \preceq \mathbf{0}; U$.
- Case $U_{11} \otimes U_{22} \preceq U_{22} \otimes U_{11}$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = U_{11} \otimes U_{22} \preceq U_{22} \otimes U_{11} = U_2'$. So the following case holds:
  (a) $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = U_{11}$ and $\mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = U_{22}$

  In the case, we can easily show that the required conditions are satisfied in a similar way to the previous cases.
- Case $U_{11} \& U_{22} \preceq U_{22} \& U_{11}$. Similar to the case $U_{11} \otimes U_{22} \preceq U_{22} \otimes U_{11}$.
- Case $\Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{22}$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{22} = U_2'$. So either of the following case holds:
  (a) $\mathcal{C} = \Diamond\mathcal{C}_1; \Diamond\mathcal{C}_2$ and $\Diamond\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = \Diamond U_{11}$ and $\mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = U_{22}$
  (b) $\mathcal{C} = [\,]$ and $V_{11}; V_{12} = \Diamond U_{11}; U_{22}$

  In the case (a), by Lemma B.3(2), we have U_{11}' such that $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] = U_{11}'$ and $(U_{11}, U_{11}') \in \mathcal{S}_3$. So, $U_1 = \Diamond U_{11}'; \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] \preceq \Diamond U_{11}' \otimes \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ holds. Thus $U_1' = \Diamond U_{11}' \otimes \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ finishes the case.

  In the case (b), from $V_{11} = \Diamond U_{11}$ and $V_{12} = U_{22}$, $U_1 = V_{11} \otimes V_{12} = \Diamond U_{11} \otimes U_{22}$ follows. So, $U_1' = \Diamond U_{11} \otimes U_{22} = U_2'$ satisfies the required conditions.
- Case $\Diamond U_{11} \otimes U_{22} \preceq \Diamond U_{11}; U_{22}$. Similar to the case $\Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{22}$.
- Case $\Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22})$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22}) = U_2'$. So the following case holds:
  (a) $\mathcal{C} = \Diamond\mathcal{C}_1; \Diamond\mathcal{C}_2$ and $\Diamond\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = \Diamond U_{11}$ and $\Diamond\mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = \Diamond U_{22}$

In the case (a), by Lemma B.3(2), we have $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] = U'_{11}$ and $\mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] = U'_{22}$ and $(U_{11}, U'_{11})\mathcal{S}_3$ and $(U_{22}, U'_{22})\mathcal{S}_3$. So, $U_1 = \Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22})$ holds. Thus $U'_1 = \Diamond(U_{11} \otimes U_{22})$ finishes the case.

- Cases $\Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22}$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22} = U'_2$. So the following case holds:

  (a) $\mathcal{C} = \Diamond\mathcal{C}_1$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = (U_{11} \otimes U_{22})$.

  In the case, by $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = (U_{11} \otimes U_{22})$, the following condition holds:

  (a1) $\mathcal{C}_1 = \mathcal{C}_{11} \otimes \mathcal{C}_{12}$ and $\mathcal{C}_{11}[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = U_{11}$ and $\mathcal{C}_{12}[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = U_{22}$

  In the case ,

$$\begin{array}{rcl} U'_2 & = & \Diamond\mathcal{C}_{11}[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] \otimes \Diamond\mathcal{C}_{12}[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] \\ U_1 & = & \Diamond(\mathcal{C}_{11}[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] \otimes \mathcal{C}_{12}[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]) \\ & \preceq & \Diamond\mathcal{C}_{11}[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] \otimes \Diamond\mathcal{C}_{12}[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] \end{array}$$

  So, for the usage context $\Diamond\mathcal{C}_{11} \otimes \Diamond\mathcal{C}_{12}$, we have $U'_2 = (\Diamond\mathcal{C}_{11} \otimes \Diamond\mathcal{C}_{12})[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]$ and $\Diamond\mathcal{C}_{11}[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] \otimes \Diamond\mathcal{C}_{12}[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] = (\Diamond\mathcal{C}_{11} \otimes \Diamond\mathcal{C}_{12})[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$. Thus $U'_1 = \Diamond\mathcal{C}_{11}[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] \otimes \Diamond\mathcal{C}_{12}[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ finishes the case.

- Cases $\Diamond U_{11};_E U_{22} \equiv \Diamond(U_{11};_E U_{22})$. Similar to the cases $\Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22})$ and $\Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22}$.

- Case $U_{11} \& U_{22} \preceq U_{11}$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = U_{11} \& U_{22} \preceq U_{11} = U'_2$. So only the following case holds:

  (a) $\mathcal{C} = \mathcal{C}_1 \& \mathcal{C}_2$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = U_{11}$ and $\mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = U_{22}$

  In the case (a), $U_1 = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] \& \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] \preceq \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}]$ holds. So, $U'_1 = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}]$ finishes the case.

- Case $\mu\alpha.U \preceq [\mu\alpha.U/\alpha]U$. It must be the case that: $\mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \mu\alpha.U \preceq [\mu\alpha.U/\alpha]U = U'_2$. So only the following case holds:

  (a) $\mathcal{C} = \mu\alpha.\mathcal{C}_1$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = U$.

  In the case (a), $U_1 = \mu\alpha.\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ holds. Since

$$([\mu\alpha.U_{11}/\alpha]U_{11}, [\mu\alpha.U_{21}/\alpha]U_{21}) \in \mathcal{S}_3$$

  holds for $U_{11} = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ and $U_{21} = \mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]$, $U'_1 = [\mu\alpha.U_{11}/\alpha]U_{11}$ satisfies the required conditions.

- Case $\dfrac{U \preceq U'}{\Diamond U \preceq \Diamond U'}$. It must be the case that: $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = \Diamond U$ and $U \preceq U'$ and $U'_2 = \Diamond U'$. So only the following case holds:

  (a) $\mathcal{C} = \Diamond\mathcal{C}_1$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{1n}; V_{2n}] = U$

  In the case (a), by the induction hypothesis, there exists U'' such that

$$\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] \preceq U'' \quad \text{and} \quad (U'', U') \in \mathcal{S}_3.$$

  So, we have $U_1 = \Diamond\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] \preceq \Diamond U''$. From $(U'', U') \in \mathcal{S}_3$, $(\Diamond U'', \Diamond U') \in \mathcal{S}_3$ follows. Thus, $U'_1 = \Diamond U''$ satisfies the required conditions.

- Case $\dfrac{U \preceq U'}{\blacklozenge U \preceq \blacklozenge U'}$. Similar to the case $\dfrac{U \preceq U'}{\Diamond U \preceq \Diamond U'}$.

- Case $\dfrac{U_{11} \preceq U'_{11}}{U_{11}; U_{22} \preceq U'_{11}; U_{22}}$. It must be the case that: $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = U_{11}; U_{22}$ and $U_{11} \preceq U'_{11}$ and $U'_2 = U'_{11}; U_{22}$.

  So either of the following cases holds:

  (a) $\mathcal{C} = \mathcal{C}_1; \mathcal{C}_2$ and $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{l1}; V_{l2}] = U_{11}$ and $\mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}] = U_{22}$

  (b) $\mathcal{C} = [\,]$ and $V_{11}; V_{12} = U_{11}; U_{22}$

  In the case (a), by the induction hypothesis, there exists U''_{11} such that

  $$\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}] \preceq U''_{11} \quad \text{and} \quad (U''_{11}, U'_{11}) \in \mathcal{S}_3.$$

  So, we have $U_1 = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{l1} \otimes V_{l2}]; \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}] \preceq U''_{11}; \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$. From $(U''_{11}, U'_{11}) \in \mathcal{S}_3$ and $U'_2 = U'_{11}; \mathcal{C}_2[V_{(l+1)1}; V_{(l+1)2}, \ldots, V_{n1}; V_{n2}]$, $(U''_{11}; \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}], U'_2) \in \mathcal{S}_3$ follows. Thus, $U'_1 = U''_{11}; \mathcal{C}_2[V_{(l+1)1} \otimes V_{(l+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ satisfies the required conditions.

  In the case (b), since $V_{11} = U_{11}$ and $V_{12} = U_{22}$, $U_1 = V_{11} \otimes V_{12} = U_{11} \otimes U_{22}$ holds. So, by the rule $\dfrac{U_{11} \preceq U'_{11}}{U_{11} \otimes U_{22} \preceq U'_{11} \otimes U_{22}}$, we have $U_1 = U_{11} \otimes U_{22} \preceq U'_{11} \otimes U_{22}$. This finishes the case.

- Cases $\dfrac{U_{11} \preceq U'_{11}}{U_{11} \otimes U_{22} \preceq U'_{11} \otimes U_{22}}$ and $\dfrac{U_{11} \preceq U'_{11}}{U_{11};_E U_{22} \preceq U'_{11};_E U_{22}}$. Similar to the case $\dfrac{U_{11} \preceq U'_{11}}{U_{11}; U_{22} \preceq U'_{11}; U_{22}}$.

- Case $\dfrac{U_2 \preceq U''_2 \quad U''_2 \preceq U'_2}{U_2 \preceq U'_2}$. By $(U_1, U_2) \in \mathcal{S}_3$ and the induction hypothesis, there exists U''_1 such that $U_1 \preceq U''_1$ and $(U''_1, U''_2) \in \mathcal{S}_3$. So, using the induction hypothesis again, we have U'_1 such that $U''_1 \preceq U'_1$ and $(U'_1, U'_2) \in \mathcal{S}$. Hence, we have $U_1 \preceq U'_1$ and $(U'_1, U'_2) \in \mathcal{S}_3$ as required. □

Now, we show that $\mathcal{S}_3$ is a weak usage simulation up to $\preceq$

Proposition B.1 *$\mathcal{S}_3$ is a weak usage simulation up to $\preceq$.*

Proof Note that if $(U_1, U_2) \in \mathcal{S}_3$, then there exist $V_{11}, V_{12}, \ldots, V_{n1}, V_{n2}$, and $\mathcal{C}$ such that $U_1 = \mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ and $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]$. We show that each condition in Definition B.1 holds as follows:

(C1) Trivial.

(C2) By induction on derivation of $U_2 \xrightarrow{l} U'_2$, with case analysis on the last rule used.

- Case (UR-Lab). $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = l$ and $U_1 = \mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ hold. So we have $\mathcal{C} = l$. In the case, it follows that $U_1 = l \stackrel{l}{\Longrightarrow} \mathbf{0}$. Thus, $U'_1 = \mathbf{0}$ finishes this case.
- Case (UR-Box). In this case, the following condition holds:
  (i) $U_2 = \Diamond\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]$ and $U_1 = \Diamond\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$.

  In the case, U'_2 must be of the form $\Diamond U'_{21}$ and $U_2 \xrightarrow{l} U'_2$ must have been derived from $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] \xrightarrow{l} U'_{21}$. So, by the induction hypothesis, there exists U'_{11} such that $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] \stackrel{l}{\Longrightarrow} U'_{11}$ and $U'_{11} \preceq \mathcal{S}_3 \preceq U'_{21}$. So, $U'_1 = \Diamond U'_{11}$ satisfies the required condition.

- Case (UR-Ubox). Similar to the case for (UR-Box).
- Case (UR-Par). In this case, the following case holds:

  (i) $U_2 = \mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{m1}; V_{m2}] \otimes \mathcal{C}_2[V_{(m+1)1}; V_{(m+1)2}, \ldots, V_{n1}; V_{n2}]$ and
  $U_1 = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}] \otimes \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$

  In the case, the followings must be derived:

  $$\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{m1}; V_{m2}] \xrightarrow{l} U'_{21} \qquad U'_2 = U'_{21} \otimes \mathcal{C}_2[V_{(m+1)1}; V_{(m+1)2}, \ldots, V_{n1}; V_{n2}]$$

  By the induction hypothesis, there exists U'_{11} such that $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}] \stackrel{l}{\Longrightarrow} U'_{11}$ and $U'_{11} \preceq \mathcal{S}_3 \preceq U'_{21}$. So, $U'_1 = U'_{11} \otimes \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ satisfies the required condition.
- Case (UR-Seq). Similar to the case for (UR-Par).
- Case (UR-SeqE). Similar to the case for (UR-Par).
- Case (UR-Hdlr). Since $\tau \notin \mathcal{A} \cup \{1\}$, this case cannot happen.
- Case (UR-Pcon). In this case, it must be the case that:

  $$U_2 \preceq U_{21} \qquad U_{21} \xrightarrow{l} U'_{21} \qquad U'_{21} \preceq U'_2$$

  By $(U_1, U_2) \in \mathcal{S}_3$, Lemma B.4 and the induction hypotheses, we have U_{11} and U'_{11} that satisfy the following diagrams:

  $$\begin{array}{ccccccc} U_2 & \preceq & U_{21} & \xrightarrow{l} & U'_{21} & \preceq & U'_2 \\ & & & & \curlyvee & & \\ \mathcal{S}_3 & & \mathcal{S}_3 & & \mathcal{S}_3 & & \\ & & & & \curlyvee & & \\ U_1 & \preceq & U_{11} & \stackrel{l}{\Longrightarrow} & U'_{11} & & \end{array}$$

  Since, $U'_{11} \preceq \mathcal{S}_3 \preceq U'_{21} \preceq U'_2$ implies $U'_{11} \preceq \mathcal{S}_3 \preceq U'_2$, $U'_1 = U'_{11}$ finishes the case.

(C4) By Lemma B.4.

(C5) By induction on derivation of $U_2 \xrightarrow{E} U'_2$, with case analysis on the last rule used.

- Case (UR-Lab). $U_2 = \mathcal{C}[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] = E$ and $U_1 = \mathcal{C}[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$ hold. So we have $\mathcal{C} = E$. In the case, it follows that $U_1 = E \stackrel{E}{\Longrightarrow} \mathbf{0}$. Thus, the required conditions are satisfied.
- Case (UR-Box). In this case, the following condition holds:

  (i) $U_2 = \Diamond\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}]$ and $U_1 = \Diamond\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}]$.

  In the case, U'_2 must be of the form $\Diamond U'_{21}$ and $U_2 \xrightarrow{E} U'_2$ must have been derived from $\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{n1}; V_{n2}] \xrightarrow{E} U'_{21}$. So, by the induction hypothesis, there exists U'_{11} such that $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] \stackrel{E}{\Longrightarrow} U'_{11}$. So, $U_1 = \Diamond\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{n1} \otimes V_{n2}] \stackrel{E}{\Longrightarrow} \Diamond U'_{11}$ satisfies the required condition.
- Case (UR-Ubox). Similar to the case for (UR-Box).
- Case (UR-Par). In this case, the following case holds:

  (i) $U_2 = \mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{m1}; V_{m2}] \otimes \mathcal{C}_2[V_{(m+1)1}; V_{(m+1)2}, \ldots, V_{n1}; V_{n2}]$ and
  $U_1 = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}] \otimes \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$

In the case, the followings must be derived:

$$\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{m1}; V_{m2}] \xrightarrow{E} U'_{21} \qquad U'_2 = U'_{21} \otimes \mathcal{C}_2[V_{(m+1)1}; V_{(m+1)2}, \ldots, V_{n1}; V_{n2}]$$

By the induction hypothesis, there exists U'_{11} such that $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}] \stackrel{E}{\Longrightarrow} U'_{11}$. So, $U_1 \stackrel{E}{\Longrightarrow} U'_{11} \otimes \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ satisfies the required condition.

- Case (UR-Seq). Similar to the case for (UR-Par).
- Case (UR-SeqE). Similar to the case for (UR-Par).
- Case (UR-Hdlr). This case cannot happen.
- Case (UR-Pcon). In this case, it must be the case that:

$$U_2 \preceq U_{21} \qquad U_{21} \xrightarrow{E} U'_{21} \qquad U'_{21} \preceq U'_2$$

By $(U_1, U_2) \in \mathcal{S}_3$, Lemma B.4 and the induction hypotheses, we have U_{11} and U'_{11} that satisfy the following diagrams:

$$\begin{array}{ccccccc} U_2 & \preceq & U_{21} & \xrightarrow{l} & U'_{21} & \preceq & U'_2 \\ & & & & \Upsilon| & & \\ \mathcal{S}_3 & & \mathcal{S}_3 & & \mathcal{S}_3 & & \\ & & & & \Upsilon| & & \\ U_1 & \preceq & U_{11} & \stackrel{l}{\Longrightarrow} & U'_{11} & & \end{array}$$

This satisfies the required conditions.

(C3) By case analysis on the last rule used to derive $U_2 \xrightarrow{\tau} U'_2$.

- Case (UR-Hdlr). In this case, the following holds:
  (i) $U_2 = \mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{m1}; V_{m2}];_E \mathcal{C}_2[V_{(m+1)1}; V_{(m+1)2}, \ldots, V_{n1}; V_{n2}]$ and $U_1 = \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}];_E \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$.

  In the case, the following derivation must be derived for a usage U'_{21}:

$$\mathcal{C}_1[V_{11}; V_{12}, \ldots, V_{m1}; V_{m2}] \xrightarrow{E} U'_{21} \qquad U'_2 = \mathcal{C}_2[V_{(m+1)1}; V_{(m+1)2}, \ldots, V_{n1}; V_{n2}]$$

  By (C5), there exists U'_{11} such that $\mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}] \stackrel{E}{\Longrightarrow} U'_{11}$. So, we have U_e , U'_e such that $\mathcal{C}_1[V_1, \ldots, V_n] \Longrightarrow U_e \xrightarrow{E} U'_e \Longrightarrow U'_{11}$. Thus, the followings hold:

$$\begin{array}{ll} & \mathcal{C}_1[V_{11} \otimes V_{12}, \ldots, V_{m1} \otimes V_{m2}];_E \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}] \\ \Longrightarrow & U_e;_E \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}] \\ \xrightarrow{\tau} & \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}] \end{array}$$

  Therefore, we have $U_1 \Longrightarrow \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ and $U'_1 = \mathcal{C}_2[V_{(m+1)1} \otimes V_{(m+1)2}, \ldots, V_{n1} \otimes V_{n2}]$ satisfies the required condition.
- Cases for other rules: Similar to (C1). □

Proof of Lemma B.1 (3). By Lemmas B.1 and B.2. □

From here, we give a proof of Lemma B.1(15). Let $\mathcal{S}_{15}$ be the following binary relation on usages.

$$\mathcal{S}_{15} = \left\{ \; (\mathcal{C}[V_1, \ldots, V_n], \mathcal{C}[V'_1, \ldots, V'_n]) \;\middle|\; \begin{array}{l} V_1, \ldots V_n \in \mathcal{U}, \text{ and } V_i \Longrightarrow V'_i, \\ \mathcal{C} \text{ is a usage context with } n \ (n \geq 0) \text{ holes.} \end{array} \right\}$$

The required property $U \leq U'$ follows if we show $\mathcal{S}_{15} \subseteq \leq$. By Lemma B.2, it suffices to show that $\mathcal{S}_{15}$ is a weak usage simulation upto $\preceq$. In order to show this, we first prove several propositions.

Lemma B.5

(1) If $(U_1, \mathbf{0}) \in \mathcal{S}_{15}$ *then* $U_1 \Longrightarrow \mathbf{0}$.

(2) If $(U_1, \Diamond U_{b2}) \in \mathcal{S}_{15}$ *then* $U_1 \Longrightarrow \Diamond U_{b1}$ *and* $(\Diamond U_{b1}, \Diamond U_{b2}) \in \mathcal{S}_{15}$ *for some usage* U_{b1}.

Proof (1): By $(U_1, \mathbf{0}) \in \mathcal{S}_{15}$, there exist $\mathcal{C}$ and $V'_1, \ldots, V'_n$ such that $\mathcal{C}[V'_1, \ldots, V'_n] = \mathbf{0}$. So, either $\mathcal{C} = \mathbf{0}$ or $(\mathcal{C} = [\,] \ \wedge \ V'_1 = \mathbf{0})$ holds. In the first case, $U_1 = \mathbf{0}$ holds. In the second case, $U_1 = V_1 \Longrightarrow \mathbf{0}$ holds. (2): By $(U_1, \Diamond U_{b2}) \in \mathcal{S}_{15}$, there exist $\mathcal{C}$ and $V'_1, \ldots, V'_n$ such that $\mathcal{C}[V'_1, \ldots, V'_n] = \Diamond U_{b2}$.

- Case $\mathcal{C} = [\,]$. It must be the case $V'_1 = \Diamond U_{b2}$. So, we have $U_1 \Longrightarrow \Diamond U_{b2}$. Thus, $U_{b1} = U_{b2}$ finishes this case.
- Case $\mathcal{C} = \Diamond \mathcal{C}_1$. It must be the case $\mathcal{C}[V'_1, \ldots, V'_n] = \Diamond \mathcal{C}_1[V'_1, \ldots, V'_n]$. So, $U_1 = \Diamond \mathcal{C}_1[V_1, \ldots, V_n]$ holds. Thus, $U_{b1} = \mathcal{C}_1[V_1, \ldots, V_n]$ finishes the case. □

Lemma B.6 *If* $(U_1, U_2) \in \mathcal{S}_{15}$ *and* $U_2 \preceq U'_2$, *then* $U_1 \Longrightarrow U'_1$ *and* $(U'_1, U'_2) \in \mathcal{S}_{15}$ *for some usage* U'_1.

Proof If $(U_1, U_2) \in \mathcal{S}_{15}$ then there exist $\mathcal{C}$ and $V_1, \ldots, V_n$ such that $U_1 = \mathcal{C}[V_1, \ldots, V_n]$ and $U_2 = \mathcal{C}[V'_1, \ldots, V'_n]$ and $V_i \Longrightarrow V'_i$ for $i = 1 \ldots n$. By induction on derivation of $U_2 = \mathcal{C}[V'_1, \ldots, V'_n] \preceq U'_2$ with case analysis on the last rule used, we show that there is a usage U'_1 that satisfies $U_1 \Longrightarrow U'_1$ and $(U'_1, U'_2) \in \mathcal{S}_{15}$.

- Case $\Diamond\mathbf{0} \preceq \mathbf{0}$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = \Diamond\mathbf{0} \preceq \mathbf{0} = U'_2$. So either of the followings case holds:

  (a) $\mathcal{C} = [\,]$ and $V'_1 = \Diamond\mathbf{0}$

  (b) $\mathcal{C} = \Diamond\mathcal{C}_1$ and $\mathcal{C}_1[V'_1, \ldots, V'_n] = \mathbf{0}$

  In the case (a), since $U_1 \Longrightarrow \Diamond\mathbf{0} \preceq \mathbf{0}$, $U'_1 = \mathbf{0}$ satisfies the required conditions. In the case (b), by Lemma B.5(1), we have $\mathcal{C}_1[V_1, \ldots, V_n] \Longrightarrow \mathbf{0}$. So, $U_1 \Longrightarrow \Diamond\mathbf{0} \preceq \mathbf{0}$ holds. Thus, $U'_1 = \mathbf{0}$ finishes the case.

- Case $\mathbf{0} \preceq \Diamond\mathbf{0}$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = \mathbf{0} \preceq \Diamond\mathbf{0} = U'_2$. By Lemma B.5(1), $\mathcal{C}[V_1, \ldots, V_n] \Longrightarrow \mathbf{0}$. So, we have $U_1 \Longrightarrow \mathbf{0} \preceq \Diamond\mathbf{0}$ and this satisfies the required conditions.

- Case $\blacklozenge\mathbf{0} \preceq \mathbf{0}$ and $\mathbf{0} \preceq \blacklozenge\mathbf{0}$. Similar to the cases $\Diamond\mathbf{0} \preceq \mathbf{0}$ and $\mathbf{0} \preceq \Diamond\mathbf{0}$.

- Case $\mathbf{0}; U \preceq U$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = \mathbf{0}; U \preceq U = U'_2$. So either of the followings case holds:

  (a) $\mathcal{C} = [\,]$ and $V'_1 = \mathbf{0}; U$

  (b) $\mathcal{C} = \mathcal{C}_1; \mathcal{C}_2$ and $\mathcal{C}_1[V'_1, \ldots, V'_l] = \mathbf{0}$ and $\mathcal{C}_2[V'_{l+1}, \ldots, V'_n] = U$

  In the case (a), since $U_1 = V_1 \Longrightarrow V'_1 = \mathbf{0}; U \preceq U$ holds, the required conditions are satisfied. In the case (b), by Lemma B.5(1), we have $\mathcal{C}_1[V_1, \ldots, V_l] \Longrightarrow \mathbf{0}$. So, $U_1 \Longrightarrow \mathbf{0}; \mathcal{C}_2[V_{l+1}, \ldots, V_n] \preceq \mathcal{C}_2[V_{l+1}, \ldots, V_n]$. Hence $U'_1 = \mathcal{C}_2[V_{l+1}, \ldots, V_n]$ satisfies the required conditions.

- Case $U \preceq \mathbf{0}; U$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = U \preceq \mathbf{0}; U = U'_2$. For the usage context $\mathbf{0}; \mathcal{C}$, $(\mathbf{0}; \mathcal{C})[V'_1, \ldots, V'_n] = U'_2$ holds. So, since $U_1 = \mathcal{C}[V_1, \ldots, V_n] \preceq \mathbf{0}; \mathcal{C}[V_1, \ldots, V_n] = (\mathbf{0}; \mathcal{C})[V_1, \ldots, V_n]$ holds, $U'_1 = (\mathbf{0}; \mathcal{C})[V_1, \ldots, V_n]$ satisfies the required conditions.

- Cases $\mathbf{0} \otimes U \equiv U$, $U; \mathbf{0} \equiv U$ and $\mathbf{0};_E U \equiv U$. Similar to the cases $\mathbf{0}; U \preceq U$ and $U \preceq \mathbf{0}; U$.

- Case $U_{11} \otimes U_{22} \preceq U_{22} \otimes U_{11}$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = U_{11} \otimes U_{22} \preceq U_{22} \otimes U_{11} = U'_2$. So either of the followings case holds:

  (a) $\mathcal{C} = [\,]$ and $V'_1 = U_{11} \otimes U_{22}$

  (b) $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$ and $\mathcal{C}_1[V'_1, \ldots, V'_l] = U_{11}$ and $\mathcal{C}_2[V'_{l+1}, \ldots, V'_n] = U_{22}$

  In each case, we can easily show that the required conditions are satisfied in a similar way to the previous cases.

- Case $U_{11} \& U_{22} \preceq U_{22} \& U_{11}$. Similar to the case $U_{11} \otimes U_{22} \preceq U_{22} \otimes U_{11}$.

- Case $\Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{22}$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = \Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{22} = U'_2$. So either of the following cases holds:

  (a) $\mathcal{C} = [\,]$ and $V'_1 = \Diamond U_{11}; U_{22}$

  (b) $\mathcal{C} = \mathcal{C}_1; \mathcal{C}_2$ and $\mathcal{C}_1[V'_1, \ldots, V'_l] = \Diamond U_{11}$ and $\mathcal{C}_2[V'_{l+1}, \ldots, V'_n] = U_{22}$

  In the case (a), since $U_1 = V_1 \Longrightarrow V'_1 = \Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{12}$, $U'_1 = \Diamond U_{11} \otimes U_{22}$ satisfies the required conditions. In the case (b), by Lemma B.5(2), we have $\mathcal{C}_1[V_1, \ldots, V_l] \Longrightarrow \Diamond U_{b1}$ and $(\Diamond U_{b1}, \Diamond U_{11}) \in \mathcal{S}_{15}$ for a usage U_{b1}. So, $U_1 \Longrightarrow \Diamond U_{b1}; \mathcal{C}_2[V_{l+1}, \ldots, V_n] \preceq \Diamond U_{b1} \otimes \mathcal{C}_2[V_{l+1}, \ldots, V_n]$ holds. Thus $U'_1 = \Diamond U_{b1} \otimes \mathcal{C}_2[V_{l+1}, \ldots, V_n]$ finishes the case.

- Case $\Diamond U_{11} \otimes U_{22} \preceq \Diamond U_{11}; U_{22}$. Similar to the case $\Diamond U_{11}; U_{22} \preceq \Diamond U_{11} \otimes U_{22}$.

- Cases $\Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22})$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = \Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22}) = U'_2$. So either of the following case holds:

  (a) $\mathcal{C} = [\,]$ and $V'_1 = \Diamond U_{11} \otimes \Diamond U_{22}$

  (b) $\mathcal{C} = \mathcal{C}_1; \mathcal{C}_2$ and $\mathcal{C}_1[V'_1, \ldots, V'_l] = \Diamond U_{11}$ and $\mathcal{C}_2[V'_{l+1}, \ldots, V'_n] = \Diamond U_{22}$

  In the case (a), we can easily show that the required conditions are satisfied in a similar way to the previous cases. In the case (b), by Lemma B.5(2), we have $\mathcal{C}_1[V_1, \ldots, V_l] \Longrightarrow \Diamond U_{b11}$ and $(\Diamond U_{b11}, \Diamond U_{11}) \in \mathcal{S}_{15}$ and $\mathcal{C}_2[V_{l+1}, \ldots, V_n] \Longrightarrow \Diamond U_{b12}$ and $(\Diamond U_{b12}, \Diamond U_{22}) \in \mathcal{S}_{15}$ for usages U_{b11} and U_{b12}. So, $U_1 \Longrightarrow \Diamond U_{b11} \otimes \Diamond U_{b12} \preceq \Diamond(U_{b11} \otimes U_{b12})$ holds. Thus $U'_1 = \Diamond(U_{b11} \otimes U_{b12})$ finishes the case.

- Cases $\Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22}$. It must be the case that: $\mathcal{C}[V'_1, \ldots, V'_n] = \Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22} = U'_2$. So either of the following cases holds:

  (a) $\mathcal{C} = [\,]$ and $V'_1 = \Diamond(U_{11} \otimes U_{22})$

  (b) $\mathcal{C} = \Diamond\mathcal{C}_1$ and $\mathcal{C}_1[V'_1, \ldots, V'_n] = U_{11} \otimes U_{22}$.

  In the case (a), since $U_1 = V_1 \Longrightarrow V'_1$, we can easily show that the required conditions are satisfied. In the case (b), by $\mathcal{C}_1[V'_1, \ldots, V'_n] = U_{11} \otimes U_{22}$, either of the following conditions holds:

  (b1) $\mathcal{C}_1 = [\,]$ and $V'_1 = U_{11} \otimes U_{22}$

  (b2) $\mathcal{C}_1 = \mathcal{C}_{11} \otimes \mathcal{C}_{12}$ and $\mathcal{C}_{11}[V'_1, \ldots, V'_l] = U_{11}$ and $\mathcal{C}_{12}[V'_{l+1}, \ldots, V'_n] = U_{22}$

  In the case (b1), $U_1 = \Diamond V_1 \Longrightarrow \Diamond V'_1 = \Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22}$ holds. So, $U'_1 = \Diamond U_{11} \otimes \Diamond U_{22} = U'_2$ satisfies the required conditions.

  In the case (b2),

  $$U'_2 = \Diamond\mathcal{C}_{11}[V'_1, \ldots, V'_l] \otimes \Diamond\mathcal{C}_{12}[V'_{l+1}, \ldots, V'_n]$$
  $$U_1 = \Diamond(\mathcal{C}_{11}[V_1, \ldots, V_l] \otimes \mathcal{C}_{12}[V_{l+1}, \ldots, V_n]) \preceq \Diamond\mathcal{C}_{11}[V_1, \ldots, V_l] \otimes \Diamond\mathcal{C}_{12}[V_{l+1}, \ldots, V_n]$$

So, for the usage context $\Diamond\mathcal{C}_{11}\otimes\Diamond\mathcal{C}_{12}$, we have $U_2' = (\Diamond\mathcal{C}_{11}\otimes\Diamond\mathcal{C}_{12})[V_1', \ldots, V_n']$ and $\Diamond\mathcal{C}_{11}[V_1, \ldots, V_l]\otimes \Diamond\mathcal{C}_{12}[V_{l+1}, \ldots, V_n] = (\Diamond\mathcal{C}_{11} \otimes \Diamond\mathcal{C}_{12})[V_1, \ldots, V_n]$. Thus $U_1' = \Diamond\mathcal{C}_{11}[V_1, \ldots, V_l] \otimes \Diamond\mathcal{C}_{12}[V_{l+1}, \ldots, V_n]$ finishes the case.

- Cases $\Diamond U_{11};_E U_{22} \equiv \Diamond(U_{11};_E U_{22})$. Similar to the cases $\Diamond U_{11} \otimes \Diamond U_{22} \preceq \Diamond(U_{11} \otimes U_{22})$ and $\Diamond(U_{11} \otimes U_{22}) \preceq \Diamond U_{11} \otimes \Diamond U_{22}$.

- Case $U_{11}\&U_{22} \preceq U_{11}$. It must be the case that: $\mathcal{C}[V_1', \ldots, V_n'] = U_{11}\&U_{22} \preceq U_{11} = U_2'$. So either of the following cases holds:

  (a) $\mathcal{C} = [\,]$ and $V_1' = U_{11}\&U_{22}$

  (b) $\mathcal{C} = \mathcal{C}_1\&\mathcal{C}_2$ and $\mathcal{C}_1[V_1', \ldots, V_l'] = U_{11}$ and $\mathcal{C}_2[V_{l+1}', \ldots, V_n'] = U_{22}$

  In the case (a), $U_1 = V_1 \Longrightarrow V_1' = U_{11}\&U_{22} \preceq U_{11}$ holds. Thus, $U_1' = U_{11}$ finishes the case. In the case (b), $U_1 = \mathcal{C}_1[V_1, \ldots, V_l]\&\mathcal{C}_2[V_{l+1}, \ldots, V_n] \preceq \mathcal{C}_1[V_1, \ldots, V_l]$ holds. So, $U_1' = \mathcal{C}_1[V_1, \ldots, V_l]$ finishes the case.

- Case $\mu\alpha.U \preceq [\mu\alpha.U/\alpha]U$. It must be the case that: $\mathcal{C}[V_1', \ldots, V_n'] = \mu\alpha.U \preceq [\mu\alpha.U/\alpha]U = U_2'$. So either of the following cases holds:

  (a) $\mathcal{C} = [\,]$ and $V_1' = \mu\alpha.U$

  (b) $\mathcal{C} = \mu\alpha.\mathcal{C}_1$ and $\mathcal{C}_1[V_1', \ldots, V_n'] = U$.

  In the case (a), by $U_1 = V_1 \Longrightarrow V_1' = \mu\alpha.U$, $U_1' = \mu\alpha.U$ satisfies the required conditions.

  In the case (b), $U_1 = \mu\alpha.\mathcal{C}_1[V_1, \ldots, V_n]$ holds. Since

  $$([\mu\alpha.U_{11}/\alpha]U_{11}, [\mu\alpha.U_{21}/\alpha]U_{21}) \in \mathcal{S}_{15}$$

  holds for $U_{11} = \mathcal{C}_1[V_1, \ldots, V_n]$ and $U_{21} = \mathcal{C}_1[V_1', \ldots, V_n']$, $U_1' = [\mu\alpha.U_{11}/\alpha]U_{11}$ satisfies the required conditions.

- Case $\dfrac{U \preceq U'}{\Diamond U \preceq \Diamond U'}$. It must be the case that: $U_2 = \mathcal{C}[V_1', \ldots, V_n'] = \Diamond U$ and $U \preceq U'$ and $U_2' = \Diamond U'$. So either of the following cases holds:

  (a) $\mathcal{C} = [\,]$ and $V_1' = \Diamond U$

  (b) $\mathcal{C} = \Diamond\mathcal{C}_1$ and $\mathcal{C}_1[V_1', \ldots, V_n'] = U$

  In the case (a), since $U_1 = V_1 \Longrightarrow V_1' = \Diamond U \preceq \Diamond U'$ holds, $U_1' = \Diamond U' = U_2'$ satisfies the required conditions. In the case (b), by the induction hypothesis, there exists U'' such that

  $$\mathcal{C}_1[V_1, \ldots, V_n] \Longrightarrow U'' \quad \text{and} \quad (U'', U') \in \mathcal{S}_{15}.$$

  So, we have $U_1 = \Diamond\mathcal{C}_1[V_1, \ldots, V_n] \Longrightarrow \Diamond U''$. From $(U'', U') \in \mathcal{S}_{15}$, $(\Diamond U'', \Diamond U') \in \mathcal{S}_{15}$ follows. Thus, $U_1' = \Diamond U''$ satisfies the required conditions.

- Case $\dfrac{U \preceq U'}{\blacklozenge U \preceq \blacklozenge U'}$. Similar to the case $\dfrac{U \preceq U'}{\Diamond U \preceq \Diamond U'}$.

- Case $\dfrac{U_{11} \preceq U_{11}'}{U_{11}; U_{22} \preceq U_{11}'; U_{22}}$. It must be the case that: $U_2 = \mathcal{C}[V_1', \ldots, V_n'] = U_{11}; U_{22}$ and $U_{11} \preceq U_{11}'$ and $U_2' = U_{11}'; U_{22}$. So either of the following cases holds:

  (a) $\mathcal{C} = [\,]$ and $V_1' = U_{11}; U_{22}$

  (b) $\mathcal{C} = \mathcal{C}_1; \mathcal{C}_2$ and $\mathcal{C}_1[V_1', \ldots, V_l'] = U_{11}$ and $\mathcal{C}_2[V_{l+1}', \ldots, V_n'] = U_{22}$

In the case (a), we can easily show that the required conditions are satisfied in a similar way to the previous case. In the case (b), by the induction hypothesis, there exists U''_{11} such that

$$\mathcal{C}_1[V_1, \ldots, V_l] \Longrightarrow U''_{11} \quad \text{and} \quad (U''_{11}, U'_{11}) \in \mathcal{S}_{15}.$$

So, we have $U_1 = \mathcal{C}_1[V_1, \ldots, V_l]; \mathcal{C}_2[V_{l+1}, \ldots, V_n] \Longrightarrow U''_{11}; \mathcal{C}_2[V_{l+1}, \ldots, V_n]$. From $(U''_{11}, U'_{11}) \in \mathcal{S}_{15}$ and $U'_2 = U'_{11}; \mathcal{C}_2[V'_{l+1}, \ldots, V'_n]$, $(U''_{11}; \mathcal{C}_2[V_{l+1}, \ldots, V_n], U'_2) \in \mathcal{S}_{15}$ follows. Thus, $U'_1 = U''_{11}; \mathcal{C}_2[V_{l+1}, \ldots, V_n]$ satisfies the required conditions.

- Cases $\dfrac{U_{11} \preceq U'_{11}}{U_{11} \otimes U_{22} \preceq U'_{11} \otimes U_{22}}$ and $\dfrac{U_{11} \preceq U'_{11}}{U_{11};_E U_{22} \preceq U'_{11};_E U_{22}}$. Similar to the case $\dfrac{U_{11} \preceq U'_{11}}{U_{11}; U_{22} \preceq U'_{11}; U_{22}}$.
- Case $\dfrac{U_2 \preceq U''_2 \quad U''_2 \preceq U'_2}{U_2 \preceq U'_2}$. By $(U_1, U_2) \in \mathcal{S}_{15}$ and the induction hypothesis, there exists U''_1 such that $U_1 \Longrightarrow U''_1$ and $(U''_1, U''_2) \in \mathcal{S}_{15}$. So, using the induction hypothesis again, we have U'_1 such that $U''_1 \Longrightarrow U'_1$ and $(U'_1, U'_2) \in \mathcal{S}_{15}$. Hence, we have $U_1 \Longrightarrow U'_1$ and $(U'_1, U'_2) \in \mathcal{S}_{15}$ as required. □

Now, we show that $\mathcal{S}_{15}$ is a weak usage simulation up to $\preceq$

Lemma B.7 *$\mathcal{S}_{15}$ is a weak usage simulation up to $\preceq$.*

Proof Note that if $(U_1, U_2) \in \mathcal{S}_{15}$, then there exist $V_1, \ldots, V_n$, $V'_1, \ldots, V'_n$ and $\mathcal{C}$ such that $U_1 = \mathcal{C}[V_1, \ldots, V_n]$ and $U_2 = \mathcal{C}[V'_1, \ldots, V'_n]$ and $V_i \Longrightarrow V'_i$ for $i = 1 \ldots n$. We show that each condition in Definition B.1 holds as follows:

(C1) Trivial.

(C2) By induction on derivation of $U_2 \xrightarrow{l} U'_2$, with case analysis on the last rule used.

- Case (UR-Lab). $U_2 = \mathcal{C}[V'_1, \ldots, V'_n] = l$ and $U_1 = \mathcal{C}[V_1, \ldots, V_n]$ hold. So we have either $\mathcal{C} = l$ or ($\mathcal{C} = [\,]$ and $V'_1 = l$). In both cases, it follows that $U_1 \overset{l}{\Longrightarrow} \mathbf{0}$. Thus, $U'_1 = \mathbf{0}$ finishes this case.
- Case (UR-Box). In this case, either of the following conditions holds:
  (i) $\mathcal{C} = [\,]$ and $U_2 = V'_1 = \Diamond U_b$, $U_1 = V_1 \Longrightarrow V'_1 = \Diamond U_b$ for a U_b
  (ii) $U_2 = \Diamond \mathcal{C}_1[V'_1, \ldots, V'_n]$ and $U_1 = \Diamond \mathcal{C}_1[V_1, \ldots, V_n]$.
  In the first case, since $U_1 \Longrightarrow U_2$ holds, $U'_1 = U'_2$ satisfies the required condition. In the second case, U'_2 must be of the form $\Diamond U'_{21}$ and $U_2 \xrightarrow{l} U'_2$ must have been derived from $\mathcal{C}_1[V'_1, \ldots, V'_n] \xrightarrow{l} U'_{21}$. So, by the induction hypothesis, there exists U'_{11} such that $\mathcal{C}_1[V_1, \ldots, V_n] \overset{l}{\Longrightarrow} U'_{11}$ and $U'_{11} \preceq \mathcal{S}_{15} \preceq U'_{21}$. So, $U'_1 = \Diamond U'_{11}$ satisfies the required condition.
- Case (UR-Ubox). Similar to the case for (UR-Box).
- Case (UR-Par). In this case, either of the following conditions holds.
  (i) $\mathcal{C} = [\,]$ and $U_2 = V'_1 = U_{11} \otimes U_{22}$, $U_1 = V_1 = U_{11} \otimes U_{22}$ for some U_{11}, U_{22}.
  (ii) $U_2 = \mathcal{C}_1[V'_1, \ldots, V'_m] \otimes \mathcal{C}_2[V'_{m+1}, \ldots, V'_n]$ and
  $U_1 = \mathcal{C}_1[V_1, \ldots, V_m] \otimes \mathcal{C}_2[V_{m+1}, \ldots, V_n]$.
  In the first case, since $U_1 \Longrightarrow U_2$ holds, $U'_1 = U'_2$ satisfies the required condition. In the second case, the followings must be derived:

$$\mathcal{C}_1[V'_1, \ldots, V'_m] \xrightarrow{l} U'_{21} \qquad U'_2 = U'_{21} \otimes \mathcal{C}_2[V'_{m+1}, \ldots, V'_n]$$

  By the induction hypothesis, there exists U'_{11} such that $\mathcal{C}_1[V_1, \ldots, V_n] \overset{l}{\Longrightarrow} U'_{11}$ and $U'_{11} \preceq \mathcal{S}_{15} \preceq U'_{21}$. So, $U'_1 = U'_{11} \otimes \mathcal{C}_2[V_{m+1}, \ldots, V_n]$ satisfies the required condition.

- Case (UR-Seq). Similar to the case for (UR-Par).
- Case (UR-SeqE). Similar to the case for (UR-Par).
- Case (UR-Hdlr). Since $\tau \notin \mathcal{A} \cup \{1\}$, this case cannot happen.
- Case (UR-Pcon). In this case, it must be the case that:

$$U_2 \preceq U_{21} \qquad U_{21} \xrightarrow{l} U'_{21} \qquad U'_{21} \preceq U'_2$$

By $(U_1, U_2) \in \mathcal{S}_{15}$, Lemma B.6 and the induction hypotheses, we have U_{11} and U'_{11} that satisfy the following diagrams:

$$\begin{array}{ccccccc} U_2 & \preceq & U_{21} & \xrightarrow{l} & U'_{21} & \preceq & U'_2 \\ & & & & \Upsilon| & & \\ \mathcal{S}_{15} & & \mathcal{S}_{15} & & \mathcal{S}_{15} & & \\ & & & & \Upsilon| & & \\ U_1 & \Longrightarrow & U_{11} & \stackrel{l}{\Longrightarrow} & U'_{11} & & \end{array}$$

Thus, $U'_1 = U'_{11}$ finishes the case.

(C4) By Lemma B.5(1).

(C5) Similar to the case for (C2).

(C3) By induction on derivation of $U_2 \xrightarrow{\tau} U'_2$, with case analysis on the last rule used.

- Case (UR-Hdlr). In this case, either of the following conditions holds.
  (i) $\mathcal{C} = [\,]$ and $U_2 = V_1 = U_{11};_E U_{12}$, $U_1 = V_1 \Longrightarrow V'_1$ for some U_{11}, U_{12}.
  (ii) $U_2 = \mathcal{C}_1[V'_1, \ldots, V'_m];_E \mathcal{C}_2[V'_{m+1}, \ldots, V'_n]$ and
  $U_1 = \mathcal{C}_1[V_1, \ldots, V_m];_E \mathcal{C}_2[V_{m+1}, \ldots, V_n]$.

  In the first case, since $U_1 \Longrightarrow U_2$ holds, $U'_1 = U'_2$ satisfies the required condition. In the second case, the following derivation must be derived for a usage U'_{21}:

$$\mathcal{C}_1[V'_1, \ldots, V'_m] \xrightarrow{E} U'_{21} \qquad U'_2 = \mathcal{C}_2[V_{m+1}, \ldots, V_n]$$

  By (C5), there exists U'_{11} such that $\mathcal{C}_1[V_1, \ldots, V_m] \stackrel{E}{\Longrightarrow} U'_{11}$. So, we have U_e and U'_e such that $\mathcal{C}_1[V_1, \ldots, V_m] \Longrightarrow U_e \xrightarrow{E} U'_e \Longrightarrow U'_{11}$. Thus, the followings hold:

$$\begin{array}{l} \mathcal{C}_1[V_1, \ldots, V_m];_E \mathcal{C}_2[V_{m+1}, \ldots, V_n] \\ \Longrightarrow \quad U_e;_E \mathcal{C}_2[V_{m+1}, \ldots, V_n] \\ \xrightarrow{\tau} \quad \mathcal{C}_2[V_{m+1}, \ldots, V_n] \end{array}$$

  Therefore, we have $U_1 \Longrightarrow \mathcal{C}_2[V_{m+1}, \ldots, V_n]$ and $U'_1 = \mathcal{C}_2[V_{m+1}, \ldots, V_n]$ satisfies the required condition.
- Cases for other rules: Similar to the corresponding cases of (C1). □

Proof of Lemma B.1 (15). By Lemma B.7 and Lemma B.2. □

Lemma B.8 *If $\varphi_1 \sqsubseteq \varphi_2$, then $(\varphi_1)^{use} \leq (\varphi_2)^{use}$.*

Proof Since the subeffect relation $\sqsubseteq$ has been defined as the partial order given by $E^? \sqsubseteq \mathbf{0} \sqsubseteq \top$ and $E^? \sqsubseteq E \sqsubseteq \top$ in Section 3.2, it is sufficient to show $E\&\mathbf{0} \leq \mathbf{0} \leq \mu\alpha.\alpha$ and $E\&\mathbf{0} \leq E \leq \mu\alpha.\alpha$.

$E\&\mathbf{0} \leq \mathbf{0}$ and $E\&\mathbf{0} \leq E$ follow immediately from $E\&\mathbf{0} \preceq \mathbf{0}, E\&\mathbf{0} \preceq E$ and the definition of subusage (Definition 3.5). By Lemma B.1(16) and $U \leq U$, we have $U \leq \mu\alpha.\alpha$ for any usage U. So, we get $\mathbf{0} \leq \mu\alpha.\alpha$ and $E \leq \mu\alpha.\alpha$. □

B.2 Basic properties of type judgment relation

We introduce several notations for convenience. For type environments Γ_1 and Γ_2 such that $dom(\Gamma_1) = dom(\Gamma_2)$, we define $\Gamma_1 \otimes \Gamma_2$ by $(\Gamma_1 \otimes \Gamma_2)(x) = \Gamma_1(x) \otimes \Gamma_2(x)$. Moreover, we define $\Gamma \otimes \emptyset = \emptyset \otimes \Gamma = \Gamma$. Next, we write $\Gamma_1 \leq \Gamma_2$, if $dom(\Gamma_1) = dom(\Gamma_2)$ and $\forall x \in dom(\Gamma_1).\Gamma_1(x) \leq \Gamma_2(x)$. Note that, if $\Gamma_1 \leq \Gamma_2$, then $\Gamma_1 \leq_\varphi \Gamma_2$ holds for any φ.

Finally, we define $U.\tau$ as the type obtained from replacing the usage of τ with U, that is:

$$U.\mathbf{bool} = \mathbf{bool} \qquad U.(\sigma_1 \xrightarrow{\varphi} \sigma_2, U') = (\sigma_1 \xrightarrow{\varphi} \sigma_2, U) \qquad U.(\mathbf{R}, U') = (\mathbf{R}, U).$$

Lemma B.9 (Inversion)

1. *If* $\Gamma \parallel \varphi \vdash \mathbf{true} : \delta$ *or* $\Gamma \parallel \varphi \vdash \mathbf{false} : \delta$, *then* $\delta = \mathbf{bool}$ *and* $\Gamma \leq_{\mathbf{0}} \emptyset$ *and* $\varphi \sqsubseteq \mathbf{0}$.
2. *If* $\Gamma \parallel \varphi \vdash x : \delta$ *then* $\Gamma \leq_{\mathbf{0}} x : \Diamond\delta$ *and* $\varphi \sqsubseteq \mathbf{0}$.
3. *If* $\Gamma \parallel \varphi \vdash \mathbf{new}^\Phi() : \delta$, *then* $\Gamma \leq_{\mathbf{0}} \emptyset$, $\varphi \sqsubseteq \mathbf{0}$ *and there exists* U *such that* $U \subseteq [\![\Phi]\!]$ *and* $(\mathbf{R}, U) \leq \delta$.
4. *If* $\Gamma \parallel \varphi \vdash \mathbf{acc}^a(M) : \delta$, *then* $\delta = \mathbf{bool}$ *and there exist* Γ_1 *and* φ_1 *such that* $\Gamma_1 \parallel \varphi_1 \vdash M : (\mathbf{R}, a)$ *and* $\Gamma \leq_{\varphi_1} \Gamma_1$ *and* $\varphi \sqsubseteq \varphi_1$.
5. *If* $\Gamma \parallel \varphi \vdash \mathbf{if}\ M_1\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3 : \delta$, *then there exist* Γ_1, Γ_2, φ_1, φ_2 *and* δ_1 *such that* $\Gamma_1 \parallel \varphi_1 \vdash M_1 : \mathbf{bool}$ *and* $\Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta_1$ *and* $\Gamma_2 \parallel \varphi_2 \vdash M_3 : \delta_1$ *and* $\Gamma \leq_{\varphi_1;\varphi_2} \Gamma_1;\Gamma_2$ *and* $\varphi \sqsubseteq \varphi_1;\varphi_2$ *and* $\delta_1 \leq \delta$.
6. *If* $\Gamma \parallel \varphi \vdash \mathbf{let}\ x = M_1\ \mathbf{in}\ M_2 : \delta$, *then there exist* $\Gamma_1, \Gamma_2, \varphi_1, \varphi_2, \sigma_1$ *and* δ_2 *such that* $\Gamma_1 \parallel \varphi_1 \vdash M_1 : \sigma_1$ *and* $\Gamma_2, x : \sigma_1 \setminus E \parallel \varphi_2 \vdash M_2 : \delta_2$ *and* $\Gamma \leq_{\varphi_1;\varphi_2} \Gamma_1;\Gamma_2$ *and* $\delta_2 \leq \delta$ *and* $\varphi \sqsubseteq (\varphi_1;\varphi_2)$.
7. *If* $\Gamma \parallel \varphi \vdash \mathbf{fun}(f, x, M) : \delta$, *then* $\varphi \sqsubseteq \mathbf{0}$ *and there exist* $U, U_1, \sigma_1, \delta_1, \delta_2$, Γ_1 *and* φ_1 *such that* $\Gamma_1, f : (\delta_1 \xrightarrow{\varphi_1} \delta_2, U_1), x : \sigma_1 \parallel \varphi_1 \vdash M : \delta_2$ *and* $\delta_1 \leq (\sigma_1 \setminus E)$, $(\delta_1 \xrightarrow{\varphi_1} \delta_2, U) \leq \delta$ *and* $\Gamma \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))}$.
8. *If* $\Gamma \parallel \varphi \vdash M_1 M_2 : \delta$, *then there exist* $\Gamma_1, \Gamma_2, \varphi_1, \varphi_2, \varphi_3, \delta_1$ *and* δ_2 *such that* $\Gamma_1 \parallel \varphi_1 \vdash M_1 : (\delta_1 \xrightarrow{\varphi_3} \delta_2, 1)$ *and* $\Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta_1$ *and* $\Gamma \leq_{(\varphi_1;\varphi_2);\varphi_3} (\Gamma_1;\Gamma_2);(\varphi_3)^{use}$ *and* $\varphi \sqsubseteq (\varphi_1;\varphi_2);\varphi_3$ *and* $\delta_2 \leq \delta$.
9. *If* $\Gamma \parallel \varphi \vdash M^{\{x\}}$, *then there exist* $\Gamma_1, \varphi_1, \delta_2$ *and* σ_1 *such that* $\Gamma_1 \parallel \varphi_1 \vdash M : \delta_2$ *and* $\Gamma \leq_{\varphi_1} \blacklozenge_x \Gamma_1$ *and* $\varphi \sqsubseteq \varphi_1$ *and* $\delta_2 \leq \delta$.
10. *If* $\Gamma \parallel \varphi \vdash \mathbf{raise} : \delta$, *then* $\Gamma \leq_E \emptyset$ *and* $\varphi \sqsubseteq E$.
11. *If* $\Gamma \parallel \varphi \vdash \mathbf{try}\ M_1\ \mathbf{with}\ M_2 : \delta$, *then there exist* $\Gamma_1, \Gamma_2, \varphi_1, \varphi_2$ *and* δ_1 *such that* $\Gamma_1 \parallel \varphi_1 \vdash M_1 : \delta_1$ *and* $\Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta_1$ *and* $\Gamma \leq_{\varphi_1;_E\varphi_2} (\Gamma_1;_E \Gamma_2)$ *and* $\varphi \sqsubseteq \varphi_1;_E \varphi_2$ *and* $\delta_1 \leq \delta$.

Proof From the typing rules. □

Lemma B.10

(1) If $\Gamma \parallel \varphi \vdash v : \delta$ *and* $Use_{\mathbf{0}}(\delta) \leq \mathbf{0}$, *then, for all* $x \in dom(\Gamma)$, $Use_{\mathbf{0}}(\Gamma(x)) \leq \mathbf{0}$ *and* $\varphi \sqsubseteq \mathbf{0}$.

(2) If $\Gamma \parallel \varphi \vdash \mathcal{E}^{\overline{try}}[\mathbf{raise}] : \delta$, *then, for all* $x \in dom(\Gamma)$, $Use_E(\Gamma(x)) \leq E$ *and* $\varphi \sqsubseteq E$.

Proof *(1)*: Follows immediately from Lemma B.9(*1, 2 and 7*).

(2): By structural induction on $\mathcal{E}^{\overline{try}}$. We show a few representative cases below.

- Case $\mathcal{E}^{\overline{try}} = [\,]$. By assumption,
$$\Gamma \parallel \varphi \vdash \mathbf{raise} : \delta.$$
By Lemma B.9(10), we have $\varphi \sqsubseteq E$ and $\Gamma \leq_E \emptyset$, which implies that $\forall x \in dom(\Gamma).\, Use_E(\Gamma(x)) \leq E$.

- Case $\mathcal{E}^{\overline{try}} = \mathbf{acc}^a(\mathcal{E}_1^{\overline{try}})$.
$$\Gamma \parallel \varphi \vdash \mathbf{acc}^a(\mathcal{E}_1^{\overline{try}}[\mathbf{raise}]) : \delta$$
By Lemma B.9(*4*), there exist Γ_1 and φ_1 such that $\Gamma_1 \parallel \varphi_1 \vdash \mathcal{E}_1^{\overline{try}}[\mathbf{raise}] : (\mathbf{R}, a)$ and $\Gamma \leq_{\varphi_1} \Gamma_1$ and $\varphi \sqsubseteq \varphi_1$.
By the induction hypothesis, we have $\varphi_1 \sqsubseteq E$ and $Use_E(\Gamma_1(x)) \leq E$ for all $x \in dom(\Gamma_1)$.
Therefore, we get $\varphi \sqsubseteq \varphi_1 \sqsubseteq E$ and
$$\begin{cases} \forall x \in dom(\Gamma_1).\, Use_E(\Gamma(x)) \leq Use_E(\Gamma_1(x)) \leq E \\ \forall x \in (dom(\Gamma) \setminus dom(\Gamma_1)).\, Use_E(\Gamma(x)) \leq (\varphi_1)^{use} \leq E \end{cases}$$

- Case $\mathcal{E}^{\overline{try}} = \mathbf{if}\ \mathcal{E}_1^{\overline{try}}\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3$.
$$\Gamma \parallel \varphi \vdash \mathbf{if}\ \mathcal{E}_1^{\overline{try}}[\mathbf{raise}]\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3 : \delta$$
By Lemma B.9(5), there exist Γ_1, Γ_2, φ_1, φ_2 and δ_1 such that $\Gamma_1 \parallel \varphi_1 \vdash \mathcal{E}_1^{\overline{try}}[\mathbf{raise}] : \mathbf{bool}$ and $\Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta_1$ and $\Gamma_2 \parallel \varphi_2 \vdash M_3 : \delta_1$ and $\Gamma \leq_{\varphi_1;\varphi_2} \Gamma_1; \Gamma_2$ and $\varphi \sqsubseteq \varphi_1; \varphi_2$ and $\delta_1 \leq \delta$.
By the induction hypothesis we have $\varphi_1 \sqsubseteq E$ and $Use_E(\Gamma_1(x)) \leq E$ for all $x \in dom(\Gamma_1)$.
Therefore, we get $\varphi \sqsubseteq \varphi_1; \varphi_2 \sqsubseteq E; \varphi_2 = E$,
$$\begin{array}{rcll} \forall x \in dom(\Gamma_1)(= \Gamma_2).\, Use_E(\Gamma(x)) & \leq & (Use_E(\Gamma_1(x)); Use_E(\Gamma_2(x))) & \\ & \leq & (E; Use_E(\Gamma_2(x))) & \\ & \leq & E \qquad \text{(by Lemma B.1(10))} & \end{array}$$
and, by $\Gamma \leq_{\varphi_1;\varphi_2} \Gamma_1; \Gamma_2$ and Lemma B.8
$$\begin{array}{rcl} \forall x \in (dom(\Gamma) \setminus dom(\Gamma_1)).\, Use_E(\Gamma(x)) & \leq & (\varphi_1; \varphi_2)^{use} \\ & \leq & (E; \varphi_2)^{use} \\ & = & (E)^{use} = E. \end{array}$$

□

Lemma B.11 *If $\Gamma \parallel \varphi \vdash M : \delta$, then there exist Γ' and φ' such that $\Gamma \leq \Gamma'$ and $\varphi \leq \varphi'$ and $\Gamma' \parallel \varphi' \vdash M : \delta$ and $(Use_{\varphi'}(\Gamma'(x)) \setminus E) \otimes (\varphi')^{use} \leq Use_{\varphi'}(\Gamma'(x))$ for all $x \in dom(\Gamma')$.*

Proof By induction on the derivation of $\Gamma \parallel \varphi \vdash M : \delta$.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Const). By (T-Const), we have $\Gamma = \emptyset$ and $\varphi = \mathbf{0}$. Therefore, $\Gamma' = \Gamma$ and $\varphi' = \varphi$ finish this case.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Var). By (T-Var), we have $\Gamma = x : \Diamond\delta$ and $\varphi = \mathbf{0}$. Since $\delta \setminus E \cong \delta$ and $(\Diamond\delta) \setminus E \leq \Diamond(\delta \setminus E)$ hold, we have $(\Diamond\delta) \setminus E \otimes (\mathbf{0})^{use} \leq \Diamond\delta$. This finishes the case.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-New). By (T-New), we have $\Gamma = \emptyset$ and $\varphi = \mathbf{0}$. Therefore, $\Gamma' = \Gamma$ and $\varphi' = \varphi$ finish this case.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Acc). By (T-Acc), $\delta = \mathbf{bool}$ and there exist M_1 and a such that $M = \mathbf{acc}^a(M_1)$ and $\Gamma \parallel \varphi \vdash M_1 : (\mathbf{R}, a)$. By the induction hypothesis, there exist Γ', φ' such that

$$\Gamma \leq \Gamma' \qquad \varphi \leq \varphi' \qquad \Gamma' \parallel \varphi' \vdash M_1 : (\mathbf{R}, a)$$
$$\forall x \in dom(\Gamma').(\mathit{Use}_{\varphi'}(\Gamma'(x)) \setminus E) \otimes (\varphi')^{use} \leq \mathit{Use}_{\varphi'}(\Gamma'(x))$$

  So, by (T-Acc), we get $\Gamma' \parallel \varphi' \vdash M : \delta$. This type judgement satisfies the required condition.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-If). By (T-If), there exist Γ_1, Γ_2, φ_1, φ_2 , M_1, M_2 and M_3 such that

$$M = \mathbf{if}\ M_1\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3$$
$$\Gamma_1 \parallel \varphi_1 \vdash M_1 : \mathbf{bool} \qquad \Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta \qquad \Gamma_2 \parallel \varphi_2 \vdash M_3 : \delta$$
$$\Gamma = \Gamma_1; \Gamma_2 \qquad \varphi = \varphi_1; \varphi_2$$

  By the induction hypothesis, there exist Γ'_1, φ'_1 and Γ'_2, φ'_2 such that

$$\Gamma_1 \leq \Gamma'_1 \qquad \varphi_1 \leq \varphi'_1 \qquad \Gamma'_1 \parallel \varphi'_1 \vdash M_1 : \mathbf{bool}$$
$$\Gamma_2 \leq \Gamma'_2 \qquad \varphi_2 \leq \varphi'_2 \qquad \Gamma'_2 \parallel \varphi'_2 \vdash M_2 : \delta \qquad \Gamma'_2 \parallel \varphi'_2 \vdash M_3 : \delta$$

$$\forall x \in dom(\Gamma'_1).(\mathit{Use}_{\varphi'_1}(\Gamma'_1(x)) \setminus E) \otimes (\varphi'_1)^{use} \leq \mathit{Use}_{\varphi'_1}(\Gamma'_1(x)) \tag{1}$$
$$\forall x \in dom(\Gamma'_2).(\mathit{Use}_{\varphi'_2}(\Gamma'_2(x)) \setminus E) \otimes (\varphi'_2)^{use} \leq \mathit{Use}_{\varphi'_2}(\Gamma'_2(x)) \tag{2}$$

  So, by (T-If), we get $\Gamma'_1; \Gamma'_2 \parallel \varphi'_1; \varphi'_2 \vdash M : \delta$. Here, by Lemma B.1(19), the expressions (1) and (2), the following relation holds for all $x \in dom(\Gamma'_1) = dom(\Gamma'_2)$:

$$(\mathit{Use}_{(\varphi'_1;\varphi'_2)}((\Gamma'_1;\Gamma'_2)(x)) \setminus E) \otimes (\varphi'_1;\varphi'_2)^{use} \leq \mathit{Use}_{(\varphi'_1;\varphi'_2)}((\Gamma'_1;\Gamma'_2)(x)).$$

  Thus, the type judgment relation $\Gamma'_1; \Gamma'_2 \parallel \varphi'_1; \varphi'_2 \vdash M : \delta$ satisfies the required condition.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Let) or (T-App). Similar to the case for (T-if).

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Fun). By (T-Fun), there exist Γ_1, U, U_1, δ_1, δ_2 and φ_1 such that

$$\Gamma = \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))} \qquad \varphi = \mathbf{0}. \tag{3}$$

  By Lemma B.1(7,12), we have

$$(\Diamond(U \setminus E)) \setminus E = \Diamond(U \setminus E);_E \mathbf{0} \preceq \Diamond((U \setminus E);_E \mathbf{0}) = \Diamond((U \setminus E) \setminus E) \cong \Diamond(U \setminus E)$$

$$(!(\Diamond(U \setminus E))) \setminus E \cong (\Diamond(!(U \setminus E))) \setminus E \preceq \Diamond((!(U \setminus E)) \setminus E) \cong \Diamond(!(U \setminus E)) \cong !(\Diamond(U \setminus E))$$

  for any U. Thus, we have $\mathit{Use}_{\mathbf{0}}(\Gamma(x)) \setminus E \otimes \mathbf{0} \leq \mathit{Use}_{\mathbf{0}}(\Gamma(x))$ for all $x \in dom(\Gamma)$.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Now). By (T-Now), there exist Γ_1, x and M_1 such that $\Gamma = \blacklozenge_x \Gamma_1$ and $M = M_1^{\{x\}}$ and $\Gamma_1 \parallel \varphi \vdash M_1 : \delta$. By the induction hypothesis, there exist Γ'_1, φ'_1 such that

$$\Gamma_1 \leq \Gamma'_1 \qquad \varphi_1 \leq \varphi'_1 \qquad \Gamma'_1 \parallel \varphi'_1 \vdash M_1 : \delta$$
$$\forall y \in dom(\Gamma'_1).(\mathit{Use}_{\varphi'_1}(\Gamma'_1(y)) \setminus E) \otimes (\varphi'_1)^{use} \leq \mathit{Use}_{\varphi'_1}(\Gamma'_1(y)). \tag{4}$$

So, by (T-Acc), we get $\blacklozenge_x \Gamma_1' \parallel \varphi_1' \vdash M : \delta$. Here, by Lemma B.1(18,9,14) and the expression (4), we have

$$\begin{array}{rcll}(\blacklozenge \mathit{Use}_{\varphi_1'}(\Gamma_1'(x)))\backslash E \otimes (\varphi_1')^{use} & \leq & \blacklozenge(\mathit{Use}_{\varphi_1'}(\Gamma_1'(x))\backslash E) \otimes \blacklozenge(\varphi_1')^{use} & \text{by Lemma B.1(14,18)} \\ & \leq & \blacklozenge((\mathit{Use}_{\varphi_1'}(\Gamma_1'(x))\backslash E) \otimes (\varphi_1')^{use}) & \text{by Lemma B.1(9)} \\ & \leq & \blacklozenge \mathit{Use}_{\varphi_1'}(\Gamma_1'(x)) & (4)\end{array}$$

Therefore, the type judgment $\blacklozenge_x \Gamma_1' \parallel \varphi_1' \vdash M : \delta$ satisfies the required condition.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Raise). By (T-Raise), we have $\Gamma = \emptyset$ and $\varphi = E$. Therefore, $\Gamma' = \Gamma$ and $\varphi' = \varphi$ finish this case.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Try). By (T-Try), there exist Γ_1, Γ_2, φ_1, φ_2 , M_1 and M_2 such that

$$M = \mathbf{try}\ M_1\ \mathbf{with}\ M_2$$
$$\Gamma_1 \parallel \varphi_1 \vdash M_1 : \delta \qquad \Gamma_2 \parallel \varphi_2 \vdash M_2 : \delta$$

By the induction hypothesis, there exist Γ_1', φ_1' and Γ_2', φ_2' such that

$$\begin{array}{rl}\Gamma_1 \leq \Gamma_1' \qquad \varphi_1 \leq \varphi_1' \qquad \Gamma_1' \parallel \varphi_1' \vdash M_1 : \delta & \\ \Gamma_2 \leq \Gamma_2' \qquad \varphi_2 \leq \varphi_2' \qquad \Gamma_2' \parallel \varphi_2' \vdash M_2 : \delta & \\ \forall x \in \mathit{dom}(\Gamma_1').(\mathit{Use}_{\varphi_1'}(\Gamma_1'(x))\backslash E) \otimes (\varphi_1')^{use} \leq \mathit{Use}_{\varphi_1'}(\Gamma_1'(x)) & (5) \\ \forall x \in \mathit{dom}(\Gamma_2').(\mathit{Use}_{\varphi_2'}(\Gamma_2'(x))\backslash E) \otimes (\varphi_2')^{use} \leq \mathit{Use}_{\varphi_2'}(\Gamma_2'(x)) & (6)\end{array}$$

So, by (T-Try), we get $\Gamma_1';_E \Gamma_2' \parallel \varphi_1';_E \varphi_2' \vdash M : \delta$. Here, by Lemma B.1(19)(2),(5) and (6), the following relation holds for all $x \in \mathit{dom}(\Gamma_1') = \mathit{dom}(\Gamma_2')$:

$$(\mathit{Use}_{(\varphi_1';_E\varphi_2')}((\Gamma_1';_E \Gamma_2')(x))\backslash E) \otimes (\varphi_1';_E \varphi_2')^{use} \leq \mathit{Use}_{(\varphi_1';_E\varphi_2')}((\Gamma_1';_E \Gamma_2')(x)).$$

Thus, the type judgment relation $\Gamma_1';_E \Gamma_2' \parallel \varphi_1';_E \varphi_2' \vdash M : \delta$ satisfies the required condition.

- Case: $\Gamma \parallel \varphi \vdash M : \delta$ is derived by rule (T-Weak). By (T-Weak), there exist φ_1, Γ_1, δ_1 such that $\varphi \sqsubseteq \varphi_1$ and $\Gamma \leq_{\varphi_1} \Gamma_1$ and $\delta_1 \leq \delta$ and $\Gamma_1 \parallel \varphi_1 \vdash M : \delta_1$. By the induction hypothesis, there exist Γ_1', φ_1' such that

$$\begin{array}{rl}\Gamma_1 \leq \Gamma_1' \qquad \varphi_1 \leq \varphi_1' \qquad \Gamma_1' \parallel \varphi_1' \vdash M : \delta' & \\ (\mathit{Use}_{\varphi_1'}(\Gamma_1'(x))\backslash E) \otimes (\varphi_1')^{use} \leq \mathit{Use}_{\varphi_1'}(\Gamma_1'(x)) & (7)\end{array}$$

Here, for $x_1, \ldots, x_n \in (\mathit{dom}(\Gamma) \setminus \mathit{dom}(\Gamma_1'))$, we can construct a type environment Γ_1''

$$\Gamma_1'' = \Gamma_1', x_1 : (\varphi_1')^{use}.(\Gamma(x_1)), \ldots, x_n : (\varphi_1')^{use}.(\Gamma(x_n))$$

Operators $U.\tau$ used above are defined in the first sentence of Appendix B.2. Then, by (T-Weak), we have $\mathit{dom}(\Gamma) = \mathit{dom}(\Gamma_1'')$ and $\Gamma_1'' \parallel \varphi_1' \vdash M : \delta$. By LemmaB.1(17) and (7), letting $\Gamma' = \Gamma_1''$ and $\varphi' = \varphi_1'$ finishes this case. □

Lemma B.12 *If* $\Gamma, x : \sigma_1 \parallel \varphi \vdash M : \delta$, *then* $\Gamma, x : (\sigma_1 \backslash E) \otimes (\varphi)^{use} \parallel \varphi \vdash M : \delta$.

Proof By Lemma B.11, there exist Γ', σ_1' and φ' such that $\Gamma \leq \Gamma'$ and $\sigma \leq \sigma'$ and $\varphi \leq \varphi'$ and

$$\Gamma', x : \sigma_1' \parallel \varphi' \vdash M : \delta \qquad \text{and} \qquad (\Gamma' \backslash E) \otimes (\varphi')^{use} \leq \Gamma' \qquad \text{and} \qquad (\sigma_1' \backslash E) \otimes (\varphi')^{use} \leq \sigma_1'.$$

Hence, we have $\Gamma', x : (\sigma_1' \backslash E) \otimes (\varphi')^{use} \parallel \varphi' \vdash M : \delta$ by (T-Weak). It follows that $\Gamma, x : (\sigma_1 \backslash E) \otimes (\varphi)^{use} \parallel \varphi \vdash M : \delta$ from $\Gamma \leq \Gamma'$, $\sigma \leq \sigma'$, $\varphi \leq \varphi'$ and (T-Weak). □

Lemma B.13 *If* $\Gamma_1 \leq_\varphi \Gamma_2$, *then* $\Gamma_1; (\varphi')^{use} \leq_{(\varphi;\varphi')} \Gamma_2; (\varphi')^{use}$.

Proof By definitions of $\Gamma_1 \leq_\varphi \Gamma_2$ and the subtype relation. □

Lemma B.14 *If $U_1 \leq 1$, then $\Delta^{\mathbf{fun}}_{(U_1,U_2,\Gamma)} \leq \Delta^{\mathbf{fun}}_{(U_2,U_2,\Gamma)} \otimes \Gamma$.*

Proof By $U_1 \leq 1$, we have $1 \in [\![U_1]\!]$. If $1 \notin U_2$ and $[\![U_1]\!] \subseteq \{\epsilon, 1, 1\downarrow\}$, then $\Delta^{\mathbf{fun}}_{(U_1,U_2,\Gamma)} = \Gamma$ and $\Delta^{\mathbf{fun}}_{(U_2,U_2,\Gamma)} = \emptyset$. Therefore, $\Delta^{\mathbf{fun}}_{(U_1,U_2,\Gamma)} = \Delta^{\mathbf{fun}}_{(U_2,U_2,\Gamma)} \otimes \Gamma = \Gamma$. In the case where either $1 \in U_2$ or $[\![U_1]\!] \not\subseteq \{\epsilon, 1, 1\downarrow\}$ holds, $\Delta^{\mathbf{fun}}_{(U_1,U_2,\Gamma)} = !\Gamma$ and $!\Gamma \leq \Delta^{\mathbf{fun}}_{(U_2,U_2,\Gamma)}$ hold. Hence, we have $\Delta^{\mathbf{fun}}_{(U_1,U_2,\Gamma)} = !\Gamma \leq !\Gamma \otimes \Gamma \leq \Delta^{\mathbf{fun}}_{(U_2,U_2,\Gamma)} \otimes \Gamma$. □

Lemma B.15 *If $\Gamma \parallel \varphi \vdash v : \delta$, then there exists a type environment Γ' such that $\Gamma \leq \Diamond\Gamma'$ and $\Diamond\Gamma' \parallel \varphi \vdash v : \delta$.*

Proof A value is either a boolean value (**true** or **false**), a variable or a function.

- Case $v = \mathbf{true}$ or $v = \mathbf{false}$.

  $$\Gamma \parallel \varphi \vdash v : \mathbf{bool}$$
  $$\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$$

  By Lemma B.9(1), $\Gamma \leq_{\mathbf{0}} \emptyset$ and $\varphi \sqsubseteq \mathbf{0}$ hold. By $\Gamma \leq_{\mathbf{0}} \emptyset$, we have

  $$x_1 : \sigma_1, \ldots, x_n : \sigma_n \leq x_1 : \mathbf{0}.\sigma_1, \ldots, x_n : \mathbf{0}.\sigma_n.$$

  Let $\Gamma_1 = x_1 : \mathbf{0}.\sigma_1, \ldots, x_n : \mathbf{0}.\sigma_n$. Then, from $\Diamond\mathbf{0} \equiv \mathbf{0}$, it follows that $\Gamma \leq \Gamma_1 \equiv \Diamond\Gamma_1$.

  Moreover, by $\varphi \sqsubseteq \mathbf{0}$, (T-Const) and (T-Weak), we have

  $$x_1 : \mathbf{0}.\sigma_1, \ldots, x_n : \mathbf{0}.\sigma_n \parallel \varphi \vdash v : \mathbf{bool}$$

  Hence, we get $\Diamond\Gamma_1 \parallel \varphi \vdash v : \mathbf{bool}$. $\Gamma' = \Gamma_1$ finishes this case.

- Case $v = x$

  $$\Gamma \parallel \varphi \vdash x : \delta$$
  $$\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$$

  By Lemma B.9(2), we have $x_1 : \sigma_1, \ldots, x_n : \sigma_n \leq_{\mathbf{0}} x : \Diamond\delta$ and $\varphi \sqsubseteq \mathbf{0}$ hold.

  By $x_1 : \sigma_1, \ldots, x_n : \sigma_n \leq_{\mathbf{0}} x : \Diamond\delta$, there exists i such that $x_i = x$ and $\sigma_i \leq \Diamond\delta$ and $\sigma_j \leq \mathbf{0}.\sigma_j$ $(i \neq j)$.

  Let $\Gamma_1 = x : \delta, x_{j_1} : \mathbf{0}.\sigma_{j_1}, \ldots, x_{j_{(n-1)}} : \mathbf{0}.\sigma_{j_{(n-1)}}$, $\{1, .., i-1, i+1, .., n\} = \{j_1, .., j_{(n-1)}\}$. Then, from $\Diamond\mathbf{0} \equiv \mathbf{0}$, it follows that $\Gamma \leq \Gamma_1 \equiv \Diamond\Gamma$.

  Moreover, by $\varphi \sqsubseteq \mathbf{0}$, (T-Var) and (T-Weak), we have $\Diamond\Gamma_1 \parallel \varphi \vdash v : \delta$. $\Gamma' = \Gamma_1$ finishes this case.

- Case $v = \mathbf{fun}(x, f, M)$

  $$\Gamma \parallel \varphi \vdash \mathbf{fun}(f, x, M) : \delta$$
  $$\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$$

  By Lemma B.9(7), $\varphi \sqsubseteq \mathbf{0}$ and there exist $U, U_1, \sigma_1, \delta_1, \delta_2, \Gamma_1$ and φ_1 such that $\Gamma_1, f : (\delta_1 \xrightarrow{\varphi_1} \delta_2, U_1)$, $x : \sigma_1 \parallel \varphi_1 \vdash M : \delta_2$ and $\delta_1 \leq (\sigma_1 \setminus E)$ and $(\delta_1 \xrightarrow{\varphi_1} \delta_2, U) \leq \delta$ and $\Gamma \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U,U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))}$.

  Here, by $\Gamma \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U,U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))}$, we can assume

  $$\Gamma_1 = x_1 : \sigma_{11}, \ldots, x_k : \sigma_{1k} \ (k \leq n)$$

  without loss of generarity.

By $\varphi \sqsubseteq \mathbf{0}$, $(\delta_1 \xrightarrow{\varphi_1} \delta_2, U) \leq \delta$, (T-Weak) and (T-Fun), we have

$$\begin{array}{c} x_1 : \Delta^{\mathbf{fun}}_{(U,U_1 \setminus E, \Diamond(\sigma_{11} \setminus E))}, \ldots, x_k : \Delta^{\mathbf{fun}}_{(U,U_1 \setminus E, \Diamond(\sigma_{1k} \setminus E))}, \\ x_{k+1} : \mathbf{0}.\sigma_{k+1}, \ldots x_n : \mathbf{0}.\sigma_n \end{array} \quad \| \varphi \vdash \mathbf{fun}(f, x, M) : \delta$$

Here, by the definition of $\Delta^{\mathbf{fun}}_{(U,U_1 \setminus E, \Diamond(\sigma_{1i} \setminus E))}$ and $\Diamond\mathbf{0} \equiv \mathbf{0}$ and Lemma B.1(7), we have a type σ'_{1i} such that $\Delta^{\mathbf{fun}}_{(U,U_1 \setminus E, \Diamond(\sigma_{1i} \setminus E))} \cong \Diamond\sigma'_{1i}$ $(i \in \{1 \ldots n\})$.

Therefore, we have $\sigma'_{11}, \ldots, \sigma'_{1k}$ such that

$$\begin{array}{c} x_1 : \Diamond\sigma'_{11}, \ldots, x_k : \Diamond\sigma'_{1k}, \\ x_{k+1} : (\mathbf{0}.\sigma_{k+1}), \ldots, x_n : (\mathbf{0}.\sigma_n) \end{array} \quad \| \varphi \vdash \mathbf{fun}(f, x, M) : \delta$$

Hence, $\Gamma' = x_1 : \sigma'_{11}, \ldots, x_k : \sigma'_{1k}, x_{k+1} : (\mathbf{0}.\sigma_{k+1}), \ldots, x_n : (\mathbf{0}.\sigma_n)$ satisfies the required condition.

B.3 Proofs of Lemmas 4.2 and 4.3

In this section, we prove Lemmas 4.2 and 4.3.

Proof of Lemma 4.2.

- For the first part, by $\varphi \vdash (H, v)$, we have

$$\begin{array}{c} x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n) \parallel \varphi \vdash v : \delta \\ dom(H) = \{x_1, \ldots, x_n\} \\ [\![U_1]\!] \subseteq H(x_1), \ldots, [\![U_n]\!] \subseteq H(x_n) \end{array}$$

  From Lemma B.10(1), it follows that $U_1 \leq \mathbf{0}, \ldots, U_n \leq \mathbf{0}$. Therefore, we get $\downarrow \in [\![\mathbf{0}]\!] \subseteq [\![U_i]\!] \subseteq H(x_i)$ for all $i \in \{1, \ldots, n\}$.

- For the second part, by $\varphi \vdash (H, \mathcal{E}^{\overline{try}}[\mathbf{raise}])$, we have

$$\begin{array}{c} x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n) \parallel \varphi \vdash \mathcal{E}^{\overline{try}}[\mathbf{raise}] : \delta \\ dom(H) = \{x_1, \ldots, x_n\} \\ [\![U_1]\!] \subseteq H(x_1), \ldots, [\![U_n]\!] \subseteq H(x_n) \end{array}$$

  From Lemma B.10(2), it follows that $U_1 \leq E, \ldots, U_n \leq E$. Therefore, we get $\downarrow \in [\![E]\!] \subseteq [\![U_i]\!] \subseteq H(x_i)$ for all $i \in \{1, \ldots, n\}$.

Proof of Lemma 4.3. Evaluation of a well-annotated term may get stuck only in the following three cases:

- Case $M = \mathcal{E}[\mathbf{acc}^a(v)]$ and v is not a variable. By $\varphi \vdash (H, \mathcal{E}[\mathbf{acc}^a(v)])$, there are $x_1, \ldots, x_n$, $U_1, \ldots, U_n$ and φ' such that

$$\begin{array}{c} x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n) \parallel \varphi' \vdash \mathbf{acc}^a(v) : \mathbf{bool} \\ dom(H) = \{x_1, \ldots, x_n\} \end{array}$$

  Hence, by (T-Acc), v must have a resource type. Therefore, v must be a variable, hence a contradiction.

- Case $M = \mathcal{E}[v_1 v_2]$ and v_1 is not a function. By $\varphi \vdash (H, \mathcal{E}[v_1 v_2])$, there are $x_1, \ldots, x_n$, $U_1, \ldots, U_n$ and φ' such that

$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n) \parallel \varphi' \vdash v_1 v_2 : \mathbf{bool}$$
$$dom(H) = \{x_1, \ldots, x_n\}$$

  Hence, by (T-App), v_1 must have a function type. Therefore, v_1 must be of the form $\mathbf{fun}(f', x', M')$, hence a contradiction.

- Case $M = \mathcal{E}[\mathbf{if}\ v\ \mathbf{then}\ M_1\ \mathbf{else}\ M_2]$ and v is a not boolean.

  This case is rejected by a similar argument. □

B.4 Substitution Lemma

We will the following substitution lemma:

> If $\Gamma_1, x : \sigma_1 \parallel \varphi_1 \vdash M : \delta_2$ and $\Gamma_2 \parallel \mathbf{0} \vdash v : \sigma_1 \setminus E$ and $dom(\Gamma_1) = dom(\Gamma_2)$ then $\Gamma_1 \otimes \Gamma_2 \parallel \varphi_1 \vdash [v/x]M : \delta_2$.

Unfortunately, this property cannot be directly proved by induction on the derivation of $\Gamma_1, x : \sigma_1 \parallel \varphi_1 \vdash M : \delta_2$ – the type environment $\Gamma_1 \otimes \Gamma_2$ in the conclusion is too weak when that statement is used as the induction hypothesis. Thus, we strengthen the property for induction to work.

For this purpose, we first extend usage expressions by introducing a new usage $(U_1|U_2)$ which is similar to $U_1 \otimes U_2$ but synchronizes on the transition E. We then prove the following properties (Lemma B.19 and B.22)

> If $\Gamma_1, x : \tau_x, y : \tau_y \parallel \varphi_1 \vdash M : \delta_2$, then $\Gamma_1, y : (\tau_y|\tau_x) \parallel \varphi_1 \vdash [y/x]M : \delta_2$.

and

> If $\Gamma_1, f : (\sigma_f, U_1) \parallel \varphi_1 \vdash M : \delta_2$ and $\Gamma_2 \parallel \mathbf{0} \vdash \mathbf{fun}(f', x', M') : (\sigma_f, 1)$, then $\Gamma_1|(U_1 \odot \Gamma_2) \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f]M : \delta_2$

Here, $U_1 \odot \Gamma_2$, which intuitively means multiplication of Γ_2 by U_1, will be defined later. Using these lemmas, we can prove the substitution lemma.

We introduce a new expression $(U_1|U_2)$ to the syntax of usage expressions

$$U ::= \cdots \mid (U_1|U_2).$$

Then, the structural pre-order on usages $U_1 \preceq U_2$ and the transition relation $U_1 \xrightarrow{l} U_2$ are extended:

Definition B.2 (Additional Pre-order relations for $(U_1|U_2)$) *The relation $\preceq$ on usages is extended to the one closed under the additional relations (recall that $U_1 \equiv U_2$ means $U_1 \preceq U_2$ and $U_2 \preceq U_1$):*

$$\Diamond(U_1|U_2) \equiv \Diamond U_1|\Diamond U_2 \quad (U_1|U_2) \equiv U_2|U_1 \quad \mathbf{0} \equiv (\mathbf{0}|\mathbf{0})$$

Definition B.3 (Additional Usage Reduction rules for $(U_1|U_2)$) *The transition relation on usages is extended to the one closed under the additional relations:*

$$\frac{U_1 \xrightarrow{l} U_1' \quad l \neq E}{U_1|U_2 \xrightarrow{l} U_1'|U_2} \qquad \frac{U_1 \xrightarrow{E} U_1' \quad U_2 \xrightarrow{E} U_2'}{U_1|U_2 \xrightarrow{E} U_1'|U_2'}$$

The second rule means that $(U_1|U_2)$ has a transition labeled E if both usages U_1 and U_2 can make transition on E simultaneously.

Note. *I general, we must prove old lemmas (like Lemma B.1) again for the new usage expression* $(U_1|U_2)$*. However, we omit those proofs since these are similar to the case* $U_1 \otimes U_2$ *of corresponding lemmas*

From here, we use notations $\Delta^{\mathbf{fun}}_{(U,U_1,U')}$ and $\Delta^{\mathbf{fun}}_{(U,U_1,\sigma)}$ which are defined in a manner similar to $\Delta^{\mathbf{fun}}_{(U,U_1,\Gamma)}$ as follows:

$$\Delta^{\mathbf{fun}}_{(U_2,U_1,U)} = \begin{cases} \mathbf{0} & \text{if } 1 \notin [\![U_2]\!] \\ U & (1 \in [\![U]\!] \subseteq \{\epsilon, 1, 1\downarrow\}) \wedge (1 \notin [\![U_1]\!]) \\ !U & \text{otherwise} \end{cases}$$

$$\Delta^{\mathbf{fun}}_{(U_2,U_1,\sigma)} = \begin{cases} \mathbf{0}.\sigma & \text{if } 1 \notin [\![U_2]\!] \\ \sigma & (1 \in [\![U]\!] \subseteq \{\epsilon, 1, 1\downarrow\}) \wedge (1 \notin [\![U_1]\!]) \\ !\sigma & \text{otherwise} \end{cases}$$

Here, $\mathbf{0}.\sigma$ is defined at the beginning of Appendix B.2.

Moreover, we add $\sigma_1|\sigma_2$ and $\Gamma_1|\Gamma_2$ to the binary operations on types and type environments in Definition 3.11. Here, the operations $\sigma_1|\sigma_2$ and $\Gamma_1|\Gamma_2$ are defined in a manner similar to other operations in the definition, namely $;, \&, ;_E$.

Lemma B.16 *The subusage relation* $\leq$ *satisfies the following properties:*

1. $(U_1;U_2)|(U_3;U_4) \leq (U_1|U_3);(U_2|U_4)$,
2. $(U_1;_E U_2)|(U_3;_E U_4) \leq (U_1|U_3);_E (U_2|U_4)$,
3. $(!U_1|!U_2) \leq !(U_1|U_2)$,
4. $(\blacklozenge U_1|\blacklozenge U_2) \leq \blacklozenge(U_1|U_2)$,
5. $((\varphi)^{use}|(\varphi)^{use}) \leq (\varphi)^{use}$
6. $U_1 \otimes U_2 \leq U_1|(U_2 \otimes (\varphi)^{use})$ *for any* φ.

Proof Similarly to the proof of Lemma B.1.

Lemma B.17 *If* $\Gamma \parallel \varphi \vdash M{:}\delta$*, then there exist* Γ' *and* φ' *such that* $\Gamma \leq \Gamma'$ *and* $\varphi \sqsubseteq \varphi'$ *and* $\Gamma' \parallel \varphi' \vdash M{:}\delta$ *and* $((Use_{\varphi'}(\Gamma'(x)))|(\varphi')^{use}) \leq Use_{\varphi'}(\Gamma'(x))$ *for all* $x \in dom(\Gamma')$. □

Proof Similar to Lemma B.11

Lemma B.18 *If* $\Gamma, x : \sigma_1 \parallel \varphi \vdash M : \delta$*, then* $\Gamma, x : (\sigma_1|(\varphi)^{use}) \parallel \varphi \vdash M : \delta$.

Proof By Lemma B.17, there exist Γ', σ'_1 and φ' such that $\Gamma \leq \Gamma'$ and $\sigma \leq \sigma'$ and $\varphi \leq \varphi'$ and

$$\Gamma', x : \sigma'_1 \parallel \varphi' \vdash M : \delta \qquad \text{and} \qquad (\Gamma'|(\varphi')^{use}) \leq \Gamma' \qquad \text{and} \qquad (\sigma'_1|(\varphi')^{use}) \leq \sigma'_1.$$

Hence, we have $\Gamma', x : (\sigma'_1|(\varphi')^{use}) \parallel \varphi' \vdash M : \delta$ by (T-Weak). It follows that $\Gamma, x : (\sigma_1|(\varphi)^{use}) \parallel \varphi \vdash M : \delta$ from $\Gamma \leq \Gamma'$, $\sigma \leq \sigma'$, $\varphi \leq \varphi'$ and (T-Weak). □

Lemma B.19 (Substitution of Variables)
If $\Gamma, y : \sigma_y, x : \sigma_x \parallel \varphi \vdash M : \delta$ *and* $\sigma_x|\sigma_y$ *is well-defined, then* $\Gamma, y : (\sigma_y|\sigma_x) \parallel \varphi \vdash [x/y]M : \delta$.

Proof By induction on the structure of M. We show a few representative cases below:

- Case $M = \mathbf{try}\ M_1\ \mathbf{with}\ M_2$ By Lemma B.9(11), $\varphi \sqsubseteq \varphi_1;_E \varphi_2$, there exist $\Gamma_1, \Gamma_2, \sigma_{y1}, \sigma_{y2}, \sigma_{x1}, \sigma_{x2}$, δ_2, φ_1 and φ_2 such that $\Gamma \leq \Gamma_1;_E \Gamma_2$ and $\sigma_x \leq \sigma_{x1};_E \sigma_{x2}$ and $\sigma_y \leq \sigma_{y1};_E \sigma_{y2}$ and $\delta_2 \leq \delta$ and $\varphi \sqsubseteq \varphi_1;_E \varphi_2$ and

$$\Gamma_1, y : \sigma_{y1}, x : \sigma_{x1} \parallel \varphi_1 \vdash M_1 : \delta_2$$
$$\Gamma_2, y : \sigma_{y2}, x : \sigma_{x2} \parallel \varphi_2 \vdash M_2 : \delta_2$$

Thus, by the induction hypothesis, we have

$$\Gamma_1, y : (\sigma_{y1}|\sigma_{x1}) \parallel \varphi_1 \vdash [y/x]M_1 : \delta_2$$
$$\Gamma_2, y : (\sigma_{y2}|\sigma_{x2}) \parallel \varphi_2 \vdash [y/x]M_2 : \delta_2.$$

By rule (T-Try),

$$\Gamma_1;_E \Gamma_2, y : (\sigma_{y1}|\sigma_{x1});_E (\sigma_{y2}|\sigma_{x2}) \parallel \varphi_1;_E \varphi_2 \vdash \mathbf{try}\ [|y!|/x]M_1\ \mathbf{with}\ [y/x]M_2.$$

By Lemma B.16(2),

$$(\sigma_y|\sigma_x) \leq ((\sigma_{y1};_E \sigma_{y2})|(\sigma_{x1};_E \sigma_{x2})) \leq (\sigma_{y1}|\sigma_{x1});_E (\sigma_{y2}|\sigma_{x2}).$$

By (T-Weak) and $\varphi \sqsubseteq \varphi_1;_E \varphi_2$ and $\delta_2 \leq \delta$, we have

$$\Gamma, y : (\sigma_y|\sigma_x) \parallel \varphi \vdash [y/x](\mathbf{try}\ M_1\ \mathbf{with}\ M_2) : \delta.$$

- Case $M = \mathbf{fun}(f, z, M_1)$. By Lemma B.9(11), $\varphi \sqsubseteq \mathbf{0}$ and there exist $U, U_1, \sigma_1, \delta_1, \delta_2, \Gamma_1, \sigma_{x1}, \sigma_{y1}$ and φ_1 such that $\delta_1 \leq \sigma_1 \setminus E$ and $(\delta_1 \xrightarrow{\varphi_1} \delta_2, U) \leq \delta$ and

$$\Gamma_1, y : \sigma_{y1}, x : \sigma_{x1}, f : (\delta_1 \xrightarrow{\varphi_1} \delta_2, U_1), z : \sigma_1 \parallel \varphi_1 \vdash M_1 : \delta_2$$

and $\Gamma \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \diamond(\Gamma_1 \setminus E))}$ and $\sigma_y \leq \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \diamond(\sigma_{y1} \setminus E))}$ and $\sigma_x \leq \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \diamond(\sigma_{x1} \setminus E))}$.
Then, by the induction hypothesis, we have

$$\Gamma_1, y : (\sigma_{y1}|\sigma_{x1}), f : (\delta_1 \xrightarrow{\varphi_1} \delta_2, U_1), z : \sigma_1 \parallel \varphi_1 \vdash [y/x]M_1 : \delta_2.$$

By rule (T-Fun),

$$\Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, (\diamond(\Gamma_1 \setminus E)))}, y : \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, (\diamond((\sigma_{y1}|\sigma_{x1}) \setminus E)))} \parallel \mathbf{0} \vdash \mathbf{fun}(f, z, [y/x]M_1) : (\delta_1 \xrightarrow{\varphi_1} \delta_2, U)$$

Then,

$$\begin{array}{lll} \sigma_x|\sigma_y & \leq & (\Delta^{\mathbf{fun}}_{(U_1, U_2 \setminus E, \diamond(\sigma_{y1} \setminus E))}|\Delta^{\mathbf{fun}}_{(U_1, U_2 \setminus E, \diamond(\sigma_{x1} \setminus E))}) \\ & \leq & \Delta^{\mathbf{fun}}_{(U_1, U_2 \setminus E, (\diamond(\sigma_{y1} \setminus E)|\diamond(\sigma_{x1} \setminus E)))} \quad \text{by Lemma B.16(3)} \\ & \equiv & \Delta^{\mathbf{fun}}_{(U_1, U_2 \setminus E, (\diamond((\sigma_{y1} \setminus E)|(\sigma_{x1} \setminus E))))} \quad \text{by Definition B.2} \\ & \leq & \Delta^{\mathbf{fun}}_{(U_1, U_2 \setminus E, (\diamond((\sigma_{y1}|\sigma_{x1}) \setminus E)))} \quad \text{by Lemma B.16(2)} \end{array}$$

Hence, by (T-Weak), $(\delta_1 \xrightarrow{\varphi_1} \delta_2, U_1) \leq \delta$ and $\varphi \sqsubseteq \mathbf{0}$, we have

$$\Gamma, y : (\sigma_y|\sigma_x) \parallel \varphi \vdash [y/x](\mathbf{fun}(f, z, M_1)) : \delta. \qquad \square$$

We will give the substitution lemma for function values. In order to prove the lemma, we define $U \odot U_1$, $U \odot \tau$ and $U \odot \Gamma$ as follows:

$$U \odot U_1 = [\mu\alpha.\alpha/a, U_1/1]U \quad U \odot \tau = [\mu\alpha.\alpha/a, \tau/1]U \quad U \odot \Gamma = [\mu\alpha.\alpha/a, \Gamma/1]U$$

where, $[\mu\alpha.\alpha/a]U$ means the usage expression obtained by replacing all occurrences of access label a in U with $\mu\alpha.\alpha$.

For example, $(1\&(1;E)) \odot (R;C) = (R;C)\&(R;C;E)$ and $(1\&E) \odot (x:(\mathbf{R},R;C), y:(\mathbf{R},(W\&\mathbf{0})), z:\mathbf{bool}) = x:(\mathbf{R},(R;C)\&E), y:(\mathbf{R},(W\&\mathbf{0})\&E), z:\mathbf{bool}$, $(1;W;C) \odot (R;C) = R;C;(\mu\alpha.\alpha);(\mu\alpha.\alpha)$.

Lemma B.20

1. *If $U_1 \leq U_2$, then $U_1 \odot \Diamond U \leq U_2 \odot \Diamond U$.*
2. *If $U_1 \leq U_2$, then $U \odot U_1 \leq U \odot U_2$.*

Proof Similar to Lemma B.1. □

Lemma B.21 *If $U \cong U \setminus E$, then the followings hold.*

1. $\mathbf{0} \leq U \odot \mathbf{0}$
2. $\Delta^{\mathbf{fun}}_{(U,U_2,U_1)} \leq U \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}$

Proof We assume $U \cong U \setminus E$ and define a function on usages as follows:

$$Times_{Fun}(U) = \begin{cases} \mathbf{0} & \text{if } 1 \notin [\![U]\!] \\ 1 & \text{if } 1 \in [\![U]\!] \subseteq \{\epsilon, 1, 1\downarrow\} \\ !1 & \text{otherwise} \end{cases}$$

By $U \cong U \setminus E$, $[\mu\alpha.\alpha/a]U$ can make only τ or 1 transitions. Hence, we have $Times_{Fun}(U) \leq [\mu\alpha.\alpha/a]U$ by the definition of $Times_{Fun}(\cdot)$. Moreover, $[\mu\alpha.\alpha/a]U \odot U' = U \odot U'$ holds by the definition of $U \odot U'$. So, we have $Times_{Fun}(U) \odot U' \leq U \odot U'$ for any U, U'.

We first prove 1 as follows

$$\mathbf{0} \equiv \Diamond\mathbf{0} \cong [\Diamond\mathbf{0}/1](Times_{Fun}(U)) \leq [\mu\alpha.\alpha/a, \Diamond\mathbf{0}/1]U = U \odot (\Diamond\mathbf{0}) \equiv U \odot \mathbf{0}.$$

Next, we prove 2.

- Case $1 \notin [\![U]\!]$: By Lemma B.20(1),

$$\begin{array}{rcl} \Delta^{\mathbf{fun}}_{(U,U_2,U_1)} & = & \mathbf{0} = (\mathbf{0} \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \\ & = & (Times_{Fun}(U) \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \leq (U \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \end{array}$$

- Case $1 \in [\![U]\!] \subseteq \{\epsilon, 1, 1\downarrow\}$ and $1 \notin [\![U_2]\!]$: By Lemma B.20(1),

$$\begin{array}{rcl} \Delta^{\mathbf{fun}}_{(U,U_2,U_1)} & = & U_1 = (1 \odot U_1) = (1 \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \\ & = & (Times_{Fun}(U) \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \leq (U \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \end{array}$$

- Otherwise: By Lemma B.20(1),

$$\begin{array}{rcl} \Delta^{\mathbf{fun}}_{(U,U_2,U_1)} & = & !U_1 \cong !!U_1 = (!1 \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \\ & = & (Times_{Fun}(U) \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \leq (U \odot \Delta^{\mathbf{fun}}_{(1,U_2,U_1)}) \end{array}$$

Lemma B.22 (Substitution of Functions)

If $\Gamma_1, f:(\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, U_1) \parallel \varphi_1 \vdash M:\delta_2$ and $\Gamma_2 \parallel \varphi_2 \vdash \mathbf{fun}(f', x', M'):(\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, 1)$ and $dom(\Gamma_1) = dom(\Gamma_2)$ and $\Gamma_1|\Gamma_2$ is well-defined, then $\Gamma_1|(U_1 \odot \Gamma_2) \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f]M : \delta_2$

Proof By induction on the structure of M. We show a few representative cases below:

- Case $M = \mathbf{try}\ M_1\ \mathbf{with}\ M_2$. By Lemma B.15, there is Γ_2' such that

$$dom(\Gamma_2') = dom(\Gamma_2) \quad \Gamma_2 \leq \diamond\Gamma_2'$$
$$\diamond\Gamma_2' \parallel \varphi_2 \vdash \mathbf{fun}(f', x', M') : (\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, 1).$$

  By Lemma B.9, there exist $\Gamma_{11}, \Gamma_{12}, U_{11}, U_{12}$, δ_3, φ_{11} and φ_{12} such that $\Gamma_1 \leq \Gamma_{11};_E \Gamma_{21}$ and $U_1 \leq U_{11};_E U_{12}$ and $\delta_3 \leq \delta_2$ and $\varphi_1 \sqsubseteq \varphi_{12};_E \varphi_{12}$ and

$$dom(\Gamma_{11}) = dom(\Gamma_1) \quad \Gamma_{11}, f : (\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, U_{11}) \parallel \varphi_{11} \vdash M_1 : \delta_3$$
$$dom(\Gamma_{12}) = dom(\Gamma_1) \quad \Gamma_{12}, f : (\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, U_{12}) \parallel \varphi_{12} \vdash M_2 : \delta_3$$

  By the induction hypothesis, $\delta_3 \leq \delta_2$ and (T-Weak), we have

$$\Gamma_{11}|(U_{11} \odot \diamond\Gamma_2') \parallel \varphi_{11} \vdash [\mathbf{fun}(f', x', M')/f]M_1 : \delta_2$$
$$\Gamma_{12}|(U_{12} \odot \diamond\Gamma_2') \parallel \varphi_{12} \vdash [\mathbf{fun}(f', x', M')/f]M_2 : \delta_2.$$

  By rule (T-Try), we have

$$(\Gamma_{11}|(U_{11} \odot \diamond\Gamma_2'));_E (\Gamma_{12}|(U_{12} \odot \diamond\Gamma_2')) \parallel \varphi_1;_E \varphi_2$$
$$\vdash \mathbf{try}\ [\mathbf{fun}(f', x', M')/f]M_1\ \mathbf{with}\ [\mathbf{fun}(f', x', M')/f]M_2 : \delta_2.$$

  By Lemma B.16(2) and (T-Weak), we have

$$(\Gamma_{11};_E \Gamma_{12})|((U_{11} \odot \diamond\Gamma_2');_E (U_{12} \odot \diamond\Gamma_2')) \parallel \varphi_1;_E \varphi_2$$
$$\vdash \mathbf{try}\ [\mathbf{fun}(f', x', M')/f]M_1\ \mathbf{with}\ [\mathbf{fun}(f', x', M')/f]M_2 : \delta_2$$

  In general, for any U, it holds that $(U_{11} \odot U);_E (U_{12} \odot U) = (U_{11};_E U_{12}) \odot U$. Hence, we have

$$(\Gamma_{11};_E \Gamma_{12})|((U_{11};_E U_{12}) \odot \diamond\Gamma_2') \parallel \varphi_1;_E \varphi_2$$
$$\vdash \mathbf{try}\ [\mathbf{fun}(f', x', M')/f]M_1\ \mathbf{with}\ [\mathbf{fun}(f', x', M')/f]M_2 : \delta_2.$$

  By Lemma B.20(1), (T-Weak) and $\Gamma_1 \leq \Gamma_{11};_E \Gamma_{12}$ and $U_1 \leq U_{11};_E U_{12}$ and $\varphi_1 \sqsubseteq \varphi_{11};_E \varphi_{12}$, we have

$$\Gamma_1|(U_1 \odot \diamond\Gamma_2') \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f](\mathbf{try}\ M_1\ \mathbf{with}\ M_2) : \delta_2.$$

  By (T-Weak), Lemma B.20(2) and $\Gamma_2 \leq \diamond\Gamma_2'$, we get

$$\Gamma_1|(U_1 \odot \Gamma_2) \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f](\mathbf{try}\ M_1\ \mathbf{with}\ M_2) : \delta_2.$$

- Case $M = \mathbf{fun}(g, x_1, M_1)$. By Lemma B.15 and (T-Fun), there is Γ_2' such that

$$dom(\Gamma_2') = dom(\Gamma_2) \quad \Gamma_2 \leq \diamond\Gamma_2'$$
$$\diamond\Gamma_2' \parallel \mathbf{0} \vdash \mathbf{fun}(f', x', M') : (\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, 1).$$

  By Lemma B.9, we have $\varphi_1 \sqsubseteq \mathbf{0}$ and there exist $\Gamma_1', U_1', U_o, U_g$, δ_{g1} and δ_{g2} such that $\delta_1 \leq \sigma_1 \setminus E$ and $(\delta_{g1} \xrightarrow{\varphi_g} \delta_{g2}, U) \leq \delta_2$ and

$$\Gamma_1 \leq \Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \diamond(\Gamma_1' \setminus E))} \qquad U_1 \leq \Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \diamond(U_1' \setminus E))} \qquad dom(\Gamma) = dom(\Gamma_1')$$

$$\Gamma_1', f : (\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, U_1'), g : (\delta_{g1} \xrightarrow{\varphi_g} \delta_{g2}, U_g), x_1 : \sigma_1 \parallel \varphi_g \vdash M_1 : \delta_{g2}.$$

By (T-Weak) and $dom(\Gamma_1') = dom(\Gamma_2')$, we have

$$\Diamond\Gamma_2', g : (\delta_{g1} \xrightarrow{\varphi_g} \delta_{g2}, \mathbf{0}), x_1 : \mathbf{0}.\sigma_1 \parallel \mathbf{0} \vdash \mathbf{fun}(f', x', M') : (\delta_{f1} \xrightarrow{\varphi_f} \delta_{f2}, 1).$$

So, by the induction hypothesis, we have

$$\begin{array}{l}(\Gamma_1'|(U_1' \odot \Diamond\Gamma_2')), g : (\delta_{g1} \xrightarrow{\varphi_g} \delta_{g2}, (U_g|(U_1' \odot \mathbf{0}))), x_1 : (Use(\sigma_1)|(U_1' \odot \mathbf{0})).\sigma_1 \parallel \varphi_g \\ \qquad\qquad \vdash [\mathbf{fun}(f', x', M')/f]M_1 : \delta_{g2}.\end{array}$$

Here, for any U, U', it follows that

$$\begin{array}{rcl} U \setminus E & \cong & (U \setminus E) \otimes \mathbf{0} \leq ((U \setminus E)|\mathbf{0}) \leq ((U \setminus E)|((U' \setminus E) \odot \mathbf{0})) = ((U \setminus E)|((U' \odot \mathbf{0}) \setminus E)) \\ & \leq & (U|(U' \odot \mathbf{0})) \setminus E \end{array} \quad (8)$$

from Lemma B.21(1) and Lemma B.16(2,6). So, we have $\delta_1 \leq \sigma_1 \setminus E \leq ((Use(\sigma_1)|(U_1' \odot \mathbf{0})).\sigma_1) \setminus E$. Therefore, from (T-Fun), it follows that

$$\Delta^{\mathbf{fun}}_{(U_o, ((U_g|(U_1' \odot \mathbf{0})) \setminus E), \Diamond((\Gamma_1'|(U_1' \odot \Diamond\Gamma_2')) \setminus E))} \parallel \mathbf{0} \vdash \mathbf{fun}(g, x_1, [\mathbf{fun}(f', x', M')/f]M_1) : (\delta_{g1} \xrightarrow{\varphi_g} \delta_{g2}, U_o).$$

By $(\delta_{g1} \xrightarrow{\varphi_g} \delta_{g2}, U) \leq \delta_2$ and $\mathbf{fun}(g, x_1, [\mathbf{fun}(f', x', M')/f]M_1) = [\mathbf{fun}(f', x', M')/f]\mathbf{fun}(g, x_1, M_1)$,

$$\Delta^{\mathbf{fun}}_{(U_o, ((U_g|(U_1' \odot \mathbf{0})) \setminus E), \Diamond((\Gamma_1'|(U_1' \odot \Diamond\Gamma_2')) \setminus E))} \parallel \mathbf{0} \vdash [\mathbf{fun}(f', x', M')/f]\mathbf{fun}(g, x_1, M_1) : \delta_2.$$

holds.

Therefore, we have

$$\begin{array}{rcll} \Gamma_1|(U_1 \odot \Gamma_2) & \leq & \Gamma_1|(U_1 \odot \Diamond\Gamma_2') & \text{(by (T-Weak))} \\ & \leq & \Gamma_1|\Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond(U_1' \setminus E))} \odot \Diamond\Gamma_2') & \text{(by Lemma B.20(1))} \\ & = & \Gamma_1|\Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond((U_1' \odot \Diamond\Gamma_2') \setminus E))} & \text{(by Definition of } \odot) \\ & \leq & \Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond(\Gamma_1' \setminus E))}|\Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond((U_1' \odot \Diamond\Gamma_2') \setminus E))} & \text{(by (T-Weak))} \\ & \leq & \Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond(\Gamma_1' \setminus E)|(\Diamond((U_1' \odot \Diamond\Gamma_2') \setminus E)))} & \text{(by Lemma B.16(3))} \\ & \leq & \Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond((\Gamma_1' \setminus E)|((U_1' \odot \Diamond\Gamma_2') \setminus E)))} & \text{(by Definition B.2)} \\ & \leq & \Delta^{\mathbf{fun}}_{(U_o, U_g \setminus E, \Diamond((\Gamma_1'|(U_1' \odot \Diamond\Gamma_2')) \setminus E))} & \text{(by Lemma B.16(2))} \\ & \leq & \Delta^{\mathbf{fun}}_{(U_o, (U_g|(U' \odot \mathbf{0})) \setminus E, \Diamond((\Gamma_1'|(U_1' \odot \Diamond\Gamma_2')) \setminus E))} & \text{(by the equation (8)).} \end{array}$$

So, we get

$$\Gamma_1|(U_1 \odot \Gamma_2) \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f]\mathbf{fun}(g, x_1, M_1) : \delta_2.$$

- Case $M = M_1 M_2$. By Lemma B.15, there is Γ_2' such that

$$\begin{array}{l} dom(\Gamma_2) = dom(\Gamma_2') \quad \Gamma_2 \leq \Diamond\Gamma_2' \\ \Diamond\Gamma_2' \parallel \varphi_2 \vdash \mathbf{fun}(f', x', M') : (\sigma_f, 1). \end{array}$$

By Lemma B.9, there exist $\Gamma_{11}, \Gamma_{21}, U_{11}, U_{12}, \varphi_{11}, \varphi_{12}, \varphi_3, \delta_1$ and δ_3 such that $\Gamma_1 \leq (\Gamma_{11}; \Gamma_{12}); (\varphi_3)^{use}$ and $U_1 \leq (U_{11}; U_{12}); (\varphi_3)^{use}$ and $\varphi \leq (\varphi_{11}; \varphi_{12}); \varphi_3$ and $\delta_3 \leq \delta_2$ and

$$\begin{array}{ll} dom(\Gamma_{11}) = dom(\Gamma_1) & \Gamma_{11}, f : (\sigma_f, U_{11}) \parallel \varphi_{11} \vdash M_1 : (\delta_1 \xrightarrow{\varphi_3} \delta_3, 1) \\ dom(\Gamma_{12}) = dom(\Gamma_1) & \Gamma_{12}, f : (\sigma_f, U_{12}) \parallel \varphi_{12} \vdash M_2 : \delta_1. \end{array}$$

By the induction hypothesis, we have

$$\Gamma_{11}|(U_{11} \odot \Diamond\Gamma'_2) \parallel \varphi_{11} \vdash [\mathbf{fun}(f', x', M')/f]M_1 : (\delta_1 \xrightarrow{\varphi_3} \delta_3, 1)$$
$$\Gamma_{12}|(U_{12} \odot \Diamond\Gamma'_2) \parallel \varphi_{12} \vdash [\mathbf{fun}(f', x', M')/f]M_2 : \delta_1.$$

By rule (T-App), we have

$$(\Gamma_{11}|(U_{11} \odot \Diamond\Gamma'_2)); (\Gamma_{12}|(U_{11} \odot \Diamond\Gamma'_2)); (\varphi_3)^{use} \parallel \varphi_{11}; \varphi_{12}; \varphi_3$$
$$\vdash ([\mathbf{fun}(f', x', M')/f]M_1)([\mathbf{fun}(f', x', M')/f]M_2) : \delta_3.$$

By Lemma B.16(1) and (T-Weak), we have

$$((\Gamma_{11}; \Gamma_{12})|((U_{11} \odot \Diamond\Gamma'_2); (U_{12} \odot \Diamond\Gamma'_2))); (\varphi_3)^{use} \parallel \varphi_{11}; \varphi_{12}; \varphi_3$$
$$\vdash ([\mathbf{fun}(f', x', M')/f]M_1)([\mathbf{fun}(f', x', M')/f]M_2) : \delta_3.$$

By (T-Weak) and Lemma B.16(1),(5), we have

$$(((\Gamma_{11}; \Gamma_{12}); (\varphi_3)^{use})|((U_{11} \odot \Diamond\Gamma'_2); (U_{12} \odot \Diamond\Gamma'_2)); (\varphi_3)^{use}) \parallel \varphi_{11}; \varphi_{12}; \varphi_3$$
$$\vdash ([\mathbf{fun}(f', x', M')/f]M_1)([\mathbf{fun}(f', x', M')/f]M_2) : \delta_3.$$

In general, for any U and φ, it holds that $(U_{11} \odot U); (U_{12} \odot U); (\varphi)^{use} = ((U_{11}; U_{12}) \odot U); (\varphi)^{use} = (U_{11}; U_{12}; (\varphi)^{use}) \odot U$. Therefore, we have

$$(((\Gamma_{11}; \Gamma_{12}); (\varphi_3)^{use})|(((U_{11}; U_{12}); (\varphi_3)^{use}) \odot \Diamond\Gamma'_2)) \parallel \varphi_{11}; \varphi_{12}; \varphi_3$$
$$\vdash ([\mathbf{fun}(f', x', M')/f]M_1)([\mathbf{fun}(f', x', M')/f]M_2) : \delta_3$$

Finally, by Lemma B.20(1), (T-Weak) and $\Gamma_1 \leq (\Gamma_{11}; \Gamma_{12}); (\varphi_3)^{use}$ and $U_1 \leq (U_{11}; U_{12}); (\varphi_3)^{use}$ and $\varphi_1 \sqsubseteq (\varphi_{11}; \varphi_{12}); \varphi_3$, we have

$$\Gamma_1|(U_1 \odot \Diamond\Gamma'_2) \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f](M_1M_2) : \delta_2.$$

By (T-Weak), Lemma B.20(2) and $\Gamma_2 \leq \Diamond\Gamma'_2$, we get

$$\Gamma_1|(U_1 \odot \Gamma_2) \parallel \varphi_1 \vdash [\mathbf{fun}(f', x', M')/f](M_1M_2) : \delta_2.$$

□

Lemma B.23 *If* $\Gamma \parallel \varphi \vdash \mathbf{fun}(f, x, M) : (\delta_1 \xrightarrow{\varphi_f} \delta_2, U)$ *then there exists* Γ' *such that* $\Diamond\Gamma' \parallel \varphi \vdash \mathbf{fun}(f, x, M) : (\delta_1 \xrightarrow{\varphi_f} \delta_2, 1)$ *and* $\Gamma \leq U \odot \Diamond\Gamma'$.

Proof By Lemma B.9(7), there exist U_1, σ_1, Γ_1 and φ_1 such that

$$\delta_1 \leq \sigma_1 \setminus E \tag{9}$$
$$\Gamma \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))} \tag{10}$$
$$\Gamma_1, f : (\delta_1 \xrightarrow{\varphi_f} \delta_2, U_1), x : \sigma_1 \parallel \varphi \vdash M : \delta_2 \tag{11}$$

By (T-fun), (9) and (11), we have

$$\Delta^{\mathbf{fun}}_{(1, U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))} \parallel \mathbf{0} \vdash \mathbf{fun}(f, x, M) : (\delta_1 \xrightarrow{\varphi_f} \delta_2, 1)$$

Here, from Lemma B.1(7) and Lemma B.21(2), it follows that

$$\Gamma \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U, U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))} \leq U \odot \Delta^{\mathbf{fun}}_{(1, U_1 \setminus E, \Diamond(\Gamma_1 \setminus E))} \cong U \odot (\Diamond(\Delta^{\mathbf{fun}}_{(1, U_1 \setminus E, (\Gamma_1 \setminus E))})).$$

So, $\Gamma' = \Delta^{\mathbf{fun}}_{(1, U_1 \setminus E, (\Gamma_1 \setminus E))}$ finishes this case. □

Now, we are ready to to prove the following substitution lemma.

Lemma B.24 (Substitution Lemma)
If $\Gamma_1, x{:}\sigma_1 \parallel \varphi_1 \vdash M{:}\delta_2$ *and* $\Gamma_2 \parallel \varphi_2 \vdash v{:}\sigma_1 \setminus E$ *and* $dom(\Gamma_1) = dom(\Gamma_2)$ *and* $\Gamma_1 \otimes \Gamma_2$ *is well-defined, then* $\Gamma_1 \otimes \Gamma_2 \parallel \varphi_1 \vdash [v/x]M : \delta_2$.

Proof We prove this lemma by case analysis on value v.

- Case: v is either **true** or **false**. The conclusion follows by straightforward induction on the derivation of $\Gamma_1, x : \sigma_1 \parallel \varphi_1 \vdash M : \delta_2$.

- Case: v is a variable y.

  We assume $y \in dom(\Gamma_1)$ without loss of generality. We therefore assume $\Gamma_1 = \Gamma_{11}, y : \sigma_y$.

$$\Gamma_{11}, y : \sigma_y, x : \sigma_1 \parallel \varphi_1 \vdash M : \delta_2 \tag{12}$$
$$\Gamma_2 \parallel \varphi_2 \vdash y : \sigma_1 \backslash E. \tag{13}$$

  By applying Lemmas B.12 and B.18 to the type judgment (12) above, we have

$$(\Gamma_{11}|(\varphi_1)^{use}), y : \sigma_y, x : (\sigma_1 \backslash E) \otimes (\varphi_1)^{use} \parallel \varphi_1 \vdash M : \delta_2 \tag{14}$$

  By Lemma B.19, we have

$$(\Gamma_{11}|(\varphi_1)^{use}), y : (\sigma_y|((\sigma_1 \backslash E) \otimes (\varphi_1)^{use})) \parallel \varphi_1 \vdash [y/x]M : \delta_2. \tag{15}$$

  By applying Lemma B.9 to the type judgment (13) above,

$$\Gamma_2 \leq_{\varphi_2} y : \Diamond(\sigma_1 \backslash E)$$

  Then, by Lemma B.13,

$$\Gamma_2; (\varphi_1)^{use} \quad \leq_{(\mathbf{0};\varphi_1)} \quad y : \Diamond(\sigma_1 \backslash E); (\varphi_1)^{use} \leq y : \Diamond(\sigma_1 \backslash E) \otimes (\varphi_1)^{use} \leq y : (\sigma_1 \backslash E) \otimes (\varphi_1)^{use}$$

  Thus, we have

$$\Gamma_2; (\varphi_1')^{use} \leq_{(\mathbf{0};\varphi_1')} y : (\sigma_1 \backslash E) \otimes (\varphi_1)^{use}. \tag{16}$$

  Now consider the type environment $\Gamma_1|(\Gamma_2; (\varphi_1)^{use})$, where $\Gamma_1 = \Gamma_{11}, y : \sigma_y$. By (16),

$$\begin{cases} (\Gamma_1|(\Gamma_2; (\varphi_1)^{use}))(y) & = \sigma_y|(\Gamma_2; (\varphi_1)^{use})(y) \\ & \leq \sigma_y|((\sigma_1 \backslash E) \otimes (\varphi_1)^{use}) \\ (\Gamma_1|(\Gamma_2; (\varphi_1)^{use}))(z) & = \Gamma_{11}(z)|(\Gamma_2; (\varphi_1)^{use})(z) \\ & \leq \Gamma_{11}(z)|(\mathbf{0}; \varphi_1)^{use} \\ & \leq \Gamma_{11}(z)|(\varphi_1)^{use} \\ & = (\Gamma_{11}|(\varphi_1)^{use})(z) \qquad \text{if } z \neq y \end{cases}$$

  Therefore, by (T-Weak), we have

$$\Gamma_1|(\Gamma_2; (\varphi_1)^{use}) \parallel \varphi_1 \vdash [y/x]M : \delta_2.$$

  By Lemma B.16(6) and $\Gamma_1|(\Gamma_2 \otimes (\varphi_1)^{use}) \leq \Gamma_1|(\Gamma_2; (\varphi_1)^{use})$ we have

$$\Gamma_1 \otimes \Gamma_2 \parallel \varphi_1 \vdash [y/x]M : \delta_2.$$

- Case v is a function $\mathbf{fun}(f, z, M_1)$.

  By Lemma B.23, there exists Γ_2' such that

$$\Diamond\Gamma_2' \parallel \varphi_2 \vdash v : 1.(\sigma_1 \backslash E)$$
$$\Gamma_2 \leq (U_1 \backslash E) \odot \Diamond\Gamma_2' \qquad U_1 = Use(\sigma_1) \tag{17}$$

By Lemma B.22, we have

$$\Gamma_1|(U_1 \odot \Diamond\Gamma_2') \parallel \varphi_1 \vdash [v/x]M : \delta_2.$$

By Lemma B.12, we also have

$$\Gamma_1|(((U_1 \odot \Diamond\Gamma_2')\backslash E) \otimes (\varphi_1)^{use}) \parallel \varphi_1 \vdash [v/x]M : \delta_2.$$

Here, in general, for any U, U', φ, it holds that $((U\backslash E) \odot U')\backslash E = (U\backslash E) \odot U'$ and $(U \odot U') \otimes (\varphi)^{use} = (U \otimes \varphi) \odot U'$. Therefore, we get

$$\Gamma_1|((((U_1\backslash E) \otimes (\varphi_1)^{use})) \odot \Diamond\Gamma_2') \parallel \varphi_1 \vdash [v/x]M : \delta_2.$$

It remains to show $\Gamma_1 \otimes \Gamma_2 \leq \Gamma_1|((((U_1\backslash E) \otimes (\varphi_1)^{use})) \odot \Diamond\Gamma_2')$, which follows from:

$$\begin{array}{rcll} \Gamma_1 \otimes \Gamma_2 & \leq & \Gamma_1|(\Gamma_2 \otimes (\varphi_1)^{use}) & \text{(by Lemma B.16(6))} \\ & \leq & \Gamma_1|(((U_1\backslash E) \odot \Diamond\Gamma_2') \otimes (\varphi_1)^{use}) & \text{(by (17))} \\ & \leq & \Gamma_1|(((U_1\backslash E) \otimes (\varphi_1)^{use}) \odot \Diamond\Gamma_2') & \text{(by the definition of } \odot) \end{array}$$

B.5 Proof of Lemma 4.4

Lemma B.25 *If $\Gamma \parallel \varphi \vdash \mathcal{E}[\mathbf{new}^\Phi()] : \delta$ and z does not occur in $\mathcal{E}$ or Γ, then $\Gamma, z : (\mathbf{R}, U) \parallel \varphi \vdash \mathcal{E}[z] : \delta$ for some U such that $[\![U]\!] \subseteq \Phi$.*

Proof By structural induction on $\mathcal{E}$. We show a few representative cases below.

- Case $\mathcal{E} = [\,]$.

  $$\mathcal{E}[\mathbf{new}^\Phi()] = \mathbf{new}^\Phi()$$

  By Lemma B.9(3), $\Gamma \leq_{\mathbf{0}} \emptyset$ and $\varphi \sqsubseteq \mathbf{0}$ and there exists U such that $[\![U]\!] \subseteq \Phi$ and $(\mathbf{R}, U) \leq \delta$.

  From (T-Var), we have

  $$z : (\mathbf{R}, \Diamond U) \parallel \mathbf{0} \ \vdash z : (\mathbf{R}, U)$$

  From (T-Weak), $\Gamma \leq_{\mathbf{0}} \emptyset$ and $\varphi \sqsubseteq \mathbf{0}$, we have $\Gamma, z : (\mathbf{R}, \Diamond U) \parallel \varphi \vdash z : \delta$.

  $[\![\Diamond U]\!] = [\![U]\!] \subseteq \Phi$ finishes the case.

- Case $\mathcal{E} = (\mathbf{let}\ x = \mathcal{E}_1\ \mathbf{in}\ M_1)$.

  $$\mathcal{E}[\mathbf{new}^\Phi()] = (\mathbf{let}\ x = \mathcal{E}_1[\mathbf{new}^\Phi()]\ \mathbf{in}\ M_1)$$

  By Lemma B.9(6), there exist Γ_1, Γ_2, φ_1, φ_2, δ_1, σ_x such that $\Gamma \leq_{\varphi_1;\varphi_2} \Gamma_1;\Gamma_2$ and $\varphi \sqsubseteq \varphi_1;\varphi_2$ and $\delta_1 \leq \sigma_x\backslash E$ and

  $$\Gamma_1 \parallel \varphi_1 \vdash \mathcal{E}_1[\mathbf{new}^\Phi()] : \delta_1$$
  $$\Gamma_2, x : \sigma_x \parallel \varphi_2 \vdash M_1 : \delta$$

  By the induction hypothesis, there exists U such that

  $$[\![U]\!] \subseteq \Phi$$
  $$\Gamma_1, z : (\mathbf{R}, U) \parallel \varphi_1 \vdash \mathcal{E}_1[z] : \delta_1$$

  Hence, from (T-Let), we have

  $$\Gamma_1;\Gamma_2, z : (\mathbf{R}, U; (\varphi_2)^{use}) \parallel \varphi_1;\varphi_2 \vdash \mathbf{let}\ x = \mathcal{E}_1[z]\ \mathbf{in}\ M_1 : \delta$$

  Moreover, from (T-Weak), we get

  $$\Gamma, z : (\mathbf{R}, U; (\varphi_2)^{use}) \parallel \varphi \vdash \mathbf{let}\ x = \mathcal{E}[z]\ \mathbf{in}\ M_1 : \delta$$

  Here, for any usage U and effect φ, we have $[\![U; (\varphi)^{use}]\!] \subseteq [\![U]\!]$ since $[\![U; \mu\alpha.\alpha]\!] \subseteq [\![U]\!] = [\![U; 0]\!] = [\![U; E]\!] = [\![U; (E\&\mathbf{0})]\!]$. Therefore, $[\![U; (\varphi_2)^{use}]\!] \subseteq [\![U]\!] \subseteq \Phi$ holds as required.

Lemma B.26 *If $\Gamma, x : (\mathbf{R}, U) \parallel \varphi \vdash \mathcal{E}[\mathbf{acc}^a(x)] : \delta$, then $\Gamma, x : (\mathbf{R}, U') \parallel \varphi \vdash \mathcal{E}[b] : \delta$ for some U' such that $U \leq a; U'$.*

Proof By structural induction on $\mathcal{E}$. We show a few representative cases below.

- Case $\mathcal{E} = [\,]$.

$$\mathcal{E}[\mathbf{acc}^a(x)] = \mathbf{acc}^a(x)$$

By Lemma B.9(4), there exist Γ_1, U_x and φ_1 such that $\Gamma_1, x : (\mathbf{R}, U_x) \parallel \varphi_1 \vdash x : (\mathbf{R}, a)$ and $\Gamma \leq_{\varphi_1} \Gamma_1$ and $U \leq U_x$ and $\varphi \sqsubseteq \varphi_1$.

By Lemma B.9(2), $\Gamma_1, x : (\mathbf{R}, U_x) \leq_{\mathbf{0}} (\mathbf{R}, \Diamond a)$ and $\varphi_1 \sqsubseteq \mathbf{0}$. Hence we have $\Gamma \leq_{\varphi_1} \Gamma_1 \leq_{\mathbf{0}} \emptyset$ and $U_x \leq \Diamond a$ and $\varphi \sqsubseteq \varphi_1 \sqsubseteq \mathbf{0}$.

On the other hand, by (T-Bool)

$$\emptyset \parallel \mathbf{0} \vdash b : \mathbf{bool}$$

Hence, by (T-Weak), $\Gamma_1 \leq_{\mathbf{0}} \emptyset$, $\varphi_1 \sqsubseteq \mathbf{0}$ and $\delta = \mathbf{bool}$,

$$\Gamma_1, x : (\mathbf{R}, \mathbf{0}) \parallel \varphi_1 \vdash b : \delta.$$

Therefore, by (T-Weak), $\Gamma \leq_{\varphi_1} \Gamma_1$ and $\varphi \sqsubseteq \varphi_1$, we have

$$\Gamma, x : (\mathbf{R}, \mathbf{0}) \parallel \varphi \vdash b : \delta.$$

Let $U' = \mathbf{0}$. Then $U \leq U_x \leq \Diamond a \leq a \equiv a; \mathbf{0} = a; U'$ finishes the case.

- Case $\mathcal{E} = v\mathcal{E}_1$.

$$\mathcal{E}[\mathbf{acc}^a(x)] = v\mathcal{E}_1[\mathbf{acc}^a(x)]$$

By Lemma B.9(8), there exist Γ_1, Γ_2, U_1, U_2, φ_1, φ_2, φ_3 and δ_1 such that $\Gamma \leq_{\varphi_1;\varphi_2;\varphi_3} \Gamma_1; \Gamma_2; (\varphi_3)^{use}$ and $U \leq U_1; U_2; (\varphi_3)^{use}$ and $\varphi \sqsubseteq \varphi_1; \varphi_2; \varphi_3$ and

$$\begin{array}{l} \Gamma_1, x : (\mathbf{R}, U_1) \parallel \varphi_1 \vdash v : (\delta_1 \xrightarrow{\varphi_3} \delta, 1) \\ \Gamma_2, x : (\mathbf{R}, U_2) \parallel \varphi_2 \vdash \mathcal{E}_1[\mathbf{acc}^a(x)] : \delta_1. \end{array}$$

By Lemma B.15, (T-Weak), we have U_1' such that

$$\begin{array}{c} U_1 \leq \Diamond U_1' \\ \Gamma_1, x : (\mathbf{R}, \Diamond U_1') \parallel \varphi_1 \vdash v : (\delta_1 \xrightarrow{\varphi_3} \delta, 1). \end{array}$$

By the induction hypothesis, we have U_2' such that

$$\begin{array}{c} U_2 \leq a; U_2' \\ \Gamma_2, x : (\mathbf{R}, U_2') \parallel \varphi_2 \vdash \mathcal{E}[b] : \delta_1. \end{array}$$

Hence, by using (T-App) and (T-Weak), we get

$$\Gamma, x : (\mathbf{R}, \Diamond U_1'; U_2'; (\varphi_3)^{use}) \parallel \varphi \vdash v\mathcal{E}[b] : \delta$$

Here, by Lemma B.1(3,4) we have

$$\begin{array}{rcl} U & \leq & U_1; U_2; (\varphi_3)^{use} \leq (\Diamond U_1'); (a; U_2'); (\varphi_3)^{use} \equiv (\Diamond U_1') \otimes (a; U_2'; (\varphi_3)^{use}) \\ & \leq & ((\Diamond U_1') \otimes a); (U_2'; (\varphi_3)^{use}) \equiv (a \otimes (\Diamond U_1')); (U_2'; (\varphi_3)^{use}) \\ & \equiv & a; (\Diamond U_1'); U_2'; (\varphi_3)^{use}. \end{array}$$

So, $U' = \Diamond U_1'; U_2'; (\varphi_3)^{use}$ satisfies the required condition. This finishes the case.

Proof of Lemma 4.4. By a case analysis on the reduction rule used. We show a few representative cases below.

- Case (R-New).

$$M = \mathcal{E}[\mathbf{new}^{\Phi}()]$$
$$H' = H \uplus \{z \mapsto \Phi\} \text{ , } z \text{ is a fresh variable} \quad M' = \mathcal{E}[z]$$

By $\varphi \vdash (H, \mathcal{E}[\mathbf{new}^{\Phi}()]) : \delta$, there exist $x_1, \ldots, x_n$ and $U_1, \ldots, U_n$ such that

$$dom(H) = \{x_1, \ldots, x_n\}$$
$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n) \parallel \varphi \ \vdash \mathcal{E}[\mathbf{new}^{\Phi}()] : \delta$$
$$[\![U_1]\!] \subseteq H(x_1), \ldots, [\![U_n]\!] \subseteq H(x_n).$$

By Lemma B.25, there exists U_z such that

$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n), z : (\mathbf{R}, U_z) \parallel \varphi \ \vdash \mathcal{E}[z] : \delta$$
$$[\![U_z]\!] \subseteq \Phi$$

Hence, we have

$$dom(H') = \{x_1, \ldots, x_n, z\}$$
$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n), z : (\mathbf{R}, U_z) \parallel \varphi \ \vdash \mathcal{E}[z] : \delta$$
$$[\![U_1]\!] \subseteq H'(x_1), \ldots, [\![U_n]\!] \subseteq H'(x_n) \quad [\![U_z]\!] \subseteq \Phi = H'(z).$$

Therefore $\varphi \vdash (H \uplus \{z \mapsto \Phi\}, \mathcal{E}[z]) : \delta$ holds.

- Case (R-Acc).

$$H = H_1 \uplus \{x \mapsto \Phi\} \quad M = \mathcal{E}[\mathbf{acc}^a(x)]$$
$$H' = H_1 \uplus \{x \mapsto \Phi^{-a}\} \quad M' = \mathcal{E}[b]$$

By $\varphi \vdash (H_1 \uplus \{x \mapsto \Phi\}, \mathcal{E}[\mathbf{acc}^a(x)]) : \delta$, there exist $x_1, \ldots, x_n$ and $U_1, \ldots, U_n$, U_x such that

$$dom(H) = \{x_1, \ldots, x_n, x\}$$
$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n), x : (\mathbf{R}, U_x) \parallel \varphi \ \vdash \mathcal{E}[\mathbf{acc}^a(x)] : \delta$$
$$[\![U_1]\!] \subseteq H(x_1), \ldots, [\![U_n]\!] \subseteq H(x_n), [\![U_x]\!] \subseteq H(x) = \Phi$$

By Lemma B.26, there exist U'_x and b such that

$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n), x : (\mathbf{R}, U'_x) \parallel \varphi \ \vdash \mathcal{E}[b] : \delta$$
$$U_x \leq a; U'_x$$

From Lemma B.1(21), we have $[\![U'_x]\!] \subseteq [\![U_x]\!]^{-a}$. Hence, we have

$$dom(H) = \{x_1, \ldots, x_n, x\}$$
$$x_1 : (\mathbf{R}, U_1), \ldots, x_n : (\mathbf{R}, U_n), x : (\mathbf{R}, U'_x) \parallel \varphi \ \vdash \mathcal{E}[b] : \delta$$
$$[\![U_1]\!] \subseteq H'(x_1), \ldots, [\![U_n]\!] \subseteq H'(x_n),$$
$$[\![U'_x]\!] \subseteq [\![U_x]\!]^{-a} = \Phi^{-a} = (H_1 \uplus \{x \mapsto \Phi^{-a}\})(x) = H'(x)$$

Therefore $\varphi \vdash (H_1 \uplus \{x \mapsto \Phi^{-a}\}, \mathcal{E}[b]) : \delta$ holds.

- Case (R-Try-Rai).

$$M = \mathcal{E}[\mathbf{try}\ \mathcal{E}^{\overline{try}}[\mathbf{raise}]\ \mathbf{with}\ M_1]$$
$$H' = H \quad M' = \mathcal{E}[M_1]$$

So, it suffices to show

$$\Gamma \parallel \varphi \vdash \mathcal{E}[\mathbf{try}\ \mathcal{E}^{\overline{try}}[\mathbf{raise}]\ \mathbf{with}\ M_1] : \delta$$

implies

$$\Gamma \parallel \varphi \vdash \mathcal{E}[M_1] : \delta$$

by induction on the structure of $\mathcal{E}$. Since the induction step is straightforward, we show only the base case, where $\mathcal{E} = [\,]$.

The following is assumed:

$$\Gamma \parallel \varphi \vdash \mathbf{try}\ \mathcal{E}^{\overline{try}}[\mathbf{raise}]\ \mathbf{with}\ M_1 : \delta.$$

By Lemma B.9(11), there exist $\Gamma_1, \Gamma_2, \varphi_1, \varphi_2$ and δ' such that $\Gamma \leq \Gamma_1 ;_E \Gamma_2$ and $\varphi \sqsubseteq \varphi_1 ;_E \varphi_2$ and $\delta' \leq \delta$ and

$$\Gamma_1 \parallel \varphi_1 \vdash \mathcal{E}^{\overline{try}}[\mathbf{raise}] : \delta'$$
$$\Gamma_2 \parallel \varphi_2 \vdash M_1 : \delta'$$

It follows that $\forall x \in dom(\Gamma_1).(Use_E(\Gamma_1(x)) \leq E)$ and $\varphi_1 \sqsubseteq E$ from Lemma B.10(2).

Hence, we have $\Gamma \leq \Gamma_1 ;_E \Gamma_2 \leq \Gamma_2$, and $\varphi \sqsubseteq (\varphi_1 ;_E \varphi_2) \sqsubseteq \varphi_2$ by Lemma B.1(10) and Definition 3.7.

So, by (T-Weak), we get $\Gamma \parallel \varphi \vdash M_1 : \delta$.

- Case (R-App).

$$M = \mathcal{E}[\mathbf{fun}(f, x, M_1)v]$$
$$H' = H \quad M' = \mathcal{E}[[\mathbf{fun}(f, x, M_1)/f, v/x]M_1]$$

So, it suffices to show

$$\Gamma \parallel \varphi \vdash \mathcal{E}[\mathbf{fun}(f, x, M_1)v] : \delta$$

implies

$$\Gamma \parallel \varphi \vdash \mathcal{E}[[\mathbf{fun}(f, x, M_1)/f, v/x]M_1] : \delta$$

by induction on the structure of $\mathcal{E}$. Since the induction step is straightforward, we show only the base case, where $\mathcal{E} = [\,]$.

The following is assumed in the case $\mathcal{E} = [\,]$:

$$\Gamma \parallel \varphi \vdash \mathbf{fun}(f, x, M_1)v : \delta.$$

By Lemma B.9(8), there exist Γ_1, Γ_2, $\varphi_1, \varphi_2, \varphi_3$, δ_1 and δ_2 such that

$$\Gamma \leq \Gamma_1 ; \Gamma_2 ; (\varphi_3)^{use} \tag{18}$$
$$\varphi \sqsubseteq \varphi_1 ; \varphi_2 ; \varphi_3 \tag{19}$$
$$\Gamma_1 \parallel \varphi_1 \vdash \mathbf{fun}(f, x, M_1) : (\delta_1 \xrightarrow{\varphi_3} \delta_2, 1) \tag{20}$$
$$\Gamma_2 \parallel \varphi_2 \vdash v : \delta_1 \tag{21}$$
$$\delta_2 \leq \delta. \tag{22}$$

By Lemma B.15 and (20), there exists Γ_1' such that $\Gamma_1 \leq \diamond\Gamma_1'$ and

$$\diamond\Gamma_1' \parallel \varphi_1 \vdash \mathbf{fun}(f, x, M_1) : (\delta_1 \xrightarrow{\varphi_3} \delta_2, 1) \tag{23}$$

By (23) and Lemma B.9(7), there exist U_{f1}, U_{f2}, $\sigma_{f1}, \delta_{f1}, \delta_{f2}$ and $\Gamma_{M_1}, \varphi_{M_1}$ such that

$$\varphi_1 \sqsubseteq \mathbf{0} \tag{24}$$
$$\Gamma_{M_1}, f : (\delta_{f1} \xrightarrow{\varphi_{M_1}} \delta_{f2}, U_{f2}), x : \sigma_{f1} \parallel \varphi_{M_1} \vdash M_1 : \delta_{f2} \tag{25}$$
$$\delta_{f1} \leq (\sigma_{f1} \setminus E) \tag{26}$$
$$(\delta_{f1} \xrightarrow{\varphi_{M_1}} \delta_{f2}, U_{f1}) \leq (\delta_1 \xrightarrow{\varphi_3} \delta_2, 1) \tag{27}$$
$$\diamond\Gamma_1' \leq_{\mathbf{0}} \Delta^{\mathbf{fun}}_{(U_{f1}, (U_{f2} \setminus E), \diamond(\Gamma_{M_1} \setminus E))} \tag{28}$$

By (27), we have $\delta_1 = \delta_{f1} \leq \sigma_{f1} \setminus E$, $\delta_{f2} = \delta_2$, $\varphi_3 \sqsubseteq \varphi_{M_1}$ and $U_{f1} \leq 1$.
By (25),(26) and (T-Fun),

$$\Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))} \parallel \mathbf{0} \vdash \mathbf{fun}(f, x, M_1) : (\delta_{f1} \xrightarrow{\varphi_{M_1}} \delta_{f2}, U_{f2} \setminus E)$$

By (T-Weak),

$$\Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))}, x : \mathbf{0}.\sigma_{f1} \parallel \mathbf{0} \vdash \mathbf{fun}(f, x, M_1) : (\delta_{f1} \xrightarrow{\varphi_{M_1}} \delta_{f2}, U_{f2} \setminus E) \tag{29}$$

From Lemma B.12 and (25), it follows that

$$(\Gamma_{M_1} \setminus E) \otimes (\varphi_{M_1})^{use}, f : (\delta_{f1} \xrightarrow{\varphi_{M_1}} \delta_{f2}, U_{f2}), x : \sigma_{f1} \parallel \varphi_{M_1} \vdash M_1 : \delta_{f2} \tag{30}$$

By Lemma B.24, (29) and (30),

$$\begin{array}{c} \Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))} \otimes ((\Gamma_{M_1} \setminus E) \otimes (\varphi_{M_1})^{use}), x : \sigma_{f1} \otimes \mathbf{0} \parallel \varphi_{M_1} \\ \vdash [\mathbf{fun}(f, x, M_1)/f]M_1 : \delta_{f2} \end{array} \tag{31}$$

Here, we have

$$(\diamond\Gamma_1'; (\varphi_{M_1})^{use}), x : \sigma_{f1} \leq_{\varphi_{M_1}} (\Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))} \otimes ((\Gamma_{M_1} \setminus E) \otimes (\varphi_{M_1})^{use})), x : \sigma_{f1}$$

by

$$\begin{array}{lll} & (\diamond\Gamma_1'; (\varphi_{M_1})^{use}), x : \sigma_{f1} & \\ \leq_{\varphi_{M_1}} & (\Delta^{\mathbf{fun}}_{(U_{f1},(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))}; (\varphi_{M_1})^{use}) & \left(\text{by } \begin{array}{l}\text{Lemma B.13}\\ \text{and (28)}\end{array}\right) \\ \leq & (\Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))} \otimes (\diamond(\Gamma_{M_1} \setminus E))); (\varphi_{M_1})^{use} & \left(\text{by } \begin{array}{l}\text{Lemma B.14}\\ \text{and } U_{f1} \leq 1\end{array}\right) \\ \leq & \Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))} \otimes (\diamond(\Gamma_{M_1} \setminus E); (\varphi_{M_1})^{use}) & \text{(by Lemma B.1(4))} \\ \leq & \Delta^{\mathbf{fun}}_{((U_{f2} \setminus E),(U_{f2} \setminus E),\diamond(\Gamma_{M_1} \setminus E))} \otimes (\diamond(\Gamma_{M_1} \setminus E) \otimes (\varphi_{M_1})^{use}) & \text{(by the definition of } \equiv) \end{array}$$

Therefore, by (T-Weak), we get

$$(\diamond\Gamma_1'); (\varphi_{M_1})^{use}, x : \sigma_{f1} \parallel \varphi_{M_1} \vdash [\mathbf{fun}(f, x, M_1)/f]M_1 : \delta_{f2}. \tag{32}$$

So, from $dom(\diamond\Gamma_1') = dom(\Gamma_1) = dom(\Gamma_2)$, Lemma B.24, (21), (32) and $\delta_1 \leq \sigma_{f1} \setminus E$, it follows that

$$((\diamond\Gamma_1'); (\varphi_{M_1})^{use}) \otimes \Gamma_2 \parallel \varphi_{M_1} \vdash [\mathbf{fun}(f, x, M_1)/f, v/x]M_1 : \delta_{f2}.$$

By Lemma B.1(4) and (T-Weak),

$$((\diamond\Gamma_1') \otimes \Gamma_2); (\varphi_{M_1})^{use} \parallel \varphi_{M_1} \vdash [\mathbf{fun}(f, x, M_1)/f, v/x]M_1 : \delta_{f2}. \tag{33}$$

By Lemma B.10 and (21), we have $\varphi_2 \sqsubseteq \mathbf{0}$. From $\varphi_1 \sqsubseteq \mathbf{0}$ and $\varphi_3 \sqsubseteq \varphi_{M_1}$, it follows that $\varphi_1; \varphi_2; \varphi_3 \sqsubseteq \varphi_{M_1}$.
By (T-Weak), (22) and (33), we have

$$((\diamond\Gamma_1') \otimes \Gamma_2); (\varphi_3)^{use} \parallel \varphi_1; \varphi_2; \varphi_3 \vdash [\mathbf{fun}(f, x, M_1)/f, v/x]M_1 : \delta$$

By $(\diamond\Gamma_1') \otimes \Gamma_2 \equiv (\diamond\Gamma_1'); \Gamma_2$, $\Gamma_1 \leq \diamond\Gamma_1'$, $\Gamma \leq \Gamma_1; \Gamma_2; (\varphi_3)^{use}$ and $\varphi \sqsubseteq \varphi_1; \varphi_2; \varphi_3$, we get

$$\Gamma \parallel \varphi \vdash [\mathbf{fun}(f, x, M_1)/f, v/x]M_1 : \delta.$$

as required.
□