Gradual Typing for Effect Handlers (Extended Version)*

MAX S. NEW, University of Michigan, USA
ERIC GIOVANNINI, University of Michigan, USA
DANIEL R. LICATA, Wesleyan University, USA

We present a gradually typed language, GrEff, with effects and handlers that supports migration from unchecked to checked effect typing. This serves as a simple model of the integration of an effect typing discipline with an existing effectful typed language that does not track fine-grained effect information. Our language supports a simple module system to model the programming model of gradual migration from unchecked to checked effect typing in the style of Typed Racket.

The surface language GrEff is given semantics by elaboration to a core language Core GrEff. We equip Core GrEff with an inequational theory for reasoning about the semantic error ordering and desired program equivalences for programming with effects and handlers. We derive an operational semantics for the language from the equations provable in the theory. We then show that the theory is sound by constructing an operational logical relations model to prove the graduality theorem. This extends prior work on embedding-projection pair models of gradual typing to handle effect typing and subtyping.

CCS Concepts: • **Software and its engineering** → **Functional languages**; • **Theory of computation** → *Operational semantics*; *Type structures*.

Additional Key Words and Phrases: gradual typing, effect handlers, graduality, operational semantics, logical relation

284

ACM Reference Format:
Max S. New, Eric Giovannini, and Daniel R. Licata. 2023. Gradual Typing for Effect Handlers (Extended Version). *Proc. ACM Program. Lang.* 7, OOPSLA2, Article 284 (October 2023), 92 pages. https://doi.org/10.1145/3622860

1 INTRODUCTION

Gradually typed programming languages are designed to support smooth migration from a lax to a strict static type discipline [Siek and Taha 2006; Tobin-Hochstadt and Felleisen 2008]. Most commonly, gradually typed languages add a static type system to an existing dynamically typed language and allow for (1) safe interoperability between the languages and (2) semantic guarantees that adding types to existing programs only results in stricter type enforcement, and no other behavioral change. More generally, gradual typing has been applied to provide a spectrum of precision in other kinds of typing disciplines such as refinement typing or effect typing [Bañados Schwerter et al. 2014; Lehmann and Tanter 2017], where the "dynamic" side is a statically typed language itself.

One particular presentation of effects and effect typing that is gaining popularity is **effect handlers** [Plotkin and Pretnar 2009]. Operationally, effect handlers are **resumable exceptions**,

*This material is based on research sponsored by the National Science Foundation under agreement number CCF-1909517

Authors' addresses: Max S. New, Computer Science and Engineering, University of Michigan, USA, maxsnew@umich.edu; Eric Giovannini, Computer Science and Engineering, University of Michigan, USA, ericgio@umich.edu; Daniel R. Licata, Mathematics and Computer Science, Wesleyan University, USA, dlicata@wesleyan.edu.

2475-1421/2023/10-ART284
https://doi.org/10.1145/3622860

code can "raise" an effect operation, which will then be handled by the closest enclosing handler, which in addition to the exception data will also receive the continuation for the raising code that can be invoked to resume at the original point where the effect was raised. Effect handlers provide an intuitive typed interface to delimited continuations, and can similarly be used to conveniently implement backtracking search, non-determinism, mutable state, and as a convenient interface to external system calls. Effect handlers have been implemented in a number of libraries and experimental languages, and more recently have been incorporated as a built-in feature into OCaml 5, and have been proposed as an extension to WASM [Brachthäuser et al. 2020; Contributors [n.d.]; Cooper et al. 2006; Kiselyov et al. 2013; Leijen 2014; Lindley et al. 2017; Sivaramakrishnan et al. 2021].

Designers of languages supporting effect handlers, much like designers of languages with exceptions, are left with a choice of whether the type system should merely validate that the input and output types of effect operations are respected, or if an *effect typing* system should be employed to determine that a particular effect can only be raised when the context is known to implement a handler for it. On the one hand, checked effects allow programmers to easily reason about which effects can be raised by subprocedures and ensure they are handled appropriately, rather than being caught by the runtime system and causing the program to crash. On the other hand, strict checking may necessitate large code changes when code is extended to raise new operations, and even in languages such as Java that support both checked and unchecked exceptions, unchecked exceptions are preferred in many scenarios. Furthermore, when adding effect typing to a language that does not already support it, even correct existing libraries may not typically pass the necessarily conservative static type checker. It may be infeasible to rewrite large amounts of existing library code to precisely track effect usage. Gradual typing provides a linguistic framework for designing languages where a programmer is not entirely locked in to one system or another: they might use unchecked exceptions in one module and checked exceptions in another, while supporting well-defined interoperability with useful error messages at runtime if there is an effect raised in a context where it is not expected. Further, a gradually typed language provides a path for gradually *migrating* code from less precise to more precise static type checking. This potential for gradual typing to be used in this way to incorporate effect typing disciplines into existing languages has been eloquently discussed in prior work by Phil Wadler [Wadler 2021].

In this work we present the design and semantics of GrEff, a gradual language with effect handlers that supports gradual migration from unchecked effects to precise effect typing. The untracked sublanguage of GrEff is designed to be similar to SML and Java's treatment of exceptions: new effect operations are declared with specified input and output types, and these can be imported and used to raise and handle those operations in other modules, but which effects are raised by a function is not tracked by the type system. In addition, GrEff supports *tracked* function types $A \rightarrow_{\sigma} B$ where the input values must be of type A, output values will be of type B, and the function may raise any of and only the effects in the set σ. The untracked function type is modeled then as a type $A \rightarrow_{?} B$ which has a "dynamic" effect type, in the sense that it may raise any effect, possibly including unknown effect operations declared in some independent module of the program. Since our main focus in this work is on providing a foundation for extending existing statically typed languages such as OCaml 5 with effect types, we have chosen not to support full dynamic typing in the design of GrEff. However, the design should easily accommodate supporting fully dynamic value typing in addition to the dynamic effect typing using standard gradual typing techniques.

In GrEff, new effect operations can be declared in each module, just as new exceptions can be declared in Java and ML-style languages. When an effect is declared in a module, it is given an associated *request* and *response* type. For instance, an effect for reading a boolean state would be `get : Unit --> Bool`, the user provides a trivial value as the request and receives a boolean value as the response, while an effect for writing to boolean state would be `set : Bool --> Unit`. Similar to ML and Java, GrEff takes a *nominal* approach to effect operations: each effect operation has an associated request and response type that are used to determine when an effect is properly raised or handled. However, having a single, global assignment from effect names to request/response types is problematic from the perspective of *gradual* migration from untracked to tracked effects. In a completely nominal form of effect typing, if an effect operation is used in many different modules with imprecise typing, and one module is migrated to use a more precise version of the effect's request/response type, then we would need to migrate all modules to use the more precise type. Instead, gradual migration should allow for this to be done a single module at a time. To achieve this, in GrEff, we take a *locally nominal* but *globally structural* approach to the typing of effect operations. That is, *locally*, within each module, the request and response type for an effect are fixed, and all `raise` and `handle` constructs are checked with the same typing. On the other hand, *globally*, different modules across the program can associate different types to the same effect operation. At module boundaries, i.e., imports and exports, modules are statically allowed to interoperate if they agree on the precisely typed portion of the effects they share. If one module is more precise than the other, then dynamic runtime monitoring is inserted in the implementation to ensure that the runtime behavior agrees with the static typing, raising an error if the dynamically typed code violates the imposed runtime type discipline.

There are two aspects in designing a *sound* gradually typed language: designing the syntax and gradual type checking of the surface language and designing the corresponding core language and semantics. The syntax should support a simple process for migrating from an imprecise to a precise style, satisfying the *static gradual guarantee* [Siek et al. 2015]. We designed the surface language with the goal of modeling program migration from dynamic to static effect typing. For this reason we include a simple module system in the style of Typed Racket [Tobin-Hochstadt and Felleisen 2008] so that we can express that different portions of the program have different views on how the effect operations are typed. Once the design of the base language is fixed, we design the gradual type checking using techniques from prior work to arrive at a gradual type system that satisfies the static gradual guarantee[Garcia et al. 2016; Siek and Taha 2006].

Next, the core language provides a definition for the runtime semantics. The semantics should admit useful type-based reasoning principles for precisely typed code, even in the presence of interaction with imprecisely typed components. Further, the aforementioned migration process should have a predictable impact on program semantics: migrating to more precise checking may result in new errors being identified (statically or dynamically), but otherwise should not impact program behavior, a property known as the *dynamic gradual guarantee* or *graduality* [New and Ahmed 2018; Siek et al. 2015]. To design the core language and runtime semantics, we follow the prior work ([New and Ahmed 2018; New et al. 2020, 2019]) which established a recipe for designing a new gradual core language to satisfy the graduality theorem and validate strong type-based equational reasoning principles. Their approach is to *axiomatize* the type-based reasoning principles as equations and the graduality theorem as inequalities, where casts are defined not by specifying their operational behavior *a priori* but instead by assuming they are given by least upper bounds/greatest lower bounds. Then the operational behavior of the casts can be *derived* from the inequational theory. An operational or denotational model must then be constructed to prove the theory is consistent, which implies the graduality theorem. But since the operational semantics is *derived* from the inequational theory, this also establishes a stronger theorem that the observable behavior of the casts is *uniquely determined* by the desired type-based reasoning and graduality, showing that any observably different cast semantics must violate one or more of the axioms.

For designing our core language, called Core GrEff, we extend this recipe, which previously been demonstrated on simple and polymorphic types, to apply also to *effect casts* and *subtyping* of value and effect types. We then show that every rule of an operational semantics is derivable from the least upper bound/greatest lower bound specifications of casts as well as congruence rules and an *effect forwarding* principle for handlers. The effect forwarding principle states that a handler clause that simply re-raises the effect it handles with the same continuation can be removed without changing the observable behavior of the system, an intuitive principle as well as a highly desirable compiler optimization.

In this work, we extend prior step-indexed logical relations models for proving graduality to handle effects and subtyping, by showing that the runtime casts satisfy the properties of being *embedding-projection pairs* [New and Ahmed 2018]. In doing so, we show how to combine effect and value embedding-projection pairs within the same system, and how they interact. Additionally, we identify new semantic principles for the interaction between subtyping and runtime casts.

The contributions of the paper are as follows:

(1) We define a gradually typed language GrEff supporting migration from unchecked to checked effects and handlers.
(2) We prove this language satisfies the static gradual guarantee and the dynamic gradual guarantee (graduality).
(3) We give the language a semantics by elaboration into a core language, core GrEff.
(4) We axiomatize the desired graduality and program equivalence properties of the core language by giving an inequational theory. We then derive from this an operational semantics by orienting certain equations in the theory, showing that the operational behavior is derivable from the graduality and extensionality principles.
(5) We prove type soundness and graduality by constructing a logical relations model, extending prior work on embedding-projection pair semantics to effects and subtyping.

2 OVERVIEW OF GREFF

Before discussing the syntax and semantics of GrEff, we provide an informal introduction to its features and how it supports a gradual migration from unchecked to checked effect handlers. As an example, consider the implementation of a simple threading library using effect handlers. We start with a system using unchecked effect types in an ASCII syntax in Figure 1. We split this program across three modules: first, a module `Operations` defines the effects we will be using in our other modules. These are the effects that the threads use: `print` for displaying output so that we can observe the interleaving of threads, `yield`, which yields back control to the scheduler, and most importantly, `fork`, which allows for a thread to spawn new threads. Each effect declaration `effect e : Req --> Resp` is annotated with two types: the type of *requests* to the ambient handler, and the type of expected *responses* from the ambient handler. For instance, the request type for print is a `string` to be printed, and the response is unit. In a more realistic setting, the response type might be a boolean to say if the printing succeeded, or an unsigned integer to say how many bytes were successfully printed. For yield, the request and response are both unit. For fork, the response type is again unit and the request type is a thunk `1 -[?]-> 1` where the ? is the type of *effects* the function may raise when called. In this case, ? indicates the thunk might raise any effect.

Next, module `Scheduler` defines a scheduler as a handler for the provided effects. For simplicity the implementation relies on some built-in list implementation, and shallow handlers, a simple extension to our formalism which uses the more complex deep handlers. The scheduler loop takes a queue of threads, represented as thunks, and runs them in a round-robin fashion, taking in a string consisting of everything printed so far and returning a final string that contains everything printed by running the threads. If there are no threads in the queue, the scheduler returns the string unchanged. Otherwise, it pops off the first thunk in the queue and executes it, handling effects

```
module Operations where
  effect print : str --> 1
  effect yield : 1 --> 1
  effect fork  : (1 -[?]-> 1) --> 1
module Scheduler where
  import Operations.print : str --> 1
  import Operations.yield : 1 --> 1
  import Operations.fork  : (1 -[?]-> 1) --> 1
  define sch-loop : List (1 -[?]-> 1) -[?]-> str -[?]-> str = lambda q.
    match q with
      empty           -> lambda s. s
      cons(thunk, q') -> shallow-handle thunk() with
        ret _ -> sch-loop q'
        print(s, k) -> lambda s'. sch-loop (cons k q') (s ++ s')
        yield(_, k) -> sch-loop (snoc q' k)
        fork(new,k) -> sch-loop (cons k (snoc q' new))
  define scheduler : (1 -[?]-> 1) -[?]-> str = lambda thunk.
    sch-loop (cons thunk empty) ""
module Main where
  import Operations.print : str --> 1
  import Operations.yield : 1 --> 1
  import Operations.fork  : (1 -[?]-> 1) --> 1
  import Scheduler.scheduler : (1 -[?]-> 1) -[?]-> str
  define letters : 1 -[?]-> 1 =
    print("a"); yield(); print("b"); ()
  define numbers : 1 -[?]-> 1 =
    print("1"); fork(letters); print("2"); ()
  define main: 1 -[?]-> str =
    scheduler(numbers)
```

Fig. 1. GrEff Threading Program with Imprecise Types

that it raises. The `ret` clause handles the case that the thread terminates without performing any effects, in which case, the scheduler executes the remaining threads in the queue. In the `print(s,k)` clause, the `s` parameter is the `str` to be printed by the thread, and the `k` is the continuation for the program point where the `print` was raised. The scheduler handles this case by taking in the string accumulator, appending the printed string to the back of it, and continuing the scheduling with the continuation `k` at the front of the queue. In the `yield(_,k)` clause, the scheduler continues with the continuation at the back of the queue. Finally, in the `fork(new,k)` clause, the scheduler continues with the continuation thread `k` at the front of the queue and the new thread `new` at the back of the queue. Then this loop is run by a wrapper `scheduler` function which calls the scheduler loop with a singleton queue and an initial empty string accumulator.

Finally, we have the `Main` module, which uses the scheduler defined in the `Scheduler` module with a thunk that uses the effects defined in the `Operations` to implement a program that prints a simple message using threads whose output will depend on the scheduler's behavior.

The imprecision of the effect typing in this program means that programmers have to rely on documentation or understanding of the code to understand what effects might be raised when

```
module Operations where
  effect print : str --> 1
  effect yield : 1 --> 1
  effect fork  : (1 -[fork,print,yield]-> 1) --> 1
module Scheduler where
  import Operations.print : str --> 1
  import Operations.yield : 1 --> 1
  import Operations.fork  : (1 -[fork,print,yield]-> 1) --> 1
  define sch-loop : List (1 -[fork,print,yield]-> 1) -[]-> str -[]-> str = ...
  define scheduler : (1 -[fork,print,yield]-> 1) -[]-> str = ...
module Main where
  import Operations.print : str --> 1
  import Operations.yield : 1 --> 1
  import Operations.fork  : (1 -[fork,print,yield]-> 1) --> 1
  import Scheduler.scheduler : (1 -[fork,print,yield]-> 1) -[]-> str
  define letters : 1 -[print,yield]-> 1 =
    print("a"); yield(); print("b"); ()
  define numbers : 1 -[fork,print]-> 1 =
    print("1"); fork(letters); print("2"); ()
  define main: str =
    scheduler(numbers)
```

Fig. 2. GrEff Threading Program with Precise Typing

they import a function from another module. With effect typing, this information can be expressed precisely using effect annotations on the functions themselves. For instance, in the declaration of the `fork` operation, the request is a thunk that when launched as a thread itself may raise further effects such as manipulating shared state, `yield`ing to other threads, or `fork`ing additional threads. However with imprecise effect tracking, the `scheduler` procedure has the uninformative type `(1 -[?]-> 1) -[?]-> 1` so we cannot specify in the type which operations the scheduler will handle and which it will propagate forward.

GrEff allows as well for the introduction of *precise* effect types to express these choices in the type structure. In figure 2, we show a fully precisely typed version of the same threading program (with implementations, which are unchanged, now elided). This allows us to specify in the `Scheduler` module that the scheduler expects threads that can (1) `print` a string, (2) `yield` to the other threads and (3) `fork` further threads with the same effects. To express this, the scheduler module changes the type to `(1 -[fork,print,yield]-> 1) -[]-> str` expressing that the scheduler will be passed a thunk that may `fork`, `print` or `yield`, but will itself return a string without raising any effects. Additionally, we can express that *fork*ed threads should only raise these three effects as well. This is expressed by annotating the *import* statement, which defines fork as a recursive[1] effect type whose response type is trivial and whose request type is that of thunks that can raise the three provided effects. This typing will then be used by all occurrence of the fork effect, in raise or handlers, within this module. The types are also changed in the main module, where the `letters` thunk can be given a type expressing it only prints and yields, whereas `numbers` thunk only forks

[1]though recursive effect types are natural here, we do not support them in our core language and leave this extension to future work

and prints. These are compatible with the types in scheduler using an effect subtyping that allows functions that use fewer effects to be used in a context that can handle more.

Since GrEff is a *gradual* effect language, a programmer who started with the imprecise program does not need to fully type the entire program before running it. Instead, the programmer can *gradually* migrate from the imprecise style to the more precise style, for example one module at a time. In fact, any of the $2^3 = 8$ combinations of the imprecise versions and precise versions of the three modules presented here will pass the GrEff gradual type-and-effect checker. For instance, we might start with adding precise effect typing to the `Operations` module to specify the effects that a forked thread can have. Whereas in a non-gradual type system, this would require changing the consumer modules to use the more precise typing, in GrEff, the import statements allow for the uses within the module to continue to use the imprecise typing, and at the module boundary it is checked that the precise components of the declared type for the `fork` effect match the precise components of the declaration in the defining module. On the other hand, we can keep the `Operations` module imprecisely typed, and instead add typing to the `Scheduler` module first. This is again unusual compared to a conventional typed language, we have declared a nominal effect type in one module, but used it at a different type in a client module. The import statements allow for the gradual migration of the client code without changing the original library.

The module system plays a crucial role in allowing for the programmer to independently choose between migrating the declaration site of the nominal effects and its uses. If we were in a purely expression-oriented language, then any change to the effect declaration, even in a gradual language, would change the typing of all uses of the effect. Here we use the module boundaries in the style of Typed Racket as a way to formally specify different expectations of what the type of the nominal effect operations should be in different portions of the codebase. This design fixes the types of the effects within a module, in keeping with the common nominal type system for exceptions in the ML family of languages. An advantage of this design is that it is clear to the programmer at all times what the type of an effect is in an expression. Further, this makes it clear what migration of effect types means: the programmer can independently change the precision of the effect types for each module one at a time, and there is never any confusion about what the "current types" of an effect is.

However note that it is not the case that the only gain or loss of precision happens at module boundaries. Within a module, gradual type casts of function values can occur. For instance, if you pass a value of type `A -[ ? ]-> B` to a function that expects an input of type `A -[ fork ]-> B` then a downcast will be inserted to ensure only fork effects are raised.

3 SURFACE AND CORE GREFF

In this section, we introduce the syntax and typing of GrEff along with its elaboration into a core language, Core GrEff. GrEff includes a module system and nominal effect operations, as well as a gradual type checking algorithm that allows for a mix of dynamic and static effect tracking. Core GrEff, on the other hand, is a simpler expression language with a declarative type system where all gradual type casts (but not subtyping) are explicit in the term. The high-level features of GrEff are elaborated away into core GrEff. Because Core GrEff is simpler, we describe its syntax and typing first, and then describe GrEff and its type-checking/elaboration algorithm.

3.1 Syntax and Typing of Core GrEff

We give an overview of the Core GrEff syntax in Figure 3. Core GrEff expression syntax include typical lambda calculus syntax for variables, let-bindings, functions and booleans. Additionally, there is a term $\mho$ that represents a runtime error produced by a failed cast. Next, it includes forms for raising an effect operation $\texttt{raise}\ \varepsilon(M)$ and handling effect operations $\texttt{handle}\ M\,\{\texttt{ret}\ x.N \mid \phi\}$.

We use ε to stand for an element of some fixed countable set of effect names. The handler includes a clause `ret` $x.N$ to handle a return value for M as well as clauses for handling effects ϕ. Abstracting from syntactic details, ϕ is modeled as a finitely supported partial function (written $\rightharpoonup_{\text{fin}}$) from effect names to terms, which all have two free variables x and k for the payload of the effect raised and its continuation. That is, if syntactically a handler has a clause $\varepsilon(x, k) \mapsto N_\varepsilon$, we model this by having $\phi(\varepsilon) = N_\varepsilon$. Next, Core GrEff includes four explicit gradual type cast forms: downcasts ($\langle A \twoheadleftarrow B \rangle M$) and upcasts ($\langle B \leftarrowtail A \rangle M$) for value types, as well as analogous casts for effect types ($\langle \sigma \twoheadleftarrow \tau \rangle M$ and $\langle \tau \leftarrowtail \sigma \rangle M$).

The value types A, B, C classify runtime values: in this simple calculus, just booleans and functions, where functions are typed with respect to a domain, codomain as well as an effect type σ which classifies what effects the function may raise when it is called. The effect types are either ? to indicate dynamically tracked effects, or a concrete effect type. A concrete effect type says which effect names ε can be raised, and when they are raised, what is the type of the request A the raising party provides and what is the type of responses B with which the handling party can resume. Abstracting from syntactic details, this is defined to be a finitely supported partial mapping from names to pairs of value types (i.e., an element of the Cartesian product $\text{ValueType}^2 = \text{ValueType} \times \text{ValueType}$). To model that an effect ε can be raised with request type A and response type B we would define $\sigma_c(\varepsilon) = (A, B)$, which we will notate more suggestively as $\varepsilon : A \rightsquigarrow B \in \sigma_c$. As shown in Section 2, programs declare which effect names can be used, and with which associated request and response types. To track this information in typing core GrEff expressions, we type check all GrEff expressions against a *Signature* Σ which associates a pair of *non-tracking* types to each name. By a *non-tracking* type $A_?$, we mean a value type that only use ? effect types. Additionally, expressions are type-checked with respect to an ordinary typing context Γ. Finally, we define typical notions of value and evaluation context to encode a call-by-value, left-to-right evaluation order. Most notably, all casts are evaluation contexts, and function casts are values, i.e. "proxies" that delay type enforcement until an application is performed.

The use of non-tracking types in the signature is a design decision in the semantics of GrEff: it means that when an effect is declared in a module, it fully specifies only the non-effect typing portions of the request and response types. When a module imports an effect, it is only checked that the new request and response type are *consistent* with the exporting module. Since effect types can be re-exported and the consistency relation is not transitive, this means that in general the types used in one module will not be consistent with those of the module where it was originally declared. However, transitive closure of consistency[2] *does* ensure that the types have the same non-tracking portion, and so it is sensible to define the valid instances of the effect type to be any that agree on this non-tracking portion of the type. An alternative would be for the signature to have a fully specified type and limit all uses of the effect to be at least as precise as the original declaration. However we argue that this is not in the spirit of gradual typing: for instance it might be the case that module P **p**rovides an effect declaration, module I is an **i**ntermediate that re-exports the effect and module C is a **c**lient of I that uses the effect but does not directly interact with P. Say P, I, C all initially use untracked effects, but then C becomes typed and so specifies precise effect typing for the effect. The program functions properly and eventually P is additionally made more precise but in such a way that the effect implementation is incompatible with the usage in C. In GrEff this does not lead to a static error, because C and P are not directly communicating along a precisely typed interface, but rather through an intermediary I that uses imprecise typing. Indeed, it may be the case that I uses the effect differently between C and P and there is no runtime type

[2]Note that in a gradual language with a dynamic type, the transitive closure of consistency is the total relation, but because there is no dynamic value type the relation here is non-trivial.

$$
\begin{array}{rcl}
\text{Terms } M, N & ::= & x \mid \lambda x.M \mid M\,M' \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}\ M\{N\}\{N\} \\
 & & \mid \mathtt{let}\ x = M\ \mathtt{in}\ N \mid \mathtt{raise}\ \varepsilon(M) \mid \mathtt{handle}\ M\,\{\mathtt{ret}\ x.N \mid \phi\} \\
 & & \mid \langle B \leftarrowtail A\rangle M \mid \langle A \twoheadleftarrow B\rangle M \mid \langle \tau \leftarrowtail \sigma\rangle M \mid \langle \sigma \twoheadleftarrow \tau\rangle M \mid \mho \\
\text{Handler clause } \phi & \in & \text{Name} \rightharpoonup_{\text{fin}} \text{Term} \\
\text{Value Types } A, B, C & ::= & A \to_{\sigma} B \mid \mathtt{bool} \\
\text{Effect Types } \sigma, \tau & ::= & ? \mid \sigma_c \\
\text{Concrete Effect Types } \sigma_c & \in & \text{Name} \rightharpoonup_{\text{fin}} \text{ValueType}^2 \\
\text{Signature } \Sigma & \in & \text{Name} \rightharpoonup_{\text{fin}} \text{NonTrackingType}^2 \\
\text{Non-tracking Types } A_? & ::= & A_? \to_? A_? \mid \mathtt{bool} \\
\text{Typing Contexts } \Gamma & ::= & \cdot \mid \Gamma, x : A \\
\text{Values } V & ::= & x \mid \lambda x : A.M \mid \mathtt{true} \mid \mathtt{false} \\
 & & \mid \langle A \to_{\sigma} B \leftarrowtail A' \to_{\sigma'} B'\rangle V \mid \langle A' \to_{\sigma'} B' \twoheadleftarrow A \to_{\sigma} B\rangle V \\
\text{Evaluation Context } E & ::= & \bullet \mid \langle B \leftarrowtail A\rangle E \mid \langle A \twoheadleftarrow B\rangle E \mid \langle \tau \leftarrowtail \sigma\rangle E \mid \langle \sigma \twoheadleftarrow \tau\rangle E \\
 & & \mid \mathtt{raise}\ \varepsilon(E) \mid \mathtt{handle}\ E\,\{\mathtt{ret}\ x.N \mid \phi\} \mid E\,M \mid V\,E \\
 & & \mid \mathtt{if}\ E\{N_t\}\{N_f\} \mid \mathtt{let}\ x = E\ \mathtt{in}\ N
\end{array}
$$

Fig. 3. Core GrEff Syntax

error. However, if I becomes precisely typed, it must specify its interpretation of the effect and will result in a static error with either C or P.

Next, we present *declarative* term typing rules in Figure 4. The main judgment $\Sigma \mid \Gamma \vdash_{\sigma} M : A$ says that under the assumptions Γ, M can raise effects drawn from σ, and produce a final value of type A. We follow the convention that whenever we form the judgment $\Sigma \mid \Gamma \vdash_{\sigma} M : A$ we must already have established that the types in Γ, A, σ are well-formed under the signature Σ. First, we include a subsumption rule for value and effect subtyping, which we will soon define. The rules for value forms (variable, booleans, and lambdas) all have an arbitrary effect type σ because they do not raise any effects themselves. The runtime cast error $\mho$ can be given any value or effect type. The let, application and if rules simply require that all the sub-terms use the same effect type, though subsumption can be used to combine effects. The raise rule says that the effect being raised needs to be in the current effect type and the payload of the request must also have the same effect type.

Next, the rule for typing a handler works as follows. First, the output value type is B and output effect type is τ, while for the scrutinee M the corresponding types are A and σ. First, we check that the return clause N has the same output types as the handler overall, when its input x has the type of the output of M. Next, for each effect operation $\varepsilon : A_{\varepsilon} \rightsquigarrow B_{\varepsilon}$ raised by M, either the effect is not handled by ϕ, in which case it must be included in the final effect type, or it is handled by ϕ. If it is handled by ϕ, then the clause $\phi(\varepsilon)$ must be well typed with a request value $x : A_{\varepsilon}$ and a continuation that takes responses and has output effect and value types that match the term overall $k : B_{\varepsilon} \to_{\tau} B$. Lastly, we include the rules for type and effect upcasts and downcasts. Whenever a *type precision* relationship $A \sqsubseteq B$ holds (to be defined), we get an *up*cast from the more precise type A to the more imprecise type B and a corresponding downcast from B to A.

Finally, finishing out the syntax, in Figure 5, we define three judgments on types: well-formedness, subtyping and type precision. Well-formedness $\Sigma \vdash A$ and $\Sigma \vdash \sigma$ checks that the types used in effect operations erase to the types associated in the signature. Here we use the notation $|A|$ to mean the erasure of effect typing information in that we replace any effect type subterms σ with dynamic ?. Subtyping works as usual for booleans and functions, contravariant in domain of the function type, but covariant in the codomain and effect. Subtyping for effect types includes both a *width* subtyping aspect: a smaller type can raise fewer operations, as well as a *depth* aspect that is

$$\frac{\Sigma \mid \Gamma \vdash_\sigma M : A \qquad \Sigma \mid \Gamma \vdash A \leq B \qquad \sigma \leq \tau}{\Sigma \mid \Gamma \vdash_\tau M : B} \qquad \frac{\Gamma(x) = A}{\Sigma \mid \Gamma \vdash_\sigma x : A} \qquad \Sigma \mid \Gamma \vdash_\sigma \mho : A$$

$$\Sigma \mid \Gamma \vdash_\sigma \texttt{true}, \texttt{false} : \texttt{bool} \qquad \frac{\Sigma \mid \Gamma, x : A \vdash_\tau M : B}{\Sigma \mid \Gamma \vdash_\sigma \lambda x.M : A \to_\tau B} \qquad \frac{\begin{array}{c}\Sigma \mid \Gamma \vdash_\sigma M : A \\ \Sigma \mid \Gamma, x : A \vdash_\sigma N : B\end{array}}{\Sigma \mid \Gamma \vdash_\sigma \texttt{let}\ x = M\ \texttt{in}\ N : B}$$

$$\frac{\begin{array}{c}\Sigma \mid \Gamma \vdash_\sigma M : A \to_\sigma B \\ \Sigma \mid \Gamma \vdash_\sigma N : A\end{array}}{\Sigma \mid \Gamma \vdash_\sigma M\,N : B} \qquad \frac{\begin{array}{c}\Sigma \mid \Gamma \vdash_\sigma M : \texttt{bool} \\ \Sigma \mid \Gamma \vdash_\sigma N_t : B \qquad \Sigma \mid \Gamma \vdash_\sigma N_f : B\end{array}}{\Sigma \mid \Gamma \vdash_\sigma \texttt{if}\ M\{N_t\}\{N_f\} : B} \qquad \frac{\Sigma \mid \Gamma \vdash_\sigma M : A \qquad \epsilon : A \rightsquigarrow B \in \sigma}{\Sigma \mid \Gamma \vdash_\sigma \texttt{raise}\ \epsilon(M) : B}$$

$$\frac{\begin{array}{c}\Sigma \mid \Gamma \vdash_\sigma M : A \\ \Sigma \mid \Gamma, x : A \vdash_\tau N : B \\ (\forall (\varepsilon : A_\varepsilon \rightsquigarrow B_\varepsilon) \in \sigma.\ (\varepsilon \notin \mathrm{dom}(\phi) \wedge (\varepsilon : A_\varepsilon \rightsquigarrow B_\varepsilon) \in \tau) \\ \vee (\varepsilon \notin \mathrm{dom}(\phi) \wedge \Sigma \mid \Gamma, x : A_\varepsilon, k : B_\varepsilon \to_\tau B \vdash_\tau \phi(\varepsilon) : B))\end{array}}{\Sigma \mid \Gamma \vdash_\tau \texttt{handle}\ M\,\{\texttt{ret}\ x.N \mid \phi\} : B} \qquad \frac{\Sigma \mid \Gamma \vdash_\sigma M : A \qquad A \sqsubseteq B}{\Sigma \mid \Gamma \vdash_\sigma \langle B \nwarrow A \rangle M : B}$$

$$\frac{\Sigma \mid \Gamma \vdash_\sigma M : B \qquad A \sqsubseteq B}{\Sigma \mid \Gamma \vdash_\sigma \langle A \swarrow B \rangle M : A} \qquad \frac{\Sigma \mid \Gamma \vdash_\sigma M : A \qquad \sigma \sqsubseteq \sigma'}{\Sigma \mid \Gamma \vdash_{\sigma'} \langle \sigma' \nwarrow \sigma \rangle M : A} \qquad \frac{\Sigma \mid \Gamma \vdash_{\sigma'} M : A \qquad \sigma \sqsubseteq \sigma'}{\Sigma \mid \Gamma \vdash_\sigma \langle \sigma \swarrow \sigma' \rangle M : A}$$

Fig. 4. Core GrEff Typing

covariant in the request type and *contravariant* in the response type. This variance makes sense from the perspective of the party *producing* the request, to match the function type subtyping. Finally, type precision $A \sqsubseteq B$ tracks instead how "dynamic" or "imprecise" a type is. For functions it is covariant in every argument, and for effect types, the dynamic effect is the most imprecise and for two concrete effect sets, it has a depth rule that that is covariant in request and response positions. In a more standard gradual language with full dynamic typing, in addition to the dynamic effect type we would have a dynamic value type $?_v$ that is similarly maximally imprecise among value types.

3.2 Syntax and Elaboration of GrEff

We present the syntax for the surface language GrEff in Figure 6. To distinguish surface GrEff syntactic forms from similar core GrEff forms we use an underscore of s for **s**urface GrEff forms. A GrEff program P consists of a sequence of modules ending in a single "main" module. Each module m consists of two parts: first, the effect definitions and then the value definitions, whose types annotations may use the effects previously defined in that module. An effect definition is either a declaration of a new effect operation `effect` $\varepsilon : A_s \rightsquigarrow B_s$ or an import of an existing effect operation `import-eff` $m.\varepsilon : A_s \rightsquigarrow B_s$. In either case, the declaration includes the request type A_s and the response type B_s for the effect within the current module. An effect import brings an effect defined in another module into the current scope, but with a possibly different request and response type. To support *gradual* migration, these types are allowed to have a different level of precision than the original, but on subterms where both types are precise they must match. After the effect declarations are the value definitions which are also either a definition of a new value `define` $x = V_s$ or an import of a value declared in a different module at a possibly different type

$$\Sigma \vdash \texttt{bool} \qquad \frac{\Sigma \vdash A \qquad \Sigma \vdash \sigma \qquad \Sigma \vdash B}{\Sigma \vdash A \rightarrow_\sigma B} \qquad \Sigma \vdash ? \qquad \frac{\begin{array}{c}\forall \varepsilon : A \rightsquigarrow B \in \sigma_c. \\ (\varepsilon : |A| \rightsquigarrow |B| \in \Sigma). \land \Sigma \vdash A \land \Sigma \vdash B)\end{array}}{\Sigma \vdash \sigma_c}$$

$$\texttt{bool} \le \texttt{bool} \qquad \frac{A' \le A \qquad \sigma \le \sigma' \qquad B \le B'}{A \rightarrow_\sigma B \le A' \rightarrow_{\sigma'} B'} \qquad ? \le ? \qquad \frac{\begin{array}{c}\forall \varepsilon : A_\sigma \rightsquigarrow B_\sigma \in \sigma_c. \exists A_\tau, B_\tau. \\ \varepsilon : A_\tau \rightsquigarrow B_\tau \in \tau_c \land A_\sigma \le A_\tau \land B_\tau \le A_\tau\end{array}}{\sigma_c \le \tau_c}$$

$$\texttt{bool} \sqsubseteq \texttt{bool} \qquad \frac{A \sqsubseteq A' \qquad \sigma \sqsubseteq \sigma' \qquad B \sqsubseteq B'}{A \rightarrow_\sigma B \sqsubseteq A' \rightarrow_{\sigma'} B'} \qquad \sigma \sqsubseteq ? \qquad \frac{\begin{array}{c}\mathrm{dom}(\sigma_c) = \mathrm{dom}(\sigma'_c) \\ \forall \varepsilon : A \rightsquigarrow B \in \sigma_c. \exists A', B'. \\ \varepsilon : A' \rightsquigarrow B' \in \sigma'_c \land A \sqsubseteq A' \land B \sqsubseteq B'\end{array}}{\sigma_c \sqsubseteq \sigma'_c}$$

Fig. 5. Well formed types and effects, Type and Effect Precision

$$\begin{array}{rcl}
\text{Programs } P & ::= & L; \cdots L_{main} \\
\text{Modules } L & ::= & \texttt{module } m \; \{b\} \\
\text{Module Body } b & ::= & \cdot \mid D; b \\
\text{Main Module } L_{main} & ::= & \texttt{main } \{b; M_s\} \\
\text{Module reference } r & ::= & m.x \mid m.\varepsilon \\
\text{Declaration } D & ::= & \texttt{import-eff } r : A_s \rightsquigarrow B_s \mid \texttt{effect } \varepsilon : A_s \rightsquigarrow B_s \\
 & & \mid \texttt{define } x = V \mid \texttt{import-val } r \texttt{ as } x : A \\
\text{Surface Value Types } A_s, B_s, C_s & ::= & A \rightarrow_\sigma B \mid \texttt{bool} \\
\text{Surface Effect Types } \sigma_s, \tau_s & ::= & ? \mid \sigma_{sc} \\
\text{Operation Set } \sigma_{sc}, \tau_{sc} & \in & \mathcal{P}_{\mathrm{fin}}(\mathrm{Name}) \\
\text{Surface Values } V_s & ::= & x \mid \lambda x : A_s . M_s \mid \texttt{true} \mid \texttt{false} \\
\text{Surface Terms } M_s, N_s & ::= & x \mid \texttt{raise } \varepsilon(M_s) \mid \texttt{handle}_{C_s \,!\, \sigma_s} \, M_s \, \{\texttt{ret } x.N_s \mid \phi_s\} \\
 & & \mid \lambda x.M_s \mid M_s \, M'_s \mid \texttt{true} \mid \texttt{false} \mid \texttt{if } M_s\{N_s\}\{N'_s\} \\
 & & \mid M_s :: A_s \mid M_s :: \sigma_s \\
\text{Handler clauses } \phi_s & \in & \mathrm{Name} \rightharpoonup_{\mathrm{fin}} \mathrm{SurfTerm} \\
\text{Program Typing Contexts } \Delta & ::= & \cdot \mid \Delta, m \mapsto \Gamma_s \\
\text{Module Typing Contexts } \Gamma_s & ::= & \cdot \mid \Gamma_s, \varepsilon : A \rightsquigarrow B \mid \Gamma_s, x : A
\end{array}$$

Fig. 6. GrEff Syntax

`import-val` r `as` $x : A_s$. For simplicity, all effects and values are public and can be imported by later modules. Finally a program ends with a main module, which consists of the same kind of effect and value declarations, followed by a final main expression.

Surface GrEff types differ from core GrEff types in that effect types are *nominal*: a concrete effect set σ_{sc} is simply a finite set of names such as $\{\texttt{fork}, \texttt{yield}, \texttt{print}\}$ where the types of the effect names are determined by the declaration in the current module. The elaboration process adds the relevant type information to match the more structural typing of core GrEff. Surface GrEff terms and values are for the most part similar to the core GrEff forms except that they may include syntactic type annotations in order to support the algorithmic gradual type system of the surface language.

$$\frac{\Sigma \mid \Delta \mid \cdot \vdash b \Rightarrow \Sigma'; \gamma; \Gamma_s \qquad \Gamma_s \vdash M_s \Rightarrow M : \sigma\,!\,A}{\Sigma \mid \Delta \vdash \texttt{main}\ b\ M_s \Rightarrow \Sigma' \vdash_\sigma \texttt{let}\ \Gamma_s = \gamma\ \texttt{in}\ M : A}$$

$$\frac{\Sigma \mid \Delta \mid \cdot \vdash b \Rightarrow \Sigma'; \gamma; \Gamma_s \qquad \Sigma, \Sigma' \mid \Delta, m \mapsto \Gamma_s \vdash P \Rightarrow \Sigma'' \vdash_\sigma M : A}{\Sigma \mid \Delta \vdash \texttt{module}\ m\ b;\ P \Rightarrow \Sigma', \Sigma'' \vdash_\sigma \texttt{let}\ \Gamma_s = \gamma\ \texttt{in}\ M : A} \qquad \frac{}{\Sigma \mid \Delta \mid \Gamma_s \vdash \cdot \Rightarrow \cdot; \cdot; \cdot}$$

$$\frac{\varepsilon \notin \Sigma \qquad \Gamma_s \vdash A_s \Rightarrow A \qquad \Gamma_s \vdash B_s \Rightarrow B}{\Sigma \mid \Delta \mid \Gamma_s \vdash \texttt{effect}\ \varepsilon : A_s \rightsquigarrow B_s \Rightarrow (\varepsilon : |A| \rightsquigarrow |B|); \cdot; \varepsilon : A \rightsquigarrow B}$$

$$\frac{\Sigma \mid \Delta \mid \Gamma_s \vdash D \Rightarrow \Sigma'; \gamma'; \Gamma_s{}' \qquad \Sigma, \Sigma' \mid \Delta \mid \Gamma_s, \Gamma_s{}' \vdash b \Rightarrow \Sigma''; \gamma''; \Gamma_s{}''}{\Sigma \mid \Delta \mid \Gamma_s \vdash D; b \Rightarrow \Sigma', \Sigma''; \gamma', \gamma''; \Gamma_s{}', \Gamma_s{}''}$$

$$\frac{\Delta(m) \ni \varepsilon : A' \rightsquigarrow B' \qquad \Gamma_s \vdash A_s \Rightarrow A \qquad \Gamma_s \vdash B_s \Rightarrow B \qquad A \sim A' \qquad B \sim B'}{\Sigma \mid \Delta \mid \Gamma_s \vdash \texttt{import-eff}\ m.\varepsilon : A_s \rightsquigarrow B_s \Rightarrow \cdot; \cdot; \varepsilon : A \rightsquigarrow B}$$

$$\frac{\Gamma_s \vdash V_s \Rightarrow V : \emptyset\,!\,A}{\Sigma \mid \Delta \mid \Gamma_s \vdash \texttt{define}\ x = V_s \Rightarrow \cdot; V/x; x : A}$$

$$\frac{\Delta(m) \ni x : A' \qquad \Gamma_s \vdash A_s \Rightarrow A \qquad A' \lesssim A}{\Sigma \mid \Delta \mid \Gamma_s \vdash \texttt{import-val}\ m.x\ \texttt{as}\ y : A_s \Rightarrow \cdot; \langle A \Leftarrow A' \rangle x/y; y : A}$$

Fig. 7. GrEff Typing/Elaboration, Module Language

Finally we define the typing contexts for programs Δ and modules Γ_s, which are used in the elaboration/type checking process. A program typing context Δ associates module names m to their module typing contexts. The module typing context Γ_s contains both typings for values and effect names. Note that the *types* in the module typing context are *core* GrEff types because these types are the result of elaboration of surface GrEff types.

Next, we present the combination type checker and elaborator from GrEff into core GrEff. We view GrEff programs as essentially a description of an effect signature Σ and a closed expression well-typed under that signature. The module system is a way to manage the declaration of new effect operations in the signature and a way to manage the typing of effect operations by giving nominal associations to request and response types rather than solely the structural typing in core GrEff. We describe the elaboration of the module language in Figure 7. The top-level judgment $\Sigma \mid \Delta \vdash P \Rightarrow \Sigma' \vdash_\sigma M : A$ says that under the starting signature Σ and previously defined modules Δ, we can elaborate P to a *core GrEff* term M with *core GrEff* effect type σ and *core GrEff* value type A that is well-typed under the extension of the signature by Σ'. To elaborate a complete program, we initialize this with empty signature and module typing ($\cdot \mid \cdot \vdash P \Rightarrow \Sigma \vdash_\sigma M : A$). This expresses that not only does a program denote a core GrEff program, but it also has a "side effect" of allocating new effect names Σ'. A module is elaborated with the judgment $\Sigma \mid \Delta \mid \Gamma_s \vdash b \Rightarrow \Sigma'; \gamma'; \Gamma_s'$. The outputs of this judgment are the newly allocated effects of the module Σ', the names of effect operations and types for values the module defines Γ_s' and the definitions of all the values the module defines, given as a core GrEff substitution γ' from variable names in Γ_s' to core GrEff values of their associated types. Then to elaborate a program consisting of several modules, we first elaborate the modules and then elaborate the remainder of the program and finally combine the two by let-binding all of the variables declared in the module, which we write as a shorthand $\texttt{let}\ \Gamma_s = \gamma\ \texttt{in}\ M$. A module is elaborated by combining the results of elaborating each declaration. A new effect declaration

$$\langle A \Leftarrow B \rangle M = \langle A \swarrow |A| \rangle \langle |B| \nwarrow B \rangle M \qquad \langle \sigma \Leftarrow \tau \rangle M = \langle \sigma \swarrow ? \rangle \langle ? \nwarrow \tau \rangle M \qquad \frac{\Gamma \ni x : A}{\Gamma \vdash x \Rightarrow x : \emptyset\,!\,A}$$

$$\Gamma \vdash \mathtt{true} \Rightarrow \mathtt{true} : \emptyset\,!\,\mathtt{bool} \qquad \Gamma \vdash \mathtt{false} \Rightarrow \mathtt{false} : \emptyset\,!\,\mathtt{bool}$$

$$\frac{\Gamma \vdash M_s \Rightarrow M : \sigma\,!\,A' \qquad \Gamma \vdash A_s \Rightarrow A \qquad A' \lesssim A}{\Gamma \vdash M_s :: A_s \Rightarrow \langle A \Leftarrow A' \rangle M : \sigma\,!\,A} \qquad \frac{\Gamma \vdash M_s \Rightarrow M : \sigma'\,!\,A \qquad \Gamma \vdash \sigma_s \Rightarrow \sigma \qquad \sigma' \lesssim \sigma}{\Gamma \vdash M_s :: \sigma_s \Rightarrow \langle \sigma \Leftarrow \sigma' \rangle M : \sigma\,!\,A}$$

$$\frac{\Gamma \vdash M_s \Rightarrow M : \sigma_m\,!\,\mathtt{bool} \qquad \Gamma \vdash N_s \Rightarrow N : \sigma_n\,!\,B \qquad \Gamma \vdash N'_s \Rightarrow N' : \sigma'_n\,!\,B' \qquad C = B \mathrel{\tilde{\vee}} B' \qquad \sigma = \sigma_m \mathrel{\tilde{\vee}} \sigma_n \mathrel{\tilde{\vee}} \sigma'_n}{\Gamma \vdash \mathtt{if}\ M_s\{N_s\}\{N'_s\} \Rightarrow \mathtt{if}\ \langle \sigma \Leftarrow \sigma_m \rangle M\{\langle \sigma \Leftarrow \sigma_n \rangle \langle C \Leftarrow B \rangle N\}\{\langle \sigma \Leftarrow \sigma'_n \rangle \langle C \Leftarrow B' \rangle N'\} : \sigma\,!\,C}$$

$$\frac{\Gamma \vdash A_s \Rightarrow A \qquad \Gamma, x : A \vdash M_s \Rightarrow M : \sigma\,!\,B}{\Gamma \vdash \lambda x : A_s.M_s \Rightarrow \lambda x.M : \emptyset\,!\,A \to_\sigma B} \qquad \frac{\Gamma \vdash M_s \Rightarrow M : \sigma_m\,!\,A \to_{\sigma_o} B \qquad \Gamma \vdash N_s \Rightarrow N : \sigma_n\,!\,A' \qquad A' \lesssim A \qquad \sigma = \sigma_m \mathrel{\tilde{\vee}} \sigma_n \mathrel{\tilde{\vee}} \sigma_o}{\Gamma \vdash M_s\,N_s \Rightarrow (\langle \sigma \Leftarrow \sigma_m \rangle M)(\langle A \Leftarrow A' \rangle \langle \sigma \Leftarrow \sigma_n \rangle N) : \sigma\,!\,B}$$

$$\frac{\Gamma \vdash M_s \Rightarrow M : \sigma_m\,!\,A' \qquad \Gamma \ni \varepsilon : A \rightsquigarrow B \qquad A' \lesssim A \qquad \sigma = \sigma_m \mathrel{\tilde{\vee}} \{\varepsilon : A \rightsquigarrow B\}}{\Gamma \vdash \mathtt{raise}\ \varepsilon(M_s) \Rightarrow \mathtt{let}\ x = \langle \sigma \Leftarrow \sigma_m \rangle M\ \mathtt{in}\ \langle \sigma \Leftarrow \{\varepsilon : A \rightsquigarrow B\} \rangle \mathtt{raise}\ \varepsilon(\langle A \Leftarrow A' \rangle x) : \sigma\,!\,B}$$

$$\frac{\begin{array}{c}\Gamma \vdash C_s \Rightarrow C \qquad \Gamma \vdash \sigma_s \Rightarrow \sigma \qquad \Gamma \vdash M_s \Rightarrow M : \sigma_m\,!\,A_m \\ \Gamma, x : A_m \vdash N_s \Rightarrow N : \sigma_n\,!\,C_n \qquad \sigma_n \lesssim \sigma \qquad C_n \lesssim C \\ \mathrm{dom}(\phi_\Leftarrow) = \mathrm{dom}(\phi_s) \qquad \Gamma \vdash \mathrm{handleTy}(\sigma_m, \sigma, \mathrm{dom}(\phi_s)) = \sigma'_m \\ (\forall \varepsilon \in \mathrm{dom}(\phi_s).\ \exists(\varepsilon : A_\varepsilon \rightsquigarrow B_\varepsilon) \in \Gamma. \\ \Gamma, x : A_\varepsilon, k : B_\varepsilon \to_\sigma C \vdash \phi_s(\varepsilon) \Rightarrow N_\varepsilon : \sigma_\varepsilon\,!\,C_\varepsilon \\ \sigma_\varepsilon \lesssim \sigma \qquad C_\varepsilon \lesssim C \qquad \phi_\Leftarrow(\varepsilon) = \langle \sigma \Leftarrow \sigma_\varepsilon \rangle \langle C \Leftarrow C_\varepsilon \rangle N_\varepsilon)\end{array}}{\begin{array}{c}\Gamma \vdash \mathtt{handle}_{\sigma_s\,!\,C_s}\ M_s\ \{\mathtt{ret}\ x.N_s \mid \phi_s\} \Rightarrow \\ \mathtt{handle}\ \langle \sigma'_m \Leftarrow \sigma_m \rangle M\ \{\mathtt{ret}\ x.\langle \sigma \Leftarrow \sigma_n \rangle \langle C \Leftarrow C_n \rangle N \mid \phi_\Leftarrow\} : \sigma\,!\,C\end{array}}$$

$$\frac{\mathrm{dom}(\sigma_c) \subseteq \mathrm{dom}(\tau_c) \cup \sigma_{sc}}{\Gamma \vdash \mathrm{handleTy}(\sigma_c, \tau_c, \sigma_{sc}) = \tau_c \cup \Gamma(\sigma_{sc})} \qquad \Gamma \vdash \mathrm{handleTy}(?, \tau_c, \sigma_{sc}) = \tau_c \cup \Gamma(\sigma_{sc})$$

$$\Gamma \vdash \mathrm{handleTy}(\sigma_c, ?, \sigma_{sc}) = \sigma_c|_{\sigma_{sc}} \uplus |\Gamma(\mathrm{dom}(\sigma_c) - \sigma_{sc})| \qquad \Gamma \vdash \mathrm{handleTy}(?, ?, \sigma_{sc}) = ?$$

Fig. 8. GrEff Typing/Elaboration, Expression Language

checks that the name is not previously declared, and then recursively elaborates the syntactic types declared for request and response and then adds these to the allocated effects as well as the local effect names declared in the module. When adding to the signature, we take erasure of the types because signatures use untracked types. Next, to import an effect from a different module, the types given for the effect are checked to be compatible with the types declared in the other module. Note that for simplicity of presentation, all effects must be used with the same name in all modules.

More flexible renaming mechanisms can easily be supported in a realistic implementation. Here the compatibility judgment $A \sim A'$ is defined on core GrEff types as the conjunction of gradual subtyping in both directions, $A \lesssim A'$ and $A' \lesssim A$, to be defined soon. This compatibility check ensures that any imports from that module using this effect name will succeed. We check gradual subtyping in both directions because the effect may be used in both positive and negative positions in a later import. This effect name is added to the local names only, and not the signature, because it is using an already allocated effect name. Next, defining a value simply elaborates the value and adds its type to the output typing and associates the value to that name. Importing a value is similar, except that we check that the declared type is a gradual subtype, and so can be coerced by the cast $\langle A \Leftarrow A' \rangle$, whose definition will be described shortly.

Next, we define the elaboration of the expression language in Figure 8. The judgment $\Gamma_s \vdash M_s \Rightarrow M : \sigma \,!\, A$ says that under the typing of names given by Γ_s, the GrEff expression M_s elaborates to the core GrEff function M, which will be well-typed with inferred core GrEff effect type σ and value type A. All forms essentially elaborate to similar forms in core GrEff, but with suitable casts inserted. First, we define the translation of value type casts $\langle A \Leftarrow B \rangle M$ and effect type casts $\langle \sigma \Leftarrow \tau \rangle M$ as an upcast followed by a downcast. For the effect cast, these casts go through the dynamic effect type, but for two value types there is no single most dynamic effect type so we again use the erasure operation. Note that this will only be well-typed in case $|B| = |A|$, which is ensured whenever $A \lesssim B$, which is a precondition for inserting a cast. This is not necessarily the most efficient implementation of the cast, we discuss optimizations in Section 4.3

Next, variables, boolean values and function values elaborate to themselves with an empty effect type $\emptyset$. The let-binding form shows how different effect types are combined: the effect types of M_s and N_s are combined using a gradual join $\tilde{\vee}$ (to be defined shortly), and casts are inserted into the elaborations of M_s and N_s to give them this effect type. The ascription forms simply check that the appropriate kind of type satisfies a gradual subtyping judgment and inserts a cast. This uses the elaboration of types $\Gamma \vdash A_s \Rightarrow A$, defined below. The if rule checks that the condition has boolean type and gives the output value type as the gradual join of the branches, and the output effect type as the gradual join with the condition expression as well, matching prior work [Garcia et al. 2016]. The application rule is similar except that the argument is cast to have the type of the domain of the function and the effect type of the function is joined with the effect types of the terms. Next, we have the raise form, which elaborates to a raise but first let-binds the request term and casts the raise term to have an effect type that is the join of the request term's effect type and the operation's type. Finally, we have the most complex case, the handle form. The handle form elaborates to a handle form in the core language with casts inserted in each case to make them agree with the ascribed value type C_s and effect type σ_s. The request variables and input to the continuations are given by looking up the effect in Γ_s, while the output is given by the ascription. The most complex part of this elaboration is the cast needed for the scrutinee M_s. In the core language, we need that all of the effects that M raises but are *not* caught by the handle are in the output type σ_s. But when σ_s is dynamic and M_s has concrete effect type or vice-versa, this is not necessarily true, so in these cases a cast must be inserted that effectively handles all of the "other" effects. This definition is given below in a special elaboration of handle scrutinees ($\Gamma \vdash \mathrm{handleTy}(\sigma, \tau, \sigma_{sc}) = \sigma_o$). Here, the type σ is the elaborated type of the scrutinee, τ is the elaborated type of the result of the handle expression, and σ_s is the set of effect names caught by the handler, where we write $\Gamma(\sigma_{sc})$ for the map that looks up the currently associated types for each operation in σ_{sc}. First, if σ and τ are both precise collections of effects, then we check that all of the effects it raises are either caught or still occur in the output type, and we insert a subtyping cast. Second, if σ, the type of the scrutinee is imprecise, then we downcast it to include only the union of the output effects and the caught

$$\Gamma \vdash \texttt{bool} \Rightarrow \texttt{bool} \qquad \frac{\Gamma \vdash A_s \Rightarrow A \qquad \Gamma \vdash B_s \Rightarrow B \qquad \Gamma \vdash \sigma_s \Rightarrow \sigma}{\Gamma \vdash A_s \to_{\sigma_s} B_s \Rightarrow A \to_\sigma B} \qquad \Gamma \vdash ? \Rightarrow ? \qquad \frac{\mathrm{dom}(\sigma_c) = \sigma_s \qquad \forall \varepsilon \in \sigma_c.\ \sigma_c(\varepsilon) = \Gamma(\varepsilon)}{\Gamma \vdash \sigma_s \Rightarrow \sigma_c}$$

$$\texttt{bool} \lesssim \texttt{bool} \qquad \frac{A' \lesssim A \qquad \sigma \lesssim \sigma' \qquad B \lesssim B'}{A \to_\sigma B \lesssim A' \to_{\sigma'} B'} \qquad ? \lesssim \sigma \qquad \sigma \lesssim ? \qquad \frac{\begin{array}{c}\forall \varepsilon : A_\sigma \rightsquigarrow B_\sigma \in \sigma_c. \exists A_\tau, B_\tau. \\ \varepsilon : A_\tau \rightsquigarrow B_\tau \in \tau_c \\ \wedge A_\sigma \lesssim A_\tau \\ \wedge B_\tau \lesssim B_\sigma\end{array}}{\sigma_c \lesssim \tau_c}$$

$$\begin{aligned}
\texttt{bool} \mathbin{\tilde{\vee}} \texttt{bool} &= \texttt{bool} \\
(A \to_\sigma B) \mathbin{\tilde{\vee}} (A' \to_{\sigma'} B') &= (A \mathbin{\tilde{\wedge}} A') \to_{\sigma \tilde{\vee} \sigma'} (B \mathbin{\tilde{\vee}} B') \\
? \mathbin{\tilde{\vee}} \sigma &= ? \\
\sigma \mathbin{\tilde{\vee}} ? &= ? \\
\sigma_c \mathbin{\tilde{\vee}} \tau_c &= \{\varepsilon : A \rightsquigarrow B \mid \varepsilon : A \rightsquigarrow B \in \sigma_c \wedge \varepsilon \notin \mathrm{dom}(\tau_c)\} \\
&\quad \cup \{\varepsilon : A' \rightsquigarrow B' \mid \varepsilon \notin \mathrm{dom}(\sigma_c) \wedge \varepsilon : A' \rightsquigarrow B' \in \tau_c\} \\
&\quad \cup \{\varepsilon : A \mathbin{\tilde{\vee}} A' \rightsquigarrow B \mathbin{\tilde{\wedge}} B' \mid \varepsilon : A \rightsquigarrow B \in \sigma_c \wedge \varepsilon : A' \rightsquigarrow B' \in \tau_c\}
\end{aligned}$$

$$\begin{aligned}
\texttt{bool} \mathbin{\tilde{\vee}} \texttt{bool} &= \texttt{bool} \\
(A \to_\sigma B) \mathbin{\tilde{\wedge}} (A' \to_{\sigma'} B') &= (A \mathbin{\tilde{\vee}} A') \to_{\sigma \tilde{\wedge} \sigma'} (B \mathbin{\tilde{\wedge}} B') \\
? \mathbin{\tilde{\wedge}} \sigma &= \sigma \\
\sigma \mathbin{\tilde{\wedge}} ? &= \sigma \\
\sigma_c \mathbin{\tilde{\wedge}} \tau_c &= \{\varepsilon : A \mathbin{\tilde{\wedge}} A' \rightsquigarrow B \mathbin{\tilde{\vee}} B' \mid \varepsilon : A \rightsquigarrow B \in \sigma_c \wedge \varepsilon : A' \rightsquigarrow B' \in \tau_c\}
\end{aligned}$$

Fig. 9. Type Elaboration, Gradual Subtyping and Join/Meet

effects, otherwise erroring. Third, if the scrutinee is precise but the result $\tau = ?$ is dynamic, then we need to upcast all of the unhandled effect operations to their dynamic versions. This is expressed by having the result type be the combination ($\uplus$) of the effects who are handled as is, written $\sigma_c|_{\sigma_s}$ with the most dynamic version of any other effects that are not handled $|\Gamma(\mathrm{dom}(\sigma_c) - \sigma_s)|$. Here $\sigma_c|_{\sigma_s}$ means the restriction of the partial function σ_c to only be defined on the set σ_{sc}. Finally, if the scrutinee and the goal are both imprecise then we put a trivial identity cast to ? on the scrutinee.

Finally, Figure 9 describes the elaboration of types, gradual subtyping and gradual join and meet. Value and effect type elaboration $\Gamma_s \vdash A_s \Rightarrow A$ is mostly structural. The elaboration of a concrete effect set is essentially a "map" over the fields of the concrete effect set, saying the elaborated concrete effect type has the exact same names as the surface effect set, and they are associated to the request and response types of the effect operation based on the current module context Γ_s. Next, we describe the mostly standard *gradual* subtyping of value types $A \lesssim B$ and effect types $\sigma \lesssim \tau$ to determine when a dynamic cast $\langle B \Leftarrow A\rangle$ or $\langle \tau \Leftarrow \sigma\rangle$ would reduce to subtyping on the precise portions of the types. Note that we define gradual subtyping of types in the core language i.e., after elaboration, so that we can compare effect types across module boundaries that use different typings for the effect names. With this intuition, the definition is like that of subtyping, except that the dynamic effect type is a gradual subtype and supertype of all other effect types.

Lastly, we define gradual join and meet of types and effects as a partial function. The gradual join of types is defined similarly to prior work, with the covariant positions in the function type recursively being joined, while the contravariant position, the domain uses the gradual meet. The gradual join of two concrete effect rows takes the union of the effects used in each type, where the common effect names have to be joined as well. Here the request is covariant, and recursively joined and the response type is contravariantly and so recursively the gradual meet is used. On concrete effect types, the gradual meet is similarly defined as an intersection of the effects used, where the requests and responses are handled dually. Finally, taking the gradual join with the dynamic effect always returns the dynamic effect and taking the gradual meet always returns the original type. This can be justified by the AGT methodology by interpreting the concretization of the gradual effect type as the set of all possible fully static effect types. Following the AGT methodology in this way ensures the static gradual guarantee is satisfied.

We conclude by noting the following syntactic properties of elaboration, which follow by structural induction.

Lemma 3.1 (Elaboration is a function). *If* $\cdot \mid \cdot \vdash P \Rightarrow \Sigma \vdash_{\sigma} M : A$ *and* $\cdot \mid \cdot \vdash P \Rightarrow \Sigma' \vdash_{\sigma'} M' : A'$ *then* $\Sigma = \Sigma'$ *and* $M = M'$ *and* $\sigma = \sigma'$ *and* $A = A'$.

Lemma 3.2 (Elaborated terms are Well-typed). *If* $\cdot \mid \cdot \vdash P \Rightarrow \Sigma \vdash_{\sigma} M : A$, *then* $\Sigma \mid \cdot \vdash_{\sigma} M : A$.

4 AXIOMATICS AND OPERATIONAL SEMANTICS

Next we turn to the semantic aspects of GrEff: how expressions are evaluated, what simplifications/optimizations are correct to perform, and that the graduality principle holds for the language. We formalize these three aspects axiomatically in the form of an *inequational theory* for reasoning about Core GrEff programs. That is, we define a notion of inequality $M \sqsubseteq N$ between expressions called *term precision*, which is a kind of extension of the notion of type precision to expressions. The semantic interpretation of this inequality is that M has the same behavior as N with respect to output and termination, except in that it may raise a dynamic type error when N does not. From this notion of inequality we get an induced equivalence relation $M \equiv N$ that specifies when M and N have the same behavior. Term precision and the induced equivalence are used to model our desired semantic ideas: an expression M can be evaluated to a value V when the equivalence $M \equiv V$ holds, M can be simplified/optimized to N when $M \equiv N$ holds, and the graduality principle states that when M is rewritten in the surface language to some M' that has more precise typing information, then a corresponding relationship $M' \sqsubseteq M$ should hold: adding more precise type information results in more precise dynamic type checking. With this in mind, we axiomatize the valid optimizations known from effect handlers as well as desired inequalities from prior work on graduality in our inequational theory.

Axioms are only useful if we can construct models in which they are satisfied. For GrEff, we do this by constructing an *operational* semantics that specifies more precisely how to evaluate programs and then define notions of observational equivalence and an error ordering to model $\equiv$ and $\sqsubseteq$ and prove that all of the axioms are valid in this operational model. We will construct this operational semantics, based on the axiomatics: we show in Section 4.2 that every reduction $M \mapsto N$ is justified by a provable equivalence $M \equiv N$ in the inequational theory. For many rules this is very straightforward, e.g., β reduction of functions is justified by a corresponding β equation. The most utility we get from the axioms in this case is for the cast reductions: cast reductions for handlers are justified not by a direct corresponding rule in the axioms, but instead by extensionality

(η) principles for handlers combined with a least upper bound/greatest lower bound property of casts identified in prior work as being key to the graduality property [New and Licata 2018]. This shows that the operational behavior we define has a canonical status: if certain optimizations for handlers are to be valid, and the graduality property is desired, then the cast reductions we define must be used.

4.1 Axiomatics

We present a selection of the rules of the inequational theory of term precision in Figure 10. The full rules are provided in the appendix [New et al. 2023]. The form of the inequality judgment is $\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \tau} M \sqsubseteq N : A \sqsubseteq B$, which says that M is more precise, or, roughly, "errors more" than N. This is a kind of *heterogeneous* inequality relation in that M and N are not required to have the same type: M must have value type A and effect type σ and N must have value type B and effect type τ under the context $\Gamma^{\sqsubseteq}$ and $A \sqsubseteq B$ and $\sigma \sqsubseteq \tau$ must hold. We allow for M and N to be open terms, typed with respect to the typing context $\Gamma^{\sqsubseteq}$. The typing context $\Gamma^{\sqsubseteq}$ is like an ordinary typing context Γ, except that variables are typed $x : A \sqsubseteq B$ where the left type A is the type x has in the left term M and B is the type for N. For the context to be well formed, each of the $A \sqsubseteq B$ must be provable.

First, we add reflexivity and transitivity rules, where in the transitivity rule both the value and effect type are allowed to vary simultaneously. Next, we give two rules for modeling errors: first $\mho$ is the least element in the ordering, which models the graduality property, and second that all evaluation contexts are strict with respect to errors. The latter uses equivalence $\equiv$, which is defined as a shorthand: $M \equiv N$ means that both $M \sqsubseteq N$ and $N \sqsubseteq M$ are true. In this case, we elide the typing, but both sides are assumed to be well typed under the same context and typing $\Gamma \vdash_{\sigma} M, N : A$. Next we have computation (β) and reasoning (η) rules for each type. For functions and if, these are standard call-by-value $\beta\eta$ rules, so we instead show only the handle rules. There are two β rules for handle. If the term being handled is a value, then the return clause is used. If the term being handled is a raise of an effect ε, it is equivalent to the handler clause $\phi(\varepsilon)$ where the continuation is the captured continuation surrounding the original handler term. We require this to be a let, but note that we have additional rules that imply that any evaluation context that doesn't handle can be re-written as a let. We then have two reasoning (η) rules for handle. First, if M is handled by a handler with no effect clauses, then the handler is equivalent to a let-binding. This can be combined with standard rules for let binding to show that any term is equivalent to a handler with no clauses $M \equiv \texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \emptyset\}$. We call this the *non-handling* principle. Second, we have a rule that says that any clause that simply re-raises its operation with the same continuation it was passed can be dropped from the handler, as this is the same behavior as not catching the term at all. We call this the *effect forwarding* principle, as it says that forwarding an effect to the ambient context is equivalent to not handling it explicitly at all. Combined with the non-handling principle, any term M with effect type σ can be shown equivalent to $\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\sigma}\}$ where ϕ_{σ} simply forwards all the effects in σ. We next show rules describing the interaction of subtyping with value type casts, the full system includes analogous rules for effect types. The first says that an upcast followed by a subtyping coercion is less than a subtyping coercion followed by an upcast, and the downcast rule is similar. Finally, we have rules specifying the behavior of value and effect casts. These rules characterize upcasts as least upper bounds and downcasts as greatest lower bounds. The first rule shows that the downcast is a lower bound and the second that it is the greatest. The upcasts have similar rules, and we include analogous rules for effect casts as well. These lub/glb properties are adapted from prior work on axiomatics for gradual typing [New et al. 2019], but now incorporate the ordering on both effect and value typing. We found that this general form of the rule, where the effect is allowed to differ ($\sigma \sqsubseteq \sigma'$) while performing a

$$\frac{\Gamma \vdash_{\sigma} M : A}{\Gamma \vdash_{\sigma \sqsubseteq \sigma} M \sqsubseteq M : A \sqsubseteq A} \qquad \frac{\begin{array}{c}\Gamma^{\sqsubseteq} \vdash_{\sigma_1 \sqsubseteq \sigma_2} M_1 \sqsubseteq M_2 : A_1 \sqsubseteq A_2 \qquad \Gamma'^{\sqsubseteq} \vdash_{\sigma_2 \sqsubseteq \sigma_3} M_2 \sqsubseteq M_3 : A_2 \sqsubseteq A_3 \\ \mathrm{rhs}(\Gamma^{\sqsubseteq}) = \mathrm{lhs}(\Gamma'^{\sqsubseteq}) \quad \mathrm{lhs}(\Gamma^{\sqsubseteq}) = \mathrm{lhs}(\Gamma''^{\sqsubseteq}) \quad \mathrm{rhs}(\Gamma'^{\sqsubseteq}) = \mathrm{rhs}(\Gamma''^{\sqsubseteq})\end{array}}{\Gamma''^{\sqsubseteq} \vdash_{\sigma_1 \sqsubseteq \sigma_3} M_1 \sqsubseteq M_3 : A_1 \sqsubseteq A_3}$$

$$\frac{\Gamma \vdash_{\sigma} M : A}{\Gamma \vdash_{\sigma \sqsubseteq \sigma} \mho \sqsubseteq M : A \sqsubseteq A} \qquad E[\mho] \equiv \mho$$

$$\frac{\Gamma^{\sqsubseteq}, x : A \sqsubseteq A' \vdash_{\tau \sqsubseteq \tau'} M \sqsubseteq M' : B \sqsubseteq B'}{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} \lambda x.M \sqsubseteq \lambda x.M' : A \to_{\tau} B \sqsubseteq A' \to_{\tau'} B'} \qquad \frac{\begin{array}{c}\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq M' : A \to_{\sigma} B \sqsubseteq A' \to_{\sigma'} B' \\ \Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} N \sqsubseteq N' : A \sqsubseteq A'\end{array}}{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M\,N \sqsubseteq M'\,N' : B \sqsubseteq B'}$$

$$\mathtt{handle}\ V\,\{\mathtt{ret}\ y.M \mid \phi\} \equiv M[V/y] \qquad \begin{array}{c}\mathtt{handle}\ (\mathtt{let}\ o = \mathtt{raise}\ \varepsilon(x)\ \mathtt{in}\ N_k)\,\{\mathtt{ret}\ y.M \mid \phi\} \\ \equiv \phi(\varepsilon)[\lambda o.\mathtt{handle}\ N_k\,\{\mathtt{ret}\ y.M \mid \phi\}/k]\end{array}$$

$$\begin{array}{c}\mathtt{handle}\ M\,\{\mathtt{ret}\ x.N \mid \emptyset\} \\ \equiv \mathtt{let}\ x = M\ \mathtt{in}\ N\end{array} \qquad \frac{\begin{array}{c}\forall \varepsilon \in \mathrm{dom}(\phi).\ \psi(\varepsilon) = \phi(\varepsilon) \\ \forall \varepsilon \in \mathrm{dom}(\psi). \varepsilon \notin \mathrm{dom}(\phi) \Rightarrow \psi(\varepsilon) = k(\mathtt{raise}\ \varepsilon(x))\end{array}}{\mathtt{handle}\ M\,\{\mathtt{ret}\ y.N \mid \phi\} \equiv \mathtt{handle}\ M\,\{\mathtt{ret}\ y.N \mid \psi\}}$$

$$\frac{\begin{array}{c}A \leq A' \qquad B \leq B' \\ \Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq N : A\end{array}}{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} \langle B \leftarrowtail A \rangle M \sqsubseteq \langle B' \leftarrowtail A' \rangle N : B' \sqsubseteq B'} \qquad \frac{\begin{array}{c}A \leq A' \qquad B \leq B' \\ \Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq N : B \sqsubseteq B\end{array}}{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} \langle A' \twoheadleftarrow B' \rangle M \sqsubseteq \langle A \twoheadleftarrow B \rangle N : A' \sqsubseteq A'}$$

$$\frac{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq N : B \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} \langle A \twoheadleftarrow B \rangle M \sqsubseteq N : A \sqsubseteq B} \qquad \frac{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq N : A \sqsubseteq B}{\Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq \langle A \twoheadleftarrow B \rangle N : A \sqsubseteq A}$$

Fig. 10. Inequational Theory

value cast, is essential for proving the commutativity of value and effect casts, which is used in the derivation of the operational semantics and also valid in our logical relations model.

4.2 Operational Semantics

Next, we show a selection of the rules of the operational semantics $M \mapsto M'$ in Figure 11, eliding the standard call-by-value rules for booleans, functions and let-bindings. We capture the left-to-right, call-by-value evaluation order by using evaluation contexts defined in Section 3.1. First, we have the β rules for handlers. When a handler encloses a value, we execute the return clause. When a raise occurs, we search for the closest enclosing handler that handles the raised effect and capture the intermediate evaluation context in the continuation passed to the appropriate handler. We capture this with the relation $E'\#\varepsilon$ which says that the evaluation context does not handle the given operation.

The next rules concern the behavior of effect casts. First, all effect casts are the identity on values. Next, when upcasting a raise, we re-raise the effect, but *up*cast the request and *down*cast the response according to the types in the output effect type. An effect downcast works dually if the effect occurs in the result effect type. However, if the effect does not occur in the output effect type (which can only occur if the input effect type is ?), then an error is raised. Finally, we have the function downcast. Recall that a function cast applied to a value itself is a value, and only

$$E[\texttt{handle}\ V\ \{\texttt{ret}\ x.N \mid \phi\}] \mapsto E[N[V/x]]$$

$$\frac{\varepsilon \in \mathrm{dom}(\phi) \qquad E'\#\varepsilon}{\begin{array}{c}E[\texttt{handle}\ E'[\texttt{raise}\ \varepsilon(V)]\ \{\texttt{ret}\ x.N \mid \phi\}] \\ \mapsto E[\phi(\varepsilon)[V/x][(\lambda y.\texttt{handle}\ (E'[y])\ \{\texttt{ret}\ x.N \mid \phi\})/k]]\end{array}}$$

$$E[(\lambda x.M)V] \mapsto E[M[V/x]] \qquad E[\texttt{let}\ x = V\ \texttt{in}\ M] \mapsto E[M[V/x]]$$

$$E[\langle \sigma' \leftarrowtail \sigma \rangle V] \mapsto E[V] \qquad \frac{\varepsilon : A \leadsto B \in \sigma \qquad \varepsilon : A' \leadsto B' \in \sigma' \qquad E'\#\varepsilon}{\begin{array}{c}E[\langle \sigma' \leftarrowtail \sigma \rangle E'[\texttt{raise}\ \varepsilon(V)]] \mapsto \\ E[\texttt{let}\ x = \langle B \twoheadleftarrow B' \rangle \texttt{raise}\ \varepsilon(\langle A' \leftarrowtail A \rangle V)\ \texttt{in}\ \langle \sigma' \leftarrowtail \sigma \rangle E'[x]]\end{array}}$$

$$E[\langle \sigma \twoheadleftarrow \sigma' \rangle V] \mapsto E[V] \qquad \frac{\varepsilon : A \leadsto B \in \sigma \qquad \varepsilon : A' \leadsto B' \in \sigma' \qquad E'\#\varepsilon}{\begin{array}{c}E[\langle \sigma \twoheadleftarrow \sigma' \rangle E'[\texttt{raise}\ \varepsilon(V)]] \mapsto \\ E[\texttt{let}\ x = \langle B' \leftarrowtail B \rangle \texttt{raise}\ \varepsilon(\langle A \twoheadleftarrow A' \rangle V)\ \texttt{in}\ \langle \sigma \twoheadleftarrow \sigma' \rangle E'[x]]\end{array}}$$

$$\frac{\varepsilon \notin \sigma \qquad E'\#\varepsilon}{E[\langle \sigma \twoheadleftarrow ? \rangle E'[\texttt{raise}\ \varepsilon(V)]] \mapsto \mho} \qquad \begin{array}{c}E[(\langle (A \to_\sigma B) \twoheadleftarrow (A' \to_{\sigma'} B') \rangle V_f)\ V] \mapsto \\ E[\langle B \twoheadleftarrow B' \rangle \langle \sigma \twoheadleftarrow \sigma' \rangle (V_f \langle A' \leftarrowtail A \rangle V)]\end{array}$$

Fig. 11. Operational semantics of Core GrEff

reduces when applied to a value. When this occurs in a downcast, as shown, the result reduces to applying the original function to an *up*casted version of the input and *down*cast of the output, where this time we cast both value and effect types. Note the order of the value and effect casts on the output is arbitrarily chosen: because value casts only affect values and effect casts only affect effect operations, the two possible orders are equivalent. The elided cast for function upcasts is precisely dual, and finally there is a trivial cast rule for the identity cast on booleans.

We conclude the operational semantics with the following theorem, which establishes that the operational rules are all valid equational reasoning principles in any system that models the inequational theory.

Theorem 4.1. *If* $\cdot \vdash_\emptyset M, N : A$ *and* $M \mapsto N$ *then* $M \equiv N$ *is provable in the axiomatic semantics.*

The full proof is in the appendix [New et al. 2023], but we give an overview of how the behavior of effect casts $\langle \sigma \twoheadleftarrow \sigma' \rangle M$ is derived in particular. The core of the argument is to show that the downcast is equivalent to a particular handler, and then derive the operational reductions from the β reductions for handlers. The handler is $\langle \sigma \twoheadleftarrow \sigma' \rangle M \equiv \texttt{handle}\ M\ \{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}\}$ where the $\phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}$ handles precisely the effects in σ' and for each such $\varepsilon : A'_\sigma \leadsto B'_\sigma \in \sigma'$, the handling clause is defined as

$$\phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}(\varepsilon) = \begin{cases} \mho & \varepsilon \notin \mathrm{dom}(\sigma) \\ k(\langle B'_\sigma \leftarrowtail B_\sigma \rangle \texttt{raise}\ \varepsilon(\langle A_\sigma \twoheadleftarrow A'_\sigma \rangle x)) & \varepsilon : A_\sigma \leadsto B_\sigma \in \sigma \end{cases}$$

That is, if the effect is not present in σ, the handler errors, and otherwise it re-raises the effect to its context with mediating casts. The raising party raises a request value x of type $A_{\sigma'}$ and expects a response of type $B_{\sigma'}$, but the ambient handler expects ε requests to have type A_σ and

$$\begin{array}{ccccc}
A_h & \sqsubseteq & D_h & \sqsupseteq & B \\
\text{\rotatebox{90}{$\leq$}} & & \text{\rotatebox{90}{$\leq$}} & & \text{\rotatebox{90}{$\leq$}} \\
A & \sqsubseteq & D_l & \sqsupseteq & B_l
\end{array}$$

Fig. 12. Situation derivable from $A \lesssim B$

responds with B_σ values, so when re-raising, we need to *down*cast the request and upcast the resulting response which is then passed to the original continuation k. Then we show that $\langle \sigma \twoheadleftarrow \sigma' \rangle M \equiv \texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}\}$ by showing an ordering each way. For the $\langle \sigma \twoheadleftarrow \sigma' \rangle M \sqsubseteq \texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}\}$ case, we apply the effect forwarding principle to transform the left-hand side to $\texttt{handle}\ \langle \sigma \twoheadleftarrow \sigma' \rangle M\,\{\texttt{ret}\ x.x \mid \phi_\sigma\}$. Then we apply congruence for handlers, with the cases of the right-hand side that handle effects not in σ being irrelevant. Then the remaining clauses are all of the same syntactic structure except for upcasts and downcasts, and so the proof follows by congruence and the upcast/downcast rules. To show $\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}\} \sqsubseteq \langle \sigma \twoheadleftarrow \sigma' \rangle M$, we first apply the downcast right rule to eliminate the cast on the right. Then to show $\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}\} \sqsubseteq M$ we again use the effect forwarding principle to rewrite the right-hand side as $\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\sigma'}\}$. We again apply handler congruence, with the cases where $\varepsilon \in \sigma$ analogous to the prior argument. In the remaining cases $\phi_{\langle \sigma \twoheadleftarrow \sigma' \rangle}(\varepsilon) \sqsubseteq \phi_{\sigma'}(\varepsilon)$ where $\varepsilon \notin \sigma$, we have the left hand side is an error, and so the argument follows by the fact that the error is the minimum in the ordering.

While the converse of Theorem 4.1 is not literally true that equivalent terms reduce to each other operationally, the graduality proof in Section 5 does imply that if $M \equiv N$ then M and N are contextually equivalent with respect to the operational semantics.

4.3 Subtyping, Gradual Subtyping and Coercions

The elaboration defined in Section 3.2 inserts casts of the form $\langle A \twoheadleftarrow |A| \rangle \langle |B| \leftarrowtail B \rangle M$ when a gradual subtyping $A \lesssim B$ is used in the type-checker. If we think of $|A|$ as the type of programs in the untracked language, this says to cast a program from one type to another, we should cast it to an untracked type and then to the other effect-tracking type, similar to prior work on cast calculi based on upcasts and downcasts [New and Ahmed 2018]. This is a reasonable cast if we think of the untracked language as our "operational ground truth", and so we should prove that any other translation is extensionally equivalent to this one. However, operationally, this can be quite a wasteful translation, as a cast can result in proxying at runtime, while *subtyping* coercions have no runtime behavior, and so are zero cost. For instance, if $A \lesssim B$ is true because in fact $A \leq B$, then there need not be any runtime cast at all. For this reason, we would prefer to optimize the cast based on the subtyping information in the proof of $A \lesssim B$. Since A may be more imprecise than B in some subterms and vice-versa, the structure of the cast should still be an upcast followed by a downcast, but with the possibility that we use implicit subtyping coercions at some points. There are three places we might insert the implicit subtyping coercion: before the upcast, between the upcast and downcast and after the downcast. From the proof of $A \lesssim B$, we can extract types and subtyping/precision derivations as in Figure 12.

On the left we have a "pure subtyping" component of the gradual subtpying proof coming from A, and on the right we we have the pure subtyping component coming from B. In the middle we have two "dynamic" types also related by subtyping. There are then three paths from A to B in this diagram, which generate three different potential casts with implicit subtyping coercions ensuring they are well-typed as taking A to B: (1) Up and then right twice $\langle B \twoheadleftarrow D_h \rangle \langle D_h \leftarrowtail A_h \rangle$ (2) Right, up and then right: $\langle B \twoheadleftarrow D_h\rangle\langle D_l \leftarrowtail A\rangle$ (3) Right twice and then up: $\langle B_l \twoheadleftarrow D_l\rangle\langle D_l \leftarrowtail A\rangle$. Fortunately we can choose whichever is operationally preferable: each of these casts is equivalent as a function from A to B and they are all equivalent to the ground truth cast $\langle B \twoheadleftarrow |A|\rangle\langle |A| \leftarrowtail A\rangle$. The above discussion applies equally well to effect casts, which are even simpler in that the "ground-truth" always factors through the single most imprecise effect type: the dynamic effect type.

5 SOUNDNESS AND GRADUALITY

In this section we establish that the axiomatic semantics of core GrEff has a sound model in terms of its operational semantics. This establishes two key properties: equivalent terms ($M \equiv N$) are contextually equivalent in the operational semantics, and the graduality property holds. First, we review the definition of the graduality property, and then we give a logical relations model and prove that any provable inequality $M \sqsubseteq N$ implies that the terms are related in the logical relation.

5.1 Static and Dynamic Gradual Guarantees

GrEff is designed to support a smooth *migration* from imprecise to precise typing. The static gradual guarantee [Siek et al. 2015] formalizes a syntactic element of this idea of a smooth migration. The static gradual guarantee informally says that increasing the precision of type annotations on a program can only make it *harder* to satisfy the static type checker, or viewed the other way around, decreasing the precision of type annotations can only make it *easier* to satisfy the static type checker. Then the dynamic gradual guarantee, also known as *graduality*, establishes the semantic counterpart: increasing the precision of type annotations on a program should only make it *harder* to terminate without a *dynamic* type error, and furthermore except where there are dynamic type errors, the behavior of the program should match the original. These properties can be formalized as a form of *monotonicity* of the elaboration of the syntactic programs of surface GrEff into the semantically meaningful core GrEff programs as follows. First, we define a syntactic term precision ordering $\sqsubseteq^{\mathrm{syn}}$ on untyped GrEff programs as the congruence closure of the type precision ordering. Then the static gradual guarantee says that this is a monotone *partial* function from the syntactic term precision ordering to the axiomatic inequality on core GrEff terms:

Theorem 5.1 (Static Gradual Guarantee). *If $P \sqsubseteq^{syn} P'$, then if $\cdot \mid \cdot \vdash P \Rightarrow \Sigma \vdash_\sigma M : A$, then there exist M', σ', A' such that $\cdot \mid \cdot \vdash P' \Rightarrow \Sigma \vdash_{\sigma'} M' : A'$ such that $\cdot \vdash_{\sigma \sqsubseteq \sigma'} M \sqsubseteq M' : A \sqsubseteq A'$.*

Then the dynamic gradual guarantee says that this extends to monotonicity in the following *semantic* ordering on core GrEff terms:

Definition 5.2 (Error Ordering on Closed Programs). Given $\cdot \vdash_\emptyset M, M' : \mathtt{bool}$, define $M \sqsubseteq^{\mathrm{sem}} M'$ to hold when one of the following is satisfied (1) $M \mapsto^* \mho$, (2) $M \Uparrow$ and $M' \Uparrow$, (3) $M \mapsto^* \mathtt{true}$ and $M' \mapsto^* \mathtt{true}$ (4) $M \mapsto^* \mathtt{false}$ and $M' \mapsto^* \mathtt{false}$.

Theorem 5.3 (Dynamic Gradual Guarantee). *If $\Sigma \mid \cdot \vdash_{\emptyset \sqsubseteq \emptyset} M \sqsubseteq M' : \mathtt{bool}$, then $M \sqsubseteq^{sem} M'$.*

This theorem is stated in terms of closed terms of a fixed type, but to prove it we need a stronger inductive hypothesis, i.e., the logical relation for open terms. The resulting theorem that any inequality provable in the theory implies the semantic ordering is called *graduality*, as it is analogous in structure to the *parametricity* theorem in parametric polymorphism. Then the dynamic gradual guarantee follows as a corollary.

5.2 Logical Relation

We begin by introducing the notion of *precision derivations* in Figure 13, which will be used extensively in the definition of the logical relation. A derivation $c : A \sqsubseteq A'$ represents a proof

$$\Sigma \vdash \mathtt{bool} : \mathtt{bool} \sqsubseteq \mathtt{bool} \qquad \frac{\Sigma \vdash d_i : A \sqsubseteq A' \qquad \Sigma \vdash d_e : \sigma \sqsubseteq \sigma' \qquad \Sigma \vdash d_o : B \sqsubseteq B'}{\Sigma \vdash d_i \to_{d_e} d_o : A \to_\sigma B \sqsubseteq A' \to_{\sigma'} B'}$$

$$\Sigma \vdash ? : ? \sqsubseteq ? \qquad \frac{\begin{array}{c}\mathrm{supp}(d_c) = \mathrm{supp}(\sigma_c) = \mathrm{supp}(\sigma'_c) \\ (\forall \varepsilon : c \rightsquigarrow d \in d_c, \varepsilon : A \rightsquigarrow B \in \sigma_c, \varepsilon : A' \rightsquigarrow B' \in \sigma'_c. \\ \Sigma \vdash c : A \sqsubseteq A' \qquad \Sigma \vdash d : B \sqsubseteq B')\end{array}}{\Sigma \vdash d_c : \sigma_c \sqsubseteq \sigma'_c} \qquad \frac{\Sigma \vdash d_c : \sigma_c \sqsubseteq \Sigma|_{\mathrm{supp}(\sigma_c)}}{\Sigma \vdash \mathtt{inj}(d_c) : \sigma_c \sqsubseteq ?}$$

$$\frac{\varepsilon : c \rightsquigarrow d \in \Sigma}{\Sigma \vdash \varepsilon : c \rightsquigarrow d \in ?} \qquad \frac{\Sigma \vdash \varepsilon : c' \rightsquigarrow d' \in d_c \qquad c = \mathtt{inj}(c') \qquad d = \mathtt{inj}(d')}{\Sigma \vdash \varepsilon : c \rightsquigarrow d \in \mathtt{inj}(d_c)}$$

Fig. 13. Type and Effect Precision Derivations

that $A \sqsubseteq A'$, and is built up inductively using the rules in the figure. Likewise, $d_\sigma : \sigma \sqsubseteq \sigma'$ is an inductively constructed proof witnessing the fact that σ is more precise than σ'. The benefit to making these derivations explicit in the syntax is that we can perform induction over them. As part of the definition of effect precision derivation, we use the notion of an effect operation being "in" a precision derivation $\varepsilon : c \rightsquigarrow d \in d_c$. For when d_c itself is a partial function this is just as with earlier usage, but when $d_c = ?$ or $d_c = \mathtt{inj}(d'_c)$ we use the definition at the bottom of the figure.

The assignment of derivations to type and effect precision given in Figure 13 is equivalent to the definition of precision given in Figure 5, in the sense that the choice does not affect provability:

Lemma 5.4 (Correctness of Precision Derivation Assignment). *Assuming $\Sigma \vdash A$ and $\Sigma \vdash B$, the following are equivalent*

- *$A \sqsubseteq A'$ is provable in the system in Figure 5*
- *There exists a derivation $\Sigma \vdash c : A \sqsubseteq A'$ in the system in Figure 13.*

Similarly, assuming $\Sigma \vdash \sigma$ and $\Sigma \vdash \sigma'$, the following are equivalent

- *$\sigma \sqsubseteq \sigma'$ is provable in the system in Figure 5*
- *There exists a derivation $\Sigma \vdash c_e : \sigma \sqsubseteq \sigma'$ in the system in Figure 13.*

We also have that precision derivations are unique if they exist:

Lemma 5.5 (Uniqueness of Precision Derivations). *If $A \sqsubseteq B$, then there is exactly one value type precision derivation c such that $c : A \sqsubseteq B$. Likewise, if $\sigma \sqsubseteq \sigma'$, then there is exactly one effect type precision derivation d_σ such that $d_\sigma : \sigma \sqsubseteq \sigma'$.*

The definition of the logical relation is given in Figure 15. Following prior work on logical relations for graduality, the relation is indexed not by types, but by type precision derivations. For a type precision derivation c, define c^l and c^r to be the types such that $c : c^l \sqsubseteq c^r$, and analogously for effect types.

Figure 14 defines the notions of well typed value, term, and evaluation-context atoms. These are used in the definition of the step-indexed logical relation for graduality. Given a value type precision derivation c, the set VAtom c consists of pairs of values (V_1, V_2) such that V_1 has type c^l and V_2 has type c^r. Similarly, given types A^l and A^r and an effect type precision derivation d_σ, the set TAtom $A^l\ A^r\ d_\sigma$ consists of pairs of terms (M_1, M_2) with value types A^l and A^r and effect types d^l_σ and d^r_σ, respectively. An evaluation context can be thought of as a term with a hole, which when filled yields another term. For our purposes, an evaluation context corresponds to a continuation that accepts a *value* and returns a term. The type of the hole is the type of the input value to the continuation. The set $\mathrm{ECtxAtom}\, c\, (\sigma^l \,!\, A^l)\, (\sigma^r \,!\, A^r)$ consists of pairs of such evaluation contexts whose input value types are c^l and c^r respectively, and whose output value and effect types are A^l, A^r and σ^l, σ^r, respectively.

$$\begin{array}{rcl}
\mathrm{VAtom}\, c & := & \{(V^l, V^r) : \mathtt{val}(V^l) \wedge \mathtt{val}(V^r) \wedge \\
& & \quad (\Sigma \mid \cdot \mid \cdot \vdash_{\emptyset} V^l : c^l) \wedge (\Sigma \mid \cdot \mid \cdot \vdash_{\emptyset} V^r : c^r)\} \\[1em]
\mathrm{TAtom}\, A^l\, A^r\, d_\sigma & := & \{(M^l, M^r) : \\
& & \quad (\Sigma \mid \cdot \mid \cdot \vdash_{d^l_\sigma} M^l : A^l) \wedge (\Sigma \mid \cdot \mid \cdot \vdash_{d^r_\sigma} M^r : A^r)\} \\[1em]
\mathrm{ECtxAtom}\, c\, (\sigma^l \,!\, A^l)\, (\sigma^r \,!\, A^r) & := & \{(x^l.M^l, x^r.M^r) : \\
& & \quad (\Sigma \mid x^l : c^l \mid \cdot \vdash_{\sigma^l} M^l : A^l) \wedge (\Sigma \mid x^r : c^r \mid \cdot \vdash_{\sigma^r} M^r : A^r)\}
\end{array}$$

Fig. 14. Well typed atoms

$$\begin{array}{rcl}
(M_1, M_2) \in (\blacktriangleright R)_j & \iff & j = 0 \vee (j = k + 1 \wedge (M_1, M_2) \in R_k) \\
(V_1, V_2) \in \mathcal{V}^{\sim}_j[\![\mathtt{bool}]\!] & \iff & (V_1, V_2) \in \mathrm{VAtom}\, \mathtt{bool} \wedge \\
& & (V_1 = V_2 = \mathtt{true}) \vee (V_1 = V_2 = \mathtt{false}) \\
(V_1, V_2) \in \mathcal{V}^{\sim}_j[\![d_i \to_{d_\sigma} d_o]\!] & \iff & (V_1, V_2) \in \mathrm{VAtom}\, (d_i \to_{d_\sigma} d_o) \wedge \\
& & \forall k \le j. \forall (V_{i1}, V_{i2}) \in \mathcal{V}^{\sim}_k[\![d_i]\!]. \\
& & (V_1\, V_{i1}, V_2\, V_{i2}) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!](\mathcal{V}^{\sim}[\![d_o]\!]) \\
(M_1, M_2) \in \mathcal{E}^{\preceq}_j[\![d_\sigma]\!](R, A^l, A^r) & \iff & (M_1, M_2) \in \mathrm{TAtom}\, A^l\, A^r\, d_\sigma \wedge (M_1 \mapsto^{j+1} \\
& & \vee (\exists k \le j.\ (M_1 \mapsto^{j-k} \mho) \\
& & \quad \vee (\exists (N_1, N_2) \in \mathcal{R}^{\preceq}_k[\![d_\sigma]\!] R \wedge \\
& & \quad\quad M_1 \mapsto^{j-k} N_1 \wedge M_2 \mapsto^* N_2))) \\
(M_1, M_2) \in \mathcal{E}^{\succeq}_j[\![d_\sigma]\!](R, A^l, A^r) & \iff & (M_1, M_2) \in \mathrm{TAtom}\, A^l\, A^r\, d_\sigma \wedge (M_2 \mapsto^{j+1} \\
& & \vee (\exists k \le j. (M_2 \mapsto^{j-k} \mho \wedge M_1 \mapsto^* \mho) \\
& & \quad \vee (\exists N_2. M_2 \mapsto^{j-k} N_2 \wedge M_1 \mapsto^* \mho) \\
& & \quad \vee (\exists (N_1, N_2) \in \mathcal{R}^{\succeq}_k[\![d_\sigma]\!] R \wedge \\
& & \quad\quad M_2 \mapsto^{j-k} N_2 \wedge M_1 \mapsto^* N_1))) \\
(M_1, M_2) \in \mathcal{R}^{\sim}_j[\![d_\sigma]\!](R, A^l, A^r) & \iff & (M_1, M_2) \in \mathrm{TAtom}\, A^l\, A^r\, d_\sigma \wedge \\
& & ((\mathtt{val}(M_1) \wedge \mathtt{val}(M_2) \wedge (M_1, M_2) \in Rj) \\
& & \quad \vee (\exists \epsilon : c \rightsquigarrow d \in d_\sigma, E^l \# \epsilon, E^r \# \epsilon, V^l, V^r. \\
& & \quad (V^l, V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![c]\!])_j \wedge \\
& & \quad (x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright \mathcal{K}^{\sim}[\![d]\!])_j \\
& & \quad\quad (\mathcal{E}^{\sim}[\![d_\sigma]\!](R, A^l, A^r), (d^l_\sigma \,!\, A^l), (d^r_\sigma \,!\, A^r)) \wedge \\
& & \quad M_1 = E^l[\mathtt{raise}\ \epsilon(V^l)] \wedge M_2 = E^r[\mathtt{raise}\ \epsilon(V^r)])) \\
(x^l.M^l, x^r.M^r) \in & \iff & (M^l, M^r) \in \mathrm{ECtxAtom}\, c\, (\sigma^l \,!\, A^l)\, (\sigma^r \,!\, A^r) \wedge \\
\mathcal{K}^{\sim}_j[\![c]\!](S, (\sigma^l \,!\, A^l), (\sigma^r \,!\, A^r)) & & \forall k \le j. (V^l, V^r) \in \mathcal{V}^{\sim}_k[\![c]\!]. \\
& & \quad (M^l[V^l/x^l], M^r[V^r/x^r]) \in S_k \\
(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma^{\sqsubseteq}]\!] & \iff & \forall (x_1 \sqsubseteq x_2 : c) \in \Gamma^{\sqsubseteq}. (\gamma_1(x_1), \gamma_2(x_2)) \in \mathcal{V}^{\sim}_j[\![c]\!]
\end{array}$$

Fig. 15. Logical Relation

At first glance, it may seem as though we do not need to employ step-indexing in the logical relation. That is, it might seem that we could simply define the relation by induction on the structure of the derivation of $A \sqsubseteq A'$. However, this would not suffice — there is indeed recursion in the logical relation, specifically in the *result* relation $\mathcal{R}^{\sim}[\![\cdot]\!]$. This is discussed further below.

Given a step-indexed relation R, we define an operator $\blacktriangleright R$ (pronounced "later R") as follows: Terms M_1 and M_2 are related in $\blacktriangleright R$ at index n if and only if either n is zero, or $n \geq 1$ and M_1 and M_2 are related in R at index $n-1$.

Many of the details of the logical relation are similar to prior work, especially [New et al. 2020], so we highlight the handling of effect types, which is novel. In addition to the usual expression and value relations $\mathcal{E}^{\sim}[\![\cdot]\!]$ and $\mathcal{V}^{\sim}[\![\cdot]\!]$, we have a *result* relation $\mathcal{R}^{\sim}[\![\cdot]\!]$ and a *continuation* relation $\mathcal{K}^{\sim}[\![\cdot]\!]$. In our language, a **result** is either a value, or an evaluation context E wrapping a raise of an operation ε, such that $E\#\varepsilon$. The result relation specifies the conditions for two such results to be related.

Each of the relations is parameterized by a precision derivation. In the case of the expression and result relations, this is an effect precision derivation, while for values and continuations, it is a value type precision derivation. This is analogous to the usual approach whereby a logical relation is indexed by a type. But instead of using types, we use precision derivations, i.e., the proof that the type of the LHS term is more precise than the type of the RHS term. These derivations are used implicitly to constrain the types of the LHS and RHS terms. For instance, in the value relation for function types, the requirement that $(V_1, V_2) \in \mathrm{VAtom}\, d_i \rightarrow_{d_\sigma} d_o$ ensures not only that V_1 and V_2 have function type, but that the type of V_1 is more precise than the type of V_2.

As in previous work on logical relations for graduality, the expression logical relation $\mathcal{E}^{\sim}[\![\cdot]\!]$ is split into two relations $\mathcal{E}^{\preceq}[\![\cdot]\!]$ and $\mathcal{E}^{\succeq}[\![\cdot]\!]$. The former counts the steps taken by the left-hand term, while the latter counts steps taken by the right-hand term. This is captured by the quantitative small-step reduction $M \mapsto^{j} N$ which means M takes exactly j steps to reduce to N. The other logical relations are also split into two versions in the same way. Despite needing two one-sided versions of each relation, we are for the most part able to abstract over their differences: most of the lemmas we prove hold for both versions with no adjustment needed. Notable exceptions are transitivity and the anti- and forward reduction lemmas: these lemmas make crucial use of step counting, so naturally the side whose steps we are counting makes a difference.

We note that in the definition of the value relation for function types, $\mathcal{V}^{\sim}[\![d_o]\!]$ without the step-index should be interpreted as a partial application, i.e., it is a function from step indices to relations.

For the sake of clarity, we briefly outline the definition of the two one-sided expression relations. In both relations, the first clause is a "time-out" condition. In the case when we're counting steps on the left (i.e., $\mathcal{E}^{\preceq}[\![\cdot]\!]$), this states that if M_1 takes $j+1$ or more steps, then it is automatically related at step index j to M_2. An analogous rule holds when counting steps on the right: if M_2 takes $j+1$ or more steps, then it is related to M_1 at step index j. The next clauses relate to errors. In the case of $\mathcal{E}^{\preceq}[\![\cdot]\!]$, if M_1 errors in at most j steps, then it is related to M_2 regardless of the behavior of M_2. This models the axiom that error is the most precise term. In the case of $\mathcal{E}^{\succeq}[\![\cdot]\!]$, if M_2 errors in at most j steps, we ensure that M_1 *also* errors (in any number of steps, since we're counting steps on the right). We also allow for the case where M_2 reduces to a result in at most j steps, and M_1 errors. An equivalent way to phrase these rules that clarifies the similarity between the two versions of the expression relation is that *if* M_1 errors (in any number of steps), then it is related to M_2 in $\mathcal{E}^{\succeq}[\![\cdot]\!]\,j$ provided that M_2 steps in at most j steps to either an error, or a result. Finally, the last clauses concern the case when both M_1 and M_2 step to results, where as usual in $\mathcal{E}^{\preceq}[\![\cdot]\!]\,j$ we require that M_1 takes at most j steps and in $\mathcal{E}^{\succeq}[\![\cdot]\!]\,j$ we require that M_2 takes at most j steps.

In both cases, we check that the results to which they step are related in the result relation at the appropriate step index.

One novel aspect of our logical relation is the *result* relation $\mathcal{R}^{\sim}[\![d_\sigma]\!]$. The result relation relates terms M_1 and M_2 – of type A^l and A^r respectively – representing either two values or two "evaluations" of raised operations. The result relation is parameterized by a step-indexed relation R between *values* of type A^l and A^r (the types of M_1 and M_2). (Ultimately, R will end up being instantiated as $\mathcal{V}^{\sim}[\![c]\!]$ for some c.) M_1 and M_2 are related by $\mathcal{R}^{\sim}[\![d_\sigma]\!]$ at step index j when either (1) both terms are values and are related by R at index j, or (2) there exists an effect $\varepsilon : c \rightsquigarrow d$ in d_σ, values V^l and V^r related *later*, and evaluation contexts (i.e., continuations – see below) E^l and E^r related *later*, such that M_1 is equal to raising the effect and then wrapping it in the continuation, and likewise for M_2. Recall that d_σ is an effect precision derivation; "membership" in such a derivation is defined inductively on the structure of the derivation (the formal definition is given in the appendix [New et al. 2023]).

Observe that the result relation is recursive: If d_σ is the dynamic effect type ? then the definitions of c and d may in general include ? in them. Thus, in order to maintain well-foundedness, when we refer to the value and continuation relations in this part of the definition we need to "decrement the step index" (hence the use of the later operator).

The relation $\mathcal{K}^{\sim}[\![\cdot]\!]$ relates evaluation contexts E_1 and E_2, similar to prior work on logical relations for continuations [Asai 2005]. As mentioned above, evaluation contexts represent continuations that accept values. To enforce that the continuations accept *values* only, and not arbitrary terms, the inputs to the continuation relation are actually *terms* M^l and M^r with free variables x^l and x^r, respectively. E_1 and E_2 also have "output" types (A^l and A^r) and "output" effect sets (σ^l and σ^r). When values are plugged into E_1 and E_2, the result is two terms having types A^l and A^r and effect sets σ^l and σ^r, respectively.

5.3 Proof of Graduality

Our goal is to prove that the inequational theory is sound with respect to the logical relation. First we define the notion of two terms being related semantically:

$$\Gamma^{\sqsubseteq} \vDash_{d_\sigma} M_1 \sqsubseteq M_2 \in c := \forall \sim \in \{\preceq, \succeq\}. \forall j \in \mathbb{N}. \forall (\gamma_1, \gamma_2) \in \mathcal{V}_j^{\sim}[\![\Gamma]\!]. (M_1[\gamma_1], M_2[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

That is, M_1 and M_2 are related if for all j and all substitutions of values γ_1 and γ_2 related at j, the resulting terms are related in $\mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$, where this needs to hold both when $\sim$ is $\preceq$ and when it is $\succeq$. Our goal is then to prove the following:

Theorem 5.6 (Graduality). *If* $\Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq N : c$ *then* $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} M \sqsubseteq N \in c$

We provide here a high-level overview of the proof; the complete proofs are in the appendix [New et al. 2023]. We begin by establishing variants of standard anti- and forward-reduction lemmas as well as monadic bind. We also prove a Löb induction principle to structure the induction over step-indices. With these lemmas, we first prove soundness of each of the congruence rules for term precision, by uses of the monadic bind lemma along with the reduction lemmas. Next, we prove soundness of the rules of the equational theory, e.g., the β and η laws, and transitivity. Finally, we prove soundness of the rules for casts and subtyping.

6 DISCUSSION

Prior Work on Gradual Effects. The most significant prior work on gradual effects is the work of Bañados Schwerter and collaborators [Bañados Schwerter et al. 2014], who defined a gradual effect system based on the generic effect calculus of Marino and Millstein [2009] using an early version of the *abstracting gradual typing* (AGT) framework for gradual type systems[Garcia et al.

2016]. While we based GrEff on effect handlers rather than the generic effect calculus, there are significant similarities in the typing: function types and typing judgments are indexed by a set of effect operations in each system. The most significant syntactic difference is that their framework is parameterized by a fixed effect theory, whereas GrEff has explicit support for declaration of new effects in the program. In particular, this means that their system does not need to support modules containing different views of the same nominal effect as we did. They additionally support a form of partially tracked functions, in GrEff syntax this would look like $A \to_{\varepsilon,?} B$, a function type where the function is known specifically to possibly raise the effect ε in addition to raising other effects. In GrEff this partial tracking would ensure that any effects raised with the name ε match the module's local view of the effect typing of ε. Finally, on the semantic side, this prior work proves only a type safety proof, whereas here we have proven graduality and the correctness of type-based optimizations and handler optimizations.

Another related area of research is on gradual typing with delimited continuations, which are mutually expressible with effect handlers [Forster et al. 2019; Piróg et al. 2019]. Takikawa and co-authors propose a gradual type system and semantics via contracts for a language with delimited continuations using typed prompts [Takikawa et al. 2013]. They consider only value types and untracked function types that do not say which prompts are expected to be present. They show that a naive contract based implementation is unsound because a dynamically typed program can interact with a typed prompt and therefore the prompts themselves must be equipped with contracts, even though it does not correspond to any value being imported. In core GrEff, this unsoundness is ruled out by using intrinsic typing: the problem corresponds to raising an effect operation with a different type than the type expected by the closest handler, which is precisely what the effect type system tracks. Wrapping the prompt in contracts is behaviorally equivalent to what is achieved by our effect type casts. Sekiyama, Ueda and Igarashi present a blame calculus for a language with shift and reset [Sekiyama et al. 2015]. The blame calculus is analogous to our core GrEff language, and uses a type and effect system for the answer types of shift/reset. They do not develop a surface language that elaborates to this blame calculus like our GrEff, and there is no analog of effect operations in shift/reset-based systems so there are no nominal aspects of their language. Additionally, while they have an effect system to keep track of answer types, they do not have effect casts.

Prior Approaches to Gradual Nominal Datatypes. We are also not the first to consider the combination of gradual and nominal typing. The closest match to our design is in Typed Racket's support for typed structs. In Racket, a struct is a kind of record type that (by default) is generative in that it creates a new type tag distinct from all others. Typed Racket supports import of untyped Racket structs into Typed Racket, where types are assigned to the fields, and values of the struct type are then wrapped in contracts accordingly. This is quite close to our treatment of nominal effect operations which can be thought of as adding new cases to the dynamic effect monad rather than dynamic type. Our type system is more complex however, since in our system modules can use dynamically typed effects whereas in Typed Racket, there is no syntactic type for dynamically typed values, when imported into typed code the system must give a completely precise type. Malewski and co-authors present a design for gradual typing with nominal algebraic datatypes [Malewski et al. 2021]. Their focus is on the gradual migration from datatypes whose cases are open-ended to datatypes with a fixed set of constructors. They do not consider the use-case we have where different modules have different typings for the same nominal constructor.

Prior Work on Subtyping. Much prior work on incorporating subtyping with gradual types has focused on the static typing aspects [Castagna et al. 2019; Garcia and Cimini 2015; Siek and Taha 2007; Wadler and Findler 2009]. The most significant prior semantic work on subtyping and gradual typing is the Abstracting Gradual Typing work [Garcia et al. 2016] which proves the dynamic gradual guarantee for a system with subtyping developed using the AGT methodology. In this work we establish equivalence between multiple different ways to combine gradual type casts and subtyping coercions, summarized in Figure 12, which are derivable from our newly identified cast/coercion ordering principle in our equational theory (Figure 10).

Towards a Practical Language Design. GrEff is intended as a proof-of-concept language design to provide the semantic foundation for extending a language such as OCaml 5 with gradual effect typing. We discuss the current mismatches with OCaml's design and how these might be rectified. First, OCaml uses *extensible variant* types for effects and exceptions, whereas in GrEff effects are not first-class values. This should not be difficult to support as the variant type can be treated somewhat similarly to a dynamic type. Next, OCaml supports *recursive* effect types, meaning that the request or response of an effect can refer to the effect being defined. For instance, this allows for a variant of our coroutine example where forked threads can fork further threads. This would complicate the metatheory of GrEff but should work in principle. The logical relation already supports a form of recursive effect type in the form of the dynamic type, and so this could be extended to arbitrary recursive definitions using step-indexing in a similar fashion. A final syntactic difference is that OCaml is based on Hindley-Milner-style polymorphic type schemes, whereas GrEff is based on a simple type system. It may be possible to adapt previous work for gradual typing in unification-based type systems[Castagna et al. 2019; Garcia and Cimini 2015; Siek and Vachharajani 2008].

Implementing gradual effects brings its own challenges. Our derivation of the operational semantics is based on proving that effect casts can be implemented as handlers, and so can be implemented by a source-to-source transformation. However, such an implementation may suffer from similar performance issues as other naive wrapper semantics, which can be solved by defunctionalizing the casts [Herman et al. 2010]. Additionally, strong gradual typing between fully dynamically typed and static code can result in high performance penalties [Takikawa et al. 2016] even with space efficient implementations. However since effect casts would not be as pervasive in typical programs as value type casts, it is not obvious that the same pathological behaviors would arise in gradually effect typed OCaml programs. This is a clear empirical question to be addressed in future work.

Guarded Recursion as an alternative to Explicit Step-Indexing. The later operator $\blacktriangleright$ was originally studied by Nakano [Nakano 2000] as a modality for expressing guarded recursive types and this has been used along with the principle of Löb-induction ($(\blacktriangleright P \Rightarrow P) \Rightarrow P$) to develop domain-specific logics for step-indexed logical relations [Dreyer et al. 2009]. This allows for proofs to be carried out without explicit reference to step indices. More generally, the mathematical area of *synthetic guarded domain theory* (SGDT) has extended this approach from higher-order logic to a full modal dependent type theory [Bahr et al. 2017; Birkedal et al. 2011]. Such an approach might considerably simplify the construction of a logical relations model by avoiding the explicit threading of steps, at the cost of using a non-standard meta-logic, and so would be an interesting avenue for future work. However, it is not clear how to adapt the final graduality property from Section 5.3, which quantifies over all step indices to this setting.

REFERENCES

Kenichi Asai. 2005. Logical relations for call-by-value delimited continuations. In *Revised Selected Papers from the Sixth Symposium on Trends in Functional Programming, TFP 2005, Tallinn, Estonia, 23-24 September 2005 (Trends in Functional Programming, Vol. 6)*, Marko C. J. D. van Eekelen (Ed.). 63–78.

Felipe Bañados Schwerter, Ronald Garcia, and Éric Tanter. 2014. A Theory of Gradual Effect Systems. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming* (Gothenburg, Sweden) *(ICFP '14)*. 283–295. https://doi.org/10.1145/2692915.2628149

Patrick Bahr, Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. 2017. The clocks are ticking: No more delays!. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–12. https://doi.org/10.1109/LICS.2017.8005097

Lars Birkedal, Rasmus Ejlers Mogelberg, Jan Schwinghammer, and Kristian Stovring. 2011. First Steps in Synthetic Guarded Domain Theory: Step-Indexing in the Topos of Trees. In *lics11*. https://doi.org/10.1109/LICS.2011.16

Jonathan Immanuel Brachthäuser, Philipp Schuster, and Klaus Ostermann. 2020. Effekt: Capability-passing style for type- and effect-safe, extensible effect handlers in Scala. *J. Funct. Program.* 30 (2020), e8. https://doi.org/10.1017/S0956796820000027

Giuseppe Castagna, Victor Lanvin, Tommaso Petrucciani, and Jeremy G. Siek. 2019. Gradual Typing: A New Perspective. *Proc. ACM Program. Lang.* 3, POPL, Article 16 (jan 2019), 32 pages. https://doi.org/10.1145/3290329

WasmFX Contributors. [n.d.]. WasmFX: Effect Handlers for WebAssembly. https://wasmfx.dev/ Accessed: 2020-11-10.

Ezra Cooper, Sam Lindley, Philip Wadler, and Jeremy Yallop. 2006. Links: Web Programming Without Tiers. In *Formal Methods for Components and Objects, 5th International Symposium, FMCO 2006, Amsterdam, The Netherlands, November 7-10, 2006, Revised Lectures (Lecture Notes in Computer Science, Vol. 4709)*. 266–296. https://doi.org/10.1007/978-3-540-74792-5_12

Derek Dreyer, Amal Ahmed, and Lars Birkedal. 2009. Logical Step-Indexed Logical Relations. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*. 71–80. https://doi.org/10.1109/LICS.2009.34

Yannick Forster, Ohad Kammar, Sam Lindley, and Matija Pretnar. 2019. On the expressive power of user-defined effects: Effect handlers, monadic reflection, delimited control. *J. Funct. Program.* 29 (2019), e15. https://doi.org/10.1017/S0956796819000121

Ronald Garcia and Matteo Cimini. 2015. Principal Type Schemes for Gradual Programs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, Sriram K. Rajamani and David Walker (Eds.). ACM, 303–315. https://doi.org/10.1145/2676726.2676992

Ronald Garcia, Alison M. Clark, and Éric Tanter. 2016. Abstracting Gradual Typing. In *ACM Symposium on Principles of Programming Languages (POPL)*. https://doi.org/10.1145/2837614.2837670

David Herman, Aaron Tomb, and Cormac Flanagan. 2010. Space-Efficient Gradual Typing. *Higher Order Symbol. Comput.* 23, 2 (jun 2010), 167–189. https://doi.org/10.1007/s10990-011-9066-z

Oleg Kiselyov, Amr Sabry, and Cameron Swords. 2013. Extensible effects: an alternative to monad transformers. In *Proceedings of the 2013 ACM SIGPLAN Symposium on Haskell, Boston, MA, USA, September 23-24, 2013*. ACM, 59–70. https://doi.org/10.1145/2503778.2503791

Nico Lehmann and Éric Tanter. 2017. Gradual Refinement Types. In *ACM Symposium on Principles of Programming Languages (POPL)*. https://doi.org/10.1145/3009837.3009856

Daan Leijen. 2014. Koka: Programming with Row Polymorphic Effect Types. In *Proceedings 5th Workshop on Mathematically Structured Functional Programming, MSFP@ETAPS 2014, Grenoble, France, 12 April 2014 (EPTCS, Vol. 153)*. 100–126. https://doi.org/10.4204/EPTCS.153.8

Sam Lindley, Conor McBride, and Craig McLaughlin. 2017. Do be do be do. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*. ACM, 500–514. https://doi.org/10.1145/3009837.3009897

Stefan Malewski, Michael Greenberg, and Éric Tanter. 2021. Gradually structured data. *Proc. ACM Program. Lang.* 5, OOPSLA (2021), 1–29. https://doi.org/10.1145/3485503

Daniel Marino and Todd D. Millstein. 2009. A generic type-and-effect system. In *Proceedings of TLDI'09: 2009 ACM SIGPLAN International Workshop on Types in Languages Design and Implementation, Savannah, GA, USA, January 24, 2009*, Andrew Kennedy and Amal Ahmed (Eds.). ACM, 39–50. https://doi.org/10.1145/1481861.1481868

H. Nakano. 2000. A modality for recursion. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.99CB36332)*. 255–266. https://doi.org/10.1109/LICS.2000.855774

Max S. New and Amal Ahmed. 2018. Graduality from Embedding-Projection Pairs. In *International Conference on Functional Programming (ICFP), St. Louis, Missouri*. https://doi.org/10.1145/3236768

Max S. New, Eric Giovannini, and Daniel R. Licata. 2023. Gradual Typing for Effect Handlers (Extended Version). (2023). https://maxsnew.com/docs/greff-extended.pdf

Max S. New, Dustin Jamner, and Amal Ahmed. 2020. Graduality and parametricity: together again for the first time. *Proc. ACM Program. Lang.* 4, POPL (2020), 46:1–46:32. https://doi.org/10.1145/3371114

Max S. New and Daniel R. Licata. 2018. Call-by-name Gradual Type Theory. In *Formal Structures for Computation and Deduction, Oxford England*. https://doi.org/10.4230/LIPIcs.FSCD.2018.24

Max S. New, Daniel R. Licata, and Amal Ahmed. 2019. Gradual Type Theory. In *ACM Symposium on Principles of Programming Languages (POPL), Cascais, Portugal*. https://doi.org/10.1145/3290328

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

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

Taro Sekiyama, Soichiro Ueda, and Atsushi Igarashi. 2015. Shifting the Blame - A Blame Calculus with Delimited Control. In *Programming Languages and Systems - 13th Asian Symposium, APLAS 2015, Pohang, South Korea, November 30 - December 2, 2015, Proceedings (Lecture Notes in Computer Science, Vol. 9458)*, Xinyu Feng and Sungwoo Park (Eds.). Springer, 189–207. https://doi.org/10.1007/978-3-319-26529-2_11

Jeremy Siek, Micahel Vitousek, Matteo Cimini, and John Tang Boyland. 2015. Refined Criteria for Gradual Typing. In *1st Summit on Advances in Programming Languages (SNAPL 2015)*. https://doi.org/10.4230/LIPIcs.SNAPL.2015.274

Jeremy G. Siek and Walid Taha. 2006. Gradual Typing for Functional Languages. In *Scheme and Functional Programming Workshop (Scheme)*. 81–92.

Jeremy G. Siek and Walid Taha. 2007. Gradual Typing for Objects. In *European Conference on Object-Oriented Programming (ECOOP)*. https://doi.org/10.1007/978-3-540-73589-2_2

Jeremy G. Siek and Manish Vachharajani. 2008. Gradual typing with unification-based inference. In *Proceedings of the 2008 Symposium on Dynamic Languages, DLS 2008, July 8, 2008, Paphos, Cyprus*, Johan Brichau (Ed.). ACM, 7. https://doi.org/10.1145/1408681.1408688

K. C. Sivaramakrishnan, Stephen Dolan, Leo White, Tom Kelly, Sadiq Jaffer, and Anil Madhavapeddy. 2021. Retrofitting effect handlers onto OCaml. In *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*. ACM, 206–221. https://doi.org/10.1145/3453483.3454039

Asumu Takikawa, Daniel Feltey, Ben Greenman, Max S. New, Jan Vitek, and Matthias Felleisen. 2016. Is Sound Gradual Typing Dead?. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (St. Petersburg, FL, USA) *(POPL '16)*. Association for Computing Machinery, New York, NY, USA, 456–468. https://doi.org/10.1145/2837614.2837630

Asumu Takikawa, T. Stephen Strickland, and Sam Tobin-Hochstadt. 2013. Constraining Delimited Control with Contracts. In *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7792)*, Matthias Felleisen and Philippa Gardner (Eds.). Springer, 229–248. https://doi.org/10.1007/978-3-642-37036-6_14

Sam Tobin-Hochstadt and Matthias Felleisen. 2008. The Design and Implementation of Typed Scheme. In *ACM Symposium on Principles of Programming Languages (POPL), San Francisco, California*. https://doi.org/10.1145/1328438.1328486

Philip Wadler. 2021. GATE: Gradual Effect Types. In *Leveraging Applications of Formal Methods, Verification and Validation - 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021, Rhodes, Greece, October 17-29, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13036)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer, 335–345. https://doi.org/10.1007/978-3-030-89159-6_21

Philip Wadler and Robert Bruce Findler. 2009. Well-typed programs can't be blamed. In *European Symposium on Programming (ESOP)* (York, UK). 1–16. https://doi.org/10.1007/978-3-642-00590-9_1

A (IN)EQUATIONAL THEORY

In this section we describe the full inequational theory and then prove several derivable theorems in the theory.

Note that for brevity, we use some shorthands: rather than writing out the full $\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{\sigma \sqsubseteq \tau} M \sqsubseteq N : A \sqsubseteq B$, (1) we elide $\Sigma \mid \Gamma^{\sqsubseteq}$, and all rules should be interpreted as holding under an arbitrary such contexts (2) rather than write $\sigma \sqsubseteq \tau$ and $A \sqsubseteq B$, we use instead precision derivations d_σ, c and (3) whenever it is clear, we elide the types as well, especially for equational rules.

First we need general call-by-value reasoning principles.

$$\frac{M[x : A] \equiv N[x : A] \qquad V \equiv V' : A}{M[V/x] \equiv N[V'/x]}\ \textsc{ValSubst} \qquad \texttt{let}\ x = y\ \texttt{in}\ N \equiv N[y/x]\ \textsc{MonadUnitL}$$

$$\texttt{let}\ x = M\ \texttt{in}\ x \equiv M\ \textsc{MonardUnitR}$$

$$\texttt{let}\ y = (\texttt{let}\ x = M\ \texttt{in}\ N)\ \texttt{in}\ P \equiv \texttt{let}\ x = M\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ P\ \textsc{MonadAssoc}$$

$$M[x : \texttt{bool}] \equiv \texttt{if}\ x\{M[\texttt{true}/x]\}\{M[\texttt{false}/x]\}\ \textsc{BoolEta}$$

$$\texttt{if true}\{N_t\}\{N_f\} \equiv N_t\ \textsc{BoolBetaTru} \qquad \texttt{if false}\{N_t\}\{N_f\} \equiv N_f\ \textsc{BoolBetaFalse}$$

$$\texttt{if}\ M\{N_t\}\{N_f\} \equiv \texttt{let}\ x = M\ \texttt{in if}\ x\{N_t\}\{N_f\}\ \textsc{IfEval} \qquad (\lambda x.M)V \equiv M[V/x]\ \textsc{FunBeta}$$

$$(V : A \to B) \equiv \lambda x.V x\ \textsc{FunEta} \qquad M\,N \equiv \texttt{let}\ x = M\ \texttt{in let}\ y = N\ \texttt{in}\ x\,y\ \textsc{AppEval}$$

Next, the rules specifically for raise and handlers:

$$\texttt{handle}\ x\,\{\texttt{ret}\ y.M \mid \phi\} \equiv M[x/y]\ \textsc{HandleBetaRet}$$

$$\begin{array}{c}\texttt{handle}\ (\texttt{let}\ o = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ N_k)\,\{\texttt{ret}\ y.M \mid \phi\} \equiv \\ \phi(\varepsilon)[\lambda o.\texttt{handle}\ N_k\,\{\texttt{ret}\ y.M \mid \phi\}/k]\end{array}\ \textsc{HandleBetaRaise}$$

$$\texttt{raise}\ \varepsilon(M) \equiv \texttt{let}\ x = M\ \texttt{in raise}\ \varepsilon(x)\ \textsc{RaiseEval}$$

$$\texttt{handle}\ M\,\{\texttt{ret}\ x.N \mid \emptyset\} \equiv \texttt{let}\ x = M\ \texttt{in}\ N\ \textsc{HandleEmpty}$$

$$\frac{\forall \varepsilon \in \mathrm{dom}(\phi).\ \psi(\varepsilon) = \phi(\varepsilon) \qquad \forall \varepsilon \in \mathrm{dom}(\psi). \varepsilon \notin \mathrm{dom}(\phi) \Rightarrow \psi(\varepsilon) = k(\texttt{raise}\ \varepsilon(x))}{\texttt{handle}\ M\,\{\texttt{ret}\ y.N \mid \phi\} \equiv \texttt{handle}\ M\,\{\texttt{ret}\ y.N \mid \psi\}}\ \textsc{HandleExt}$$

Next, the congruence rules

$$\frac{x_1 \sqsubseteq x_2 : c \in \Gamma^{\sqsubseteq}}{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} x_1 \sqsubseteq x_2 : c}\ \textsc{VarCong} \qquad \frac{}{\vdash_{d_\sigma} \texttt{true} \sqsubseteq \texttt{true} : \texttt{bool}}\ \textsc{TrueCong}$$

$$\frac{}{\vdash_{d_\sigma} \texttt{false} \sqsubseteq \texttt{false} : \texttt{bool}}\ \textsc{FalseCong} \qquad \frac{x_1 \sqsubseteq x_2 : c \vdash_{d_{\sigma'}} M_1 \sqsubseteq M_2 : d}{\vdash_{d_\sigma} \lambda x_1.M_1 \sqsubseteq \lambda x_2.M_2 : c \rightarrow_{d_{\sigma'}} d}\ \textsc{LambdaCong}$$

$$\frac{\vdash_{d_\sigma} M_1 \sqsubseteq M_2 : c \rightarrow_{d_\sigma} d \qquad \vdash_{d_\sigma} N_1 \sqsubseteq N_2 : c}{\vdash_{d_\sigma} M_1\, N_1 \sqsubseteq M_2\, N_2 : d}\ \textsc{AppCong}$$

$$\frac{\begin{array}{c}\vdash_{d_\sigma} M_1 \sqsubseteq M_2 : c \\ x_1 \sqsubseteq x_2 : c \vdash_{d_\sigma} N_1 \sqsubseteq N_2 : d\end{array}}{\vdash_{d_\sigma} \texttt{let}\ x_1 = M_1\ \texttt{in}\ N_1 \sqsubseteq \texttt{let}\ x_2 = M_2\ \texttt{in}\ N_2 : d}\ \textsc{LetCong}$$

$$\frac{\begin{array}{c}\vdash_{d_\sigma} M \sqsubseteq M' : \texttt{bool} \\ \vdash_{d_\sigma} N_t \sqsubseteq N_t' : c \qquad \vdash_{d_\sigma} N_f \sqsubseteq N_f' : c\end{array}}{\vdash_{d_\sigma} \texttt{if}\ M\{N_t\}\{N_f\} \sqsubseteq \texttt{if}\ M'\{N_t'\}\{N_f'\} : c}\ \textsc{IfCong}$$

$$\frac{\begin{array}{c}c : A_1 \sqsubseteq A_2 \qquad d : B_1 \sqsubseteq B_2 \\ \varepsilon : c \rightsquigarrow d \in d_\sigma \qquad \vdash_{d_\sigma} M_1 \sqsubseteq M_2 : c\end{array}}{\vdash_{d_\sigma} \texttt{raise}\ \varepsilon(M_1) \sqsubseteq \texttt{raise}\ \varepsilon(M_2) : d}\ \textsc{RaiseCong}$$

$$\frac{\begin{array}{c}\vdash_{d_\sigma} M \sqsubseteq M' : c \qquad y : c \vdash_{d_\tau} N \sqsubseteq N' : d \\ \forall \varepsilon : d_i \rightsquigarrow d_o \in d_\sigma.(\varepsilon \notin \mathrm{dom}(\phi) \wedge \varepsilon \notin \mathrm{dom}(\phi') \wedge \varepsilon : d_i \rightsquigarrow d_o \in d_\tau) \vee \\ x : d_i, k : d_o \rightarrow_{d_\tau} d \vdash_{d_\tau} \phi(\varepsilon) \sqsubseteq \phi'(\varepsilon) : d\end{array}}{\vdash_{d_\tau} \texttt{handle}\ M\,\{\texttt{ret}\ y.N \mid \phi\} \sqsubseteq \texttt{handle}\ M'\,\{\texttt{ret}\ y.N' \mid \phi'\} : d}\ \textsc{HandleCong}$$

Next, the rules for errors

$$\frac{\vdash_{d_\sigma{}^r} M : c^r}{\vdash_{d_\sigma} \mho \sqsubseteq M : c}\ \textsc{ErrBot} \qquad E[\mho] \equiv \mho\ \textsc{ErrStrict}$$

The generic rules for casts

$$\frac{\vdash_{d_\sigma} M \sqsubseteq N : (c : A \sqsubseteq B) \qquad c : A \sqsubseteq A}{\vdash_{d_\sigma} \langle B \leftarrowtail A \rangle M \sqsubseteq N : B}\ \textsc{ValUpL} \qquad \frac{\vdash_\sigma M : A \qquad c : A \sqsubseteq B}{\vdash_\sigma M \sqsubseteq \langle B \leftarrowtail A \rangle M : c}\ \textsc{ValUpR}$$

$$\langle B \leftarrowtail A \rangle M \equiv \texttt{let}\ x = M\ \texttt{in}\ \langle B \leftarrowtail A \rangle x \quad \textsc{ValUpEval} \qquad \frac{c : A \sqsubseteq B \qquad \vdash_\sigma N : B}{\vdash_\sigma \langle A \twoheadleftarrow B \rangle N \sqsubseteq N : c}\ \textsc{ValDnL}$$

$$\frac{\vdash_{d_\sigma} M \sqsubseteq N : (c : A \sqsubseteq B)}{\vdash_{d_\sigma} M \sqsubseteq \langle A \twoheadleftarrow B \rangle N : A}\ \textsc{ValDnR} \qquad \langle A \twoheadleftarrow B \rangle M \equiv \texttt{let}\ x = M\ \texttt{in}\ \langle A \twoheadleftarrow B \rangle x\ \textsc{ValDnEval}$$

$$\frac{\vdash_{d_\sigma} M \sqsubseteq N : c \qquad d_\sigma : \sigma \sqsubseteq \tau}{\vdash_\tau \langle \tau \leftarrowtail \sigma \rangle M \sqsubseteq N : c}\ \textsc{ValUpL} \qquad \frac{\vdash_\sigma M : A \qquad d_\sigma : \sigma \sqsubseteq \tau}{\vdash_{d_\sigma} M \sqsubseteq \langle \tau \leftarrowtail \sigma \rangle M : c}\ \textsc{ValUpR}$$

$$\frac{d_\sigma : \sigma \sqsubseteq \tau \qquad \vdash_\tau N : A}{\vdash_{d_\sigma} \langle \sigma \twoheadleftarrow \tau \rangle N \sqsubseteq N : A}\ \textsc{EffDnL} \qquad \frac{d_\sigma : \sigma \sqsubseteq \tau \vdash_{d_\sigma} M \sqsubseteq N : c}{\vdash_\sigma M \sqsubseteq \langle \sigma \twoheadleftarrow \tau \rangle N : c}\ \textsc{EffDnR}$$

And the subtyping rules

$$\frac{\begin{array}{c}\vdash_{d_\sigma} M \sqsubseteq N : c \qquad d_\sigma : \sigma \sqsubseteq \tau \qquad c : A \sqsubseteq B \\ d'_\sigma : \sigma' \sqsubseteq \tau' \qquad c' : A' \sqsubseteq B' \\ \sigma \leq \sigma' \qquad A \leq A' \qquad \tau \leq \tau' \qquad B \leq B'\end{array}}{\vdash_{d'_\sigma} M \sqsubseteq N : c'}\ \textsc{SubtyMon}$$

$$\frac{c : A \sqsubseteq B \qquad c' : A' \sqsubseteq B' \qquad c \leq c' \qquad \vdash_\sigma M : A}{\vdash_\sigma \langle B \leftarrowtail A \rangle M \equiv \langle B' \leftarrowtail A' \rangle M : B'}\ \textsc{ValUpSub}$$

$$\frac{c : A \sqsubseteq B \qquad c' : A' \sqsubseteq B' \qquad c \leq c' \qquad \vdash_\sigma N : B}{\vdash_\sigma \langle A \twoheadleftarrow B \rangle N \equiv \langle A' \twoheadleftarrow B' \rangle N : \sigma\,!\,A'}\ \textsc{ValDnSub}$$

$$\frac{c_\sigma : \sigma \sqsubseteq \tau \qquad c' : \sigma' \sqsubseteq \tau' \qquad c_\sigma \leq c'_\sigma \qquad \vdash_\sigma M : A}{\vdash_{\tau'} \langle \tau \leftarrowtail \sigma \rangle M \equiv \langle \tau' \leftarrowtail \sigma' \rangle M : A}\ \textsc{EffUpSub}$$

$$\frac{c_\sigma : \sigma \sqsubseteq \tau \qquad c' : \sigma' \sqsubseteq \tau' \qquad c_\sigma \leq c'_\sigma \qquad \vdash_\tau N : A}{\vdash_{\sigma'} \langle \sigma \twoheadleftarrow \tau \rangle N \equiv \langle \sigma' \twoheadleftarrow \tau' \rangle N : A}\ \textsc{EffDnSub}$$

In Figure 16, we list some derivable reasoning principles for our inequational theory, which follow by analogous proofs to prior work.

We can show the following properties of the interaction between subtyping and casts axiomatically:

Lemma A.1. *The following hold:*

(1) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\sigma} \langle B' \leftarrowtail A' \rangle M \sqsubseteq \langle B \leftarrowtail A \rangle N : B'$.
(2) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\sigma} \langle A \twoheadleftarrow B \rangle M \sqsubseteq \langle A' \twoheadleftarrow B' \rangle N : A'$.
(3) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma'_2} \langle \sigma'_2 \leftarrowtail \sigma'_1 \rangle P \sqsubseteq \langle \sigma_2 \leftarrowtail \sigma_1 \rangle Q : c$.
(4) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma'_1} \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle P \sqsubseteq \langle \sigma'_1 \twoheadleftarrow \sigma'_2 \rangle Q : c$.

Proof.

$$\langle A \nwarrow A\rangle M \equiv M \qquad \langle \sigma \nwarrow \sigma\rangle M \equiv M \qquad \langle A \swarrow A\rangle M \equiv M \qquad \langle \sigma \swarrow \sigma\rangle M \equiv M$$

$$\langle C \nwarrow B\rangle\langle B \nwarrow A\rangle M \equiv \langle C \nwarrow A\rangle M \qquad \langle A \swarrow B\rangle\langle B \swarrow C\rangle M \equiv \langle A \swarrow C\rangle M$$

$$\langle \sigma'' \nwarrow \sigma'\rangle\langle \sigma' \nwarrow \sigma\rangle M \equiv \langle \sigma'' \nwarrow \sigma\rangle M \qquad \langle \sigma \swarrow \sigma'\rangle\langle \sigma' \swarrow \sigma''\rangle M \equiv \langle \sigma \swarrow \sigma''\rangle M$$

$$\langle B \nwarrow A\rangle\langle \sigma' \nwarrow \sigma\rangle M \equiv \langle \sigma' \nwarrow \sigma\rangle\langle B \nwarrow A\rangle M \qquad \langle A \swarrow B\rangle\langle \sigma \swarrow \sigma'\rangle M \equiv \langle \sigma \swarrow \sigma'\rangle\langle A \swarrow B\rangle M$$

Fig. 16. Provable Uniqueness Theorems

We have

$$\dfrac{\dfrac{\dfrac{\vdash_{d_\sigma} M \sqsubseteq N : A}{\vdash_{d\sigma} M \sqsubseteq \langle B \nwarrow A\rangle N : A \sqsubseteq B}\ (\text{ValUpR})}{\vdash_{d_\sigma} M \sqsubseteq \langle B \nwarrow A\rangle N : A' \sqsubseteq B'}\ (\text{Subtyping})}{\vdash_{d\sigma} \langle B' \nwarrow A'\rangle M \sqsubseteq \langle B \nwarrow A\rangle N : B'}\ (\text{ValUpL})$$

Dual to the above.
We have

$$\dfrac{\dfrac{\dfrac{\vdash_{\sigma_1} P \sqsubseteq Q : c}{\vdash_{d_\sigma} P \sqsubseteq \langle \sigma_2 \nwarrow \sigma_1\rangle Q : c}\ (\text{EffUpR})}{\vdash_{d'_\sigma} P \sqsubseteq \langle \sigma_2 \nwarrow \sigma_1\rangle Q : c}\ (\text{Subtyping})}{\vdash_{\sigma'_2} \langle \sigma'_2 \nwarrow \sigma'_1\rangle P \sqsubseteq \langle \sigma_2 \nwarrow \sigma_1\rangle Q : c}\ (\text{EffUpL})$$

Dual to the above. □

B OPERATIONAL SEMANTICS

An evaluation context $E_{\#\varepsilon}$ is one in which none of the handler clauses in the spine of the context handles ε.

B.1 Operational Semantics from First Principles

Now we show that every operational reduction is justified by our inequational theory.

Lemma B.1 (Effect Casts are Handlers). *Let $\sigma \sqsubseteq \tau$ where σ is a concrete effect set. Then the upcast $\langle \tau \nwarrow \sigma\rangle$ is equivalent to a handler in that for any $M : \sigma\,!\,A$:*

$$\langle \tau \nwarrow \sigma\rangle M \equiv \texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma\rangle}\}$$

where for each $\varepsilon \in dom(\sigma)$

$$x, k \vdash \phi_{\langle \tau \nwarrow \sigma\rangle}(\varepsilon) = k(\langle B_\sigma \swarrow B_\tau\rangle \texttt{raise}\ \varepsilon(\langle A_\tau \nwarrow A_\sigma\rangle))$$

where $\varepsilon : A_\sigma \rightsquigarrow B_\sigma \in \sigma$ and $\varepsilon : A_\tau \rightsquigarrow B_\tau \in \tau$.

Similarly, the downcast $\langle \sigma \swarrow \tau\rangle$ is equivalent to a handler in that for any $N : \tau\,!\,A$:

$$\langle \sigma \swarrow \tau\rangle M \equiv \texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow \tau\rangle}\}$$

where for each $\varepsilon \in dom(\tau)$, if $\varepsilon \in dom(\sigma)$, then

$$x, k \vdash \phi_{\langle \sigma \swarrow \tau\rangle}(\varepsilon) = k(\langle B_\tau \nwarrow B_\sigma\rangle \texttt{raise}\ \varepsilon(\langle A_\sigma \swarrow A_\tau\rangle))$$

$$\frac{\varepsilon \in \mathrm{dom}(\phi) \qquad E'\#\varepsilon}{\begin{array}{c}E[\mathsf{handle}\ E'[\mathtt{raise}\ \varepsilon(V)]\ \{\mathtt{ret}\ x.N \mid \phi\}] \\ \mapsto E[\phi(\varepsilon)[V/x][(\lambda y.\mathsf{handle}\ (E'[y])\ \{\mathtt{ret}\ x.N \mid \phi\})/k]]\end{array}}$$

$$\frac{}{E[\mathsf{handle}\ V\ \{\mathtt{ret}\ x.N \mid \phi\}] \mapsto E[N[V/x]]}\ \textsc{HandleVal}$$

$$\frac{}{E[(\lambda x.M)V] \mapsto E[M[V/x]]}\ \textsc{Lam} \qquad \frac{}{E[\mathtt{let}\ x = V\ \mathtt{in}\ M] \mapsto E[M[V/x]]}\ \textsc{Let}$$

$$\frac{}{E[\mho] \mapsto \mho}\ \textsc{Err} \qquad \frac{}{E[\mathtt{if\ true}\{N_t\}\{N_f\}] \mapsto E[N_t]}\ \textsc{IfTrue}$$

$$\frac{}{E[\mathtt{if\ false}\{N_t\}\{N_f\}] \mapsto E[N_f]}\ \textsc{IfFalse} \qquad \frac{}{E[\langle \sigma' \leftarrowtail \sigma\rangle V] \mapsto E[V]}\ \textsc{EffUpDnCastVal}$$

$$\frac{}{E[\langle \sigma \twoheadleftarrow \sigma'\rangle V] \mapsto E[V]}\ \textsc{EffDnCastVal}$$

$$\frac{\varepsilon \in \sigma' \qquad E'\#\varepsilon}{\begin{array}{c}E[\langle \sigma' \leftarrowtail \sigma\rangle E'[\mathtt{raise}\ \varepsilon(V)]] \mapsto \\ E[\mathtt{let}\ x = \langle B \twoheadleftarrow B'\rangle \mathtt{raise}\ \varepsilon(\langle A' \leftarrowtail A\rangle V)\ \mathtt{in}\ \langle \sigma' \leftarrowtail \sigma\rangle E'[x]]\end{array}}\ \textsc{EffUpCast}$$

$$\frac{\varepsilon \in \sigma \qquad E'\#\varepsilon}{\begin{array}{c}E[\langle \sigma \twoheadleftarrow \sigma'\rangle E'[\mathtt{raise}\ \varepsilon(V)]] \mapsto \\ E[\mathtt{let}\ x = \langle B' \leftarrowtail B\rangle \mathtt{raise}\ \varepsilon(\langle A \twoheadleftarrow A'\rangle V)\ \mathtt{in}\ \langle \sigma \twoheadleftarrow \sigma'\rangle E'[x]]\end{array}}\ \textsc{GoodEffDnCast}$$

$$\frac{\varepsilon \notin \sigma \qquad E'\#\varepsilon}{E[\langle \sigma \twoheadleftarrow ?\rangle E'[\mathtt{raise}\ \varepsilon(V)]] \mapsto E[\mho]}\ \textsc{BadEffDnCast}$$

$$E[\updownarrow \mathsf{bool} M] \mapsto E[M] \quad \textsc{BoolUpDnCast}$$

$$\frac{}{E[(\langle (A' \to_{\sigma'} B') \leftarrowtail (A \to_\sigma B)\rangle V_f)\, V] \mapsto E[\langle B' \leftarrowtail B\rangle \langle \sigma' \leftarrowtail \sigma\rangle (V_f \langle A \twoheadleftarrow A'\rangle V)]}\ \textsc{FunUpCast}$$

$$\frac{}{E[(\langle (A \to_\sigma B) \twoheadleftarrow (A' \to_{\sigma'} B')\rangle V_f)\, V] \mapsto E[\langle B \twoheadleftarrow B'\rangle \langle \sigma \twoheadleftarrow \sigma'\rangle (V_f \langle A' \leftarrowtail A\rangle V)]}\ \textsc{FunDnCast}$$

Fig. 17. Full Operational Semantics

and if $\varepsilon \notin dom(\sigma)$, then

$$\phi_{\langle \sigma \twoheadleftarrow \tau\rangle}(\varepsilon) = \mho$$

Proof. First for the upcast case

- We want to show

$$\langle \tau \leftarrowtail \sigma\rangle M \sqsubseteq \mathsf{handle}\ M\ \{\mathtt{ret}\ x.x \mid \phi_{\langle \tau \leftarrowtail \sigma\rangle}\}$$

  By UpL, it is sufficient to show

$$M \sqsubseteq \mathsf{handle}\ M\ \{\mathtt{ret}\ x.x \mid \phi_{\langle \tau \leftarrowtail \sigma\rangle}\}$$

$$\Delta ::= \bullet : (\sigma \,!\, A) \qquad \frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A \to_\sigma B \qquad \Sigma \mid \Gamma \mid \cdot \vdash_\sigma N : A}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E\, N : B}$$

$$\frac{\Sigma \mid \Gamma \vdash_\sigma V : A \to_\sigma B \qquad \Sigma \mid \Gamma \mid \bullet : (\sigma_i \,!\, C) \vdash_\sigma E : A}{\Sigma \mid \Gamma \mid \bullet : (\sigma_i \,!\, C) \vdash_\sigma V\, E : B}$$

$$\frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : \mathtt{bool} \qquad \Sigma \mid \Gamma \vdash_\sigma N_t B \qquad \Sigma \mid \Gamma \vdash_\sigma N_f B}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma \mathtt{if}\ E\{N_t\}\{N_f\} : B}$$

$$\frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A \qquad \varepsilon : A \rightsquigarrow B \in \sigma}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma \mathtt{raise}\ \varepsilon(E) : B}$$

$$\frac{\begin{array}{c}\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A \\ \Sigma \mid \Gamma, x : A \vdash_\tau N : B \\ (\forall (\varepsilon : A_\varepsilon \rightsquigarrow B_\varepsilon) \in \sigma.\ (\varepsilon \notin \mathrm{dom}(\phi) \wedge (\varepsilon : A_\varepsilon \rightsquigarrow B_\varepsilon) \in \tau) \\ \vee(\Sigma \mid \Gamma, x : A_\varepsilon, k : B_\varepsilon \to_\tau B \vdash_\tau \phi(\varepsilon) : B))\end{array}}{\Sigma \mid \Gamma \vdash_\tau \mathtt{handle}\ E\, \{\mathtt{ret}\ x.N \mid \phi\} : B}$$

$$\frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A \qquad \Sigma \mid \Gamma, x : A \vdash_\sigma N : B}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma \mathtt{let}\ x = E\ \mathtt{in}\ N : B} \qquad \frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A' \qquad \Sigma \mid \Gamma \vdash A' \leq A}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A}$$

$$\frac{\Sigma \mid \Gamma \mid \Delta \vdash_\Delta \sigma' : EA \qquad \Sigma \mid \Gamma \vdash \sigma' \leq \sigma}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A} \qquad \frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A \qquad \Sigma \vdash A \sqsubseteq B}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma \langle B \leftarrowtail A \rangle E : B}$$

$$\frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : B \qquad \Sigma \vdash A \sqsubseteq B}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma \langle A \twoheadleftarrow B \rangle E : A} \qquad \frac{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma E : A \qquad \Sigma \vdash \sigma \sqsubseteq \sigma'}{\Sigma \mid \Gamma \mid \Delta \vdash_{\sigma'} \langle \sigma' \leftarrowtail \sigma \rangle E : A}$$

$$\frac{\Sigma \mid \Gamma \mid \Delta \vdash_{\sigma'} E : A \qquad \Sigma \vdash \sigma \sqsubseteq \sigma'}{\Sigma \mid \Gamma \mid \Delta \vdash_\sigma \langle \sigma \twoheadleftarrow \sigma' \rangle E : A}$$

Fig. 18. Typing Rules for Evaluation Contexts

$$\varepsilon \# \bullet \qquad \frac{\varepsilon \# E}{\varepsilon \# (\langle B \leftarrowtail A \rangle E)} \qquad \frac{\varepsilon \# E}{\varepsilon \# (\langle A \twoheadleftarrow B \rangle E)} \qquad \frac{\varepsilon \# E \qquad \varepsilon \notin \sigma \qquad \varepsilon \notin \sigma'}{\varepsilon \# (\langle \sigma' \leftarrowtail \sigma \rangle E)}$$

$$\frac{\varepsilon \# E \qquad \varepsilon \notin \sigma'}{\varepsilon \# (\langle \sigma \twoheadleftarrow \sigma' \rangle E)} \qquad \frac{\varepsilon \# E \qquad \varepsilon' \text{ any effect}}{\varepsilon \# (\mathtt{raise}\ \varepsilon'(E))} \qquad \frac{\varepsilon \# E \wedge \varepsilon \notin \mathrm{dom}(\phi)}{\varepsilon \# (\mathtt{handle}\ E\, \{\mathtt{ret}\ x.N \mid \phi\})} \qquad \frac{\varepsilon \# E}{\varepsilon \# (E\, M)}$$

$$\frac{\varepsilon \# E}{\varepsilon \# (V\, E)} \qquad \frac{\varepsilon \# E}{\varepsilon \# (\mathtt{if}\ E\{N_t\}\{N_f\})} \qquad \frac{\varepsilon \# E}{\varepsilon \# (\mathtt{let}\ x = E\ \mathtt{in}\ N)}$$

Fig. 19. Apartness of Effect from an Evaluation Context

But by the handler η rule, this is equivalent to showing

$$\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_\sigma\} \sqsubseteq \texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\}$$

where $\mathrm{dom}(\phi_\sigma) = \mathrm{dom}(\sigma)$ and $\phi_\sigma(\varepsilon) = k(\texttt{raise}\ \varepsilon(x))$. Then by congruence, we need to show that for each $\varepsilon \in \mathrm{dom}(\sigma)$,

$$k(\texttt{raise}\ \varepsilon(x)) \sqsubseteq k(\langle B_\sigma \swarrow B_\tau\rangle \texttt{raise}\ \varepsilon(\langle A_\sigma \nwarrow\ )\rangle A_\tau x)$$

which follows from UpR/DnR and congruence rules

- We want to show

$$\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\} \sqsubseteq \langle \tau \nwarrow \sigma\rangle M$$

By handler η it is sufficient to show

$$\texttt{handle}\ M\,\{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\} \sqsubseteq \texttt{handle}\ \langle \tau \nwarrow \sigma\rangle M\,\{\texttt{ret}\ x.x \mid \phi_\tau\}$$

where $\mathrm{dom}(\phi_\tau) = \mathrm{dom}(\tau)$ and $\phi_\tau(\varepsilon) = k(\texttt{raise}\ \varepsilon(x))$. Then $M \sqsubseteq \langle \tau \nwarrow \sigma\rangle M$ by UpR and so by congruence we need only to show for each $\varepsilon \in \sigma$ that

$$\phi_{\langle \tau \nwarrow \sigma \rangle}(\varepsilon) \sqsubseteq \phi_\tau(\varepsilon)$$

which follows by a similar argument to the previous case.

Next, the downcast cases.

- We want to show

$$\texttt{handle}\ N\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow \tau \rangle}\} \sqsubseteq \langle \sigma \swarrow \tau\rangle N$$

By DnR, it is sufficient to show

$$\texttt{handle}\ N\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow \tau \rangle}\} \sqsubseteq N$$

By handler η this is equivalent to showign

$$\texttt{handle}\ N\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow \tau \rangle}\} \sqsubseteq \texttt{handle}\ N\,\{\texttt{ret}\ x.x \mid \phi_\tau\}$$

That is, for any $\varepsilon \in \mathrm{dom}(\tau)$ that

$$\phi_{\langle \sigma \swarrow \tau \rangle}(\varepsilon) \sqsubseteq \phi_\tau(\varepsilon)$$

There are two cases

(1) If $\varepsilon \in \mathrm{dom}(\sigma)$, then we need to show

$$k(\langle B_\tau \nwarrow B_\sigma\rangle \texttt{raise}\ \varepsilon(\langle A_\tau \nwarrow A_\sigma\rangle x)) \sqsubseteq k(\texttt{raise}\ \varepsilon(x))$$

which follows by congruence and DnL/UpL rules.

(2) If $\varepsilon \notin \mathrm{dom}(\sigma)$, then we need to show

$$\mho \sqsubseteq k(\texttt{raise}\ \varepsilon(x))$$

which is immediate.

- We want to show

$$\langle \sigma \swarrow \tau\rangle N \sqsubseteq \texttt{handle}\ N\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow \tau \rangle}\}$$

By handler η this is equivalent to showing

$$\texttt{handle}\ (\langle \sigma \swarrow \tau\rangle N)\,\{\texttt{ret}\ x.x \mid \phi_\sigma\} \sqsubseteq \texttt{handle}\ N\,\{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow \tau \rangle}\}$$

By congruence and DnL this reduces to showing for each $\varepsilon \in \mathrm{dom}(\sigma)$ that

$$\phi_\sigma(\varepsilon) \sqsubseteq \phi_{\langle \sigma \swarrow \tau \rangle}(\varepsilon)$$

since $\varepsilon \in \text{dom}(\sigma)$, these are each of the form:

$$k(\texttt{raise } \varepsilon(x)) \sqsubseteq k(\langle B_\tau \leftarrowtail B_\sigma\rangle \texttt{raise } \varepsilon(\langle A_\tau \leftarrowtail A_\sigma\rangle x))$$

which follows by congruence and DnR/UpR rules.

□

Lemma B.2 (Derivation of Function Casts).

$$\langle A' \to_\tau B' \leftarrowtail A \to_\sigma B\rangle f \equiv \lambda x.\langle B' \leftarrowtail B\rangle\langle \tau \leftarrowtail \sigma\rangle(f(\langle A \twoheadleftarrow A'\rangle x))$$

And similarly,

$$\langle A \to_\sigma B \twoheadleftarrow A' \to_\tau B'\rangle f \equiv \lambda x.\langle B \twoheadleftarrow B'\rangle\langle \sigma \twoheadleftarrow \tau\rangle(f(\langle A' \leftarrowtail A\rangle x))$$

Proof. We show the upcast cases, the downcast cases are precisely dual.

(1) We want to show

$$\langle A' \to_\tau B' \leftarrowtail A \to_\sigma B\rangle f \sqsubseteq \lambda x.\langle B' \leftarrowtail B\rangle\langle \tau \leftarrowtail \sigma\rangle(f(\langle A \twoheadleftarrow A'\rangle x))$$

By UpL, it is sufficient to show

$$f \sqsubseteq \lambda x.\langle B' \leftarrowtail B\rangle\langle \tau \leftarrowtail \sigma\rangle(f(\langle A \twoheadleftarrow A'\rangle x))$$

By η equivalence for functions it is sufficient to show

$$\lambda x.f x \sqsubseteq \lambda x.\langle B' \leftarrowtail B\rangle\langle \tau \leftarrowtail \sigma\rangle(f(\langle A \twoheadleftarrow A'\rangle x))$$

Which follows by congruence rules and UpR/DnR rules.

(2) We want to show

$$\lambda x.\langle B' \leftarrowtail B\rangle\langle \tau \leftarrowtail \sigma\rangle(f(\langle A \twoheadleftarrow A'\rangle x)) \sqsubseteq \langle A' \to_\tau B' \leftarrowtail A \to_\sigma B\rangle f$$

By function η it is sufficient to show

$$\lambda x.\langle B' \leftarrowtail B\rangle\langle \tau \leftarrowtail \sigma\rangle(f(\langle A \twoheadleftarrow A'\rangle x)) \sqsubseteq \lambda y.(\langle A' \to_\tau B' \leftarrowtail A \to_\sigma B\rangle f)y$$

Which follows by congruence and UpL/DnL/UpR rules.

□

Lemma B.3. *If $x, k \vdash \phi(\varepsilon) = k(\texttt{raise } \varepsilon(x))$, then*

$$\texttt{handle raise } \varepsilon(x)\,\{\texttt{ret } y.N \mid \phi\} \equiv \texttt{let } o = (\texttt{raise } \varepsilon(x)) \texttt{ in handle } o\,\{\texttt{ret } y.N \mid \phi\}$$

Proof.

$$\begin{aligned}\texttt{handle raise } \varepsilon(x)\,\{\texttt{ret } y.N \mid \phi\} &\equiv \texttt{handle (let } o = \texttt{raise } \varepsilon(x) \texttt{ in } o\,\{\texttt{ret } y.N \mid \phi\}\\ &\equiv (\lambda o.\texttt{handle } o\,\{\texttt{ret } y.N \mid \phi\})(\texttt{raise } \varepsilon(x))\\ &\equiv \texttt{let } o = (\texttt{raise } \varepsilon(x)) \texttt{ in handle } o\,\{\texttt{ret } y.N \mid \phi\}\end{aligned}$$

□

This lemma is useful for the cast cases of the following, as it reduces to showing the cast is equivalent to one whose ε case is just a re-raise.

Lemma B.4. *If $E \# \varepsilon$, then*

$$E[\texttt{raise } \varepsilon(x)] \equiv \texttt{let } y = \texttt{raise } \varepsilon(x) \texttt{ in } E[y]$$

Proof. By induction on $\varepsilon \# E$

- $\varepsilon \# \bullet$

$$\texttt{raise } \varepsilon(x) \equiv \texttt{let } y = \texttt{raise } \varepsilon(x) \texttt{ in } y$$

- $$\frac{\varepsilon \# E}{\varepsilon \# (\langle B \nwarrow A \rangle E)}$$

$$\begin{aligned}
\langle B \nwarrow A \rangle E[\texttt{raise } \varepsilon(x)] &\equiv \texttt{let } y = E[\texttt{raise } \varepsilon(x)] \texttt{ in } \langle B \nwarrow A \rangle y \\
&\equiv \texttt{let } y = (\texttt{let } z = (\texttt{raise } \varepsilon(x)) \texttt{ in } E[z]) \texttt{ in } \langle B \nwarrow A \rangle y \\
&\equiv \texttt{let } z = (\texttt{raise } \varepsilon(x)) \texttt{ in let } y = E[z] \texttt{ in } \langle B \nwarrow A \rangle y \\
&\equiv \texttt{let } z = (\texttt{raise } \varepsilon(x)) \texttt{ in } \langle B \nwarrow A \rangle E[z]
\end{aligned}$$

- $$\frac{\varepsilon \# E}{\varepsilon \# (\langle A \swarrow B \rangle E)}$$
  Similar to previous.
- $$\frac{\varepsilon \# E}{\varepsilon \# (\texttt{raise } \varepsilon'(E))}$$

$$\begin{aligned}
\texttt{raise } \varepsilon'(E[\texttt{raise } \varepsilon'(x)]) &\equiv \texttt{raise } \varepsilon'((\texttt{let } z = \texttt{raise } \varepsilon'(x) \texttt{ in } E[z])) \\
&\equiv \texttt{let } z = \texttt{raise } \varepsilon'(x) \texttt{ in } E[z] \texttt{raise } \varepsilon'(())
\end{aligned}$$

- $$\frac{\varepsilon \# E \qquad \varepsilon \notin \mathrm{dom}(\phi)}{\varepsilon \# (\texttt{handle } E \, \{\texttt{ret } y.N \mid \phi\})}$$
  Define ψ to be the extension of ϕ with the case $\psi(\varepsilon) = k(\texttt{raise } \varepsilon(x))$.

$$\begin{aligned}
\texttt{handle } E[\texttt{raise } \varepsilon(x)] \, \{\texttt{ret } y.N \mid \phi\} &\equiv \texttt{handle } E[\texttt{raise } \varepsilon(x)] \, \{\texttt{ret } y.N \mid \psi\} \\
&\equiv \texttt{handle } (\texttt{let } z = (\texttt{raise } \varepsilon(x)) \texttt{ in } E[z]) \, \{\texttt{ret } y.N \mid \psi\} \\
&\equiv (\lambda o.\texttt{handle } E[o] \, \{\texttt{ret } y.N \mid \psi\})(\texttt{raise } \varepsilon(x)) \\
&\equiv (\texttt{let } o = (\texttt{raise } \varepsilon(x)) \texttt{ in handle } E[o] \, \{\texttt{ret } y.N \mid \psi\}) \\
&\equiv (\texttt{let } o = (\texttt{raise } \varepsilon(x)) \texttt{ in handle } E[o] \, \{\texttt{ret } y.N \mid \phi\})
\end{aligned}$$

- $$\frac{\varepsilon \# E}{\varepsilon \# (E \, M)}$$

$$\begin{aligned}
(E[\texttt{raise } \varepsilon(x)])M &\equiv \texttt{let } f = E[\texttt{raise } \varepsilon(x)] \texttt{ in let } y = M \texttt{ in } f \, y \\
&\equiv \texttt{let } f = (\texttt{let } z = \texttt{raise } \varepsilon(x) \texttt{ in } E[z]) \texttt{ in let } y = M \texttt{ in } f \, y \\
&\equiv \texttt{let } z = \texttt{raise } \varepsilon(x) \texttt{ in let } f = E[z] \texttt{ in let } y = M \texttt{ in } f \, y \\
&\equiv \texttt{let } z = \texttt{raise } \varepsilon(x) \texttt{ in } (E[z]) \, M
\end{aligned}$$

- $$\frac{\varepsilon \# E}{\varepsilon \# (V \, E)}$$

$$\begin{aligned}(V\,E[\texttt{raise}\ \varepsilon(x)]) &\equiv \texttt{let}\ f = V\ \texttt{in}\ \texttt{let}\ y = E[\texttt{raise}\ \varepsilon(x)]\ \texttt{in}\ f\,y\\ &\equiv \texttt{let}\ f = V\ \texttt{in}\ \texttt{let}\ y = (\texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ E[z])\ \texttt{in}\ f\,y\\ &\equiv \texttt{let}\ y = (\texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ E[z])\ \texttt{in}\ V\,y\\ &\equiv \texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ \texttt{let}\ y = (E[z])\ \texttt{in}\ V\,y\\ &\equiv \texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ V\,(E[z])\end{aligned}$$

- $\dfrac{\varepsilon\#E}{\varepsilon\#(\texttt{if}\ E\{N_t\}\{N_f\})}$

$$\begin{aligned}\texttt{if}\ E[\texttt{raise}\ \varepsilon(x)]\{N_t\}\{N_f\} &\equiv \texttt{let}\ y = (E[\texttt{raise}\ \varepsilon(x)])\ \texttt{in}\ \texttt{if}\ y\{N_t\}\{N_f\}\\ &\equiv \texttt{let}\ y = (\texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ E[z])\ \texttt{in}\ \texttt{if}\ y\{N_t\}\{N_f\}\\ &\equiv \texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ \texttt{let}\ y = (E[z])\ \texttt{in}\ \texttt{if}\ y\{N_t\}\{N_f\}\\ &\equiv \texttt{let}\ z = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ \texttt{if}\ E[z]\{N_t\}\{N_f\}\end{aligned}$$

- $\dfrac{\varepsilon\#E}{\varepsilon\#(\texttt{let}\ x = E\ \texttt{in}\ N)}$

$$\begin{aligned}\texttt{let}\ y = E[\texttt{raise}\ \varepsilon(x)]\ \texttt{in}\ N &\equiv \texttt{let}\ y = \texttt{let}\ z = (\texttt{raise}\ \varepsilon(x))\ \texttt{in}\ E[z]\ \texttt{in}\ N\\ &\equiv \texttt{let}\ z = (\texttt{raise}\ \varepsilon(x))\ \texttt{in}\ \texttt{let}\ y = E[z]\ \texttt{in}\ N\end{aligned}$$

□

Theorem B.5 (Soundness of Operational Semantics). *If $M \mapsto^* M'$ then $M \equiv M'$ is derivable in the inequational theory.*

Proof. (1) The value handle, boolean/function β reductions and error reduction are immediate by axioms.

(2)

$$\frac{E\#\varepsilon}{\texttt{handle}\ E[\texttt{raise}\ \varepsilon(V)]\ \{\texttt{ret}\ y.N \mid \phi\} \equiv \phi(\varepsilon)[V/x, \lambda o.\texttt{handle}\ E[o]\ \{\texttt{ret}\ y.N \mid \phi\}/k]}$$

$$\begin{aligned}\texttt{handle}\ E[\texttt{raise}\ \varepsilon(V)]\ \{\texttt{ret}\ y.N \mid \phi\} &\equiv \texttt{handle}\ (\texttt{let}\ z = \texttt{raise}\ \varepsilon(V)\ \texttt{in}\ E[z])\ \{\texttt{ret}\ y.N \mid \phi\} && \text{(LemmaB.4)}\\ &\equiv \phi(\varepsilon)[V/x, \lambda o.\texttt{handle}\ E[o]\ \{\texttt{ret}\ y.N \mid \phi\}/k]\end{aligned}$$

(3)

$$\langle \tau \leftarrowtail \sigma\rangle V \equiv V$$

By the following:

$$\begin{aligned}\langle \tau \leftarrowtail \sigma\rangle V &\equiv \texttt{handle}\ V\ \{\texttt{ret}\ x.x \mid \phi_{\langle \tau \leftarrowtail \sigma\rangle}\} && \text{(Lemma B.1)}\\ &\equiv V && \text{(Handle } \beta)\end{aligned}$$

(4)
$$\langle \sigma \swarrow \tau \rangle V \equiv V$$
is similar to the previous.

(5)
$$\frac{\varepsilon : A \rightsquigarrow B \in \sigma \qquad \varepsilon : A' \rightsquigarrow B' \in \tau \qquad E\#\varepsilon}{\langle \tau \nwarrow \sigma \rangle E[\texttt{raise}\ \varepsilon(V)] \equiv \texttt{let}\ x = \langle B \swarrow B' \rangle \texttt{raise}\ \varepsilon(\langle A' \nwarrow A \rangle V)\ \texttt{in}\ \langle \tau \nwarrow \sigma \rangle E[x]}$$

$$\begin{aligned}
\langle \tau \nwarrow \sigma \rangle E[\texttt{raise}\ \varepsilon(V)] &\equiv \texttt{handle}\ (E[\texttt{raise}\ \varepsilon(V)])\ \{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\} && \text{(Lemma B.1)}\\
&\equiv \texttt{handle}\ (\texttt{let}\ z = \texttt{raise}\ \varepsilon(V)\ \texttt{in}\ E[z])\ \{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\} && \text{(Lemma B.4)}\\
&\equiv \phi_{\langle \tau \nwarrow \sigma \rangle}(\varepsilon)[V/x, \lambda o.\texttt{handle}\ E[o]\ \{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\}]\\
&= (\lambda o.\texttt{handle}\ E[o]\ \{\texttt{ret}\ x.x \mid \phi_{\langle \tau \nwarrow \sigma \rangle}\})(\langle B \swarrow B' \rangle \texttt{raise}\ \varepsilon(\langle A' \nwarrow A \rangle V))\\
&\equiv (\lambda o.\langle \tau \nwarrow \sigma \rangle E[o])(\langle B \swarrow B' \rangle \texttt{raise}\ \varepsilon(\langle A' \nwarrow A \rangle V))\\
&\equiv \texttt{let}\ o = (\langle B \swarrow B' \rangle \texttt{raise}\ \varepsilon(\langle A' \nwarrow A \rangle V))\ \texttt{in}\ \langle \tau \nwarrow \sigma \rangle E[o]
\end{aligned}$$

(6)
$$\frac{\varepsilon : A \rightsquigarrow B \in \sigma \qquad \varepsilon : A' \rightsquigarrow B' \in \tau \qquad E\#\varepsilon}{\langle \sigma \swarrow \tau \rangle E[\texttt{raise}\ \varepsilon(V)] \equiv \texttt{let}\ x = \langle B' \nwarrow B \rangle \texttt{raise}\ \varepsilon(\langle A \swarrow A' \rangle V)\ \texttt{in}\ \langle \sigma \swarrow \tau \rangle E[x]}$$
Similar to previous

(7)
$$\frac{\varepsilon \notin \sigma \qquad E\#\varepsilon}{\langle \sigma \swarrow ? \rangle E[\texttt{raise}\ \varepsilon(V)] \equiv \mho}$$

$$\begin{aligned}
\langle \sigma \swarrow ? \rangle E[\texttt{raise}\ \varepsilon(V)] &\equiv \texttt{handle}\ (E[\texttt{raise}\ \varepsilon(V)])\ \{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow ? \rangle}\} && \text{(Lemma B.1)}\\
&\equiv \texttt{handle}\ (\texttt{let}\ z = (\texttt{raise}\ \varepsilon(V))\ \texttt{in}\ E[z])\ \{\texttt{ret}\ x.x \mid \phi_{\langle \sigma \swarrow ? \rangle}\} && \text{(Lemma B.4)}\\
&\equiv \mho
\end{aligned}$$

(8)
$$\langle \texttt{bool} \nwarrow \texttt{bool} \rangle V \equiv V$$
By the identity rule.

(9)
$$\langle \texttt{bool} \swarrow \texttt{bool} \rangle V \equiv V$$
By the identity rule.

(10)
$$(\langle (A' \to_\tau B') \nwarrow (A \to_\sigma B) \rangle V_f)\ V \equiv \langle B' \nwarrow B \rangle \langle \tau \nwarrow \sigma \rangle (V_f \langle A \swarrow A' \rangle V)$$
By the following:

$$\begin{aligned}
(\langle (A' \to_\tau B') \nwarrow (A \to_\sigma B) \rangle V_f)\ V &\equiv ((\lambda x.\langle B' \nwarrow B \rangle \langle \tau \nwarrow \sigma \rangle (V_f(\langle A \swarrow A' \rangle x)))\ V && \text{(Lemma B.2)}\\
&\equiv \langle B' \nwarrow B \rangle \langle \tau \nwarrow \sigma \rangle (V_f(\langle A \swarrow A' \rangle)V) && (\beta \to)
\end{aligned}$$

(11) Similar to previous.

□

$$\texttt{bool} \leq \texttt{bool} \qquad \frac{d_i \leq c_i \qquad c_e \leq d_e \qquad c_o \leq d_o}{c_i \to_{c_e} c_o \leq d_i \to_{d_e} d_o} \qquad ? \leq ? \qquad \frac{c \leq d}{\texttt{inj}(c) \leq \texttt{inj}(d)}$$

$$\frac{\begin{array}{c}\mathrm{dom}(d_c) \subseteq \mathrm{dom}(d_c') \\ \forall \varepsilon : c \rightsquigarrow d \in d_c.\varepsilon : c' \rightsquigarrow d' \in d_c' \wedge c \leq c' \wedge d' \leq d\end{array}}{d_c \leq d_c'} \qquad \frac{c \leq \texttt{inj}(\Sigma)}{c \leq ?} \qquad \frac{c \leq d}{c \leq \texttt{inj}(d)}$$

Fig. 20. Subtyping of Precision Derivations

Theorem B.6 (Adequacy). *If* $\cdot \vdash_\emptyset M \equiv M' : \texttt{bool}$ *is derivable in the equational theory than for any* $R \in \{\texttt{true}, \texttt{false}, \mho\}$

$$M \mapsto^* R \iff M' \mapsto^* R$$

Corollary B.7 (Consistency). $\texttt{true} \equiv \texttt{false}$ *is not derivable.*

Theorem B.8 (Graduality). *If* $\cdot \vdash_\emptyset M \sqsubseteq M' : \texttt{bool}$ *Then for any* $R \in \{\texttt{true}, \texttt{false}\}$,

$$M \mapsto^* R \Rightarrow M' \mapsto^* R$$

and for any $R' \in \{\texttt{true}, \texttt{false}, \mho\}$,

$$M' \mapsto^* R' \implies M \mapsto^* R'$$

C ELABORATION OF GRADUAL SUBTYPING

First, we define in Figure 20 a subtyping of precision derivations.

Lemma C.1. *If* $A \lesssim B$ *then there exist types* A_h, D_h, D_l, B_l *with*

(1) $c_l : A \sqsubseteq D_l$ *and* $c_h : A_h \sqsubseteq D_h$ *satisfying* $c_l \leq c_h$
(2) $d_l : B_l \sqsubseteq D_l$ *and* $d_h : B \sqsubseteq D_h$ *satisfying* $d_l \leq d_h$
(3) $e_l : D_l \sqsubseteq D$ *and* $e_h : D_h \sqsubseteq D$ *with* $e_l \leq e_h$ *where* $D = |A| = |B|$.

Proof. By induction on the proof of $A \lesssim A'$. □

Then the four different choices of cast are all equivalent in the inequational theory:

Lemma C.2. *Given* $A, A_h, B, B_l, D_l, D_h, D, c_l, c_h, d_l, d_h, e_l, e_h$ *as in the output of the previous lemma, for any* $\Gamma \vdash M : \sigma \,!\, A$*, the following four terms are equivalent at type* B.

(1) $\langle B \twoheadleftarrow D_h\rangle\langle D_h \leftarrowtail A_h\rangle M$
(2) $\langle B \twoheadleftarrow D_h\rangle\langle D_l \leftarrowtail A\rangle M$
(3) $\langle B_l \twoheadleftarrow D_l\rangle\langle D_l \leftarrowtail A\rangle M$
(4) $\langle B \twoheadleftarrow D\rangle\langle D \leftarrowtail A\rangle M$

Proof. (1) To show (1) is equivalent to (2), it suffices to show

$$\langle D_h \leftarrowtail A_h\rangle M \equiv \langle D_l \leftarrowtail A\rangle M$$

which is an instance of the subtyping/cast rule since $c_l \sqsubseteq c_h$.

(2) Similarly to show (2) is equivalent to (3) follows from $d_l \leq d_h$
(3) Lastly we show (4) is equivalent to (2). By cast functoriality,

$$\langle B \twoheadleftarrow D\rangle\langle D \leftarrowtail A\rangle M \equiv \langle B \twoheadleftarrow D_h\rangle\langle D_h \twoheadleftarrow D\rangle\langle D \leftarrowtail D_l\rangle\langle D_l \leftarrowtail A\rangle M$$

And by retraction the middle cast $\langle D_h \twoheadleftarrow D\rangle\langle D \leftarrowtail D_l\rangle$ is the identity.

□

D GRADUALITY

Our main goal is to prove the soundness of the inequational theory with respect to the logical relation. That is

Theorem D.1 (Graduality). *If* $\Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq N : c$ *then* $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} M \sqsubseteq N : c$

Proof. By induction on the term precision derivation.

(1) (ValSubst) Lemma D.29
(2) (MonadUnitL) Lemma D.30
(3) (MonadUnitR) Lemma D.31
(4) (MonadAssoc) Lemma D.32
(5) (BoolBeta) Lemmas D.34 and D.35
(6) (BoolEta) Lemma D.33
(7) (IfEval) Lemma D.36
(8) (FunBeta) Lemma D.37
(9) (FunEta) Lemma D.38
(10) (AppEval) Lemma D.39
(11) (HandleBetaRet) Lemma D.40
(12) (HandleBetaRaise) Lemma D.41
(13) (HandleEmpty) Lemma D.43
(14) (HandleExt) Lemma D.44
(15) (RaiseEval) Lemma D.42
(16) (Variable) Lemma D.21
(17) (Let) Lemma D.25
(18) (Boolean) Lemma D.20
(19) (If) Lemma D.24
(20) (Lambda) Lemma D.22
(21) (App) Lemma D.23
(22) (Raise) Lemma D.26
(23) (HandleCong) Lemma D.27
(24) (Transitivity) Lemma D.67
(25) (ErrBot) Lemma D.45
(26) (ErrStrict) Lemma D.46
(27) (SubtyMon) Lemma D.47
(28) (ValUpSub) Lemma D.59
(29) (ValDnSub) Lemma D.59
(30) (EffUpSub) Lemma D.59
(31) (EffDnSub) Lemma D.59
(32) (ValUpL) Follows from Lemma D.49.
(33) (ValUpR) Follows from Lemma D.48.
(34) (ValUpEval) Lemma D.56
(35) (ValDnR) Follows from Lemma D.51.
(36) (ValDnL) Follows from Lemma D.50.
(37) (ValDnEval) Lemma D.57
(38) (ValRetract) Lemma D.58.
(39) (EffUpL) Follows from Lemma D.53
(40) (EffUpR) Follows from Lemma D.52
(41) (EffDnR) Follows from Lemma D.55

(42) (EffDnL) Follows from Lemma D.54
(43) (EffRetract) Lemma D.58.

□

We begin with a few lemmas that will be useful in our proofs.

D.0.1 Lemmas.

Lemma D.2. *If $(V_1, V_2) \in R$, and V_1 and V_2 are values of type A^l and A^r respectively, then $(V_1, V_2) \in \mathcal{R}^{\sim}_j[\![d_\sigma]\!](R, A^l, A^r)$.*

Proof. We will establish the first disjunct in the definition of $\mathcal{R}^{\sim}[\![\cdot]\!]$. This follows by assumption. □

Lemma D.3. *If $(V_1, V_2) \in \mathcal{R}^{\sim}_j[\![d_\sigma]\!](R, A^l, A^r)$, then $(V_1, V_2) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!](R, A^l, A^r)$.*

Proof. Let $\sim \in \{<, >\}$, and suppose $(V_1, V_2) \in \mathcal{R}^{\sim}_j[\![d_\sigma]\!](R, A^l, A^r)$. Notice that regardless of whether $\sim$ is $<$ or $>$, we will be able to show the last clause in the definition of $\mathcal{E}^{\precsim}_j[\![d_\sigma]\!](R, A^l, A^r)$ or $\mathcal{E}^{\succsim}_j[\![d_\sigma]\!](R, A^l, A^r)$. In particular, we can take $k = j$, $V_1 = V_1$, and $V_2 = V_2$, noting that V_1 steps to itself in 0 steps, as does V_2. Thus, it remains to show that V_1 and V_2 are related by $\mathcal{R}^{\precsim}_j[\![d_\sigma]\!](R, A^l, A^r)$ or $\mathcal{R}^{\succsim}_j[\![d_\sigma]\!](R, A^l, A^r)$. This is true by assumption. □

Lemma D.4. *If $(V_1, V_2) \in \mathcal{V}^{\sim}_j[\![c]\!]$, then $(V_1, V_2) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$.*

Proof. By Lemma D.3 (with $R = \mathcal{V}^{\sim}[\![c]\!]$), it suffices to show that $(\sigma_1 V_1, \sigma_1 V_2) \in \mathcal{R}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. This is true by Lemma D.2, again with $R = \mathcal{V}^{\sim}[\![c]\!]$. □

Lemma D.5 (anti-reduction, one-sided). *Suppose $M_1 \mapsto^{i_1} M_1'$ and $M_2 \mapsto^{i_2} M_2'$.*
If $(M_1', M_2') \in \mathcal{E}^{\succsim}_{j-i_2}[\![d_\sigma]\!](R, A^l, A^r)$, then $(M_1, M_2) \in \mathcal{E}^{\succsim}_j[\![d_\sigma]\!](R, A^l, A^r)$.
Similarly, if $(M_1', M_2') \in \mathcal{E}^{\precsim}_{j-i_1}[\![d_\sigma]\!](R, A^l, A^r)$, then $(M_1, M_2) \in \mathcal{E}^{\precsim}_j[\![d_\sigma]\!](R, A^l, A^r)$.

Proof. We prove the first statement; the second is analogous (and in fact easier). The assumption that $(M_1', M_2') \in \mathcal{E}^{\succsim}_{j-i_2}[\![d_\sigma]\!](R, A^l, A^r)$ has four cases:

(1) $M_2' \mapsto^{j-i_2+1}$. In this case, $M_2 \mapsto^{i_2} M_2' \mapsto^{j-i_2+1}$, i.e, $M_2 \mapsto^{j+1}$. Thus, we may assert the first disjunct in the definition of $\mathcal{E}^{\succsim}_j[\![d_\sigma]\!](R, A^l, A^r)$.
(2) There exists $k \le j - i_2$ such that $M_1' \mapsto^{j-i_2-k} \mho$, and furthermore $M_1' \mapsto^* \mho$. In this case, we have that $M_2 \mapsto^{i_2} M_2' \mapsto^{j-i_2-k} \mho$, so $M_2 \mapsto^{j-k} \mho$. Also, $M_1 \mapsto^{i_1} M_1' \mapsto^* \mho$, so $M_1 \mapsto^* \mho$. Thus, we may assert the second disjunct.
(3) There exists $k \le j - i_2$ and N_2 such that $M_2' \mapsto^{j-i_2-k} N_2$ and $M_1' \mapsto^* \mho$. In this case we have $M_2 \mapsto^{i_2} M_2' \mapsto^{j-i_2-k} N_2$, so $M_2 \mapsto^{j-k} N_2$. Thus, we may assert the third disjunct.
(4) Similar to previous case.

□

Lemma D.6 (anti-reduction). *Suppose $M_1 \mapsto^{i_1} M_1'$ and $M_2 \mapsto^{i_2} M_2'$, and that $(M_1', M_2') \in \mathcal{E}^{\sim}_{j-m}[\![d_\sigma]\!](R, A^l, A^r)$, where $m = \min\{i_1, i_2\}$. Then $(M_1, M_2) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!](R, A^l, A^r)$.*

Proof. Follows from one-sided anti-reduction (Lemma D.5) and downward closure. □

Lemma D.7 (forward reduction, one-sided). *Suppose $M_1 \mapsto^{i_1} M_1'$ and $M_2 \mapsto^{i_2} M_2'$.*
If $(M_1, M_2) \in \mathcal{E}^{\succsim}_{j+i_2}[\![d_\sigma]\!](R, A^l, A^r)$, then $(M_1', M_2') \in \mathcal{E}^{\succsim}_j[\![d_\sigma]\!](R, A^l, A^r)$.
Similarly, if $(M_1, M_2) \in \mathcal{E}^{\precsim}_{j+i_1}[\![d_\sigma]\!](R, A^l, A^r)$, then $(M_1', M_2') \in \mathcal{E}^{\precsim}_j[\![d_\sigma]\!](R, A^l, A^r)$.

Proof. Follows from determinism of evaluation and a case analysis on the assumption that M_1 and M_2 are related. □

Lemma D.8 (forward reduction). *Suppose $M_1 \mapsto^{i_1} M_1'$ and $M_2 \mapsto^{i_2} M_2'$, and that $(M_1, M_2) \in \mathcal{E}^{\sim}_{j+m}[\![d_\sigma]\!](R, A^l, A^r)$, where $m = \max\{i_1, i_2\}$. Then $(M_1', M_2') \in \mathcal{E}^{\sim}_{j}[\![d_\sigma]\!](R, A^l, A^r)$.*

Proof. Follows from one-sided forward reduction (Lemma D.7) and downward closure. □

Frequently in our proofs we will encounter a situation where we know that two evaluation contexts are related in the $\mathcal{K}^{\sim}[\![\cdot]\!]$ relation, that is, substituting related values gives related outputs. On the other hand, as a cast applied to a value is not necessarily itself a value, we cannot reason directly about what happens when such semantic values are substituted into related evaluation contexts. We therefore introduce the following lemma.

Lemma D.9. *Suppose E_1 and E_2 are evaluation contexts that take values to values. Let V_1 and V_2 be values (not necessarily related) such that*

$$(E_1[V_1], E_2[V_2]) \in \mathcal{E}^{\sim}_{j}[\![d'_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

Furthermore, let $(E^l[x^l], E^r[x^r] \in \mathcal{K}^{\sim}_{j}[\![c]\!]\mathcal{E}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!])$.
Then

$$(E^l[E_1[V_1]], E^r[E_2[V_2]]) \in \mathcal{E}^{\sim}_{j}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

Proof. We show the proof for $\sim = \gtrsim$.
By assumption, we have that there exist values V_1' and V_2' such that $E_1[V_1] \mapsto^{i_1} V_1'$ and $E_2[V_2] \mapsto^{i_2} V_2'$, for some i_1 and i_2.
Thus, $E^l[E_1[V_1]] \mapsto^{i_1} E^l[V_1']$ and likewise $E^r[E_2[V_2]] \mapsto^{i_2} E^r[V_2']$.
By one-sided anti-reduction (Lemma D.5), it suffices to show that

$$(E^l[V_1'], E^r[V_2']) \in \mathcal{E}^{\gtrsim}_{j-i_2}[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![d]\!].$$

By assumption on E^l and E^r being related, it suffices to show that $(V_1', V_2') \in \mathcal{V}^{\gtrsim}_{j-i_2}[\![c]\!]$.
Now by one-sided forward reduction (Lemma D.7), it suffices to show

$$(E_1[V_1], E_2[V_2]) \in \mathcal{E}^{\gtrsim}_{(j-i_2)+i_2}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

But this is precisely our assumption, so we are finished.

□

Remark: The reason why we needed to consider cases on $\sim$ separately is that the more "generic"/two-sided anti-reduction and forward-reduction lemmas involve the min or max of the number of steps taken by the two terms. These may not be equal, in which case the arithmetic wouldn't work out. But this doesn't mean the above lemma is false. Conceptually, what is happening is that in the two-sided variants of the lemmas, $\sim$ could be either $>$ or $<$. On the other hand, the key here is that $\sim$ stays the same throughout the application of anti-reduction and forward reduction, so we are able to use the more specific, one-sided lemmas.

Lemma D.10 (time-out). *If $M_1 \mapsto^{(i+1)}$, then $(M_1, M_2) \in \mathcal{E}^{\lesssim}_{i}[\![d_\sigma]\!]R$. Similarly, if $M_2 \mapsto^{(i+1)}$, then $(M_1, M_2) \in \mathcal{E}^{\gtrsim}_{i}[\![d_\sigma]\!]R$.*

Proof. Suppose $M_1 \mapsto^{(i+1)}$. Then we may assert the first disjunct in the definition of $\mathcal{E}^{\lesssim}_{i}[\![d_\sigma]\!]R$ to conclude that $(M_1, M_2) \in \mathcal{E}^{\lesssim}_{i}[\![d_\sigma]\!]R$. Likewise, if $M_2 \mapsto^{(i+1)}$, then we may assert the first disjunct in the definition of $\mathcal{E}^{\gtrsim}_{i}[\![d_\sigma]\!]R$ to conclude that $(M_1, M_2) \in \mathcal{E}^{\gtrsim}_{i}[\![d_\sigma]\!]R$. □

We present two trivial lemmas about the later modality. We do this to cut down on tedious reasoning about step indices within other proofs.

Lemma D.11. *Let R be a monotone step-indexed relation. If $(M_1, M_2) \in R_j$, then $(M_1, M_2) \in (\blacktriangleright R)_j$.*

Proof. Suppose $(M_1, M_2) \in R_j$. If $j = 0$, then $(M_1, M_2) \in (\blacktriangleright R)_0$ trivially.
Otherwise, let $j = j' + 1$. By monotonicity of R, we have $(M_1, M_2) \in R_{j'}$, from which it follows that $(M_1, M_2) \in (\blacktriangleright R)_j$. □

Lemma D.12. *Let R be a monotone step-indexed relation, and let j be of the form $j = j' + 1$. If $(M_1, M_2) \in (\blacktriangleright R)_j$, then $(M_1, M_2) \in (\blacktriangleright R)_{j'}$.*

Proof. Suppose $(M_1, M_2) \in (\blacktriangleright R)_j$. Since $j = j' + 1$, by definition of $\blacktriangleright$ we must have that $(M_1, M_2) \in R_{j'}$. By the previous lemma (Lemma D.11), we conclude $(M_1, M_2) \in (\blacktriangleright R)_{j'}$, which is what we needed to show. □

Lemma D.13 (Reasoning with "later" when both sides step). *Suppose $M \mapsto^1 M'$ and $N \mapsto^1 N'$, and that $(M', N') \in (\blacktriangleright \mathcal{E}^{\sim}[\![d_\sigma]\!])_k R$. Then $(M, N) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!] R$.*

Proof. First suppose $k = 0$. Then by the time-out lemma (Lemma D.10), regardless of whether $\sim$ is $<$ or $>$, we have $(M, N) \in \mathcal{E}^{\sim}_0[\![d_\sigma]\!] R$.
Now suppose $k \geq 1$. Then by the definition of later, we have that $(M', N') \in \mathcal{E}^{\sim}_{k-1}[\![d_\sigma]\!] R$, so by anti-reduction we have that $(M, N) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!] R$. □

Lemma D.14 (Löb-induction). *Let $P(n)$ be a predicate indexed by a natural number n. Suppose for all natural numbers n, we have that $(\blacktriangleright^m P)(n)$ implies $P(n)$ for all $m \geq 1$. Then $P(n)$ is true for all natural numbers n.*

Proof. The proof is by induction on n. When $n = 0$, the assumption says that $(\blacktriangleright P)(0)$ implies $P(0)$ (we have taken $m = 1$). So, it suffices to show that $(\blacktriangleright P)(0)$ holds. This is true by the definition of later.
Now let $n \geq 1$ be fixed, and suppose $P(n)$ is true. We claim that $P(n+1)$ is true. By our assumption, it will suffice to show that $(\blacktriangleright P)(n + 1)$ is true. (We have again chosen $m = 1$.) By definition of later, we must show $P(n)$ is true. But $P(n)$ is true by assumption. □

We now introduce a key lemma about evaluation contexts.
Note: In the below, we omit explicit mention of the types associated to the relations that parameterize $\mathcal{E}^{\sim}[\![\cdot]\!]$ and $\mathcal{R}^{\sim}[\![\cdot]\!]$.

Lemma D.15. *If*

(1) $(M_1, M_2) \in \mathcal{E}^{\sim}_j[\![d'_\sigma]\!] S'$

(2) For all $k \leq j$ and $(N_1, N_2) \in \mathcal{R}^{\sim}_k[\![d'_\sigma]\!] S'$, we have $(E_1[N_1], E_2[N_2]) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!] S$,

then $(E_1[M_1], E_2[M_2]) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!] S$.

Proof. We prove the lemma for $\sim = >$; the other case is similar. Based on assumption (1), there are four cases:

(1) Case $M_2 \mapsto^{j+1}$. We have $E_2[M_2] \mapsto^{j+1}$, so we may assert the first disjunct in the definition of $\mathcal{E}^{\sim}_j[\![d_\sigma]\!] S$ to conclude that $(E_1[M_1], E_2[M_2]) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!] S$.
(2) Case $\exists k \leq j$ such that $M_2 \mapsto^{j-k} \mho$ and $M_1 \mapsto^* \mho$. We have $E_2[M_2] \mapsto^{j-k+1} \mho$. If $k = 0$, then we have $E_2[M_2] \mapsto^{j+1}$, so we may assert the first disjunct. Otherwise, if $k \geq 1$, then we may take $k' = k - 1$ and observe that $E_2[M_2] \mapsto^{j-k'} \mho$.

(3) Case $\exists k \le j, \exists V_2$ such that $M_2 \mapsto^{j-k} N_2$ and $M_1 \mapsto^* \mho$. We have $E_2[M_2] \mapsto^{j-k} E_2[N_2]$, so we may assert the third disjunct with $k = k$ and $N_2 = E_2[N_2]$.

(4) Case $\exists k \le j, \exists (N_1, N_2) \in \mathcal{R}^{\succeq}_k [\![d_\sigma]\!] S'$ such that $M_2 \mapsto^{j-k} N_2$ and $M_1 \mapsto^* N_1$. We have $E_1[M_1] \mapsto^{i_1} E_1[N_1]$ for some i_1, and $E_2[M_2] \mapsto^{j-k} E_2[N_2]$. By assumption (2), we have $(E_1[N_1], E_2[N_2]) \in \mathcal{E}^{\sim}_k [\![d_\sigma]\!] S$. Thus, we may assert the fourth disjunct with $V_1 = E_1[N_1]$ and $V_2 = E_2[N_2]$.

□

Lemma D.16 ("Semantic bind"). *Let $c : A \sqsubseteq A'$ and $d : B \sqsubseteq B'$. Let E_1 and E_2 be evaluation contexts such that $\Sigma \mid \Gamma \mid \bullet : (d'^l_\sigma \,!\, A) \vdash_{d^l_\sigma} E_1 : B$ and $\Sigma \mid \Gamma \mid \bullet : (d'^r_\sigma \,!\, A') \vdash_{d^r_\sigma} E_2 : B'$. Suppose*

(1) $(M_1, M_2) \in \mathcal{E}^{\sim}_j [\![d'_\sigma]\!](S', A, A')$.

(2) For all $k \le j$ and $(V_1, V_2) \in S'_k$, we have $(E_1[V_1], E_2[V_2]) \in \mathcal{E}^{\sim}_k [\![d_\sigma]\!](S, B, B')$.

(3) For all $k \le j$ and for all $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in d'_\sigma$, if E_1 catches ε or E_2 catches ε, then for all $V^l, V^r \in (\blacktriangleright \mathcal{V}^{\sim}[\![c_\varepsilon]\!])_k$ and all evaluation contexts $E^l \# \varepsilon$ and $E^r \# \varepsilon$ such that $(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright \mathcal{K}^{\sim}[\![d_\varepsilon]\!])_k(\mathcal{E}^{\sim}[\![d'_\sigma]\!](S', A, A'), (d'^l_\sigma \,!\, A), (d'^r_\sigma \,!\, A'))$, we have $(E_1[E^l[\texttt{raise}\ \varepsilon(V^l)]], E_2[E^r[\texttt{raise}\ \varepsilon(V^r)]]) \in \mathcal{E}^{\sim}_k [\![d_\sigma]\!](S, B, B')$.

Then $(E_1[M_1], E_2[M_2]) \in \mathcal{E}^{\sim}_j [\![d_\sigma]\!](S, B, B')$.

Proof. We use Löb induction (Lemma D.14). We assume that if the premises of the lemma are satisfied "later", then the conclusion holds later. We show under this assumption that the lemma holds "now".

We first apply Lemma D.15. The first hypothesis is immediate. Now let $k \le j$ and let $(N_1, N_2) \in \mathcal{R}^{\sim}_k [\![d'_\sigma]\!](S', A, A')$. We need to show that

$$(E_1[N_1], E_2[N_2]) \in \mathcal{E}^{\sim}_k [\![d_\sigma]\!](S, B, B').$$

There are two cases to consider. In the first case, N_1 and N_2 are values and $(N_1, N_2) \in \mathcal{V}^{\sim}_j [\![c]\!]$. Then by assumption (2) with $k = j$, we have $(E_1[N_1], E_2[N_2]) \in \mathcal{E}^{\sim}_j [\![d_\sigma]\!](S, B, B')$, as needed.

In the second case, there exist $\varepsilon' : c' \rightsquigarrow d' \in d'_\sigma$, $E^l \# \varepsilon'$, $E^r \# \varepsilon'$, and V^l, V^r such that $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![c']\!])_j$, and $(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright \mathcal{K}^{\sim}[\![d']\!])_j(\mathcal{E}^{\sim}[\![d'_\sigma]\!](S', A, A'), (d'^l_\sigma \,!\, A), (d'^r_\sigma \,!\, A'))$, and $N_1 = E^l[\texttt{raise}\ \varepsilon'(V^l)]$ and $N_2 = E^r[\texttt{raise}\ \varepsilon'(V^r)]$.

Let $N'_1 = E_1[N_1] = E_1[E^l[\texttt{raise}\ \varepsilon'(V^l)]]$ and $N'_2 = E_2[N_2] = E_2[E^r[\texttt{raise}\ \varepsilon'(V^r)]]$.

We need to show that

$$(N'_1, N'_2) \in \mathcal{E}^{\sim}_j [\![d_\sigma]\!](S, B, B').$$

We now consider whether one of E_1 or E_2 catches ε', or whether neither catches it. In the former case, assumption (3) immediately implies the desired result.

Now suppose neither E_1 nor E_2 catches ε. In this case, note that since $\varepsilon' \# E^l$ and $\varepsilon' \# E_1$, we have $\varepsilon' \# E_1[E^l]$. Likewise, we have $\varepsilon' \# E_2[E^r]$. It follows that N'_1 and N'_2 are stuck terms, i.e., they do not step. Thus, it suffices to show that

$$(N'_1, N'_2) \in \mathcal{R}^{\sim}_j [\![d_\sigma]\!](S, B, B').$$

We first claim $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![c']\!])_j$. Since $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![c']\!])_j$, this follows by Lemma D.12.

We now claim that

$$(x^l.(E_1[E^l[x^l]]), x^r.(E_2[E^r[x^r]])) \in (\blacktriangleright \mathcal{K}^{\sim}[\![d']\!])_j(\mathcal{E}^{\sim}[\![d_\sigma]\!](S, B, B'), (d^l_\sigma \,!\, B), (d^r_\sigma \,!\, B')).$$

To this end, let $k \leq j$ and let $(V'^l, V'^r) \in (\blacktriangleright\mathcal{V}^\sim[\![d']\!])_k$. We need to show that

$$(E_1[E^l[V'^l]], E_2[E^r[V'^r]]) \in (\blacktriangleright\mathcal{E}^\sim[\![d_\sigma]\!])_k(S, B, B').$$

By the Löb induction hypothesis, it suffices to show that the three hypotheses of the lemma hold later. We claim that $(E^l[V'^l], E^r[V'^r]) \in (\mathcal{E}^\sim_k[\![d'_\sigma]\!]\mathcal{V}^\sim[\![c]\!])$. To see this, recall our assumption that

$$(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}^\sim[\![d']\!])_j(\mathcal{E}^\sim[\![d'_\sigma]\!]\mathcal{V}^\sim[\![c]\!]).$$

Thus, we have that $(E^l[V'^l], E^r[V'^r]) \in (\blacktriangleright\mathcal{E}^\sim[\![d'_\sigma]\!])_k(\mathcal{V}^\sim[\![c]\!])$, which is what we needed to show.

□

We now introduce a few lemmas about precision derivations. We first show how we may "compose" precision derivations:

Lemma D.17 (cut admissibility for precision derivations).
- *If $c : A \sqsubseteq B$ and $d : B \sqsubseteq C$ then $c \circ d : A \sqsubseteq C$.*
- *If $d_\sigma : \sigma \sqsubseteq \sigma'$ and $d'_\sigma : \sigma' \sqsubseteq \sigma''$ then $d_\sigma \circ d'_\sigma : \sigma \sqsubseteq \sigma''$.*

Proof. We prove these statements simultaneously by induction on d and d'_σ.

- Case $d = \texttt{bool}$. We have $B = C = \texttt{bool}$, so $c = \texttt{bool}$ (the reflexivity derivation). Thus, we may take $c \circ d = \texttt{bool}$.
- Case $d = d_i \to_{d_\sigma} d_o$. Inspecting the rules in figure 13, we see that $B = B_i \to_{B_\sigma} B_o$ and $C = C_i \to_{C_\sigma} C_o$. Thus, we must have $A = A_i \to_{A_\sigma} A_o$, which means that $c = c_i \to_{c_\sigma} c_o$.
  We may take $c \circ d = (c_i \circ d_i) \to_{c_\sigma \circ d_\sigma} (c_o \circ d_o)$. By our inductive hypotheses, we have (1) $c_i \circ d_i : A_i \sqsubseteq C_i$, (2) $c_\sigma \circ d_\sigma : A_\sigma \sqsubseteq C_\sigma$, and (3) $c_o \circ d_o : A_o \sqsubseteq C_o$. Now, using the type precision formation rule for functions, we get that $(c_i \circ d_i) \to_{c_\sigma \circ d_\sigma} (c_o \circ d_o) : (A_i \to_{A_\sigma} A_o \sqsubseteq C_i \to_{C_\sigma} C_o)$.
- Case $d'_\sigma = ?$. Define $? \circ ? = ?$. Define $\texttt{Inj}(d) \circ ? = \texttt{Inj}(d)$. An concrete effect set cannot be composed with ?.
- Case $d'_\sigma = \texttt{Inj}(d)$. Note that $\sigma'' = ?$. We define $d_\sigma \circ \texttt{Inj}(d) = \texttt{Inj}(d_\sigma \circ d)$.
- Case $d'_\sigma = d'_c$: Define $(d_c \circ d'_c)$ by $\varepsilon : c \rightsquigarrow d \in (d_c \circ d'_c)$ if and only if $c = c_1 \circ c_2$ and $d = d_1 \circ d_2$ with $\varepsilon : c \rightsquigarrow_1 d_1 \in d_c$ and $\varepsilon : c \rightsquigarrow_2 d_2 \in d'_c$.

□

Lemma D.18 (reflexivity of composition). *Let $c : A \sqsubseteq B$ and $d_\sigma : \sigma \sqsubseteq \sigma'$. The following hold.*

- *$c \circ B = A \circ c = c$.*
- *$d_\sigma \circ \sigma' = \sigma \circ d_\sigma = d_\sigma$.*

Proof. Follows from the uniqueness of precision derivations. That is, $c \circ B$, $A \circ c$, and c all are all proofs of $A \sqsubseteq B$, hence are equal. □

Lemma D.19 (decomposition). *Suppose $\varepsilon : c \rightsquigarrow d \in d_\sigma \circ d'_\sigma$. Then there exist c_1, c_2 and d_1, d_2 such that $\varepsilon : c_1 \rightsquigarrow d_1 \in d_\sigma$ and $\varepsilon : c_2 \rightsquigarrow d_2 \in d'_\sigma$ and $c = c_1 \circ c_2$ and $d = d_1 \circ d_2$.*

Proof. By induction on d'_σ.

- Case $d'_\sigma = ?$. If $d_\sigma = ?$, then our assumption becomes $\varepsilon : c \rightsquigarrow d \in ? \circ ? = ?$. By definition of membership in ?, this means that $\varepsilon : c^r \rightsquigarrow d^r \in \Sigma$.
  We may take $c_1 = c$ and take c_2 to be the reflexivity derivation for $c^r \sqsubseteq c^r$. Likewise, we take $d_1 = d$ and d_2 to be the reflexivity derivation for $d^r \sqsubseteq d^r$. Note that $\varepsilon : c_2 \rightsquigarrow d_2 \in ?$,

because $c_2^r = c^r$ and $d_2^r = d^r$, and we know $\varepsilon : c^r \rightsquigarrow d^r \in \Sigma$. We also have that $c = c_1 \circ c_2$ and $d = d_1 \circ d_2$, using Lemma D.18.
If $d_\sigma = \mathtt{inj}(d_\sigma)$, then our assumption becomes $\varepsilon : c \rightsquigarrow d \in \mathtt{inj}(d_\sigma)$. By definition of membership in $\mathtt{Inj}(,)$, we have that $\varepsilon : c \rightsquigarrow d \in d_\sigma$. We may again take $c_1 = c$ and c_2 to be the reflexivity derivation for $c^r \sqsubseteq c^r$, and likewise for d_1 and d_2. The same reasoning as above applies.

- Case $d'_\sigma = \mathtt{inj}(d_\sigma)$. By definition of composition, our assumption becomes $\varepsilon : c \rightsquigarrow d \in (d_\sigma \circ \mathtt{inj}(d_\sigma)) = \mathtt{inj}(d_\sigma \circ d_\sigma)$.
By the induction hypothesis, there are c_1, c_2 and d_1, d_2 such that $\varepsilon : c_1 \rightsquigarrow d_1 \in d_\sigma$ and $\varepsilon : c_2 \rightsquigarrow d_2 \in d_\sigma$ and $c = c_1 \circ c_2$ and $d = d_1 \circ d_2$. By definition of membership in $\mathtt{Inj}(,)$, we have $\varepsilon : c_2 \rightsquigarrow d_2 \in \mathtt{inj}(d_\sigma) = d'_\sigma$.
- Case $d'_\sigma = d'_c$ (concrete effect set). Similar to previous case.

□

D.0.2 Congruence Rules. With these lemmas, we can prove the soundness of the term precision congruence rules. The proofs are by induction on the term precision derivation.

Lemma D.20 (Congruence for Booleans).

Proof. We need to show that $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} [\![\mathtt{true}]\!] \sqsubseteq [\![\mathtt{true}]\!] \in \mathtt{bool}$, and likewise for $\mathtt{false}$ (we will show this for $\mathtt{true}$ only; the reasoning for $\mathtt{false}$ is exactly the same.)
Let $\sim \in \{<, >\}$ and let $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show

$$(\mathtt{true}[\gamma_1], \mathtt{true}[\gamma_2]) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!],$$

i.e.,

$$(\mathtt{true}, \mathtt{true}) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!].$$

By Lemma D.4, it suffices to show that $(\mathtt{true}, \mathtt{true}) \in \mathcal{V}_i^{\sim}[\![\mathtt{bool}]\!]$. This is true according to the definition of the logical relation.

□

Lemma D.21 (Congruence for Variables).

Proof. We need to show that $\Gamma^{\sqsubseteq}, x_1 \sqsubseteq x_2 : c, \Gamma'^{\sqsubseteq} \vDash_{d_\sigma} x_1 \sqsubseteq x_2 \in c$.
Let $\sim \in \{<, >\}$, and let $\widehat{\Gamma}^{\sqsubseteq} = \Gamma^{\sqsubseteq}, x_1 \sqsubseteq x_2 : c, \Gamma'^{\sqsubseteq}$. Let $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\widehat{\Gamma}^{\sqsubseteq}]\!]$. We need to show

$$(x_1[\gamma_1], x_2[\gamma_2]) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

By Lemma D.4, it suffices to show that $(\gamma_1(x_1), \gamma_2(x_2)) \in \mathcal{V}_i^{\sim}[\![c]\!]$. But this follows from the fact that $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\widehat{\Gamma}^{\sqsubseteq}]\!]$. In particular, by the definition of the logical relation, since $(x_1 \sqsubseteq x_2 : c) \in \widehat{\Gamma}^{\sqsubseteq}$, we have $(\gamma_1(x_1), \gamma_2(x_2)) \in \mathcal{V}_i^{\sim}[\![c]\!]$. □

Lemma D.22 (Congruence for Lambdas).

Proof. Suppose $\Gamma^{\sqsubseteq}, x \sqsubseteq y : c \vDash_{d_{\sigma'}} M \sqsubseteq N \in d$. We need to show that $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} \lambda x.M \sqsubseteq \lambda y.N \in c \rightarrow_{d_{\sigma'}} d$.
Let $\sim \in \{<, >\}$ and let $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show

$$((\lambda x.M)[\gamma_1], (\lambda y.N)[\gamma_2]) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c \rightarrow_{d_{\sigma'}} d]\!].$$

Let $V_1 = \lambda x.M[\gamma_1]$ and $V_2 = \lambda y.N[\gamma_2]$. By Lemma D.4, it will suffice to show that $(V_1, V_2) \in \mathcal{V}_i^{\sim}[\![c \rightarrow_{d_{\sigma'}} d]\!]$. To this end, let $k \leq i$ and let $(V_{i1}, V_{i2}) \in \mathcal{V}_k^{\sim}[\![c]\!]$. We will show that $(V_1\, V_{i1}, V_2\, V_{i2}) \in \mathcal{E}_k^{\sim}[\![d_{\sigma'}]\!]\mathcal{V}^{\sim}[\![d]\!]$.

Let $M' = (M[\gamma_1])(V_{i1}/x)$ and let $N' = (N[\gamma_2])(V_{i2}/y)$. Note that $(V_1\, V_{i1}) \mapsto^1 M'$, and similarly $(V_2\, V_{i2}) \mapsto^1 N'$. Thus, if $k = 0$, then by the Time-out Lemma (Lemma D.10), we conclude that $(V_1\, V_{i1}, V_2\, V_{i2}) \in \mathcal{E}_k^{\sim}[\![d_{\sigma'}]\!]\mathcal{V}^{\sim}[\![d]\!]$.

Hence, from now on, we assume $k \geq 1$. By the Anti-reduction lemma (Lemma D.6) (with $i_1 = i_2 = 1$ and $j = k$), it will suffice to show that $(M', N') \in \mathcal{E}_{k-1}^{\sim}[\![d_{\sigma'}]\!]\mathcal{V}^{\sim}[\![d]\!]$.

This will follow by our inductive hypothesis, which says that for any $\sim \in \{<, >\}$, any natural number n, and any $(\gamma_1', \gamma_2') \in \mathcal{G}_n^{\sim}[\![\Gamma^{\sqsubseteq}, x \sqsubseteq y : c]\!]$, we have

$$(M[\gamma_1'], N[\gamma_2']) \in \mathcal{E}_n^{\sim}[\![d_{\sigma'}]\!]\mathcal{V}^{\sim}[\![d]\!].$$

Let $\gamma_1' = \gamma_1, V_{i1}/x$, let $\gamma_2' = \gamma_2, V_{i2}/y$. It is easily verified that $(\gamma_1', \gamma_2') \in \mathcal{G}_{k-1}^{\sim}[\![\Gamma^{\sqsubseteq}, x \sqsubseteq y : c]\!]$. (Doing so requires the monotonicity lemma, combined with the fact that $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$ and that $k - 1 < k \leq i$). Taking $n = k - 1$ above, and noting that $M' = M[\gamma_1']$ and $N' = N[\gamma_2']$, it follows that $(M', N') \in \mathcal{E}_{k-1}^{\sim}[\![d_{\sigma'}]\!]\mathcal{V}^{\sim}[\![d]\!]$, as we wanted to show.

□

Lemma D.23 (Congruence for Function Application).

Proof. Suppose $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} M_1 \sqsubseteq M_2 \in c \rightarrow_{d_\sigma} d$, and that $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} N_1 \sqsubseteq N_2 \in c$.

We need to show that $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} M_1\, N_1 \sqsubseteq M_2\, N_2 \in d$.

Let $\sim \in \{<, >\}$ and let $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show

$$(M_1\, N_1[\gamma_1], M_2\, N_2[\gamma_2]) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

By Lemma D.16, it will suffice to show that

(1) $(M_1[\gamma_1], M_2[\gamma_2]) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c \rightarrow_{d_\sigma} d]\!]$, and that (2) for all $k \leq i$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![c \rightarrow_{d_\sigma} d]\!]$, we have $(V_1\, N_1, V_1\, N_2) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!]$.

(1) follows immediately from our first top-level assumption.

To show (2), we again apply Lemma D.16. It follows from our second top-level assumption that $(N_1[\gamma_1], N_2[\gamma_2]) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. Now let $k' \leq k$ and $(V_1', V_2') \in \mathcal{V}_{k'}^{\sim}[\![c]\!]$. We claim that

$$(V_1\, V_1', V_2\, V_2') \in \mathcal{E}_{k'}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

This holds since $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![c \rightarrow_{d_\sigma} d]\!]$ and $(V_1', V_2') \in \mathcal{V}_{k'}^{\sim}[\![c]\!]$.

□

Lemma D.24 (Congruence for If).

Proof. Suppose:

(1) $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} [\![M]\!] \sqsubseteq [\![M']\!] \in \texttt{bool}$
(2) $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} [\![N_t]\!] \sqsubseteq [\![N_t']\!] \in c$
(3) $\Gamma^{\sqsubseteq} \vDash_{d_\sigma} [\![N_f]\!] \sqsubseteq [\![N_f']\!] \in c$

Let $\sim \in \{<, >\}$ and let $(\gamma_1, \gamma_2) \in \mathcal{G}_i^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show

$$\left(\texttt{if } M\{N_t\}\{N_f\}[\gamma_1], \texttt{if } M'\{N_t'\}\{N_f'\}[\gamma_2]\right) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

By Lemma D.16, it will suffice to show that (1) $([\![M]\!][\gamma_1], [\![M']\!][\gamma_2]) \in \mathcal{E}_i^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![\texttt{bool}]\!]$, and (2) for all $k \leq i$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![\texttt{bool}]\!]$, we have

$$(\texttt{if } V_1\{N_t[\gamma_1]\}\{N_f[\gamma_1]\}), (\texttt{if } V_2\{N'_t[\gamma_2]\}\{N'_f[\gamma_2]\}) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

We note that (1) follows by our first top-level assumption. For (2), the assumption $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![\texttt{bool}]\!]$ has two cases. If $V_1 = V_2 = \texttt{true}$, then by anti-reduction (Lemma D.6), it will suffice to show $(N_t[\gamma_1], N'_t[\gamma_2]) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. But this follows from our second top-level assumption. Similarly, if $V_1 = V_2 = \texttt{false}$, then it suffices to show that $(N_f[\gamma_1], N'_f[\gamma_2]) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$, which follows from our third top-level assumption.

□

Lemma D.25 (Congruence for Let).

Proof. This proof is similar to the function abstraction proof and is hence omitted. □

Lemma D.26 (Congruence for Raise).

Proof. Let $c : A_1 \sqsubseteq A_2$ and $d : B_1 \sqsubseteq B_2$. Suppose $\varepsilon : c \rightsquigarrow d \in d_\sigma$ and

$$\Gamma^{\sqsubseteq} \vDash_{d_\sigma} M_1 \sqsubseteq M_2 \in c.$$

We need to show that

$$\Gamma^{\sqsubseteq} \vDash_{d_\sigma} \texttt{raise } \varepsilon(M_1) \sqsubseteq \texttt{raise } \varepsilon(M_2) \in d.$$

Let $\sim \in \{<, >\}$ and $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![\Gamma]\!]$. We will show

$$(\texttt{raise } \varepsilon(M_1)[\gamma_1], \texttt{raise } \varepsilon(M_2)[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

We apply Lemma D.16. We first claim that $(M_1[\gamma_1], M_2[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. This follows by assumption. Now, let $k \le j$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![c]\!]$. We claim that

$$(\texttt{raise } \varepsilon(V_1)[\gamma_1], \texttt{raise } \varepsilon(V_2)[\gamma_2]) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

By Lemma D.3, it suffices to show that

$$(\texttt{raise } \varepsilon(V_1)[\gamma_1], \texttt{raise } \varepsilon(V_2)[\gamma_2]) \in \mathcal{R}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

We assert the second disjunct in the definition of $\mathcal{R}^{\sim}[\![\cdot]\!]$, where we take ε to be ε (which we know by assumption is in d_σ), and we take $E^l = E^r = \bullet$ and $V^l = V_1$, $V^r = V_2$.

We need to show that $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c]\!])_k$, and that

$$(x^l.(\bullet[x^l]), x^r.(\bullet[x^r])) \in (\blacktriangleright\mathcal{K}^{\sim}[\![d]\!])_k(\mathcal{E}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!])$$

To this end, let $k' \le k$ and let $(V^l, V^r) \in \mathcal{V}_{k'}^{\sim}[\![c]\!]$. We need to show

$$(V^l, V^r) \in \mathcal{E}_{k'}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

But this follows by Lemma D.4.

□

Lemma D.27 (Congruence for Handle).

$$\frac{\begin{array}{c} M \sqsubseteq M' : d_\sigma \,!\, c \qquad y : c \vdash N \sqsubseteq N' : d_\tau \,!\, d \\ \forall \varepsilon : d_i \rightsquigarrow d_o \in d_\sigma.(\varepsilon \notin dom(\phi) \wedge \varepsilon \notin dom(\phi') \wedge \varepsilon : d_i \rightsquigarrow d_o \in d_\tau) \vee \\ x : d_i, k : d_o \to_{d_\tau} d \vdash \phi(\varepsilon) \sqsubseteq \phi'(\varepsilon) : d_\tau \,!\, d \end{array}}{\texttt{handle } M\,\{\texttt{ret } y.N \mid \phi\} \sqsubseteq \texttt{handle } M'\,\{\texttt{ret } y.N' \mid \phi'\} : d_\tau \,!\, d}$$

Proof. We use Löb induction (Lemma D.14). Assume that for all $k \leq j$ and all $(\gamma_1, \gamma_2) \in (\blacktriangleright \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!])_k$ and all $(M, M') \in (\blacktriangleright \mathcal{E}^{\sim}[\![d_\sigma]\!])_k(\mathcal{V}^{\sim}[\![c]\!])$, we have

$$\begin{aligned}&(\texttt{handle}\ M\,\{\texttt{ret}\ x.N \mid \phi\}[\gamma_1],\\&\ \texttt{handle}\ M'\,\{\texttt{ret}\ x'.N' \mid \phi'\}[\gamma_2])\\&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}_j[\![d_\tau]\!]_k(\mathcal{V}^{\sim}[\![d]\!]).\end{aligned}$$

Let $(M, M') \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$.
Let $\sim \in \{<, >\}$ and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma^{\sqsubseteq}]\!]$. We need to show that

$$\begin{aligned}&(\texttt{handle}\ M\,\{\texttt{ret}\ x.N \mid \phi\}[\gamma_1],\\&\ \texttt{handle}\ M'\,\{\texttt{ret}\ x'.N' \mid \phi'\}[\gamma_2])\\&\qquad \in \mathcal{E}^{\sim}_j[\![d_\tau]\!]\mathcal{V}^{\sim}[\![d]\!].\end{aligned}$$

By monadic bind (Lemma D.16), it suffices to consider the following cases:

- Let $k \leq j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![c]\!]$. We need to show that

  $$\begin{aligned}&(\texttt{handle}\ V_1\,\{\texttt{ret}\ x.N[\gamma_1] \mid \phi[\gamma_1]\},\\&\ \texttt{handle}\ V_2\,\{\texttt{ret}\ x'.N'[\gamma_2] \mid \phi'[\gamma_2]\})\\&\qquad \in \mathcal{E}^{\sim}_j[\![d_\tau]\!]\mathcal{V}^{\sim}[\![d]\!].\end{aligned}$$

  By anti-reduction (Lemma D.6), it suffices to show that

  $$(N[\gamma_1][V_1/x], N'[\gamma_2][V_2/x']) \in \mathcal{E}^{\sim}_k[\![d_\tau]\!]\mathcal{V}^{\sim}[\![d]\!].$$

  This follows from the premise: if we let $\gamma_1' = \gamma_1, V_1/x$ and $\gamma_2' = \gamma_2, V_2/x'$, then it is easily checked that $(\gamma_1', \gamma_2') \in \mathcal{G}^{\sim}_j[\![\Gamma^{\sqsubseteq}, x \sqsubseteq x' : c]\!]$. Furthermore, $N[\gamma_1][V_1/x] = N[\gamma_1']$ and likewise for $N[\gamma_2][V_2/x']$. The premise then implies that $(N[\gamma_1][V_1/x], N'[\gamma_2][V_2/x']) \in \mathcal{E}^{\sim}_k[\![d_\tau]\!]\mathcal{V}^{\sim}[\![d]\!]$, as needed.
- Let $k \leq j$ and let $\varepsilon : d_i \rightsquigarrow d_o \in d_\sigma$ be an effect that is caught by either handler – i.e., $\varepsilon \in \mathrm{dom}(\phi)$ or $\varepsilon \in \mathrm{dom}(\phi')$. By the premise, it follows that ε is in both $\mathrm{dom}(\phi)$ and $\mathrm{dom}(\phi')$. Let $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![c_i]\!])_k$. Let $E^l \# \varepsilon$ and $E^r \# \varepsilon$ be evaluation contexts such that

  $$(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright \mathcal{K}^{\sim}[\![d_o]\!])_k(\mathcal{E}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]).$$

  We need to show that

  $$\begin{aligned}&(\texttt{handle}\ E^l[\texttt{raise}\ \varepsilon(V^l)\,\{\texttt{ret}\ x.N[\gamma_1] \mid \phi[\gamma_1]\},\\&\ \texttt{handle}\ E^r[\texttt{raise}\ \varepsilon(V^r)\,\{\texttt{ret}\ x'.N'[\gamma_2] \mid \phi'[\gamma_2]\})\\&\qquad \in \mathcal{E}^{\sim}_k[\![d_\tau]\!]\mathcal{V}^{\sim}[\![d]\!].\end{aligned}$$

  By anti-reduction, it suffices to show that

  $$\begin{aligned}&(\phi(\varepsilon)[\gamma_1][V^l/x][(\lambda y.\texttt{handle}\ E^l[y]\,\{\texttt{ret}\ x.N[\gamma_1] \mid \phi[\gamma_1]\})/k],\\&\ \phi'(\varepsilon)[\gamma_2][V^r/x'][(\lambda y.\texttt{handle}\ E^r[y]\,\{\texttt{ret}\ x'.N'[\gamma_2] \mid \phi'[\gamma_2]\})/k'])\\&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![d_\tau]\!])_k(\mathcal{V}^{\sim}[\![d]\!]).\end{aligned}$$

To show this, we apply the premise, as follows. Let $H_1 = \texttt{handle}\ E^l[y]\ \{\texttt{ret}\ x.N[\gamma_1] \mid \phi[\gamma_1]\}$ and $H_2 = \texttt{handle}\ E^r[y]\ \{\texttt{ret}\ x'.N'[\gamma_2] \mid \phi'[\gamma_2]\}$. Let $\gamma_1' = \gamma_1, V^l/x_i, (\lambda y.H_1)/k_i$ and let $\gamma_2' = \gamma_2, V^r/x_i', (\lambda y.H_2)/k_i'$. In order to apply the premise, we must prove that $(\gamma_1', \gamma_2') \in \mathcal{G}^{\sim}_{k'}[\![\Gamma^{\sqsubseteq}, x_i \sqsubseteq x_i' : d_i, k_i \sqsubseteq k_i' : d_o \to_{d_\tau} d]\!]$.
We first need to show that $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_i]\!])_k$. This holds by assumption. We now need to show that

$$((\lambda y.H_1), (\lambda y.H_2)) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_o \to_{d_\tau} d]\!])_k.$$

To this end, let $k' \leq k$ and let $(V_A, V_B) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_o]\!])_{k'}$. We need to show that

$$\begin{aligned}&((\lambda y.H_1)\ V_A, (\lambda y.H_2)\ V_B)\\ &\qquad \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\tau]\!])_{k'}(\mathcal{V}^{\sim}[\![d]\!])\end{aligned}$$

By anti-reduction, it suffices to show that

$$\begin{aligned}&(\texttt{handle}\ E^l[V_A]\ \{\texttt{ret}\ x.N \mid \phi\}, \texttt{handle}\ E^r[V_B]\ \{\texttt{ret}\ x'.N' \mid \phi'\})\\ &\qquad \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\tau]\!])_{k'}(\mathcal{V}^{\sim}[\![d]\!]).\end{aligned}$$

By the Löb induction hypothesis, it will suffice to show that

$$(E^l[V_A], E^r[V_B]) \qquad \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]).$$

Recall that by assumption, we have

$$(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}^{\sim}[\![d_o]\!])_k(\mathcal{E}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]).$$

Thus, it suffices to show that $(V_A, V_B) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_o]\!])_{k'}$, which is precisely our assumption.

□

Note that we do not need to show soundness of the term precision congruence rules involving casts. This will follow from the soundness of the upper and lower bound rules for casts.

Corollary D.28 (reflexivity). *Let M be a term such that $\Sigma \mid \Gamma \mid \Delta \vdash_\sigma M : A$. We have $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma M \sqsubseteq M : A$.*

Proof. By induction on M, using the soundness of the term precision relation already proven. □

D.0.3 Equational Rules.

Lemma D.29 (Value substitution).

$$\frac{x_1 \sqsubseteq x_2 : c \vDash_{d_\sigma} M \equiv N : d \qquad V \equiv V' : c}{M[V/x_1] \equiv N[V'/x_2]}$$

Proof. Suppose for all j and all $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma^{\sqsubseteq}, x^l \sqsubseteq x^r : c]\!]$, that

$$(x_1.M, x_2.N) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!]$$

and

$$(x_2.N, x_1.M) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

Further suppose that for all j,

$$(V, V') \in \mathcal{V}_j^{\sim}[\![c]\!]$$

and

$$(V', V) \in \mathcal{V}_j^{\sim}[\![c]\!].$$

Let j be arbitrary, and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show

$$(M[V/x_1], N[V'/x_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!]$$

and

$$(N[V'/x_2], M[V/x_1]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![d]\!].$$

The second statement is symmetric to the first, so we show only the first.

Let $\gamma_1' = (\gamma_1, x_1 = V)$ and let $\gamma_2' = (\gamma_2, x_2 = V')$.

Note that we have $M[\gamma_1'] = M[\gamma_1][V/x_1]$ and $N[\gamma_2'] = N[\gamma_2][V'/x_2]$, by definition of substitution.

By our assumption, it is sufficient to show that $(\gamma_1', \gamma_2') \in \mathcal{G}_j^{\sim}[\![\Gamma^{\sqsubseteq}, x_1 \sqsubseteq x_2 : c]\!]$.

For this, it suffices to show that $(\gamma_1'(x_1), \gamma_2'(x_2)) \in \mathcal{V}_j^{\sim}[\![c]\!]$. But $\gamma_1'(x_1) = V$ and $\gamma_2'(x_2)V'$, so we are finished.

□

Lemma D.30 (Monad Unit Left).

$$\texttt{let}\ x = y\ \texttt{in}\ N \equiv N[y/x]$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![\Gamma]\!]$. We need to show

$$(\texttt{let}\ x = y\ \texttt{in}\ N, N[y/x]) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

Since y is a variable and hence a value, we have by the operational semantics that

$$\texttt{let}\ x = y\ \texttt{in}\ N \mapsto^1 N[y/x].$$

Thus, by anti-reduction, it suffices to show that

$$(N[y/x], N[y/x]) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

But this follows by reflexivity (Corollary D.28).

□

Lemma D.31 (Monad Unit Right).

$$\texttt{let}\ x = M\ \texttt{in}\ x \equiv M$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![\Gamma]\!]$. We need to show

$$(\texttt{let}\ x = M\ \texttt{in}\ x, M) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

Since x is a variable and hence a value, we have by the operational semantics that

$$\texttt{let}\ x = M\ \texttt{in}\ x \mapsto^1 M[x/x].$$

By definition of substitution, $M[x/x] = M$. Thus, by anti-reduction, it suffices to show that

$$(M, M) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows by reflexivity (Corollary D.28).

□

Lemma D.32 (Monad Associativity).

$$\texttt{let}\ y = (\texttt{let}\ x = M\ \texttt{in}\ N)\ \texttt{in}\ P \equiv \texttt{let}\ x = M\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ P$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_{j}[\![\Gamma]\!]$. We need to show

$$(\texttt{let}\ y = (\texttt{let}\ x = M\ \texttt{in}\ N)\ \texttt{in}\ P, \texttt{let}\ x = M\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ P) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

We apply Lemma D.16, taking $E_1 = \texttt{let}\ y = (\texttt{let}\ x = \bullet\ \texttt{in}\ N)\ \texttt{in}\ P$ and $E_2 = \texttt{let}\ x = \bullet\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ P$.

We first need to show that $(M, M) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$, which is true by reflexivity (Corollary D.28).

Now, let $k \leq j$ and $(V_1, V_2) \in \mathcal{V}^{\sim}_{k}[\![A]\!]$. We need to show that

$$(\texttt{let}\ y = (\texttt{let}\ x = V_1\ \texttt{in}\ N)\ \texttt{in}\ P, \texttt{let}\ x = V_2\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ P) \in \mathcal{E}^{\sim}_{k}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

According to the operational semantics, we have

$$(\texttt{let}\ x = V_1\ \texttt{in}\ N) \mapsto^1 N[V_1/x].$$

Thus,

$$\texttt{let}\ y = (\texttt{let}\ x = V_1\ \texttt{in}\ N)\ \texttt{in}\ P \mapsto^1 \texttt{let}\ y = N[V_1/x]\ \texttt{in}\ P.$$

Similarly, we have

$$\texttt{let}\ x = V_2\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ P \mapsto^1 (\texttt{let}\ y = N\ \texttt{in}\ P)[V_2/x] = \texttt{let}\ y = N[V_2/x]\ \texttt{in}\ P[V_2/x].$$

Note that since x does not occur in P, we have $P[V_2/x] = P$.

Now, by anti-reduction, it suffices to show

$$(\texttt{let}\ y = N[V_1/x]\ \texttt{in}\ P, \texttt{let}\ y = N[V_2/x]\ \texttt{in}\ P) \in \mathcal{E}^{\sim}_{k}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

We again apply Lemma D.16, this time with $E_1 = \texttt{let}\ y = \bullet\ \texttt{in}\ P$ and $E_2 = \texttt{let}\ y = \bullet\ \texttt{in}\ P$.

We first need to show that $(N[V_1/x], N[V_2/x]) \in \mathcal{E}^{\sim}_{k}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. This follows from reflexivity (Corollary D.28) and value substitution (Lemma D.29) applied to our assumption on V_1 and V_2.

Now let $k' \leq k$ and $(V_1', V_2') \in \mathcal{V}^{\sim}_{A}[\![k']\!]$. We need to show that

$$(\texttt{let}\ y = V_1'\ \texttt{in}\ P, \texttt{let}\ y = V_2'\ \texttt{in}\ P) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By anti-reduction, it suffices to show

$$(P[V_1'/y], P[V_2'/y]) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This again follows from reflexivity and value substitution.

□

Lemma D.33 (η-expansion for Booleans).

$$M[x : \texttt{bool}] \equiv \texttt{if}\ x\{M[\texttt{true}/x]\}\{M[\texttt{false}/x]\}$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_{j}[\![\Gamma, x_1 \sqsubseteq x_2 : \mathtt{bool}]\!]$. We need to show

$$(M[\gamma_1], (\mathtt{if}\ x\{M[\mathtt{true}/x]\}\{M[\mathtt{false}/x]\})[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By definition of substitution, this is equivalent to

$$(M[\gamma_1], (\mathtt{if}\ \gamma\{_\}\{2\}(x)M[\mathtt{true}/x][\gamma_2]M[\mathtt{false}/x][\gamma_2])) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By our assumption on γ_1 and γ_2, we have that either $\gamma_1(x_1) = \gamma_2(x_2) = \mathtt{true}$ or $\gamma_1(x_1) = \gamma_2(x_2) = \mathtt{false}$.

We show only the former case; the latter is symmetric. In the former case, we need to show

$$(M[\mathtt{true}/x][\gamma_1], (\mathtt{if}\ \mathtt{true}\{M[\mathtt{true}/x][\gamma_2]\}\{M[\mathtt{false}/x][\gamma_2]\})) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By anti-reduction, it is sufficient to show

$$(M[\mathtt{true}/x][\gamma_1], M[\mathtt{true}/x][\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows by reflexivity.

□

Lemma D.34 (Boolean β reduction - true).

$$\mathtt{if}\ \mathtt{true}\{N_t\}\{N_f\} \equiv N_t$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_{j}[\![\Gamma]\!]$. We need to show

$$((\mathtt{if}\ \mathtt{true}\{N_t\}\{N_f\})[\gamma_1], N_t[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By anti-reduction, it suffices to show

$$(N_t[\gamma_1], N_t[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This holds by reflexivity.

□

Lemma D.35 (Boolean β reduction - false).

$$\mathtt{if}\ \mathtt{false}\{N_t\}\{N_f\} \equiv N_f$$

Proof. Precisely dual to the above proof. □

Lemma D.36 (Eval for If).

$$\mathtt{if}\ M\{N_t\}\{N_f\} \equiv \mathtt{let}\ x = M\ \mathtt{in}\ \mathtt{if}\ x\{N_t\}\{N_f\}\ \textit{IfEval}$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_{j}[\![\Gamma]\!]$. We need to show

$$((\mathtt{if}\ M\{N_t\}\{N_f\})[\gamma_1], (\mathtt{let}\ x = M\ \mathtt{in}\ \mathtt{if}\ x\{N_t\}\{N_f\})[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

We apply Lemma D.16, with $E_1 = \mathtt{if}\ \bullet\{N_t[\gamma_1]\}\{N_f[\gamma_1]\}$ and $E_2 = \mathtt{let}\ x = \bullet\ \mathtt{in}\ \mathtt{if}\ \gamma_2(x)\{N_t[\gamma_2]\}\{N_f[\gamma_2]\}$.

We first need to show that $(M[\gamma_1], M[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\tau]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!]$. This follows by reflexivity (Corollary D.28).

Now let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_{k}[\![\mathtt{bool}]\!]$. We need to show that

$$((\texttt{if } V_1\{N_t[\gamma_1]\}\{N_f[\gamma_1]\}), (\texttt{let } x = V_2 \texttt{ in if } \gamma_2(x)\{N_t[\gamma_2]\}\{N_f[\gamma_2]\})) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By definition of $\mathcal{V}^{\sim}[\![\texttt{bool}]\!]$, either $V_1 = V_2 = \texttt{true}$ or $V_1 = V_2 = \texttt{false}$. We consider the first case; the second is symmetric.

We need to show

$$((\texttt{if true}\{N_t[\gamma_1]\}\{N_f[\gamma_1]\}), (\texttt{let } x = \texttt{true in if } \gamma_2(x)\{N_t[\gamma_2]\}\{N_f[\gamma_2]\})) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By anti-reduction, it suffices to show

$$(N_t[\gamma_1], N_t[\gamma_2]) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows by reflexivity. □

Lemma D.37 (β-reduction for functions).

$$(\lambda x.M)V \equiv M[V/x] \textit{ FunBeta}$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma]\!]$. We need to show

$$(((\lambda x.M)\ V)[\gamma_1], (M[V/x])[\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

Since V is a value, it suffices by anti-reduction to show that

$$(M[V/x][\gamma_1], M[V/x][\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows by reflexivity. □

Lemma D.38 (η-expansion for functions). *Let V_f be a value such that $\Sigma \mid \Gamma \mid \Delta \vdash_{\emptyset} V : A \to_{\sigma'} B$. We have $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma} V_f \equiv (\lambda x.V_f x) : (A \to_{\sigma'} B)$.*

Proof. Let j be arbitrary. We need to show

$$(V_f, (\lambda x.V_f x)) \in \mathcal{E}^{\sim}_j[\![\emptyset]\!]\mathcal{V}^{\sim}[\![A \to_{\sigma'} B]\!].$$

As these are values, it suffices by Lemma D.4 to show that they are related in $\mathcal{V}^{\sim}_j[\![A \to_{\sigma'} B]\!]$. To this end, let $k \le j$ and let $(V_{i1}, V_{i2}) \in \mathcal{V}^{\sim}_k[\![A]\!]$. We claim that

$$(V_f\ V_{i1}, (\lambda x.V_f x)\ V_{i2}) \in \mathcal{E}^{\sim}_k[\![\sigma']\!]\mathcal{V}^{\sim}[\![B]\!].$$

By anti-reduction, it will suffice to show that

$$(V_f\ V_{i1}, V_f\ V_{i2}) \in \mathcal{E}^{\sim}_k[\![\sigma']\!]\mathcal{V}^{\sim}[\![B]\!].$$

By reflexivity (Corollary D.28), we know that $(V_f, V_f) \in \mathcal{E}^{\sim}_k[\![\emptyset]\!]A \to_{\sigma'} B$, and since V_f is a value, this means that $(V_f, V_f) \in \mathcal{V}^{\sim}_k[\![A \to_{\sigma'} B]\!]$. This immediately implies the desired result, since $(V_{i1}, V_{i2}) \in \mathcal{V}^{\sim}_k[\![A]\!]$. □

Lemma D.39 (AppEval).

$$M\,N \equiv \texttt{let } x = M \texttt{ in let } y = N \texttt{ in } x\,y$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma]\!]$. We need to show

$$((M\,N)[\gamma_1], (\texttt{let}\ x = M\ \texttt{in}\ \texttt{let}\ y = N\ \texttt{in}\ x\,y)[\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\tau_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].$$

We apply Lemma D.16, with $E_1 = (\bullet\, N[\gamma_2])$ and $E_2 = \texttt{let}\ x = \bullet\ \texttt{in}\ \texttt{let}\ y = N[\gamma_2]\ \texttt{in}\ x\,y$.

We first need to show that $(M[\gamma_1], M[\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\tau]\!]\mathcal{V}^{\sim}[\![A_i \to_{\tau_A} A_o]\!]$. This follows by reflexivity.

Now let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![A_i \to_{\sigma_A} A_o]\!]$. We need to show that

$$((V_1\,N[\gamma_1]), (\texttt{let}\ x = V_2\ \texttt{in}\ \texttt{let}\ y = N[\gamma_2]\ \texttt{in}\ x\,y)) \in \mathcal{E}^{\sim}_k[\![\tau_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].$$

By anti-reduction, it suffices to show

$$((V_1\,N[\gamma_1]), (\texttt{let}\ y = N[\gamma_2]\ \texttt{in}\ V_2\,y)) \in \mathcal{E}^{\sim}_k[\![\tau_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].$$

We again apply Lemma D.16, this time with $E_1 = (V_1\,\bullet)$ and $E_2 = \texttt{let}\ y = \bullet\ \texttt{in}\ V_2\,y$.

We need to show $(N[\gamma_1], N[\gamma_2]) \in \mathcal{E}^{\sim}_k[\![\tau]\!]\mathcal{V}^{\sim}[\![A_i]\!]$, which holds by reflexivity. Now let $k' \le k$ and let $(V_1', V_2') \in \mathcal{V}^{\sim}_{k'}[\![A_i]\!]$. We need to show that

$$((V_1\,V_1'), (\texttt{let}\ y = V_2'\ \texttt{in}\ V_2\,y)) \in \mathcal{E}^{\sim}_{k'}[\![\tau_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].$$

By anti-reduction, it suffices to show

$$((V_1\,V_1'), (V_2\,V_2')) \in \mathcal{E}^{\sim}_{k'}[\![\tau_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].$$

This follows from our assumptions on V_1 and V_2 and on V_1' and V_2'.

□

Lemma D.40 (HandleBetaRet).

$$\texttt{handle}\ x\,\{\texttt{ret}\ y.M \mid \phi\} \equiv M[x/y]$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma]\!]$. We need to show

$$((\texttt{handle}\ x\,\{\texttt{ret}\ y.M \mid \phi\})[\gamma_1], (M[x/y])[\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

Since x is a value, the above handle term steps, and by anti-reduction it is sufficient to show

$$((M[x/y][\gamma_1]), (M[x/y])[\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows by reflexivity.

□

Lemma D.41 (HandleBetaRaise).

$$\texttt{handle}\ (\texttt{let}\ o = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ N_k)\,\{\texttt{ret}\ y.M \mid \phi\} \equiv \phi(\varepsilon)[\lambda o.\texttt{handle}\ N_k\,\{\texttt{ret}\ y.M \mid \phi\}/k]$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma]\!]$. We need to show

$$\begin{aligned}&((\texttt{handle}\ (\texttt{let}\ o = \texttt{raise}\ \varepsilon(x)\ \texttt{in}\ N_k)\,\{\texttt{ret}\ y.M \mid \phi\})[\gamma_1],\\ &\ (\phi(\varepsilon)[\lambda o.\texttt{handle}\ N_k\,\{\texttt{ret}\ y.M \mid \phi\}/k])[\gamma_2])\\ &\qquad \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

Let $E = \texttt{let}\ o = \bullet\ \texttt{in}\ N_k[\gamma_1]$. Our goal is to show

$$\begin{aligned}&((\texttt{handle } E[\texttt{raise } \varepsilon(x)] \{\texttt{ret } y.M[\gamma_1] \mid \phi[\gamma_1]\}),\\ &\ (\phi(\varepsilon)[\gamma_2][\lambda o.\texttt{handle } N_k[\gamma_2] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}/k]))\\ &\qquad \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

Note that $E\#\varepsilon$. By anti-reduction, it suffices to show

$$\begin{aligned}&((\phi(\varepsilon)[\gamma_1][\lambda o'.\texttt{handle } E[o'] \{\texttt{ret } y.M[\gamma_1] \mid \phi[\gamma_1]\}/k]),\\ &\ (\phi(\varepsilon)[\gamma_2][\lambda o.\texttt{handle } N_k[\gamma_2] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}/k]))\\ &\qquad \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

That is, we need to show

$$\begin{aligned}&((\phi(\varepsilon)[\gamma_1][\lambda o'.\texttt{handle let } o = o' \texttt{ in } N_k[\gamma_1] \{\texttt{ret } y.M[\gamma_1] \mid \phi[\gamma_1]\}/k]),\\ &\ (\phi(\varepsilon)[\gamma_2][\lambda o.\texttt{handle } N_k[\gamma_2] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}/k]))\\ &\qquad \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

By ValSubst, it suffices to show (1) for all related $(V_{f1}, V_{f2}) \in \mathcal{V}_j^{\sim}[\![A_i \to_\sigma B]\!]$ and $\gamma_1' = \gamma_1, V_{f1}/k$ and $\gamma_2' = \gamma_2, V_{f2}/k$, we have

$$(\phi(\varepsilon)[\gamma_1'], \phi(\varepsilon)[\gamma_2']) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!],$$

and (2),

$$\begin{aligned}&((\lambda o'.\texttt{handle let } o = o' \texttt{ in } N_k[\gamma_1] \{\texttt{ret } y.M[\gamma_1] \mid \phi[\gamma_1]\}),\\ &\ (\lambda o.\texttt{handle } N_k[\gamma_2] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}))\\ &\qquad \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i \to_\sigma B]\!].\end{aligned}$$

(1) follows from reflexivity. To show (2), we will use transitivity (Lemma D.64). If $\sim$ is $<$, then note that by MonadUnitL we have

$$(\texttt{let } o = o' \texttt{ in } N_k[\gamma_1], N_k[\gamma_2][o'/o]) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!],$$

and by soundness of the congruence rules we have

$$\begin{aligned}&((\lambda o'.\texttt{handle let } o = o' \texttt{ in } N_k[\gamma_1] \{\texttt{ret } y.M[\gamma_1] \mid \phi[\gamma_1]\}),\\ &\ (\lambda o'.\texttt{handle } N_k[\gamma_2][o'/o] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}))\\ &\qquad \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i \to_\sigma B]\!].\end{aligned}$$

Then by transitivity, it will suffice to show that

$$\begin{aligned}&((\lambda o'.\texttt{handle } N_k[\gamma_2][o'/o] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}),\\ &\ (\lambda o.\texttt{handle } N_k[\gamma_2] \{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}))\\ &\qquad \in \mathcal{E}_\omega^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i \to_\sigma B]\!].\end{aligned}$$

By congruence for lambdas, it suffices to show that, given related values $(V_1, V_2) \in \mathcal{V}_\omega^{\sim}[\![A_i]\!]$, we have

$$\begin{aligned}&((\texttt{handle } N_k[\gamma_2][o'/o][V_1/o']\,\{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}),\\ &\ (\texttt{handle } N_k[\gamma_2][V_2/o]\,\{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}))\\ &\qquad \in \mathcal{E}^{\sim}_{\omega}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i \to_\sigma B]\!].\end{aligned}$$

This follows from the soundness of the congruence rules.

On the other hand, if $\sim$ is $>$, then similarly by MonadUnitL we have

$$(\texttt{let } o = o' \texttt{ in } N_k[\gamma_1], N_k[\gamma_1][o'/o]) \in \mathcal{E}^{\sim}_{\omega}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

It then suffices to show that

$$\begin{aligned}&((\lambda o'.\texttt{handle } N_k[\gamma_1][o'/o]\,\{\texttt{ret } y.M[\gamma_1] \mid \phi[\gamma_1]\}),\\ &\ (\lambda o.\texttt{handle } N_k[\gamma_2]\,\{\texttt{ret } y.M[\gamma_2] \mid \phi[\gamma_2]\}))\\ &\qquad \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i \to_\sigma B]\!],\end{aligned}$$

which again follows from the soundness of the congruence rules.

□

Lemma D.42 (RaiseEval).

$$\texttt{raise } \varepsilon(M) \equiv \texttt{let } x = M \texttt{ in raise } \varepsilon(x)$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_{j}[\![\Gamma]\!]$. We need to show

$$((\texttt{raise } \varepsilon(M))[\gamma_1], (\texttt{let } x = M \texttt{ in raise } \varepsilon(x))[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

We apply Monadic Bind (Lemma D.16), with $E_1 = \texttt{raise } \varepsilon(\bullet)$ and $E_2 = \texttt{let } x = \bullet \texttt{ in raise } \varepsilon(x)$.

We first need to show that $(M[\gamma_1], M[\gamma_2]) \in \mathcal{E}^{\sim}_{j}[\![\tau]\!]\mathcal{V}^{\sim}[\![A]\!]$. This follows from reflexivity (Corollary D.28).

Now let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_{k}[\![A]\!]$. We need to show that

$$((\texttt{raise } \varepsilon(V_1))[\gamma_1], (\texttt{let } x = V_2 \texttt{ in raise } \varepsilon(x))[\gamma_2]) \in \mathcal{E}^{\sim}_{k}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

As V_2 is a value, the above let term steps. By anti-reduction, it suffices to show

$$((\texttt{raise } \varepsilon(V_1)), (\texttt{raise } \varepsilon(V_2))) \in \mathcal{E}^{\sim}_{k}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows from our assumption on V_1 and V_2 and the soundness of the term congruence rule for raise (Lemma D.26).

□

Lemma D.43 (HandleEmpty).

$$\texttt{handle } M\,\{\texttt{ret } x.N \mid \emptyset\} \equiv \texttt{let } x = M \texttt{ in } N$$

Proof. We show one direction of the equivalence; the other is symmetric.

Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_{j}[\![\Gamma]\!]$. We need to show

$$\begin{aligned}&((\texttt{handle } M\,\{\texttt{ret } x.N \mid \emptyset\})[\gamma_1],\\ &\ (\texttt{let } x = M \texttt{ in } N)[\gamma_2])\\ &\qquad \in \mathcal{E}^{\sim}_{j}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

By Monadic Bind (Lemma D.16) and the fact that neither evaluation context catches any effects, it suffices to show that

$$\begin{aligned}&(\texttt{handle}\ V_1\ \{\texttt{ret}\ x.N[\gamma_1] \mid \emptyset\},\\ &\ \texttt{let}\ x = V_2\ \texttt{in}\ N[\gamma_2])\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!],\end{aligned}$$

where $k \leq j$ and $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![B]\!]$. By anti-reduction, it will suffice to show that

$$(N[\gamma_1][V_1/x], N[\gamma_2][V_2/x]) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

Using ValSubst, the result follows by reflexivity and our assumption on V_1 and V_2.

□

Lemma D.44.

$$\frac{\forall \varepsilon \in dom(\phi).\ \psi(\varepsilon) = \phi(\varepsilon) \qquad \forall \varepsilon \in dom(\psi). \varepsilon \notin dom(\phi) \Rightarrow \psi(\varepsilon) = k(\texttt{raise}\ \varepsilon(x))}{\texttt{handle}\ M\ \{\texttt{ret}\ y.N \mid \phi\} \equiv \texttt{handle}\ M\ \{\texttt{ret}\ y.N \mid \psi\} : \sigma\,!\,B}\ \textit{HandleExt}$$

Proof. We show one direction of the equivalence; the other is symmetric.
The proof is by Löb induction. We assume that

$$((\texttt{handle}\ M\ \{\texttt{ret}\ _.1 \mid {}'\}yN\phi)[\gamma_1], (\texttt{handle}\ M\ \{\texttt{ret}\ _.2 \mid {}'\}yN\psi)[\gamma_2]) \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_j(\mathcal{V}^{\sim}[\![B]\!]).$$

for all $k \leq j$, $(\gamma_1, \gamma_2) \in (\blacktriangleright\mathcal{G}^{\sim}[\![\Gamma]\!])_k$ and $(M_1', M_2') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!])$.
Let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma]\!]$. We need to show

$$((\texttt{handle}\ M\ \{\texttt{ret}\ _.1 \mid y\}N\phi)[\gamma_1], (\texttt{handle}\ M\ \{\texttt{ret}\ _.2 \mid y\}N\psi)[\gamma_2]) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!]$$

for all $(M_1, M_2) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$.
We apply Monadic Bind (Lemma D.16). It suffices to consider the following cases:

- Let $k \leq j$ and $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![A]\!]$. We need to show that

$$((\texttt{handle}\ V_1\ \{\texttt{ret}\ y.N[\gamma_1] \mid \phi[\gamma_1]\}), (\texttt{handle}\ V_2\ \{\texttt{ret}\ y.N[\gamma_2] \mid \psi[\gamma_2]\})) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

  This follows by anti-reduction and reflexivity.
- Let $k \leq j$ and let $\varepsilon \in \sigma$ be an effect caught by either handler, i.e., ε is in dom(ϕ) or dom(ψ). Let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon]\!])_k$, and let $E^l \# \varepsilon$ and $E^r \# \varepsilon$ such that $(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}^{\sim}[\![d_\varepsilon]\!])_k(\mathcal{E}^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!])$.
  We need to show

$$\begin{aligned}&((\texttt{handle}\ E^l[\texttt{raise}\ \varepsilon(V^l)]\ \{\texttt{ret}\ y.N[\gamma_1] \mid \phi[\gamma_1]\}),\\ &\ (\texttt{handle}\ E^r[\texttt{raise}\ \varepsilon(V^r)]\ \{\texttt{ret}\ y.N[\gamma_2] \mid \psi[\gamma_2]\}))\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

  If $\varepsilon \in$ dom(ϕ), then by the premise, we have $\psi(\varepsilon) = \phi(\varepsilon)$, so both sides step, and it suffices by anti-reduction to show

$$\begin{aligned}&(\phi(\varepsilon)[\gamma_1][V^l/x][(\lambda z.\mathtt{handle}\ E^l[z]\ \{\mathtt{ret}\ y.N[\gamma_1]\mid\phi[\gamma_1]\})/k],\\&\ \phi(\varepsilon)[\gamma_2][V^r/x][(\lambda z.\mathtt{handle}\ E^r[z]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\})/k])\\&\qquad\in(\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![B]\!]).\end{aligned}$$

By ValSubst, it suffices to show that $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon]\!])_k$, which is true by assumption, and that

$$\begin{aligned}&((\lambda z.\mathtt{handle}\ E^l[z]\ \{\mathtt{ret}\ y.N[\gamma_1]\mid\phi[\gamma_1]\}),\\&\ (\lambda z.\mathtt{handle}\ E^r[z]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\}))\\&\qquad\in(\blacktriangleright\mathcal{V}^{\sim}[\![d_\varepsilon\to_\sigma B]\!])_k.\end{aligned}$$

By congruence for lambdas, it suffices to show that, given values $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_\varepsilon]\!])_k$, we have

$$\begin{aligned}&(\mathtt{handle}\ E^l[V_1]\ \{\mathtt{ret}\ y.N[\gamma_1]\mid\phi[\gamma_1]\},\\&\ \mathtt{handle}\ E^r[V_2]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\})\\&\qquad\in(\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![B]\!]).\end{aligned}$$

This follows by the Löb induction hypothesis and our assumption on E^l and E^r.
Now assume that $\varepsilon \notin \mathrm{dom}(\phi)$. Then note that the first handle term does not step, while the second handle term steps to

$$\psi(\varepsilon)[\gamma_2][V^r/x][(\lambda z.\mathtt{handle}\ E^r[z]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\})/k].$$

By the premise, we have $\psi(\varepsilon) = k(\mathtt{raise}\ \varepsilon(x))$. Thus, by anti-reduction, it suffices to show

$$\begin{aligned}&((\mathtt{handle}\ E^l[\mathtt{raise}\ \varepsilon(V^l)]\ \{\mathtt{ret}\ y.N[\gamma_1]\mid\phi[\gamma_1]\}),\\&\ (k(\mathtt{raise}\ \varepsilon(x))[\gamma_2)[V^r/x][(\lambda z.\mathtt{handle}\ E^r[z]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\})/k])\\&\qquad\in\mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

That is, it will suffice to show

$$\begin{aligned}&((\mathtt{handle}\ E^l[\mathtt{raise}\ \varepsilon(V^l)]\ \{\mathtt{ret}\ y.N[\gamma_1]\mid\phi[\gamma_1]\}),\\&\ ((\lambda z.\mathtt{handle}\ E^r[z]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\})\ (\mathtt{raise}\ \varepsilon(V)^r)))\\&\qquad\in\mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

Neither term steps, so it suffices to show they are related in $\mathcal{R}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!]$.
We need to show that $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon]\!])_k$, which is true by assumption, and that given $k' \leq k$ and related values $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_\varepsilon]\!])_{k'}$, we have

$$\begin{aligned}&((\mathtt{handle}\ E^l[V_1]\ \{\mathtt{ret}\ y.N[\gamma_1]\mid\phi[\gamma_1]\}),\\&\ ((\lambda z.\mathtt{handle}\ E^r[z]\ \{\mathtt{ret}\ y.N[\gamma_2]\mid\psi[\gamma_2]\})\ V_2))\\&\qquad\in(\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![B]\!]).\end{aligned}$$

By anti-reduction, it suffices to show

$$\begin{aligned}&((\texttt{handle}\ E^l[V_1]\ \{\texttt{ret}\ y.N[\gamma_1] \mid \phi[\gamma_1]\}),\\ &\ (\texttt{handle}\ E^r[V_2]\ \{\texttt{ret}\ y.N[\gamma_2] \mid \psi[\gamma_2]\}))\\ &\quad \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![B]\!]).\end{aligned}$$

This follows by the Löb induction hypothesis and our assumption on E^l and E^r.

□

D.0.4 Cast, Error, and Subtyping Properties.

LEMMA D.45 (ERR-BOT).

$$\frac{M : {d_\sigma}^r \,!\, c^r}{\mho \sqsubseteq M : d_\sigma \,!\, c}$$

PROOF. Let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show

$$(\mho[\gamma_1], M[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

This follows from the definition of the logical relation: If $\sim$ is $<$ (counting steps on the left), then we are finished by the definition of the $\mathcal{E}^{\preceq}[\![\,]\!]$ relation, because $\mho \mapsto^0 \mho$.

If $\sim$ is $>$ (counting steps on the right), then we are similarly finished, because $M \mapsto^0 M$ and the left-hand term is $\mho$.

□

LEMMA D.46 (ERR-STRICT). $E[\mho] \equiv \mho$

PROOF. We show one direction of the equivalence; the other is symmetric. Let j, d_σ, and c be arbitrary. We need to show

$$(E[\mho], \mho) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

By anti-reduction, it is sufficient to show

$$(\mho, \mho) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!],$$

which is easily seen to hold by definition of the logical relation.

□

LEMMA D.47 (MONOTONICITY OF SUBTYPING). *If $c \le d$ then $\mathcal{V}[\![c]\!] \subseteq \mathcal{V}[\![d]\!]$*
Further, if $R \subseteq S$ then $\mathcal{K}[\![d]\!]R \subseteq \mathcal{K}[\![c]\!]S$,
Further, if $c_\sigma \le d_\sigma$ then both

- $\mathcal{E}[\![c]\!]R \subseteq \mathcal{E}[\![d]\!]S$
- $\mathcal{R}[\![c]\!]R \subseteq \mathcal{R}[\![d]\!]S$

PROOF. By mutual induction on the subtyping proofs. First the type subtyping cases:

(1) $\texttt{bool} \le \texttt{bool}$: trivial.
(2) $c_i \to_{c_e} c_o \le d_i to_{d_e} d_o$. Assume $(V_f, V_f') \in \mathcal{V}[\![c_i \to_{c_e} c_o]\!]$, we need to show $(V_f, V_f') \in \mathcal{V}[\![d_i \to_{d_e} d_o]\!]$. Let $(V_i, V_i') \in \mathcal{V}[\![d_i]\!]$. Then by inductive hypothesis, $(V_i, V_i') \in \mathcal{V}[\![c_i]\!]$. Therefore $(V_f V_i, V_f' V_i') \in \mathcal{E}[\![c_e]\!]\mathcal{V}[\![c_o]\!]$ and the result follows by the two inductive hypotheses.

The $\mathcal{K}[\![\cdot]\!]$ case follows by a similar argument to the function case.

The $\mathcal{E}[\![\cdot]\!]$ case follows by inductive hypothesis.

Next the $\mathcal{R}[\![\cdot]\!]$ cases:

(1) $? \leq ?$: trivial

(2) $\dfrac{c \leq \Sigma}{c \leq ?}$: trivial by definition of $\mathcal{R}[\![?]\!]$

(3) $\dfrac{c \leq d}{c \leq Inj(d)}$: trivial by definition of $\mathcal{R}[\![Inj(i,d)]\!]$

(4) $\dfrac{c \leq d}{Inj(c) \leq Inj(d)}$: trivial by definition of $\mathcal{R}[\![Inj(i,d)]\!]$

(5) $\dfrac{\begin{array}{c}\mathrm{dom}(d_c) \subseteq \mathrm{dom}(d'_c) \\ \forall \varepsilon : c \rightsquigarrow d \in d_c.\varepsilon : c' \rightsquigarrow d' \in d'_c \wedge c \leq c' \wedge d' \leq d\end{array}}{d_c \leq d'_c}$: Follows using Löb induction by the monotonicity of subtyping for the $\mathcal{V}^{\sim}[\![\cdot]\!]$ and $\mathcal{K}^{\sim}[\![\cdot]\!]$ relations.

□

We next prove generalized versions of the cast properties ValUpL, ValUpR, ValDnL, ValDnR, EffUpL, EffUpR, EffDnL, EffDnR. These are proved simultaneously by induction on the type precision derivation and by Löb-induction.

Lemma D.48 (ValUpR-general).

$$\dfrac{\begin{array}{c} c : A \sqsubseteq A' \\ e : A' \sqsubseteq A'' \\ \Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\sigma} M \sqsubseteq N : c \end{array}}{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\sigma} M \sqsubseteq \langle A'' \leftarrowtail A' \rangle N : c \circ e}$$

Proof. We need to show that

$$(M, \langle A'' \leftarrowtail A' \rangle N) \in \mathcal{E}^{\sim}_{j}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c \circ e]\!].$$

The proof is by induction on the precision derivation e. By monadic bind (Lemma D.16), with $E_1 = \bullet$ and $E_2 = \langle A'' \leftarrowtail A' \rangle \bullet$, it suffices to show

$$(V_1, \langle A'' \leftarrowtail A' \rangle V_2) \in \mathcal{E}^{\sim}_{k}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c \circ e]\!],$$

where $k \leq j$ and $(V_1, V_2) \in \mathcal{V}^{\sim}_{k}[\![c]\!]$. We continue by cases on e.

- Case $e = \mathtt{bool}$. We have $A = A' = A'' = \mathtt{bool}$, and $c = \mathtt{bool}$. Thus $c \circ e = \mathtt{bool}$. Examining the operational semantics, we see that

$$(\langle \mathtt{bool} \leftarrowtail \mathtt{bool} \rangle)(V_1) \mapsto^1 V_1.$$

  Thus, by anti-reduction, it suffices to show

$$(V_1, V_2) \in \mathcal{E}^{\sim}_{k}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!].$$

  This is true by assumption and Lemma D.4.
- Case $e = e_i \rightarrow_{e_\sigma} e_o$. We have $A' = A'_i \rightarrow_{\sigma'_A} A'_o$ and $A'' = A''_i \rightarrow_{\sigma''_A} A''_o$, and also $e_i : A'_i \sqsubseteq A''_i$ and $e_o : A'_o \sqsubseteq A''_o$.
  By inversion, we see that $c = c_i \rightarrow_{c_\sigma} c_o$. Thus, we have that $c \circ e = (c_i \rightarrow_{c_\sigma} c_o) \circ (e_i \rightarrow_{e_\sigma} e_o) = (c_i \circ e_i) \rightarrow_{c_\sigma \circ e_\sigma} (c_o \circ e_o)$.

We need to show that

$$(V_1, \langle (A_i'' \rightarrow_{\sigma_A''} A_o'') \leftarrowtail (A_i' \rightarrow_{\sigma_A'} A_o') \rangle V_2) \in \mathcal{E}_k^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![(c_i \circ e_i) \rightarrow_{c_\sigma \circ e_\sigma} (c_o \circ e_o)]\!].$$

As both terms are values, it suffices by Lemma D.4 to show they are related in $\mathcal{V}_k^{\sim}[\![(c_i \circ e_i) \rightarrow_{c_\sigma \circ e_\sigma} (c_o \circ e_o$ To this end, let $k' \le k$ and $(V^l, V^r) \in \mathcal{V}_{k'}^{\sim}[\![c_i \circ e_i]\!]$. We need to show that

$$(V_1\, V^l, (\langle (A_i'' \rightarrow_{\sigma_A''} A_o'') \leftarrowtail (A_i' \rightarrow_{\sigma_A'} A_o') \rangle V_2)\, V^r) \in \mathcal{E}_{k'}^{\sim}[\![c_\sigma \circ e_\sigma]\!]\mathcal{V}^{\sim}[\![c_o \circ e_o]\!].$$

By anti-reduction, it suffices to show that

$$(V_1\, V^l, \langle A_o'' \leftarrowtail A_o' \rangle \langle \sigma_A'' \leftarrowtail \sigma_A' \rangle (V_2\, \langle A_i' \twoheadleftarrow A_i'' \rangle V^r)) \in \mathcal{E}_{k'}^{\sim}[\![c_\sigma \circ e_\sigma]\!]\mathcal{V}^{\sim}[\![c_o \circ e_o]\!].$$

By the induction hypothesis applied twice, it suffices to show

$$(V_1\, V^l, (V_2\, \langle A_i' \twoheadleftarrow A_i'' \rangle V^r)) \in \mathcal{E}_{k'}^{\sim}[\![c_\sigma]\!]\mathcal{V}^{\sim}[\![c_o]\!].$$

Finally, it suffices by the soundness of the term precision congruence rule for function application (Lemma D.23 to show that $(V_1, V_2) \in \mathcal{V}_{k'}^{\sim}[\![c_i \rightarrow_{c_\sigma} c_o]\!]$, and that

$$(V^l, \langle A_i' \twoheadleftarrow A_i'' \rangle V^r) \in \mathcal{E}_{k'}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

The former is true by our assumption on V_1 and V_2. The latter follows by the induction hypothesis and our assumption on V^l and V^r.

□

Lemma D.49 (ValUpL-general).

$$\frac{\begin{array}{c}\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} c : A \sqsubseteq A' \\ \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} e : A' \sqsubseteq A'' \\ \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq N : c \circ e\end{array}}{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} \langle A' \leftarrowtail A \rangle M \sqsubseteq N : e}$$

Proof. Let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![\Gamma^{\sqsubseteq}]\!]$. We need to show that

$$(\langle A' \leftarrowtail A \rangle M[\gamma_1], N[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![e]\!].$$

By monadic bind (Lemma D.16), with $E_1 = \langle A' \leftarrowtail A \rangle \bullet$ and $E_2 = \bullet$, it suffices to show

$$(\langle A' \leftarrowtail A \rangle V_1, V_2) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![e]\!],$$

where $k \le j$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![c \circ e]\!]$.

We continue by cases on c. The case $c = \texttt{bool}$ is similar to that in the previous lemma, so we skip to considering the case $c = c_i \rightarrow_{c_\sigma} c_o$. By inversion, we see that $e = e_i \rightarrow_{e_\sigma} e_o$.

We have $A = A_i \rightarrow_{\hat{\sigma}} A_o$ and $A' = A_i' \rightarrow_{\hat{\sigma}'} A_o'$, and also Thus, we have that $c \circ e = (c_i \circ e_i) \rightarrow_{c_\sigma \circ e_\sigma} (c_o \circ e_o)$.

We need to show that

$$(\langle (A_i' \rightarrow_{\hat{\sigma}'} A_o') \leftarrowtail (A_i \rightarrow_{\hat{\sigma}} A_o) \rangle M[\gamma_1], N[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![e_i \rightarrow_{e_\sigma} e_o]\!].$$

Similar to before, it suffices to show that these terms are related at $\mathcal{V}_k^{\sim}[\![e_i \rightarrow_{e_\sigma} e_o]\!]$. This is similar to proof of the previous lemma, and hence omitted.

□

Lemma D.50 (ValDnL-general).

$$\frac{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} c : A \sqsubseteq A' \qquad \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} e : A' \sqsubseteq A'' \qquad \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq N : e}{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} \langle A \twoheadleftarrow A' \rangle M \sqsubseteq N : c \circ e}$$

Proof. This proof is dual to the proof of ValUpR-general (Lemma D.48) and is hence omitted. □

Lemma D.51 (ValDnR-general).

$$\frac{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} c : A \sqsubseteq A' \qquad \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} e : A' \sqsubseteq A'' \qquad \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq N : c \circ e}{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq \langle A' \twoheadleftarrow A'' \rangle N : c}$$

Proof. This proof is dual to the proof of ValUpL-general (Lemma D.49) and is hence omitted. □

Lemma D.52 (EffUpR-general).

$$\frac{d_\sigma : \sigma \sqsubseteq \sigma' \qquad d'_\sigma : \sigma' \sqsubseteq \sigma'' \qquad \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq N : c}{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma \circ d'_\sigma} M \sqsubseteq \langle \sigma'' \leftarrowtail \sigma' \rangle N : c}$$

Proof. Let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma^{\sqsubseteq}]\!]$. We need to show that

$$(M, \langle \sigma'' \leftarrowtail \sigma' \rangle N) \in \mathcal{E}^{\sim}_j[\![d_\sigma \circ d'_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

We prove this statement by Löb induction (Lemma D.14). That is, assume for all $k \le j$ and all $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma]\!])_k(\mathcal{V}^{\sim}[\![c]\!])$, we have

$$(M', \langle \sigma'' \leftarrowtail \sigma' \rangle N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma \circ d'_\sigma]\!])_k(\mathcal{V}^{\sim}[\![c]\!]).$$

Let $(M, N) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. We need to show

$$(M, \langle \sigma'' \leftarrowtail \sigma' \rangle N) \in \mathcal{E}^{\sim}_j[\![d_\sigma \circ d'_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

We proceed by cases on d'_σ. The case $d'_\sigma = ?$ is immediate, so consider $d'_\sigma = \mathtt{inj}(d_c)$, where $d_c : \sigma_c \sqsubseteq \Sigma\,|_{\mathrm{supp}(\sigma_c)}$. In this case, we know that $\sigma'' = ?$. Furthermore, we have

$$d_\sigma \circ d'_\sigma = d_\sigma \circ (\mathtt{inj}(d_c)) = \mathtt{inj}(d_\sigma \circ d_c).$$

Thus, we need to show

$$(M, \langle ? \leftarrowtail \sigma' \rangle N) \in \mathcal{E}^{\sim}_j[\![\mathtt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!].$$

By monadic bind (Lemma D.16), it will suffice to consider the following cases:

- Let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![c]\!]$. We need to show

$$(V_1, \langle ? \leftarrowtail \sigma' \rangle V_2) \in \mathcal{E}^{\sim}_k[\![\mathtt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!].$$

  By anti-reduction, it suffices to show that

$$(V_1, V_2) \in \mathcal{E}^{\sim}_k[\![\mathtt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!].$$

As V_1 and V_2 are values, it suffices by Lemma D.4 to show that $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![c]\!]$, which is true by assumption.

- Let $k \leq j$ and $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in d_\sigma$ be an effect that is caught by $\langle ? \nwarrow \sigma' \rangle\bullet$. Let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon]\!])_k$, and let $E^l\#\varepsilon$ and $E^r\#\varepsilon$ be evaluation contexts such that $(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}^{\sim}[\![d_\varepsilon]\!])_k(\mathcal{E}^{\sim}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!])$. We need to show that

$$\begin{aligned}&(E^l[\texttt{raise}\ \varepsilon(V^l)],\\ &\langle ? \nwarrow \sigma' \rangle E^r[\texttt{raise}\ \varepsilon(V^r)]) \in \mathcal{E}^{\sim}_k[\![\texttt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!].\end{aligned}$$

By anti-reduction, it suffices to show that

$$\begin{aligned}&(E^l[\texttt{raise}\ \varepsilon(V^l)],\\ &\texttt{let}\ y = \langle d^r_\varepsilon \twoheadleftarrow d^?_\varepsilon \rangle\texttt{raise}\ \varepsilon(\langle c^?_\varepsilon \nwarrow c^r_\varepsilon \rangle V^r)\ \texttt{in}\ \langle ? \nwarrow \sigma' \rangle E^r[y])\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\texttt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!].\end{aligned}$$

Let V'^r be the term to which $\langle c^?_\varepsilon \nwarrow c^r_\varepsilon \rangle V^r$ steps. By anti-reduction, it suffices to show

$$\begin{aligned}&(E^l[\texttt{raise}\ \varepsilon(V^l)],\\ &\texttt{let}\ y = \langle d^r_\varepsilon \twoheadleftarrow d^?_\varepsilon \rangle\texttt{raise}\ \varepsilon(V'^r)\ \texttt{in}\ \langle ? \nwarrow \sigma' \rangle E^r[y])\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\texttt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!].\end{aligned}$$

As neither term steps, it suffices to show they are related in $\mathcal{R}^{\sim}_k[\![\texttt{inj}(d_\sigma \circ d_c)]\!]\mathcal{V}^{\sim}[\![c]\!]$.. To this end, we need to show (1) $(V^l, V'^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon \circ c'_\varepsilon]\!])_k$, and (2) given $k' \leq k$ and $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_\varepsilon \circ d'_\varepsilon]\!])_{k'}$, we have

$$\begin{aligned}&(E^l[V_1],\\ &\texttt{let}\ y = \langle d^r_\varepsilon \twoheadleftarrow d^?_\varepsilon \rangle V_2\ \texttt{in}\ \langle ? \nwarrow \sigma' \rangle E^r[y])\\ &\qquad \in (\blacktriangleright\mathcal{E}^{\sim}[\![\texttt{Inj}(I, d_\sigma \circ d_c)]\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]).\end{aligned}$$

To show (1), it suffices by forward reduction to show that $(V^l, \langle c^?_\varepsilon \nwarrow c^r_\varepsilon \rangle V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon \circ c'_\varepsilon]\!])_k$. This follows inductively from ValUpR (which we are proving simultaneously and can therefore apply at smaller types), and our assumption on V^l and V^r.
To show (2), let V'_2 be the value to which $\langle d^r_\varepsilon \twoheadleftarrow d^?_\varepsilon \rangle V_2$ steps. It suffices by anti-reduction to show

$$\begin{aligned}&(E^l[V_1], \langle ? \nwarrow \sigma' \rangle E^r[V'_2])\\ &\qquad \in (\blacktriangleright\mathcal{E}^{\sim}[\![\texttt{Inj}(I, d_\sigma \circ d_c)]\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]).\end{aligned}$$

By the Löb induction hypothesis, it suffices to show that

$$\begin{aligned}&(E^l[V_1], E^r[V'_2])\\ &\qquad \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]).\end{aligned}$$

By our assumption on E^l and E^r, it suffices to show that $(V_1, V'_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_\varepsilon]\!])_{k'}$. By forward reduction, it suffices to show that

$$(V_1, \langle d_\varepsilon^r \twoheadleftarrow d_\varepsilon^? \rangle V_2) \in (\blacktriangleright \mathcal{E}^\sim [\![d_\sigma]\!])_{k'} (\mathcal{V}^\sim [\![d_\varepsilon]\!]).$$

Now inductively by ValDnR, it suffices to show $(V_1, V_2) \in (\blacktriangleright \mathcal{V}^\sim [\![d_\varepsilon \circ d'_\varepsilon]\!])_{k'}$, which is our assumption.

The case where d'_σ is a concrete effect precision derivation is similar to the above.

□

Lemma D.53 (EffUpL-general).

$$\frac{\begin{array}{c} d_\sigma : \sigma \sqsubseteq \sigma' \\ d'_\sigma : \sigma' \sqsubseteq \sigma'' \\ \Sigma \mid \Gamma^\sqsubseteq \vdash_{d_\sigma \circ d'_\sigma} M \sqsubseteq N : c \end{array}}{\Sigma \mid \Gamma^\sqsubseteq \vdash_{d'_\sigma} \langle \sigma' \leftarrowtail \sigma \rangle M \sqsubseteq N : c}$$

Proof. This is proved similarly to the above. □

Lemma D.54 (EffDnL-general).

$$\frac{\begin{array}{c} d_\sigma : \sigma \sqsubseteq \sigma' \\ d'_\sigma : \sigma' \sqsubseteq \sigma'' \\ \Sigma \mid \Gamma^\sqsubseteq \vdash_{d'_\sigma} M \sqsubseteq N : c \end{array}}{\Sigma \mid \Gamma^\sqsubseteq \vdash_{d_\sigma \circ d'_\sigma} \langle \sigma \twoheadleftarrow \sigma' \rangle M \sqsubseteq N : c}$$

Proof. We prove this by Löb induction (Lemma D.14). That is, assume for all $k \leq j$ and all $(M', N') \in (\blacktriangleright \mathcal{E}^\sim [\![d'_\sigma]\!])_k (\mathcal{V}^\sim [\![c]\!])$, we have

$$(\langle \sigma \twoheadleftarrow \sigma' \rangle M', N') \in (\blacktriangleright \mathcal{E}^\sim [\![d_\sigma \circ d'_\sigma]\!])_k (\mathcal{V}^\sim [\![c]\!]).$$

Let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^\sim [\![\Gamma^\sqsubseteq]\!]$, and let $(M, N) \in \mathcal{E}_j^\sim [\![d'_\sigma]\!] \mathcal{V}^\sim [\![c]\!]$. We need to show

$$(\langle \sigma \twoheadleftarrow \sigma' \rangle M, N) \in \mathcal{E}_j^\sim [\![d_\sigma \circ d'_\sigma]\!] \mathcal{V}^\sim [\![c]\!].$$

By monadic bind (Lemma D.16) and the fact that effect casts are the identity on values, it will suffice to show the following:

Let $k \leq j$ and $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in d'_\sigma$ be an effect that is caught by $\langle \sigma \twoheadleftarrow \sigma' \rangle \bullet$. Let $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^\sim [\![c_\varepsilon]\!])_k$, and let $E^l \# \varepsilon$ and $E^r \# \varepsilon$ be evaluation contexts such that $(x^l . E^l [x^l], x^r . E^r [x^r]) \in (\blacktriangleright \mathcal{K}^\sim [\![d_\varepsilon]\!])_k (\mathcal{E}^\sim [\![d'_\sigma]\!] \mathcal{V}^\sim [\![c]\!])$. We need to show that

$$\begin{aligned} &(\langle \sigma \twoheadleftarrow \sigma' \rangle E^l [\texttt{raise } \varepsilon(V^l)], \\ &E^r [\texttt{raise } \varepsilon(V^r)] N) \\ &\qquad \in \mathcal{E}_j^\sim [\![d_\sigma \circ d'_\sigma]\!] \mathcal{V}^\sim [\![c]\!]. \end{aligned}$$

Note that if $\varepsilon \notin \sigma$, then the left hand side steps to $\mho$, in which case we are finished by ErrBot (Lemma D.45). Otherwise, the proof proceeds analogously to EffUpR (Lemma D.52), with upcasts and downcasts interchanged.

□

Lemma D.55 (EffDnR-general).

$$\frac{\begin{array}{c} d_\sigma : \sigma \sqsubseteq \sigma' \\ d'_\sigma : \sigma' \sqsubseteq \sigma'' \\ \Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma \circ d_{\sigma'}} M \sqsubseteq N : c \end{array}}{\Sigma \mid \Gamma^{\sqsubseteq} \vdash_{d_\sigma} M \sqsubseteq \langle \sigma' \twoheadleftarrow \sigma'' \rangle N : c}$$

Proof. We prove this statement by Löb induction (Lemma D.14). That is, assume for all $k \leq j$ and all $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma \circ d'_\sigma]\!])_k(\mathcal{V}^{\sim}[\![c]\!])$, we have

$$(M', \langle \sigma' \twoheadleftarrow \sigma'' \rangle N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma]\!])_k(\mathcal{V}^{\sim}[\![c]\!]).$$

Let $(\gamma_1, \gamma_2) \in \mathcal{G}^{\sim}_j[\![\Gamma^{\sqsubseteq}]\!]$, and let $(M, N) \in \mathcal{E}^{\sim}_j[\![d_\sigma \circ d'_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. We need to show

$$(M, \langle \sigma' \twoheadleftarrow \sigma'' \rangle N) \in \mathcal{E}^{\sim}_j[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!].$$

By monadic bind (Lemma D.16) and the fact that effect casts are the identity on values, it will suffice to show the following:

Let $k \leq j$ and $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in d_\sigma \circ d'_\sigma$ be an effect that is caught by $\langle \sigma' \twoheadleftarrow \sigma'' \rangle \bullet$. Let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_\varepsilon]\!])_k$, and let $E^l \# \varepsilon$ and $E^r \# \varepsilon$ be evaluation contexts such that $(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}^{\sim}[\![d_\varepsilon]\!])_k(\mathcal{E}^{\sim}[\![d_\sigma \circ d'_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!])$. We need to show that

$$\begin{array}{l} (E^l[\texttt{raise}\ \varepsilon(V^l)], \\ \ \langle \sigma' \twoheadleftarrow \sigma'' \rangle E^r[\texttt{raise}\ \varepsilon(V^r)]) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]. \end{array}$$

First note that by Lemma D.19, there exist c_1, c_2, d_1, and d_2 such that $c_\varepsilon = c_1 \circ c_2$ and $d_\varepsilon = d_1 \circ d_2$ and $\varepsilon : c_1 \rightsquigarrow d_1 \in d_\sigma$ and $\varepsilon : c_2 \rightsquigarrow d_2 \in d'_\sigma$. In particular, this that $\varepsilon \in \sigma'$, so the downcast from σ'' to σ' does not fail. Let $c^L = c^l_1 (= c^l_\varepsilon)$, $c^M = c^r_1 = c^l_2$, and $c^R = c^r_2 (= c^r_\varepsilon)$, and likewise define d^L, d^M and d^R.

By anti-reduction, it suffices to show that

$$\begin{array}{l} (E^l[\texttt{raise}\ \varepsilon(V^l)], \\ \ \texttt{let}\ y = \langle d^R \leftarrowtail d^M \rangle \texttt{raise}\ \varepsilon(\langle c^M \twoheadleftarrow c^R \rangle V^r)\ \texttt{in}\ \langle \sigma' \twoheadleftarrow \sigma'' \rangle E^r[y]) \\ \quad \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]. \end{array}$$

Let V'^r be the term to which $\langle c^M \twoheadleftarrow c^R \rangle V^r$ steps. By anti-reduction, it suffices to show

$$\begin{array}{l} (E^l[\texttt{raise}\ \varepsilon(V^l)], \\ \ \texttt{let}\ y = \langle d^R \leftarrowtail d^M \rangle \texttt{raise}\ \varepsilon(V'^r)\ \texttt{in}\ \langle \sigma' \twoheadleftarrow \sigma'' \rangle E^r[y]) \\ \quad \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]. \end{array}$$

As neither term steps, it suffices to show they are related in $\mathcal{R}^{\sim}_k[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![c]\!]$. To this end, we need to show (1) $(V^l, V'^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_1]\!])_k$, and (2) given $k' \leq k$ and $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![d_1]\!])_{k'}$, we have

$$\begin{array}{l} (E^l[V_1], \\ \ \texttt{let}\ y = \langle d^R \leftarrowtail d^M \rangle V_2\ \texttt{in}\ \langle \sigma' \twoheadleftarrow \sigma'' \rangle E^r[y]) \\ \quad \in (\blacktriangleright\mathcal{E}^{\sim}[\![d_\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]). \end{array}$$

(1) follows from forward reduction and the inductive hypothesis for value types. To show (2), let V_2' be the value to which $\langle d^R \nwarrow d^M\rangle V_2$ steps. It suffices by anti-reduction to show

$$\begin{aligned}&(E^l[V_1], \langle \sigma'' \nwarrow \sigma'\rangle E^r[V_2'])\\ &\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![d_\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]).\end{aligned}$$

By the Löb induction hypothesis, it suffices to show that

$$\begin{aligned}&(E^l[V_1], E^r[V_2'])\\ &\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![d_\sigma \circ d_\sigma']\!])_{k'}(\mathcal{V}^{\sim}[\![c]\!]).\end{aligned}$$

By our assumption on E^l and E^r, it suffices to show that $(V_1, V_2') \in (\blacktriangleright \mathcal{V}^{\sim}[\![d_\varepsilon]\!])_{k'}$. By forward reduction, it suffices to show that

$$(V_1, \langle d^R \nwarrow d^M\rangle V_2) \in (\blacktriangleright \mathcal{E}^{\sim}[\![d_\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![d_\varepsilon]\!]).$$

Now inductively by ValUpR, it suffices to show $(V_1, V_2) \in (\blacktriangleright \mathcal{V}^{\sim}[\![d_1]\!])_{k'}$, which is our assumption.

The case where d_σ' is a concrete effect precision derivation is similar to the above. □

Lemma D.56 (ValUpEval).

$$\langle B \nwarrow A\rangle M \equiv \texttt{let } x = M \texttt{ in } \langle B \nwarrow A\rangle x$$

Proof. We show one direction of the equivalence; the other is symmetric. Let j be arbitrary and let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![\Gamma]\!]$. We need to show

$$((\langle B \nwarrow A\rangle M)[\gamma_1], (\texttt{let } x = M \texttt{ in } \langle B \nwarrow A\rangle x)[\gamma_2]) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By Monadic Bind (Lemma D.16) and reflexivity, it will suffice to show that for all $k \le j$ let $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![A]\!]$, we have

$$((\langle B \nwarrow A\rangle V_1), (\texttt{let } x = V_2 \texttt{ in } \langle B \nwarrow A\rangle x)) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By anti-reduction, it suffices to show

$$((\langle B \nwarrow A\rangle V_1), (\langle B \nwarrow A\rangle V_2)) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By congruence, it suffices to show

$$(V_1, V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

This follows from our assumption on V_1 and V_2.

□

Lemma D.57 (ValDnEval).

$$\langle A \swarrow B\rangle M \equiv \texttt{let } x = M \texttt{ in } \langle A \swarrow B\rangle x$$

Proof. Dual to the above. □

Lemma D.58 (cast-retraction). *let $A \sqsubseteq B$ and $\sigma \sqsubseteq \sigma'$, and let $c : A \sqsubseteq B$ and $d_\sigma : \sigma \sqsubseteq \sigma'$. Let $\Sigma \mid \Gamma^{\sqsubseteq} \vdash_\sigma M \sqsubseteq N : A$. The following hold:*

(1) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma \langle A \swarrow B\rangle\langle B \nwarrow A\rangle M \sqsubseteq N : A$
(2) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma \langle \sigma \swarrow \sigma'\rangle\langle \sigma' \nwarrow \sigma\rangle M \sqsubseteq N : A$

Proof. We prove stronger, "pointwise" version of the above statements. Namely, we assume $(M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$, and show, for example, that $(\langle A \twoheadleftarrow B\rangle\langle B \rightarrowtail A\rangle M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. The proof is by simultaneous induction on the derivations c and d_σ.

(1) Let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![A]\!]$. Suppose $(M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. We need to show

$$(\langle A \twoheadleftarrow B\rangle\langle B \rightarrowtail A\rangle M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

By monadic bind (Lemma D.16), it suffices to show that

$$(\langle A \twoheadleftarrow B\rangle\langle B \rightarrowtail A\rangle V_1, V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!],$$

where $k \le j$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![A]\!]$.
We proceed by induction on the precision derivation c. If $c = \mathtt{bool}$, then we need to show

$$(\langle \mathtt{bool} \twoheadleftarrow \mathtt{bool}\rangle\langle \mathtt{bool} \rightarrowtail \mathtt{bool}\rangle V_1, V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!].$$

According to the operational semantics, we have that

$$\langle \mathtt{bool} \twoheadleftarrow \mathtt{bool}\rangle\langle \mathtt{bool} \rightarrowtail \mathtt{bool}\rangle V_1 \mapsto^2 V_1.$$

So by anti-reduction (Lemma D.6), it suffices to show that $(V_1, V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!]$, which follows from our assumption.

If $c = c_i \rightarrow_{c_\sigma} c_o$, then $A = A_i \rightarrow_{\sigma_A} A_o$ and $B = B_i \rightarrow_{\sigma_B} B_o$. We need to show

$$\begin{aligned}(\langle (A_i \rightarrow_{\sigma_A} A_o) \twoheadleftarrow (B_i \rightarrow_{\sigma_B} B_o)\rangle\langle (B_i \rightarrow_{\sigma_B} B_o) \rightarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_1, V_2)&\\ \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i \rightarrow_{\sigma_A} A_o]\!].&\end{aligned}$$

As both of these are values, it suffices to show that they are related in $\mathcal{V}^{\sim}[\![A_i \rightarrow_{\sigma_A} A_o]\!]$. To this end, let $k' \le k$ and let $(V^l, V^r) \in \mathcal{V}_{k'}^{\sim}[\![A_i]\!]$. We need to show that

$$\begin{aligned}&((\langle (A_i \rightarrow_{\sigma_A} A_o) \twoheadleftarrow (B_i \rightarrow_{\sigma_B} B_o)\rangle\langle (B_i \rightarrow_{\sigma_B} B_o) \rightarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_1)\, V^l,\\ &V_2\, V^r)\\ &\qquad \in \mathcal{E}_{k'}^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].\end{aligned}$$

The former term steps, so by anti-reduction, it suffices to show that

$$\begin{aligned}&(\langle A_o \twoheadleftarrow B_o\rangle\langle \sigma_A \twoheadleftarrow \sigma_B\rangle(((\langle (B_i \rightarrow_{\sigma_B} B_o) \rightarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_1)\, \langle B_i \rightarrowtail A_i\rangle V^l), \qquad V_2\, V^r)\\ &\qquad \in \mathcal{E}_{k'}^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].\end{aligned}$$

Let V'^l be the value to which $\langle B_i \rightarrowtail A_i\rangle V^l$ steps. By anti-reduction, it suffices to show that

$$\begin{aligned}&(\langle A_o \twoheadleftarrow B_o\rangle\langle \sigma_A \twoheadleftarrow \sigma_B\rangle\\ &\qquad (\langle B_o \rightarrowtail A_o\rangle\langle \sigma_B \rightarrowtail \sigma_A\rangle\\ &\qquad\qquad (V_1\, \langle A_i \twoheadleftarrow B_i\rangle V'^l)),\\ &V_2\, V^r)\\ &\qquad \in \mathcal{E}_{k'}^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].\end{aligned}$$

We will appeal to transitivity (Lemma D.64). We continue by cases on $\sim$. First assume $\sim$ is $<$. Let V'^r be the value to which $\langle B_i \leftarrowtail A_i\rangle V^r$ steps. If we show (1)

$$\begin{aligned}
&(\langle A_o \twoheadleftarrow B_o\rangle\langle\sigma_A \twoheadleftarrow \sigma_B\rangle \\
&\qquad(\langle B_o \leftarrowtail A_o\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle \\
&\qquad\qquad(V_1\,\langle A_i \twoheadleftarrow B_i\rangle V'^l)), \\
&\;\langle A_o \twoheadleftarrow B_o\rangle\langle B_o \leftarrowtail A_o\rangle \\
&\qquad(\langle\sigma_A \twoheadleftarrow \sigma_B\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle \\
&\qquad\qquad(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r))) \\
&\qquad\in \mathcal{E}^{\sim}_{k'}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].
\end{aligned}$$

and (2)

$$\begin{aligned}
&(\langle A_o \twoheadleftarrow B_o\rangle\langle B_o \leftarrowtail A_o\rangle \\
&\qquad(\langle\sigma_A \twoheadleftarrow \sigma_B\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle \\
&\qquad\qquad(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)), \\
&\;V_2\,V^r) \\
&\qquad\in \mathcal{E}^{\sim}_{\omega}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!],
\end{aligned}$$

then we will be finished by transitivity.

To show (1), first note that by monotonicity of casts (Lemma D.63), it suffices to show that

$$\begin{aligned}
&(\langle\sigma_A \twoheadleftarrow \sigma_B\rangle \\
&\qquad(\langle B_o \leftarrowtail A_o\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle \\
&\qquad\qquad(V_1\,\langle A_i \twoheadleftarrow B_i\rangle V'^l)), \\
&\;\langle B_o \leftarrowtail A_o\rangle \\
&\qquad(\langle\sigma_A \twoheadleftarrow \sigma_B\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle \\
&\qquad\qquad(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r))) \\
&\qquad\in \mathcal{E}^{\sim}_{k'}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![B_o]\!].
\end{aligned}$$

Then by commutativity of casts (Corollary D.61), it suffices to show

$$\begin{aligned}
&(\langle\sigma_B \leftarrowtail \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow B_i\rangle V'^l), \\
&\;\langle\sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)) \\
&\qquad\in \mathcal{E}^{\sim}_{k'}[\![\sigma_B]\!]\mathcal{V}^{\sim}[\![A_o]\!].
\end{aligned}$$

By monotonicity of casts again, it suffices to show

$$\begin{aligned}
&((V_1\,\langle A_i \twoheadleftarrow B_i\rangle V'^l), \\
&\;(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)) \\
&\qquad\in \mathcal{E}^{\sim}_{k'}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].
\end{aligned}$$

By soundness of the precision rule for function application, it suffices to show that $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![(A_i \to_{\sigma_A} A_o)]\!]$ and that $(\langle A_i \twoheadleftarrow B_i\rangle V'^l, \langle A_i \twoheadleftarrow B_i\rangle V'^r) \in \mathcal{E}_k^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_i]\!]$. The former holds by assumption, and to show the latter, it suffices by forward reduction to show $(\langle A_i \twoheadleftarrow B_i\rangle\langle B_i \rightarrowtail A_i\rangle V^l, \langle A_i \twoheadleftarrow B_i\rangle\langle B_i \rightarrowtail A_i\rangle V^r) \in \mathcal{E}_k^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_i]\!]$. This follows from the inductive hypothesis and assumption on V^l and V^r.
To show (2), it suffices by the inductive hypothesis applied twice to show

$$\begin{aligned}&((V_2\,\langle A_i \twoheadleftarrow B_i\rangle V''^r)), V_2\,V^r)\\ &\quad \in \mathcal{E}_\omega^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!],\end{aligned}$$

By forward reduction, it suffices to show

$$\begin{aligned}&((V_2\,\langle A_i \twoheadleftarrow B_i\rangle V''^r)), V_2\,V^r)\\ &\quad \in \mathcal{E}_\omega^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!],\end{aligned}$$

By soundness of function application, it suffices to show that V_2 is related to itself at $\mathcal{V}_\omega^{\sim}[\![(A_i \to_{\sigma_A} A_o)]\!]$ and that $(\langle A_i \twoheadleftarrow B_i\rangle V''^r, V^r) \in \mathcal{E}_\omega^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_i]\!]$. The former holds by reflexivity (Corollary D.28), and to show the latter it suffices by forward reduction to show that

$$(\langle A_i \twoheadleftarrow B_i\rangle\langle B_i \rightarrowtail A_i\rangle V^r, V^r) \in \mathcal{E}_\omega^{\sim}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_i]\!],$$

which follows by the inductive hypothesis and reflexivity.
The case when $\sim$ is $<$ is analogous.

(2) Let $(\gamma_1, \gamma_2) \in \mathcal{G}_j^{\sim}[\![A]\!]$. We use Löb induction. We assume that for all $k \le j$ and all related terms $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!])$, we have

$$(\langle \sigma \twoheadleftarrow \sigma'\rangle\langle \sigma' \rightarrowtail \sigma\rangle M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!]).$$

Let $(M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. We need to show that

$$(\langle \sigma \twoheadleftarrow \sigma'\rangle\langle \sigma' \rightarrowtail \sigma\rangle M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

By monadic bind (Lemma D.16), it suffices to consider the following cases:

(a) Let $k \le j$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![A]\!]$. We need to show

$$(\langle \sigma \twoheadleftarrow \sigma'\rangle\langle \sigma' \rightarrowtail \sigma\rangle V_1, V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$$

This follows by anti-reduction and assumption.

(b) Let $k \le j$ and let $\varepsilon : c \rightsquigarrow d \in \sigma$. Let C' and D' be the types such that $\varepsilon : C' \rightsquigarrow D' \in \sigma'$. Let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![C]\!])_k$ and let $E^l\#\varepsilon$ and $E^r\#\varepsilon$ be such that

$$(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}_{)}^{\sim}[\![D]\!]_k(\mathcal{E}^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]).$$

We need to show that

$$\begin{aligned}&(\langle \sigma \twoheadleftarrow \sigma'\rangle\langle \sigma' \rightarrowtail \sigma\rangle E^l[\mathtt{raise}\ \varepsilon(V^l)], E^r[\mathtt{raise}\ \varepsilon(V^r)])\\ &\quad \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

The first term steps, so by anti-reduction it suffices to show

$$(\langle \sigma \twoheadleftarrow \sigma' \rangle(\texttt{let } x = \langle D \twoheadleftarrow D' \rangle \texttt{raise } \varepsilon(\langle C' \leftarrowtail C \rangle V^l) \texttt{ in } \langle \sigma' \leftarrowtail \sigma \rangle E^l[x]),$$
$$E^r[\texttt{raise } \varepsilon(V^r)])$$
$$\in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

Let V'^l be the value to which $\langle C' \leftarrowtail C \rangle V^l$ steps. By anti-reduction, it suffices to show

$$(\texttt{let } y = \langle D' \leftarrowtail D \rangle \texttt{raise } \varepsilon(\langle C \twoheadleftarrow C' \rangle V'^l) \texttt{ in } \langle \sigma \twoheadleftarrow \sigma' \rangle \texttt{let } x = \langle D \twoheadleftarrow D' \rangle y \texttt{ in } \langle \sigma' \leftarrowtail \sigma \rangle E^l[x],$$
$$E^r[\texttt{raise } \varepsilon(V^r)])$$
$$\in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

Let y' be the value to which $\langle D \twoheadleftarrow D' \rangle y$ steps. Let V''^l be the value to which $\langle C \twoheadleftarrow C' \rangle V'^l$ steps.
By anti-reduction, it suffices to show

$$(\texttt{let } y = \langle D' \leftarrowtail D \rangle \texttt{raise } \varepsilon(V''^l) \texttt{ in } \langle \sigma \twoheadleftarrow \sigma' \rangle \langle \sigma' \leftarrowtail \sigma \rangle E^l[y'],$$
$$E^r[\texttt{raise } \varepsilon(V^r)])$$
$$\in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

Neither term steps, so it suffices to show they are related in $\mathcal{R}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. To this end, we first show that $(V''^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![C]\!])_k$. By forward reduction, it suffices to show that $(\langle C \twoheadleftarrow C' \rangle \langle C' \leftarrowtail C \rangle V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![C]\!])_k$. This follows from the inductive hypothesis for value types and our assumption on V^l and V^r.
We now show that, given $k' \leq k$ and values $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![D]\!])_{k'}$, we have

$$(\texttt{let } y = \langle D' \leftarrowtail D \rangle V_1 \texttt{ in } \langle \sigma \twoheadleftarrow \sigma' \rangle \langle \sigma' \leftarrowtail \sigma \rangle E^l[y'],$$
$$E^r[V_2])$$
$$\in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!]).$$

Let V_1' be the value to which $\langle D' \leftarrowtail D \rangle V_1$ steps. By anti-reduction, it will suffice to show

$$(\texttt{let } y = V_1' \texttt{ in } \langle \sigma \twoheadleftarrow \sigma' \rangle \langle \sigma' \leftarrowtail \sigma \rangle E^l[y'],$$
$$E^r[V_2])$$
$$\in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!]).$$

By forward reduction, it will suffice to show

$$(\texttt{let } y = V_1' \texttt{ in } \langle \sigma \twoheadleftarrow \sigma' \rangle \langle \sigma' \leftarrowtail \sigma \rangle E^l[\langle D \twoheadleftarrow D' \rangle y],$$
$$E^r[V_2])$$
$$\in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!]).$$

By anti-reduction, it will suffice to show

$$(\langle \sigma \twoheadleftarrow \sigma' \rangle \langle \sigma' \leftarrowtail \sigma \rangle E^l[\langle D \twoheadleftarrow D' \rangle V_1'], E^r[V_2]) \\ \in (\blacktriangleright \mathcal{E}^\sim[\![\sigma]\!])_k(\mathcal{V}^\sim[\![A]\!]).$$

By the Löb induction hypothesis, it suffices to show that

$$(E^l[\langle D \twoheadleftarrow D' \rangle V_1'], E^r[V_2]) \\ \in (\blacktriangleright \mathcal{E}^\sim[\![\sigma]\!])_k(\mathcal{V}^\sim[\![A]\!]).$$

By forward reduction, it suffices to show

$$(E^l[\langle D \twoheadleftarrow D' \rangle \langle D' \leftarrowtail D \rangle V_1], E^r[V_2]) \\ \in (\blacktriangleright \mathcal{E}^\sim[\![\sigma]\!])_k(\mathcal{V}^\sim[\![A]\!]).$$

By the induction hypothesis for value types, it suffices to show

$$(E^l[V_1], E^r[V_2]) \in (\blacktriangleright \mathcal{E}^\sim[\![\sigma]\!])_k(\mathcal{V}^\sim[\![A]\!]).$$

This follows by our assumption on E^l and E^r.

□

Lemma D.59 (Gradual subtyping). *Let $c : A \sqsubseteq B$ and $c' : A' \sqsubseteq B'$ where $A \le A'$ and $B \le B'$. Let $d_\sigma : \sigma_1 \sqsubseteq \sigma_2$ and $d'_\sigma : \sigma_1' \sqsubseteq \sigma_2'$ where $\sigma_1 \le \sigma_1'$ and $\sigma_2 \le \sigma_2'$. Suppose $M \equiv N$. The following hold:*

(1)

$$\frac{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\tau} M \sqsubseteq N : A}{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\tau} \langle B \leftarrowtail A \rangle M \sqsubseteq \langle B' \leftarrowtail A' \rangle N : B'}$$

(2)

$$\frac{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\tau} M \sqsubseteq N : B}{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{d_\tau} \langle A' \twoheadleftarrow B' \rangle M \sqsubseteq \langle A \twoheadleftarrow B \rangle N : A'}$$

(3)

$$\frac{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma_1} M \sqsubseteq N : d}{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma_2'} \langle \sigma_2 \leftarrowtail \sigma_1 \rangle M \sqsubseteq \langle \sigma_2' \leftarrowtail \sigma_1' \rangle N : d}$$

(4)

$$\frac{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma_2} M \sqsubseteq N : d}{\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma_1'} \langle \sigma_1' \twoheadleftarrow \sigma_2' \rangle M \sqsubseteq \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle N : d}$$

Proof. By simultaneous induction on the derivation $c' : A' \sqsubseteq B'$ and $d'_\sigma : \sigma_1' \sqsubseteq \sigma_2'$.

(1) We need to show

$$(\langle B \leftarrowtail A \rangle M, \langle B' \leftarrowtail A' \rangle N) \in \mathcal{E}_j^\sim[\![d_\sigma]\!]\mathcal{V}^\sim[\![B']\!].$$

By monadic bind (Lemma D.16), with $E_1 = \langle B \leftarrowtail A \rangle \bullet$ and $E_2 = \langle B' \leftarrowtail A' \rangle \bullet$, it suffices to show the following.
Let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}_k^\sim[\![A']\!]$. We need to show

$$(\langle B \leftarrowtail A \rangle V_1, \langle B' \leftarrowtail A' \rangle V_2) \in \mathcal{E}_k^\sim[\![d_\sigma]\!]\mathcal{V}^\sim[\![B']\!].$$

We continue by cases on c'.
Case $c' = \texttt{bool}$. Then by inversion on the rules for subtyping of precision derivations, we have $c = \texttt{bool}$.
We need to show

$$(\langle \texttt{bool} \nwarrow \texttt{bool}\rangle V_1, \langle \texttt{bool} \nwarrow \texttt{bool}\rangle V_2) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![\texttt{bool}]\!]$$

This follows by anti-reduction and our assumption on V_1 and V_2.
Case $c' = c'_i \to_{c'_\sigma} c'_o : A'_i \to_{\sigma'_A} A'_o \sqsubseteq B'_i \to_{\sigma'_B} B'_o$.
By inversion on the rules for subtyping for precision derivations, we have that $c = c_i \to_{c_\sigma} c_o$, where $c'_i \le c_i$, and $c_\sigma \le c'_\sigma$, and $c_o \le c'_o$.
Our assumption then becomes $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![A'_i \to_{\sigma'_A} A'_o]\!]$. We need to show

$$(\langle B_i \to_{\sigma_B} B_o \nwarrow A_i \to_{\sigma_A} A_o\rangle V_1, \langle B'_i \to_{\sigma'_B} B'_o \nwarrow A'_i \to_{\sigma'_A} A'_o\rangle V_2) \in \mathcal{E}^{\sim}_k[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![B'_i \to_{\sigma'_B} B'_o]\!].$$

Since both terms are values, it suffices to show they are related in $\mathcal{V}^{\sim}_k[\![B'_i \to_{\sigma'_B} B'_o]\!]$. Let $k' \le k$ and let $(V^l, V^r) \in \mathcal{V}^{\sim}_{k'}[\![B'_i]\!]$. We need to show

$$\begin{gathered}((\langle B_i \to_{\sigma_B} B_o \nwarrow A_i \to_{\sigma_A} A_o\rangle V_1)\, V^l,\\ (\langle B'_i \to_{\sigma'_B} B'_o \nwarrow A'_i \to_{\sigma'_A} A'_o\rangle V_2)\, V^r)\\ \in \mathcal{E}^{\sim}_{k'}[\![\sigma'_B]\!]\mathcal{V}^{\sim}[\![B'_o]\!].\end{gathered}$$

By anti-reduction, it suffices to show

$$\begin{gathered}(\langle B_o \nwarrow A_o\rangle\langle \sigma_B \nwarrow \sigma_A\rangle(V_1\, \langle A_i \twoheadleftarrow B_i\rangle V^l),\\ \langle B'_o \nwarrow A'_o\rangle\langle \sigma'_B \nwarrow \sigma'_A\rangle(V_2\, \langle A'_i \twoheadleftarrow B'_i\rangle V^r))\\ \in \mathcal{E}^{\sim}_{k'}[\![\sigma'_B]\!]\mathcal{V}^{\sim}[\![B'_o]\!].\end{gathered}$$

By the induction hypothesis applied twice, it suffices to show

$$\begin{gathered}((V_1\, \langle A_i \twoheadleftarrow B_i\rangle V^l), (V_2\, \langle A'_i \twoheadleftarrow B'_i\rangle V^r))\\ \in \mathcal{E}^{\sim}_{k'}[\![\sigma'_A]\!]\mathcal{V}^{\sim}[\![A'_o]\!].\end{gathered}$$

By soundness of the term precision congruence rule for function application (Lemma D.23), it suffices to show that $(V_1, V_2) \in \mathcal{V}^{\sim}_{k'}[\![A'_i \to_{\sigma'_A} A'_o]\!]$, and that

$$(\langle A_i \twoheadleftarrow B_i\rangle V^l, \langle A'_i \twoheadleftarrow B'_i\rangle V^r) \in \mathcal{E}^{\sim}_{k'}[\![d_\sigma]\!]\mathcal{V}^{\sim}[\![A'_i]\!].$$

The former holds by assumption. To show the latter, it suffices by the admissible direction of gradual subtyping rule ValDnSub (item (2) in Lemma A.1), whose proof does not depend on the present lemma, to show that $(V^l, V^r) \in \mathcal{V}^{\sim}_{k'}[\![B'_i]\!]$. This is true by assumption.

(2) Similar to the above.
(3) We need to show

$$(\langle \sigma_2 \nwarrow \sigma_1\rangle M, \langle \sigma'_2 \nwarrow \sigma'_1\rangle N) \in \mathcal{E}^{\sim}_j[\![\sigma'_2]\!]\mathcal{V}^{\sim}[\![c]\!].$$

We use Löb induction. That is, we assume as our induction hypothesis that

$$(\langle \sigma_2 \nwarrow \sigma_1\rangle M', \langle \sigma'_2 \nwarrow \sigma'_1\rangle N') \in \blacktriangleright(\mathcal{E}^{\sim}[\![\sigma'_2]\!])_j(\mathcal{V}^{\sim}[\![c]\!]),$$

for all $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma_1']\!])_j(\mathcal{V}^{\sim}[\![c]\!])$, and we show that under this assumption, we have

$$(\langle \sigma_2 \nwarrow \sigma_1 \rangle M, \langle \sigma_2' \nwarrow \sigma_1' \rangle N) \in \mathcal{E}_j^{\sim}[\![\sigma_2']\!]\mathcal{V}^{\sim}[\![c]\!]$$

for all $(M, N) \in \mathcal{E}_j^{\sim}[\![\sigma_1']\!]\mathcal{V}^{\sim}[\![c]\!]$.

Using Monadic Bind (Lemma D.16), we have the following cases:

- Let $k \leq j$ and $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![c]\!]$. We need to show

$$(\langle \sigma_2 \nwarrow \sigma_1 \rangle V_1, \langle \sigma_2' \nwarrow \sigma_1' \rangle V_2) \in \mathcal{E}_k^{\sim}[\![\sigma_2']\!]\mathcal{V}^{\sim}[\![c]\!].$$

  This follows by anti-reduction and our assumption on V_1 and V_2.
- Let $\varepsilon : c_i \rightsquigarrow d_i \in \sigma_1$ be an effect caught by $\langle \sigma_2' \nwarrow \sigma_1' \rangle \bullet$. Let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![c_i^l]\!])_k$, and let $(E^l, E^r) \in (\blacktriangleright\mathcal{K}^{\sim}[\![d_i^l]\!])_k(\mathcal{E}^{\sim}[\![\sigma_1']\!]\mathcal{V}^{\sim}[\![c]\!])$. We need to show

$$\begin{aligned}(&\langle \sigma_2 \nwarrow \sigma_1 \rangle E^l[\texttt{raise}\ \varepsilon(V^l)], \langle \sigma_2' \nwarrow \sigma_1' \rangle E^r[\texttt{raise}\ \varepsilon(V^r)]) \\ &\in \mathcal{E}_k^{\sim}[\![\sigma_2']\!]\mathcal{V}^{\sim}[\![c]\!].\end{aligned}$$

  We continue by cases on subtyping of effect precision derivations. We show only the case d_σ' is a concrete effect precision set d_c'; the other cases follow immediately or reduce to this one.

  By inversion, we have d_σ is also a concrete effect precision set d_c where $\mathrm{dom}(d_c) \subseteq \mathrm{dom}(d_c')$ and for all $\varepsilon : c \rightsquigarrow d \in d_c$, $\varepsilon : c' \rightsquigarrow d' \in d_c'$ and $c \leq c'$ and $d' \leq d$. By anti-reduction, it suffices to show

$$\begin{aligned}&(\texttt{let}\ x = \langle d_i^l \twoheadleftarrow d_i^r \rangle \texttt{raise}\ \varepsilon(\langle c_i^r \nwarrow c_i^l \rangle V^l)\ \texttt{in}\ \langle \sigma_2 \nwarrow \sigma_1 \rangle E^l[x], \\ &\ \texttt{let}\ x = \langle d_i'^l \twoheadleftarrow d_i'^r \rangle \texttt{raise}\ \varepsilon(\langle c_i'^r \nwarrow c_i'^l \rangle V^r)\ \texttt{in}\ \langle \sigma_2' \nwarrow \sigma_1' \rangle E^r[x]) \\ &\qquad \in (\blacktriangleright\mathcal{E}_)^{\sim}[\![\sigma_2']\!]_k(\mathcal{V}^{\sim}[\![c]\!]),\end{aligned}$$

  By congruence for Let, it suffices to show (1)

$$\begin{aligned}&(\langle d_i^l \twoheadleftarrow d_i^r \rangle \texttt{raise}\ \varepsilon(\langle c_i^r \nwarrow c_i^l \rangle V^l), \\ &\ \langle d_i'^l \twoheadleftarrow d_i'^r \rangle \texttt{raise}\ \varepsilon(\langle c_i'^r \nwarrow c_i'^l \rangle V^r)) \\ &\qquad \in (\blacktriangleright\mathcal{E}_)^{\sim}[\![\sigma_2']\!]_k(\mathcal{V}^{\sim}[\![c]\!]),\end{aligned}$$

  and (2) for $(V_1, V_2) \in \blacktriangleright(\mathcal{V}^{\sim}[\![d_i]\!])_k$ we have

$$\begin{aligned}&(\langle \sigma_2 \nwarrow \sigma_1 \rangle E^l[V_1], \\ &\ \langle \sigma_2' \nwarrow \sigma_1' \rangle E^r[V_2]) \\ &\qquad \in (\blacktriangleright\mathcal{E}_)^{\sim}[\![\sigma_2']\!]_k(\mathcal{V}^{\sim}[\![c]\!]),\end{aligned}$$

  To show (1), first note that by the induction hypothesis for value types,

$$(\texttt{raise}\ \varepsilon(\langle c_i^r \nwarrow c_i^l \rangle V^l), \texttt{raise}\ \varepsilon(\langle c_i'^r \nwarrow c_i'^l \rangle V^r)) \in \mathcal{E}_k^{\sim}[\![\sigma_2']\!]\mathcal{V}^{\sim}[\![c_i']\!],$$

  and by the induction hypothesis for value types again, (1) follows. To show (2), note that $E^l[x^l]$ and $E^r[x^r]$ are related by assumption on E^l and E^r. So we may apply the Löb induction hypothesis to reach the desired conclusion.

(4) We again use Löb induction and monadic bind. In the related raises case of the bind lemma, we let $\varepsilon : c_i \rightsquigarrow d_i \in \sigma_2$ be an effect caught by $\langle \sigma_2' \nwarrow \sigma_1' \rangle \bullet$. We let $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^\sim[\![c_i^l]\!])_k$, and let $(E^l, E^r) \in (\blacktriangleright \mathcal{K}^\sim[\![d_i^l]\!])_k(\mathcal{E}^\sim[\![\sigma_1']\!]\mathcal{V}^\sim[\![c]\!])$.
We need to show

$$(\langle \sigma_2 \nwarrow \sigma_1 \rangle E^l[\texttt{raise}\ \varepsilon(V^l)], \langle \sigma_2' \nwarrow \sigma_1' \rangle E^r[\texttt{raise}\ \varepsilon(V^r)]) \\ \in \mathcal{E}_k^\sim[\![\sigma_2']\!]\mathcal{V}^\sim[\![c]\!].$$

If $\varepsilon \notin \sigma_1$, then both sides step to $\mho$. Since $\mho$ is related to itself by ErrBot (Lemma D.45), we are finished by anti-reduction.
Otherwise, the proof proceeds analogously to that of the previous case, with upcasts and downcasts interchanged.

□

Lemma D.60 (effect casts commute with pure function values). *Let E be an evaluation context such that (1) for all σ, $\Sigma \mid \Gamma \mid \bullet : (\sigma \,!\, A) \vdash_\sigma E : B$, and such that (2) $E\#\varepsilon$ for all $\varepsilon \in \Sigma$. Furthermore, suppose that (3) for all values V, there exists a value V' such that $E[V] \mapsto^* V'$.*
Let $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma_2} M \equiv N : A$.
Then $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma_1} E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle M] \equiv \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E[N] : B$, and likewise for upcasts.

Proof. We show the statement for downcasts only; the proof for upcasts is similar. Additionally, we show only one of the directions of the equivalence; the other is symmetric.
We need to show

$$(E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle M], \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E[N]) \in \mathcal{E}_j^\sim[\![\sigma_1]\!]\mathcal{V}^\sim[\![B]\!].$$

We apply monadic bind (Lemma D.16) with $E_1 = E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle \bullet]$ and $E_2 = \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E$. By assumption on M and N, will suffice to consider the following cases.

- Let $k \leq j$ and let $(V_1, V_2) \in \mathcal{V}_k^\sim[\![A]\!]$. We need to show

$$(E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle V_1], \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E[V_2]) \in \mathcal{E}_k^\sim[\![\sigma_1]\!]\mathcal{V}^\sim[\![B]\!].$$

  By the operational semantics, we have $E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle V_1] \mapsto^1 E[V_1]$.
  By anti-reduction, it suffices to show

$$(E[V_1], \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E[V_2]) \in \mathcal{E}_k^\sim[\![\sigma_1]\!]\mathcal{V}^\sim[\![B]\!].$$

  Furthermore, there exist i_1 and i_2 and values V_1' and V_2' such that $E[V_1] \mapsto^{i_1} V_1'$ and $E[V_2] \mapsto^{i_2} V_2'$.
  We also have $\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle V_2' \mapsto^1 V_2'$.
  Putting the above facts together, by anti-reduction, it suffices to show

$$(V_1', V_2') \in \mathcal{E}_k^\sim[\![\sigma_1]\!]\mathcal{V}^\sim[\![B]\!].$$

  But by forward reduction, it suffices to show that $(E[V_1], E[V_2]) \in \mathcal{E}_k^\sim[\![\sigma_1]\!]\mathcal{V}^\sim[\![B]\!]$.
  For this, it suffices (by the congruence lemmas) that V_1 and V_2 are related, which is true by assumption.
- Let $k \leq j$ and let $\varepsilon : c^r \rightsquigarrow d^r \in \sigma_2$ be an effect caught by $\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle \bullet$. Let $V^l, V^r, E^l\#\varepsilon, E^r\#\varepsilon$ be as in the statement of Lemma D.16. We need to show

$$(E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E^l[\texttt{raise}\ \varepsilon(V^l)]], \langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E[E^r[\texttt{raise}\ \varepsilon(V^r)]]) \in \mathcal{E}_k^\sim[\![\sigma_1]\!]\mathcal{V}^\sim[\![B]\!].$$

If $\varepsilon \notin \sigma_1$, then, by the operational semantics, both terms will step to $\mho$. By anti-reduction, it suffices to show that $(\mho, \mho) \in \mathcal{E}^{\sim}_{k}[\![\sigma_1]\!]\mathcal{V}^{\sim}[\![B]\!]$. This follows by ErrBot (Lemma D.45).
Now suppose $\varepsilon : c^l \rightsquigarrow d^l \in \sigma_1$. According to the operational semantics, we have

$$E[\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E^l[\texttt{raise}\ \varepsilon(V^l)]] \mapsto^1 E[E^l[\langle d^r \leftarrowtail d^l \rangle \texttt{raise}\ \varepsilon(\langle c^l \twoheadleftarrow c^r \rangle V^l)]],$$

and

$$\langle \sigma_1 \twoheadleftarrow \sigma_2 \rangle E[E^r[\texttt{raise}\ \varepsilon(V^r)]] \mapsto^1 E[E^r[\langle d^r \leftarrowtail d^l \rangle \texttt{raise}\ \varepsilon(\langle c^l \twoheadleftarrow c^r \rangle V^r)]].$$

Thus, by anti-reduction, it suffices to show

$$\begin{aligned}&(E[E^l[\langle d^r \leftarrowtail d^l \rangle \texttt{raise}\ \varepsilon(\langle c^l \twoheadleftarrow c^r \rangle V^l)]],\\ &\ E[E^r[\langle d^r \leftarrowtail d^l \rangle \texttt{raise}\ \varepsilon(\langle c^l \twoheadleftarrow c^r \rangle V^r)]])\\ &\qquad \in \mathcal{E}^{\sim}_{k}[\![\sigma_1]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

Let V'^l be the value to which $\langle c^l \twoheadleftarrow c^r \rangle V^l$ steps, and similarly let V'^r be the value to which $\langle c^l \twoheadleftarrow c^r \rangle V^r$ steps. By anti-reduction, it suffices to show

$$\begin{aligned}&(E[E^l[\langle d^r \leftarrowtail d^l \rangle \texttt{raise}\ \varepsilon(V'^l)]],\\ &\ E[E^r[\langle d^r \leftarrowtail d^l \rangle \texttt{raise}\ \varepsilon(V'^r)]])\\ &\qquad \in \mathcal{E}^{\sim}_{k}[\![\sigma_1]\!]\mathcal{V}^{\sim}[\![B]\!].\end{aligned}$$

As neither term steps, it is sufficient to show that they are related in $\mathcal{R}^{\sim}_{k}[\![\sigma_1]\!]\mathcal{V}^{\sim}[\![B]\!]$. We assert the second disjunct in the definition of $\mathcal{R}^{\sim}[\![\cdot]\!]$, taking $E^l = E[E^l[\langle d^r \leftarrowtail d^l \rangle \bullet]]$ and $E^r = E[E^r[\langle d^r \leftarrowtail d^l \rangle \bullet]]$.
We first need to show that $(V'^l, V'^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![c]\!])_k$. By forward reduction, it suffices to show that

$$(\langle c^l \twoheadleftarrow c^r \rangle V^l, \langle c^l \twoheadleftarrow c^r \rangle V^r) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma_1]\!])_k(\mathcal{V}^{\sim}[\![c^r]\!]).$$

By monotonicity of casts (lemma D.63), it suffices to show $(V^l, V^r) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma_1]\!])_k(\mathcal{V}^{\sim}[\![c^r]\!])$. This follows from our assumption about V^l and V^r.
We now need to show that

$$(x^l.E[E^l[\langle d^r \leftarrowtail d^l \rangle x^l]], x^r.E[E^r[\langle d^r \leftarrowtail d^l \rangle x^r]]) \in (\blacktriangleright \mathcal{K}^{\sim}[\![d]\!])_k(\mathcal{E}^{\sim}[\![\sigma_1]\!]\mathcal{V}^{\sim}[\![B]\!]).$$

To this end, let $k' \leq k$ and let $(V_1, V_2) \in (\blacktriangleright \mathcal{V}^{\sim}_{)}[\![d^l]\!]_{k'}$. We need to show

$$(E[E^l[\langle d^r \leftarrowtail d^l \rangle V_1]], E[E^r[\langle d^r \leftarrowtail d^l \rangle V_2]]) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma_1]\!])_{k'}(\mathcal{V}^{\sim}[\![B]\!]).$$

It will suffice by the soundness of the congruence rules to show that

$$(E^l[\langle d^r \leftarrowtail d^l \rangle V_1], E^r[\langle d^r \leftarrowtail d^l \rangle V_2]) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma_1]\!])_{k'}(\mathcal{V}^{\sim}[\![B]\!]).$$

Let V_1' and V_2' be the values to which $\langle d^r \leftarrowtail d^l \rangle V_1$ and $\langle d^r \leftarrowtail d^l \rangle V_2$ step, respectively. By anti-reduction, it suffices to show

$$(E^l[V_1'], E^r[V_2']) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma_1]\!])_{k'}(\mathcal{V}^{\sim}[\![B]\!]).$$

By assumption on E^l and E^r, it suffices to show that $(V_1', V_2') \in (\blacktriangleright\mathcal{V}^\sim[\![d^r]\!])_{k'}$. By forward reduction, it suffices to show

$$(\langle d^r \leftarrowtail d^l\rangle V_1, \langle d^r \leftarrowtail d^l\rangle V_2) \in (\blacktriangleright\mathcal{E}^\sim[\![\sigma_1]\!])_{k'}(\mathcal{V}^\sim[\![B]\!]).$$

By monotonicity of casts (lemma D.63), it suffices to show

$$(V_1, V_2) \in (\blacktriangleright\mathcal{E}^\sim[\![\sigma_1]\!])_{k'}(\mathcal{V}^\sim[\![B]\!]).$$

This follows from our assumption on V_1 and V_2.

□

Corollary D.61 (commutativity of casts). *Value casts commute with effect casts.*

Proof. This follows from D.60, because $\langle B \leftarrowtail A\rangle\bullet$ and $\langle A \twoheadleftarrow B\rangle\bullet$ satisfy the requirements in the lemma. □

Lemma D.62 (functoriality of casts). *Let M be a term such that $\Sigma \mid \Gamma \mid \cdot \vdash_\sigma M : A$. Let $c : A \sqsubseteq B$ and $e : B \sqsubseteq C$. Let $d_\sigma : \sigma \sqsubseteq \sigma'$ and let $d'_\sigma : \sigma' \sqsubseteq \sigma''$*

Suppose $\Sigma \mid \Gamma^\sqsubseteq \vDash_\sigma M \equiv N : A$. Then the following hold:

Identity properties: *Suppose $\Sigma \mid \Gamma^\sqsubseteq \vDash_\sigma M \sqsupseteq\sqsubseteq N : A$. We have*

(1) $\Sigma \mid \Gamma \vDash_\sigma \langle A \leftarrowtail A\rangle M \equiv N : A$
(2) $\Sigma \mid \Gamma \vDash_\sigma \langle A \twoheadleftarrow A\rangle M \equiv N : A$
(3) $\Sigma \mid \Gamma \vDash_\sigma \langle \sigma \leftarrowtail \sigma\rangle M \equiv N : A$
(4) $\Sigma \mid \Gamma \vDash_\sigma \langle \sigma \twoheadleftarrow \sigma\rangle M \equiv N : A$

Composition properties: *Let $c : A \sqsubseteq B$ and $e : B \sqsubseteq C$. Let $d_\sigma : \sigma \sqsubseteq \sigma'$ and $d'_\sigma : \sigma' \sqsubseteq \sigma''$. Suppose $M \sqsupseteq\sqsubseteq N$. Then*

(1) $\Sigma \mid \Gamma \vDash_\sigma \langle C \leftarrowtail A\rangle M \sqsupseteq\sqsubseteq \langle C \leftarrowtail B\rangle\langle B \leftarrowtail A\rangle N : C$
(2) $\Sigma \mid \Gamma \vDash_\sigma \langle A \twoheadleftarrow C\rangle M \sqsupseteq\sqsubseteq \langle A \twoheadleftarrow B\rangle\langle B \twoheadleftarrow C\rangle N : A$
(3) $\Sigma \mid \Gamma \vDash_{\sigma''} \langle \sigma'' \leftarrowtail \sigma\rangle M \sqsupseteq\sqsubseteq \langle \sigma'' \leftarrowtail \sigma'\rangle\langle \sigma' \leftarrowtail \sigma\rangle N : A$
(4) $\Sigma \mid \Gamma \vDash_\sigma \langle \sigma \twoheadleftarrow \sigma''\rangle M \sqsupseteq\sqsubseteq \langle \sigma \twoheadleftarrow \sigma'\rangle\langle \sigma' \twoheadleftarrow \sigma''\rangle N : A$

Proof. We prove more general, "pointwise" versions of the above statements. For instance, we show that if $(M, N) \in \mathcal{E}_j^\sim[\![\sigma]\!]\mathcal{V}^\sim[\![A]\!]$, then $(\langle A \leftarrowtail A\rangle M, N) \in \mathcal{E}_j^\sim[\![\sigma]\!]\mathcal{V}^\sim[\![A]\!]$.

Additionally, we only prove one direction of each of the equivalences (i.e., $\sqsubseteq$); the proof of the other direction is symmetric.

The statements are proven simultaneously by induction on A and σ.

- Identity properties:

(1) We need to show $(\langle A \leftarrowtail A\rangle M, N) \in \mathcal{E}_j^\sim[\![\sigma]\!]\mathcal{V}^\sim[\![A]\!]$. By monadic bind (Lemma D.16), with $E_1 = \langle A \leftarrowtail A\rangle\bullet$ and $E_2 = \bullet$, it will suffice to show the following: Let $k \leq j$ and $(V_1, V_2) \in \mathcal{V}_k^\sim[\![A]\!]$. We will show

$$(\langle A \leftarrowtail A\rangle V_1, V_2) \in \mathcal{E}_k^\sim[\![\sigma]\!]\mathcal{V}^\sim[\![A]\!].$$

We continue by induction on A. If $A = \texttt{bool}$, then we need to show

$$(\langle \texttt{bool} \leftarrowtail \texttt{bool}\rangle V_1, V_2) \in \mathcal{E}_k^\sim[\![d_\sigma]\!]\mathcal{V}^\sim[\![\texttt{bool}]\!].$$

By anti-reduction, it suffices to show $(V_1, V_2) \in \mathcal{E}_k^\sim[\![\sigma]\!]\mathcal{V}^\sim[\![\texttt{bool}]\!]$, which follows from our assumption on (V_1, V_2).

If $A = A_i \to_{\sigma_A} A_o$, we need to show

$$(\langle (A_i \to_{\sigma_A} A_o) \leftsquigarrow (A_i \to_{\sigma_A} A_o) \rangle V_1, V_2) \in \mathcal{E}^{\sim}_k [\![\sigma]\!] \mathcal{V}^{\sim} [\![A_i \to_{\sigma_A} A_o]\!].$$

As both terms are values, it suffices to show they are related in $\mathcal{V}^{\sim}_k [\![A_i \to_{\sigma_A} A_o]\!]$. So, let $k' \leq k$ and let $(V^l, V^r) \in \mathcal{V}^{\sim}_{k'} [\![A_i]\!]$. We need to show

$$((\langle (A_i \to_{\sigma_A} A_o) \leftsquigarrow (A_i \to_{\sigma_A} A_o) \rangle V_1)\, V^l, V_2\, V^r) \in \mathcal{E}^{\sim}_{k'} [\![\sigma_A]\!] \mathcal{V}^{\sim} [\![A_o]\!].$$

By anti-reduction, it suffices to show

$$(\langle A_o \leftsquigarrow A_o \rangle \langle \sigma_A \leftsquigarrow \sigma_A \rangle (V_1\, \langle A_i \twoheadleftarrow A_i \rangle V^l), V_2\, V^r) \in \mathcal{E}^{\sim}_{k'} [\![\sigma_A]\!] \mathcal{V}^{\sim} [\![A_o]\!].$$

By the induction hypothesis (applied twice), it suffices to show

$$((V_1\, \langle A_i \twoheadleftarrow A_i \rangle V^l), V_2\, V^r) \in \mathcal{E}^{\sim}_{k'} [\![\sigma_A]\!] \mathcal{V}^{\sim} [\![A_o]\!].$$

By the soundness of function application, it suffices to show that $(V_1, V_2) \in \mathcal{V}^{\sim}_{k'} [\![A_i \to_{\sigma_A} A_o]\!]$ and $(\langle A_i \twoheadleftarrow A_i \rangle V^l, V^r) \in \mathcal{E}^{\sim}_{k'} [\![\sigma_A]\!] \mathcal{V}^{\sim} [\![A_i]\!]$. The former is true by assumption and downward closure ($k' \leq k$). The latter is true by inductive hypothesis, since V^l and V^r are related.

(2) This is dual to the above.

(3) We prove this statement by Löb induction (Lemma D.14). That is, assume for all $(M', N') \in (\blacktriangleright \mathcal{E}^{\sim} [\![\sigma]\!])_j (\mathcal{V}^{\sim} [\![A]\!])$, we have $(\langle \sigma \leftsquigarrow \sigma \rangle M', N') \in (\blacktriangleright \mathcal{E}^{\sim} [\![\sigma]\!])_j (\mathcal{V}^{\sim} [\![A]\!])$. Let $(M, N) \in \mathcal{E}^{\sim}_j [\![\sigma]\!] \mathcal{V}^{\sim} [\![A]\!]$. We need to show $(\langle \sigma \leftsquigarrow \sigma \rangle M, N) \in \mathcal{E}^{\sim}_j [\![\sigma]\!] \mathcal{V}^{\sim} [\![A]\!]$. By monadic bind (Lemma D.16), with $E_1 = \langle \sigma \leftsquigarrow \sigma \rangle \bullet$ and $E_2 = \bullet$, it will suffice to consider the following cases.

- Let $k \leq j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k [\![A]\!]$. We need to show

$$(\langle \sigma \leftsquigarrow \sigma \rangle V_1, V_2) \in \mathcal{E}^{\sim}_k [\![\sigma]\!] \mathcal{V}^{\sim} [\![c]\!].$$

  Per the operational semantics, we have $\langle \sigma \leftsquigarrow \sigma \rangle V_1 \mapsto^1 V_1$, so by anti-reduction it suffices to show $(V_1, V_2) \in \mathcal{E}^{\sim}_k [\![\sigma]\!] \mathcal{V}^{\sim} [\![A]\!]$, which follows by the assumption that $(V_1, V_2) \in \mathcal{V}^{\sim}_k [\![A]\!]$.
- Let $k \leq j$ and let $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon$ be an effect caught by $\langle \sigma \leftsquigarrow \sigma \rangle \bullet$ – i.e., $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in \sigma$. Note that, as σ is a reflexivity derivation, c_ε and d_ε are also reflexivity derivations, i.e., $c^l_\varepsilon = c^r_\varepsilon$ and likewise for d_ε. For simplicity, let $C = c^l_\varepsilon$ and $D = d^l_\varepsilon$.
  Let $V^l, V^r, E^l \# \varepsilon, E^r \# \varepsilon$ be as in the statement of Lemma D.16. We need to show

$$\begin{aligned}&(\langle \sigma \leftsquigarrow \sigma \rangle E^l[\mathtt{raise}\ \varepsilon(V^l)], E^r[\mathtt{raise}\ \varepsilon(V^r)])\\ &\quad \in \mathcal{E}^{\sim}_k [\![\sigma]\!] \mathcal{V}^{\sim} [\![A]\!].\end{aligned}$$

  According to the operational semantics, we have

$$\begin{aligned}&\langle \sigma \leftsquigarrow \sigma \rangle E^l[\mathtt{raise}\ \varepsilon(V^l)] \mapsto^1\\ &\qquad \mathtt{let}\ x = \langle D \twoheadleftarrow D \rangle \mathtt{raise}\ \varepsilon(\langle C \leftsquigarrow C \rangle V^l)\ \mathtt{in}\ \langle \sigma \leftsquigarrow \sigma \rangle E^l[x]\end{aligned}$$

  So, by anti-reduction it suffices to show that

$$(\texttt{let}\ x = \langle D \twoheadleftarrow D\rangle \texttt{raise}\ \varepsilon(\langle C \nwarrow C\rangle V^l)\ \texttt{in}\ \langle \sigma \nwarrow \sigma\rangle E^l[x],\\ E^r[\texttt{raise}\ \varepsilon(V^r)])\\ \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

Let V'^l be the term to which $\langle C \nwarrow C\rangle V^l$ steps. By anti-reduction, it suffices to show that

$$(\texttt{let}\ x = \langle D \twoheadleftarrow D\rangle \texttt{raise}\ \varepsilon(V'^l)\ \texttt{in}\ \langle \sigma \nwarrow \sigma\rangle E^l[x],\\ E^r[\texttt{raise}\ \varepsilon(V^r)])\\ \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

The above terms do not step, so it suffices to show that they are related in $\mathcal{R}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. To this end, we will first show that $(V'^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![C]\!])_k$. By forward reduction, it suffices to show that $(\langle C \nwarrow C\rangle V^l, V^r) \in (\blacktriangleright\mathcal{E}^{\sim}[\![\mathcal{V}^{\sim}[\![C]\!]]\!])_k$. By the induction hypothesis, it suffices to show that $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![C]\!])_(k)$.
Now we will show that

$$(x^l.(\texttt{let}\ x = \langle D \twoheadleftarrow D\rangle x^l\ \texttt{in}\ \langle \sigma \nwarrow \sigma\rangle E^l[x]),\ x^r.E^r[x^r])\\ \in (\blacktriangleright\mathcal{K}^{\sim}[\![D]\!])_k(\mathcal{E}^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]).$$

Let $k' \leq k$ and let $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![A]\!])_{k'}$. We need to show

$$((\texttt{let}\ x = \langle D \twoheadleftarrow D\rangle V_1\ \texttt{in}\ \langle \sigma \nwarrow \sigma\rangle E^l[x]),\ E^r[V_2])\\ \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]).$$

Let V_1' be the value to which $\langle D \twoheadleftarrow D\rangle V_1$ steps. By anti-reduction, it suffices to show

$$((\texttt{let}\ x = V_1'\ \texttt{in}\ \langle \sigma \nwarrow \sigma\rangle E^l[x]),\ E^r[V_2])\\ \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]),$$

and then since the let term steps, it suffices by anti-reduction again to show

$$(\langle \sigma \nwarrow \sigma\rangle E^l[V_1'],\ E^r[V_2]) \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]),$$

By the Löb induction hypothesis, it suffices to show that

$$(E^l[V_1'], E^r[V_2]) \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!])$$

By our assumption on E^l and E^r, it suffices to show

$$(V_1', V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![A]\!])_{k'}.$$

By forward reduction, it suffices to show

$$(\langle D \twoheadleftarrow D\rangle V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\sim}[\![A]\!])_{k'}.$$

By the induction hypothesis for value types, it suffices to show

$$(V_1, V_2) \in (\blacktriangleright\mathcal{E}^{\sim}[\![A]\!])_{k'}.$$

This follows by assumption.

(4) We again use Löb induction and monadic bind.
That is, assume for all $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_j(\mathcal{V}^{\sim}[\![A]\!])$, we have $(\langle\sigma \twoheadleftarrow \sigma\rangle M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_j(\mathcal{V}^{\sim}[\![A]\!])$. We need to show

$$(\langle\sigma \twoheadleftarrow \sigma\rangle M, N) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$$

where $(M, N) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. We again use monadic bind, and as in the previous proof, the case of related values follows trivially since effect casts are the identity on values. Thus, it will suffice to show the related raises case. That is, let $k \le j$ and let $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon$ be an effect caught by $\langle\sigma \twoheadleftarrow \sigma\rangle\bullet$ – i.e., $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in \sigma$. As in the previous proof, since σ is a reflexivity derivation, c_ε and d_ε are also reflexivity derivations, so for simplicity, let $C = c^l_\varepsilon = c^r_\varepsilon$ and $D = d^l_\varepsilon = d^r_\varepsilon$.
Let $V^l, V^r, E^l\#\varepsilon, E^r\#\varepsilon$ be as in the statement of the monadic bind lemma. We need to show

$$\begin{aligned}(\langle\sigma \twoheadleftarrow \sigma\rangle E^l[\mathtt{raise}\ \varepsilon(V^l)], E^r[\mathtt{raise}\ \varepsilon(V^r)])\\ \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

Note that, since $\varepsilon \in \sigma$, the downcast cannot fail.
The remainder of the proof proceeds exactly like the previous proof, with upcasts and downcasts interchanged.

- Composition properties:

(1) We need to show $(\langle C \leftarrowtail A\rangle M, \langle C \leftarrowtail B\rangle\langle B \leftarrowtail A\rangle N) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![C]\!]$.
By monadic bind (Lemma D.16) with $E_1 = \langle C \leftarrowtail A\rangle\bullet$ and $E_2 = \langle C \leftarrowtail B\rangle\langle B \leftarrowtail A\rangle\bullet$, it will suffice to show the following: Let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![A]\!]$. We will show

$$(\langle C \leftarrowtail A\rangle V_1, \langle C \leftarrowtail B\rangle\langle B \leftarrowtail A\rangle V_2) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![C]\!].$$

If $c \circ e = \mathtt{bool}$, then $c = e = \mathtt{bool}$, and we need to show

$$(\langle\mathtt{bool} \leftarrowtail \mathtt{bool}\rangle V_1, \langle\mathtt{bool} \leftarrowtail \mathtt{bool}\rangle\langle\mathtt{bool} \leftarrowtail \mathtt{bool}\rangle V_2) \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![\mathtt{bool}]\!].$$

By anti-reduction, it suffices to show $(V_1, V_2) \in \mathcal{V}^{\sim}_j[\![\mathtt{bool}]\!]$, which follows from our assumption.
Now suppose $c \circ e = (c_i \circ e_i) \rightarrow_{(c_\sigma \circ e_\sigma)} (c_o \circ e_o)$. We need to show

$$\begin{aligned}&(\langle(C_i \rightarrow_{\sigma_C} C_o) \leftarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_1,\\ &\ \langle(C_i \rightarrow_{\sigma_C} C_o) \leftarrowtail (B_i \rightarrow_{\sigma_B} B_o)\rangle\langle(B_i \rightarrow_{\sigma_B} B_o) \leftarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_2)\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![C_i \rightarrow_{\sigma_C} C_o]\!].\end{aligned}$$

Both terms are values, so it suffices to show that they are related in $\mathcal{V}^{\sim}_k[\![C_i \rightarrow_{\sigma_C} C_o]\!]$. Let $k' \le k$ and let $(V^l, V^r) \in \mathcal{V}^{\sim}_{k'}[\![C_i]\!]$. We need to show that

$$\begin{aligned}&((\langle(C_i \rightarrow_{\sigma_C} C_o) \leftarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_1)\ V^l,\\ &\ (\langle(C_i \rightarrow_{\sigma_C} C_o) \leftarrowtail (B_i \rightarrow_{\sigma_B} B_o)\rangle\langle(B_i \rightarrow_{\sigma_B} B_o) \leftarrowtail (A_i \rightarrow_{\sigma_A} A_o)\rangle V_2)\ V^r)\\ &\qquad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].\end{aligned}$$

By anti-reduction, it suffices to show

$$
\begin{aligned}
&(\langle C_o \nwarrow A_o\rangle\langle \sigma_C \nwarrow \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow C_i\rangle V^l),\\
&\ \langle C_o \nwarrow B_o\rangle\langle \sigma_C \nwarrow \sigma_B\rangle\\
&\qquad(((\langle (B_i \to_{\sigma_B} B_o) \nwarrow (A_i \to_{\sigma_A} A_o)\rangle V_2)\,\langle B_i \twoheadleftarrow C_i\rangle V^r))\\
&\qquad\quad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].
\end{aligned}
$$

Let V'^r be the value to which $\langle B_i \twoheadleftarrow C_i\rangle V^r$ steps. By anti-reduction, it suffices to show

$$
\begin{aligned}
&(\langle C_o \nwarrow A_o\rangle\langle \sigma_C \nwarrow \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow C_i\rangle V^l),\\
&\ \langle C_o \nwarrow B_o\rangle\langle \sigma_C \nwarrow \sigma_B\rangle\\
&\qquad(((\langle (B_i \to_{\sigma_B} B_o) \nwarrow (A_i \to_{\sigma_A} A_o)\rangle V_2)\,V'^r))\\
&\qquad\quad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].
\end{aligned}
$$

By anti-reduction again, it suffices to show

$$
\begin{aligned}
&(\langle C_o \nwarrow A_o\rangle\langle \sigma_C \nwarrow \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow C_i\rangle V^l),\\
&\ \langle C_o \nwarrow B_o\rangle\langle \sigma_C \nwarrow \sigma_B\rangle\\
&\qquad((\langle B_o \nwarrow A_o\rangle\langle \sigma_B \nwarrow \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)))\\
&\qquad\quad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].
\end{aligned}
$$

We will appeal to transitivity (Lemma D.64). We continue by cases on $\sim$.

– First suppose $\sim\,=\,<$. We first claim that

$$
\begin{aligned}
&(\langle C_o \nwarrow A_o\rangle\langle \sigma_C \nwarrow \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow C_i\rangle V^l),\\
&\ \langle C_o \nwarrow B_o\rangle\langle B_o \nwarrow A_o\rangle\\
&\qquad((\langle \sigma_C \nwarrow \sigma_B\rangle\langle \sigma_B \nwarrow \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)))\\
&\qquad\quad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].
\end{aligned}
$$

By the induction hypothesis applied twice, it suffices to show

$$
\begin{aligned}
&((V_1\,\langle A_i \twoheadleftarrow C_i\rangle V^l), (V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r))\\
&\qquad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].
\end{aligned}
$$

By soundness of function application, it suffices to show that $(V_1, V_2) \in \mathcal{V}^{\sim}_{k'}[\![A_i \to_{\sigma_A} A_o]\!]$ and that

$$(\langle A_i \twoheadleftarrow C_i\rangle V^l, \langle A_i \twoheadleftarrow B_i\rangle V'^r) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i]\!].$$

The former holds by assumption and downward closure. To show the latter, it suffices by forward reduction to show that

$$(\langle A_i \twoheadleftarrow C_i\rangle V^l, \langle A_i \twoheadleftarrow B_i\rangle\langle B_i \twoheadleftarrow C_i\rangle V^r) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i]\!].$$

Now, by the induction hypothesis, it suffices to show that

$$(V^l, V^r) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![C_i]\!],$$

which follows from our assumption.

Now by transitivity, it will suffice to show

$$\begin{aligned}&(\langle C_o \leftarrowtail B_o\rangle\langle B_o \leftarrowtail A_o\rangle\\&\qquad(\langle \sigma_C \leftarrowtail \sigma_B\rangle\langle \sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)),\\&\ \langle C_o \leftarrowtail B_o\rangle\langle \sigma_C \leftarrowtail \sigma_B\rangle\\&\qquad(\langle B_o \leftarrowtail A_o\rangle\langle \sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)))\\&\qquad\qquad\in \mathcal{E}^{\gtrsim}_{\omega}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].\end{aligned}$$

By reflexivity (Corollary D.28), we have that $\langle B_o \leftarrowtail A_o\rangle\langle \sigma_C \leftarrowtail \sigma_B\rangle\langle \sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)$ is related to itself. Then by commutativity of casts (Corollary D.61), we can interchange the order of $\langle B_o \leftarrowtail A_o\rangle$ and $\langle \sigma_C \leftarrowtail \sigma_B\rangle$, and the resulting terms are related. Finally by monotonicity of casts (Lemma D.63), we can apply $\langle C_o \leftarrowtail B_o\rangle$, and the resulting terms are still related. Moreover, all of these relations hold "at ω".

– Now suppose $\sim = >$. By similar reasoning as in the previous case, we have

$$\begin{aligned}&(\langle C_o \leftarrowtail A_o\rangle\langle \sigma_C \leftarrowtail \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow C_i\rangle V^l),\\&\ \langle C_o \leftarrowtail B_o\rangle\langle B_o \leftarrowtail A_o\rangle\\&\qquad(\langle \sigma_C \leftarrowtail \sigma_B\rangle\langle \sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V^l)))\\&\qquad\qquad\in \mathcal{E}^{\sim}_{\omega}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].\end{aligned}$$

Thus, by transitivity it will suffice to show

$$\begin{aligned}&(\langle C_o \leftarrowtail B_o\rangle\langle B_o \leftarrowtail A_o\rangle\\&\qquad(\langle \sigma_C \leftarrowtail \sigma_B\rangle\langle \sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V^l)),\\&\ \langle C_o \leftarrowtail B_o\rangle\langle \sigma_C \leftarrowtail \sigma_B\rangle\\&\qquad(\langle B_o \leftarrowtail A_o\rangle\langle \sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V'^r)))\\&\qquad\qquad\in \mathcal{E}^{\gtrsim}_{k'}[\![\sigma_C]\!]\mathcal{V}^{\sim}[\![C_o]\!].\end{aligned}$$

The reasoning is analogous to that of the previous case.

(2) This is dual to the above.

(3) We prove this statement by Löb induction (Lemma D.14). That is, assume for all $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_j(\mathcal{V}^{\sim}[\![A]\!])$, we have

$$(\langle \sigma'' \leftarrowtail \sigma\rangle M, \langle \sigma'' \leftarrowtail \sigma'\rangle\langle \sigma' \leftarrowtail \sigma\rangle N) \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma'']\!])_j(\mathcal{V}^{\sim}[\![A]\!]).$$

Let $(M, N) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$. We need to show

$$(\langle \sigma'' \leftarrowtail \sigma\rangle M, \langle \sigma'' \leftarrowtail \sigma'\rangle\langle \sigma' \leftarrowtail \sigma\rangle N) \in \mathcal{E}^{\sim}_j[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].$$

By monadic bind (Lemma D.16), with $E_1 = \langle \sigma'' \leftarrowtail \sigma\rangle\bullet$ and $E_2 = \langle \sigma'' \leftarrowtail \sigma'\rangle\langle \sigma' \leftarrowtail \sigma\rangle\bullet$, it suffices to consider the following cases:

– Let $k \leq j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![A]\!]$. We need to show that

$$(\langle \sigma'' \leftarrowtail \sigma\rangle V_1, \langle \sigma'' \leftarrowtail \sigma'\rangle\langle \sigma' \leftarrowtail \sigma\rangle V_2) \in \mathcal{E}^{\sim}_j[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].$$

Since the effect cast is the identity on values, the above follows immediately by anti-reduction.

– Let $k \leq j$ and let $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in \sigma$ be an effect caught by either E_1 or E_2. Note that, as σ is a reflexivity derivation, c_ε and d_ε are also reflexivity derivations, i.e., $c^l_\varepsilon = c^r_\varepsilon$ and likewise for d_ε. For simplicity, let $C^L = c^l_\varepsilon$ and $D^L = d^l_\varepsilon$.
Let $V^l, V^r, E^l \# \varepsilon, E^r \# \varepsilon$ be as in the statement of the monadic bind lemma. We need to show

$$\begin{aligned}&(\langle \sigma'' \nwarrow \sigma\rangle E^l[\texttt{raise } \varepsilon(V^l)],\\ &\ \langle \sigma'' \nwarrow \sigma'\rangle\langle \sigma' \nwarrow \sigma\rangle E^r[\texttt{raise } \varepsilon(V^r)])\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

Let C^M and D^M be the types such that $\varepsilon : C^M \rightsquigarrow D^M \in \sigma'$ Let C^R and D^R be the types such that $\varepsilon : C^R \rightsquigarrow D^R \in \sigma''$. By anti-reduction, it suffices to show

$$\begin{aligned}&(\texttt{let } x = \langle D^L \swarrow D^R\rangle\texttt{raise } \varepsilon(\langle C^R \nwarrow C^L\rangle V^l)\texttt{ in } \langle \sigma'' \nwarrow \sigma\rangle E^l[x],\\ &\ \langle \sigma'' \nwarrow \sigma'\rangle(\texttt{let } x = \langle D^L \swarrow D^M\rangle\texttt{raise } \varepsilon(\langle C^M \nwarrow C^L\rangle V^r)\texttt{ in } \langle \sigma' \nwarrow \sigma\rangle E^r[x]))\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

Let V'^l be the value to which $\langle C^R \nwarrow C^L\rangle V^l$ steps, say in i steps. Let V'^r be the value to which $\langle C^M \nwarrow C^L\rangle V^r$ steps, say in j steps.
By anti-reduction, it suffices to show

$$\begin{aligned}&(\texttt{let } x = \langle D^L \swarrow D^R\rangle\texttt{raise } \varepsilon(V'^l)\texttt{ in } \langle \sigma'' \nwarrow \sigma\rangle E^l[x],\\ &\ \langle \sigma'' \nwarrow \sigma'\rangle(\texttt{let } x = \langle D^L \swarrow D^M\rangle\texttt{raise } \varepsilon(V'^r)\texttt{ in } \langle \sigma' \nwarrow \sigma\rangle E^r[x]))\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

Now (taking $E' = \texttt{let } x = \langle D^L \swarrow D^M\rangle \bullet \texttt{ in } \langle \sigma' \nwarrow \sigma\rangle E^r[x]$ in the EffUpCast rule), it will suffice by anti-reduction to show

$$\begin{aligned}&(\texttt{let } x = \langle D^L \swarrow D^R\rangle\texttt{raise } \varepsilon(V'^l)\texttt{ in } \langle \sigma'' \nwarrow \sigma\rangle E^l[x],\\ &\ \texttt{let } y = \langle D^M \swarrow D^R\rangle\texttt{raise } \varepsilon(\langle C^R \nwarrow C^M\rangle V'^r)\texttt{ in}\\ &\quad \langle \sigma'' \nwarrow \sigma'\rangle(\texttt{let } x = \langle D^L \swarrow D^M\rangle y\texttt{ in } \langle \sigma' \nwarrow \sigma\rangle E^r[x]))\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

Let V''^r be the value to which $\langle C^R \nwarrow C^M\rangle V'^r$ steps. By anti-reduction, it suffices to show

$$\begin{aligned}&(\texttt{let } x = \langle D^L \swarrow D^R\rangle\texttt{raise } \varepsilon(V'^l)\texttt{ in } \langle \sigma'' \nwarrow \sigma\rangle E^l[x],\\ &\ \texttt{let } y = \langle D^M \swarrow D^R\rangle\texttt{raise } \varepsilon(V''^r)\texttt{ in}\\ &\quad \langle \sigma'' \nwarrow \sigma'\rangle(\texttt{let } x = \langle D^L \swarrow D^M\rangle y\texttt{ in } \langle \sigma' \nwarrow \sigma\rangle E^r[x]))\\ &\qquad \in \mathcal{E}^{\sim}_k[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

As neither term steps, we will show that they belong to $\mathcal{R}^{\sim}_k[\![\sigma'']\!]\mathcal{V}^{\sim}[\![A]\!]$. We first need to show that

$$(V'^{l}, V''^{r}) \in (\blacktriangleright \mathcal{V}^{\sim}[\![A]\!])_k.$$

By forward-reduction, it suffices to show that

$$(\langle C^R \nwarrow C^L \rangle V^l, \langle C^R \nwarrow C^M \rangle \langle C^M \nwarrow C^L \rangle V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![A]\!])_k.$$

By the induction hypothesis for value types, it suffices to show that $(V^l, V^r) \in (\blacktriangleright \mathcal{V}^{\sim}[\![A]\!])_k$, which is true by assumption.

Now we need to show that, for all $k' \leq k$ and related values $(V_1, V_2) \in (\blacktriangleright \mathcal{V}^{\sim}[\![A]\!])_{k'}$, we have

$$\begin{aligned}
&(\texttt{let}\ x = \langle D^L \twoheadleftarrow D^R \rangle V_1\ \texttt{in}\ \langle \sigma'' \nwarrow \sigma \rangle E^l[x],\\
&\ \texttt{let}\ y = \langle D^M \twoheadleftarrow D^R \rangle V_2\ \texttt{in}\\
&\quad \langle \sigma'' \nwarrow \sigma' \rangle (\texttt{let}\ x = \langle D^L \twoheadleftarrow D^M \rangle y\ \texttt{in}\ \langle \sigma' \nwarrow \sigma \rangle E^r[x]))\\
&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma'']\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]).
\end{aligned}$$

Let V_1' and V_2' be the values to which $\langle D^L \twoheadleftarrow D^R \rangle V_1$ and $\langle D^M \twoheadleftarrow D^R \rangle V_2$ step, respectively. By anti-reduction, it will suffice to show

$$\begin{aligned}
&(\langle \sigma'' \nwarrow \sigma \rangle E^l[V_1'],\\
&\ \langle \sigma'' \nwarrow \sigma' \rangle (\texttt{let}\ x = \langle D^L \twoheadleftarrow D^M \rangle V_2'\ \texttt{in}\ \langle \sigma' \nwarrow \sigma \rangle E^r[x]))\\
&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma'']\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]).
\end{aligned}$$

Let V_2'' be the value to which $\langle D^L \twoheadleftarrow D^M \rangle V_2'$ steps. By anti-reduction, it will suffice to show

$$\begin{aligned}
&(\langle \sigma'' \nwarrow \sigma \rangle E^l[V_1'],\\
&\ \langle \sigma'' \nwarrow \sigma' \rangle (\langle \sigma' \nwarrow \sigma \rangle E^r[V_2'']))\\
&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma'']\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]).
\end{aligned}$$

Now by the Löb induction hypothesis, it suffices to show

$$(E^l[V_1'], E^r[V_2'']) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma'']\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]).$$

By assumption on E^l and E^r, it suffices to show

$$(V_1', V_2'') \in (\blacktriangleright \mathcal{V}^{\sim}[\![A]\!])_{k'}.$$

Now by forward reduction it suffices to show

$$(\langle D^L \twoheadleftarrow D^R \rangle V_1, \langle D^L \twoheadleftarrow D^M \rangle \langle D^M \twoheadleftarrow D^R \rangle V_2) \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma'']\!])_{k'}(\mathcal{V}^{\sim}[\![A]\!]).$$

This follows by the inductive hypothesis for value types and our assumption on V_1 and V_2.

(4) This is dual to the above: we use Löb induction and monadic bind, and we reach a point where we need to show

$$\begin{aligned}&(\langle \sigma \twoheadleftarrow \sigma'' \rangle E^l[\texttt{raise}\ \varepsilon(V^l)],\\ &\ \langle \sigma' \twoheadleftarrow \sigma'' \rangle \langle \sigma \twoheadleftarrow \sigma' \rangle E^r[\texttt{raise}\ \varepsilon(V^r)])\\ &\qquad \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].\end{aligned}$$

where $\varepsilon : C^R \rightsquigarrow D^R \in \sigma''$.
If $\varepsilon \notin \sigma$, then the left-hand side steps to $\mho$, as does the right-hand side. By ErrBot (Lemma D.45), $\mho$ is related to itself, so by anti-reduction, we are finished. If $\varepsilon \notin \sigma'$, then in fact, $\varepsilon \notin \sigma$ (since $\sigma \sqsubseteq \sigma'$), and so again, both sides step to $\mho$.
Otherwise, we proceed as in the proof of the previous case, with the upcasts and downcasts interchanged.

□

Lemma D.63 (monotonicity of casts). *Let $c : A \sqsubseteq B$, and $d_\sigma : \sigma \sqsubseteq \sigma'$, and let M and N be terms such that $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma M \sqsubseteq N : A$. The following hold:*

(1) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma \langle B \leftarrowtail A \rangle M \sqsubseteq \langle B \leftarrowtail A \rangle N : B$
(2) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma \langle A \twoheadleftarrow B \rangle M \sqsubseteq \langle A \twoheadleftarrow B \rangle N : A$
(3) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_{\sigma'} \langle \sigma' \leftarrowtail \sigma \rangle M \sqsubseteq \langle \sigma' \leftarrowtail \sigma \rangle N : A$
(4) $\Sigma \mid \Gamma^{\sqsubseteq} \vDash_\sigma \langle \sigma \twoheadleftarrow \sigma' \rangle M \sqsubseteq \langle \sigma \twoheadleftarrow \sigma' \rangle N : A$

Proof. As in the proof of the functoriality properties of casts, we prove stronger, "pointwise" versions of the above statements, i.e., we assume $(M, N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]$, and show, for example, that $(\langle B \leftarrowtail A \rangle M, \langle B \leftarrowtail A \rangle N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!]$.
The proof is by induction on c and d_σ.

(1) We need to show

$$(\langle B \leftarrowtail A \rangle M, \langle B \leftarrowtail A \rangle N) \in \mathcal{E}_j^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!].$$

By monadic bind (Lemma D.16), with $E_1 = E_2 = \langle B \leftarrowtail A \rangle \bullet$, it will suffice to show that

$$(\langle B \leftarrowtail A \rangle V_1, \langle B \leftarrowtail A \rangle V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B]\!],$$

where $k \le j$ and let $(V_1, V_2) \in \mathcal{V}_k^{\sim}[\![A]\!]$.
If $c = \texttt{bool}$, then we need to show

$$(\langle \texttt{bool} \leftarrowtail \texttt{bool} \rangle V_1, \langle \texttt{bool} \leftarrowtail \texttt{bool} \rangle V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![\texttt{bool}]\!].$$

By anti-reduction, it suffices to show that $(V_1, V_2) \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![\texttt{bool}]\!]$, which follows from our assumption.
If $c = c_i \to_{c_\sigma} c_o$, then we need to show

$$\begin{aligned}&(\langle (B_i \to_{\sigma_B} B_o) \leftarrowtail (A_i \to_{\sigma_A} A_o) \rangle V_1, \langle (B_i \to_{\sigma_B} B_o) \leftarrowtail (A_i \to_{\sigma_A} A_o) \rangle V_2)\\ &\qquad \in \mathcal{E}_k^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B_i \to_{\sigma_B} B_o]\!].\end{aligned}$$

As both terms are values, it suffices to show that they are related in $\mathcal{V}_k^{\sim}[\![B_i \to_{\sigma_B} B_o]\!]$. Let $k' \le k$ and let $(V^l, V^r) \in \mathcal{K}_{k'}^{\sim}[\![B_i]\!]$. We need to show

$$
\begin{aligned}
&(((\langle(B_i \to_{\sigma_B} B_o) \leftarrowtail (A_i \to_{\sigma_A} A_o)\rangle V_1)\, V^l,\\
&\ ((\langle(B_i \to_{\sigma_B} B_o) \leftarrowtail (A_i \to_{\sigma_A} A_o)\rangle V_2)\, V^r)\\
&\quad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_B]\!]\mathcal{V}^{\sim}[\![B_o]\!].
\end{aligned}
$$

By anti-reduction, it suffices to show

$$
\begin{aligned}
&(\langle B_o \leftarrowtail A_o\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle(V_1\,\langle A_i \twoheadleftarrow B_i\rangle V^l),\\
&\ \langle B_o \leftarrowtail A_o\rangle\langle\sigma_B \leftarrowtail \sigma_A\rangle(V_2\,\langle A_i \twoheadleftarrow B_i\rangle V^r))\\
&\quad \in \mathcal{E}^{\sim}_{k'}[\![\sigma_B]\!]\mathcal{V}^{\sim}[\![B_o]\!].
\end{aligned}
$$

By the inductive hypothesis applied twice, it suffices to show

$$((V_1\,\langle A_i \twoheadleftarrow B_i\rangle V^l), (V_2\,\langle A_i \twoheadleftarrow B_i\rangle V^r)) \in \mathcal{E}^{\sim}_{k'}[\![\sigma_A]\!]\mathcal{V}^{\sim}[\![A_o]\!].$$

By soundness of function application, it suffices to show that $(V_1, V_2) \in \mathcal{V}^{\sim}_{k'}[\![A_i \to_{\sigma_A} A_o]\!]$ and that $(\langle A_i \twoheadleftarrow B_i\rangle V^l, \langle A_i \twoheadleftarrow B_i\rangle V^r) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A_i]\!]$. The former is true by our assumption about V_1 and V_2. To show the latter, it suffices by the inductive hypothesis to show that $(V^l, V^r) \in \mathcal{E}^{\sim}_{k'}[\![\sigma]\!]\mathcal{V}^{\sim}[\![B_i]\!]$, which follows by our assumption.

(2) This is dual to the above.

(3) This is dual to the below, and in fact easier since these are upcasts.

(4) We prove this statement by Löb induction (Lemma D.14). That is, assume for all $(M', N') \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma']\!])_j(\mathcal{V}^{\sim}[\![A]\!])$, we have

$$(\langle\sigma \twoheadleftarrow \sigma'\rangle M, \langle\sigma \twoheadleftarrow \sigma'\rangle N) \in (\blacktriangleright\mathcal{E}^{\sim}[\![\sigma]\!])_j(\mathcal{V}^{\sim}[\![A]\!]).$$

Let $(M, N) \in \mathcal{E}^{\sim}_j[\![\sigma']\!]\mathcal{V}^{\sim}[\![A]\!]$. We need to show

$$(\langle\sigma \twoheadleftarrow \sigma'\rangle M, \langle\sigma \twoheadleftarrow \sigma'\rangle N) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

By monadic bind (Lemma D.16), it will suffice to consider the following two cases:

- Let $k \le j$ and let $(V_1, V_2) \in \mathcal{V}^{\sim}_k[\![A]\!]$. We need to show that

$$(\langle\sigma \twoheadleftarrow \sigma'\rangle V_1, \langle\sigma \twoheadleftarrow \sigma'\rangle V_2) \in \mathcal{E}^{\sim}_j[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].$$

  Since the effect cast is the identity on values, the above follows immediately by anti-reduction.
- Let $k \le j$ and let $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in \sigma'$ be an effect caught by $\langle\sigma \twoheadleftarrow \sigma'\rangle\bullet$. Recalling that σ' is shorthand for the reflexivity derivation $\sigma' \sqsubseteq \sigma'$, we have that c_ε and d_ε are themselves reflexivity (type precision) derivations; for brevity, we refer to the types as C and D.
  Let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\sim}[\![C]\!])_k$ and and let $E^l\#\varepsilon, E^r\#\varepsilon$ be such that

$$(x^l.E^l[x^l], x^r.E^r[x^r]) \in (\blacktriangleright\mathcal{K}^{\sim}[\![D]\!])_k(\mathcal{E}^{\sim}[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!]).$$

  We need to show

$$
\begin{aligned}
&(\langle\sigma \twoheadleftarrow \sigma'\rangle E^l[\texttt{raise}\ \varepsilon(V^l)],\\
&\ \langle\sigma \twoheadleftarrow \sigma'\rangle E^r[\texttt{raise}\ \varepsilon(V^r)])\\
&\quad \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].
\end{aligned}
$$

First, if $\varepsilon \notin \sigma$, then both sides step to $\mho$, and we are finished by anti-reduction since $\mho$ is related to itself by ErrBot (Lemma D.45).
Otherwise, by anti-reduction, it suffices to show

$$\begin{aligned}
&(\texttt{let}\ x = \langle D \nwarrow D\rangle \texttt{raise}\ \varepsilon(\langle C \twoheadleftarrow C\rangle V^l)\ \texttt{in}\ \langle \sigma \twoheadleftarrow \sigma'\rangle E^l[x],\\
&\ \texttt{let}\ x = \langle D \nwarrow D\rangle \texttt{raise}\ \varepsilon(\langle C \twoheadleftarrow C\rangle V^r)\ \texttt{in}\ \langle \sigma \twoheadleftarrow \sigma'\rangle E^r[x])\\
&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!]).
\end{aligned}$$

By the soundness of the term precision congruence rule for let, it suffices to show that (1)

$$\begin{aligned}
&(\langle D \nwarrow D\rangle \texttt{raise}\ \varepsilon(\langle C \twoheadleftarrow C\rangle V^l),\\
&\ \langle D \nwarrow D\rangle \texttt{raise}\ \varepsilon(\langle C \twoheadleftarrow C\rangle V^r))\\
&\qquad \in (\blacktriangleright \mathcal{E}^{\sim}[\![\sigma]\!])_k(\mathcal{V}^{\sim}[\![A]\!]).
\end{aligned}$$

and (2) for all related $(V_1, V_2) \in (\blacktriangleright \mathcal{V}^{\sim}[\![A]\!])$, we have

$$\begin{aligned}
&(\langle \sigma \twoheadleftarrow \sigma'\rangle E^l[V_1], \langle \sigma \twoheadleftarrow \sigma'\rangle E^r[V_2])\\
&\qquad \in \mathcal{E}^{\sim}_k[\![\sigma]\!]\mathcal{V}^{\sim}[\![A]\!].
\end{aligned}$$

□

D.0.5 Transitivity. We introduce the following notation. We define $(M_1, M_2) \in R_\omega$ to mean that $(M_1, M_2) \in R_k$ for all natural numbers k.

We now state and prove a “mixed transitivity” lemma, in which we allow one of the two relations in the assumption to occur at a “proper" precision derivation, while the other is constrained to occur at a reflexivity derivation.

Lemma D.64 (mixed transitivity, terms). *If (1) $(M_1, M_2) \in \mathcal{E}^{\gtrsim}_\omega[\![\sigma]\!]\mathcal{V}^{\gtrsim}[\![A]\!]$ and (2) $(M_2, M_3) \in \mathcal{E}^{\gtrsim}_j[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![c]\!]$, then $(M_1, M_3) \in \mathcal{E}^{\gtrsim}_j[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![c]\!]$.*

Similarly, if $(M_1, M_2) \in \mathcal{E}^{\lesssim}_j[\![d_\sigma]\!]\mathcal{V}^{\lesssim}[\![c]\!]$ and $(M_2, M_3) \in \mathcal{E}^{\lesssim}_\omega[\![\sigma]\!]\mathcal{V}^{\lesssim}[\![A]\!]$, then $(M_1, M_3) \in \mathcal{E}^{\lesssim}_j[\![d_\sigma]\!]\mathcal{V}^{\lesssim}[\![c]\!]$.

Proof. This is proved simultaneously with the following two lemmas on transitivity for results and values. We prove the lemma for ~=>; the other case is similar.

The proof is by Löb-induction (Lemma D.14). That is, assume that for all M_1', M_2', and M_3', if $(M_1', M_2') \in (\blacktriangleright \mathcal{E}^{\gtrsim}[\![\sigma]\!])_\omega(\mathcal{V}^{\gtrsim}[\![A]\!])$ and $(M_2', M_3') \in (\blacktriangleright \mathcal{E}^{\gtrsim}[\![d_\sigma]\!])_j(\mathcal{V}^{\gtrsim}[\![c]\!])$, then $(M_1', M_3') \in (\blacktriangleright \mathcal{E}^{\gtrsim}[\![d_\sigma]\!])_j(\mathcal{V}^{\gtrsim}[\![c]\!])$.

We proceed by considering cases on the assumption that $(M_2, M_3) \in \mathcal{E}^{\gtrsim}_j[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![c]\!]$.

In the first case, $M_3 \mapsto^{j+1}$. Then we immediately have that $(M_1, M_3) \in \mathcal{E}^{\gtrsim}_j[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![c]\!]$, via the first disjunct.

In the second case, there is $k \leq j$ such that $M_3 \mapsto^{j-k} \mho$ and $M_2 \mapsto^s \mho$, for some number of steps s. By assumption (1), we have that $(M_1, M_2) \in \mathcal{E}^{\gtrsim}_s[\![\sigma]\!]\mathcal{V}^{\gtrsim}[\![A]\!]$. By inversion, we see that the second disjunct must have been true (with $k = 0$). This means in particular that $M_1 \mapsto^* \mho$. Thus, we may conclude using the second disjunct that $(M_1, M_3) \in \mathcal{E}^{\gtrsim}_j[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![c]\!]$.

In the third case, there is $k \leq j$ and N_3 such that $M_3 \mapsto^{j-k} N_3$, and $M_2 \mapsto^s \mho$, for some number of steps s. By similar reasoning to the previous case, we may conclude using the third disjunct that $(M_1, M_3) \in \mathcal{E}^{\gtrsim}_j[\![d_\sigma]\!]\mathcal{V}^{\gtrsim}[\![c]\!]$.

Finally, in the fourth case, there exist $k \le j$ and $(N_2, N_3) \in \mathcal{R}^{\succeq}_{k}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$ such that $M_2 \mapsto^{s} N_2$ for some s, and $M_3 \mapsto^{j-k} N_3$. By assumption (1), we have that $(M_1, M_2) \in \mathcal{E}^{\succeq}_{s+i}[\![\sigma]\!]\mathcal{V}^{\succeq}[\![A]\!]$ for all $i \in \mathbb{N}$. By inversion, we see that either the third or the fourth disjunct was true, with $k = i$ in both cases (notice that $(s + i) - i = s$, which is precisely the number of steps that M_2 takes to N_2).

In the former case, we have $M_1 \mapsto^{*} \mho$ and we can then finish by asserting the third disjunct. In the latter case, there exists N_1 such that $M_1 \mapsto^{*} N_1$ and $(N_1, N_2) \in \mathcal{R}^{\succeq}_{i}[\![\sigma]\!]\mathcal{V}^{\succeq}[\![A]\!]$. Since i is arbitrary, this tells us that $(N_1, N_2) \in \mathcal{R}^{\succeq}_{\omega}[\![\sigma]\!]\mathcal{V}^{\succeq}[\![A]\!]$. To recap, we have $(N_1, N_2) \in \mathcal{R}^{\succeq}_{\omega}[\![\sigma]\!]\mathcal{V}^{\succeq}[\![A]\!]$, and $(N_2, N_3) \in \mathcal{R}^{\succeq}_{k}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$, for some $k \le j$. We want to show that $(N_1, N_3) \in \mathcal{R}^{\succeq}_{k}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$.

This follows from Lemma D.66.

□

Lemma D.65 (mixed transitivity, values). *If $(V_1, V_2) \in \mathcal{V}^{\succeq}_{\omega}[\![A]\!]$ and $(V_2, V_3) \in \mathcal{V}^{\succeq}_{j}[\![c]\!]$, then $(V_1, V_3)\mathcal{V}^{\succeq}_{j}[\![c]\!]$.*

Similarly, if $(V_1, V_2) \in \mathcal{V}^{\preceq}_{j}[\![c]\!]$ and $(V_2, V_3) \in \mathcal{V}^{\preceq}_{\omega}[\![A]\!]$, then $(V_1, V_3)\mathcal{V}^{\preceq}_{j}[\![c]\!]$.

Proof. Proved simultaneously with the homogeneous transitivity for terms (Lemma D.64) and for results (Lemma D.66). The proof is by induction on the type precision derivation c. We prove the first statement only; the other is proved similarly.

- Case $c = \texttt{bool}$. Then we have $V_1 = V_2 = V_3$ and either all are `true`, or all are `false`. In either case, V_1 is related to V_3.
- Case $c = c_i \rightarrow_{c_\sigma} c_o$. Then $A = A_i \rightarrow_{\sigma_A} A_o$ and $B = B_i \rightarrow_{\sigma_B} B_o$.
  We have $(V_1, V_2) \in \mathcal{V}^{\sim}_{\omega}[\![A_i \rightarrow_{\sigma_A} A_o]\!]$ and $(V_2, V_3) \in \mathcal{V}^{\sim}_{k}[\![c_i \rightarrow_{c_\sigma} c_o]\!]$.
  We need to show
  $$(V_1, V_3) \in \mathcal{V}^{\succeq}_{j}[\![c_i \rightarrow_{c_\sigma} c_o]\!].$$
  Let $k \le j$ and let $(V^l, V^r) \in \mathcal{V}^{\succeq}_{k}[\![c_i]\!]$. We need to show that
  $$(V_1\, V^l, V_3\, V^r) \in \mathcal{E}^{\succeq}_{k}[\![c_\sigma]\!]\mathcal{V}^{\succeq}[\![c_o]\!].$$
  By reflexivity (D.28), we know that $(V^l, V^l) \in \mathcal{V}^{\succeq}_{\omega}[\![A_i]\!]$.
  From our assumption about (V_1, V_2), it follows that
  $$(V_1\, V^l, V_2\, V^l) \in \mathcal{E}^{\succeq}_{\omega}[\![\sigma_A]\!]\mathcal{V}^{\succeq}[\![A_o]\!].$$
  From our assumption about (V_2, V_3), we have
  $$(V_2\, V^l, V_3\, V^r) \in \mathcal{E}^{\succeq}_{k}[\![c_\sigma]\!]\mathcal{V}^{\succeq}[\![c_o]\!].$$
  Now we apply the induction hypothesis (Lemma D.64) to conclude that
  $$(V_1\, V^l, V_3\, V^r) \in \mathcal{E}^{\succeq}_{k}[\![c_\sigma]\!]\mathcal{V}^{\succeq}[\![c_o]\!],$$
  as needed.

□

Lemma D.66 (mixed transitivity, results). *If (1) $(N_1, N_2) \in \mathcal{R}^{\succeq}_{\omega}[\![\sigma]\!]\mathcal{V}^{\succeq}[\![A]\!]$ and (2) $(N_2, N_3) \in \mathcal{R}^{\succeq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$, then $(N_1, N_3) \in \mathcal{R}^{\succeq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$.*

Similarly, if $(N_1, N_2) \in \mathcal{R}^{\preceq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\preceq}[\![c]\!]$ and $(N_2, N_3) \in \mathcal{R}^{\preceq}_{\omega}[\![\sigma]\!]\mathcal{V}^{\preceq}[\![A]\!]$, then $(N_1, N_3) \in \mathcal{R}^{\preceq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\preceq}[\![c]\!]$.

Proof. We prove only the first statement; the second is analogous.

Let j be fixed. We consider cases on assumption (1). There are two subcases to consider. First, N_1 and N_2 are values and $(N_1, N_2) \in \mathcal{V}^{\succeq}_{\omega}[\![A]\!]$. Then N_3 is also a value, and $(N_2, N_3) \in \mathcal{V}^{\succeq}_{j}[\![c]\!]$. By D.65, we have that $(N_1, N_3) \in \mathcal{V}^{\succeq}_{j}[\![A]\!]$.

Otherwise, there exist $\varepsilon : C \rightsquigarrow D \in \sigma$, $E_1\#epsilon$ and $E_2\#\varepsilon$, and V_1 and V_2 such that $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\succeq}[\![C]\!])_{\omega}$, and $(x_1.E_1[x_1], x_2.E_2[x_2]) \in (\blacktriangleright\mathcal{K}^{\succeq}[\![D]\!])_{\omega}(\mathcal{E}^{\succeq}[\![\sigma]\!]\mathcal{V}^{\succeq}[\![A]\!])$, and

$$N_1 = E_1[\texttt{raise}\ \varepsilon(V_1)],$$

and

$$N_2 = E_2[\texttt{raise}\ \varepsilon(V_2)].$$

Similarly, since N_2 and N_3 are related in $\mathcal{R}^{\succeq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$, it follows that $\varepsilon : c_\varepsilon \rightsquigarrow d_\varepsilon \in d_\sigma$, where $c_\varepsilon : C \sqsubseteq C'$ and $d_\varepsilon : D \sqsubseteq D'$. We also know that there exist $E_3\#\varepsilon$ and V_3 such that $(V_2, V_3) \in (\blacktriangleright\mathcal{V}^{\succeq}[\![c_\varepsilon]\!])_j$, and $(x_2.E_2[x_2], x_3.E_3[x_3]) \in (\blacktriangleright\mathcal{K}^{\succeq}[\![d_\varepsilon]\!])_j(\mathcal{E}^{\succeq}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!])$, and

$$N_3 = E_3[\texttt{raise}\ \varepsilon(V_3)].$$

Recall that we need to show

$$(E_1[\texttt{raise}\ \varepsilon(V_1)], E_3[\texttt{raise}\ \varepsilon(V_3)]) \in \mathcal{R}^{\succeq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!].$$

We assert the second disjunct in the definition of $\mathcal{R}^{\succeq}[\![\cdot]\!]$.

We first claim that $(V_1, V_3) \in (\blacktriangleright\mathcal{V}^{\succeq}[\![c_\varepsilon]\!])_j$. By transitivity for values (Lemma D.65), it suffices to show that $(V_1, V_2) \in (\blacktriangleright\mathcal{V}^{\succeq}[\![c_\varepsilon]\!])_{\omega}$ and $(V_2, V_3) \in (\blacktriangleright\mathcal{V}^{\succeq}[\![c_\varepsilon]\!])_j$. These follow by assumption.

Now we claim that

$$(x_1.E_1[x_1], x_3.E_3[x_3]) \in (\blacktriangleright\mathcal{K}^{\succeq}[\![d_\varepsilon]\!])_j(\mathcal{E}^{\succeq}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]).$$

Let $k \leq j$ and let $(V^l, V^r) \in (\blacktriangleright\mathcal{V}^{\succeq}[\![d_\varepsilon]\!])_k$. We need to show

$$(E_1[V^l], E_3[V^r]) \in (\blacktriangleright\mathcal{E}^{\succeq}[\![d_\sigma]\!])_k(\mathcal{V}^{\succeq}[\![c]\!]).$$

By the induction hypothesis (recall we are proving this simultaneously with transitivity for terms, which is being proven by Löb induction), it suffices to find a term M such that $(E_1[V^l], M) \in (\blacktriangleright\mathcal{E}^{\succeq}[\![\sigma]\!])_{\omega}(\mathcal{V}^{\succeq}[\![A]\!])$, and $(M, E_3[V_r]) \in (\blacktriangleright\mathcal{E}^{\succeq}[\![d_\sigma]\!])_k(\mathcal{V}^{\succeq}[\![c]\!])$.

By reflexivity (Corollary D.28), we have $(V^l, V^l) \in (\blacktriangleright\mathcal{V}^{\sim}[\![\,]\!])_{\omega}$.

Then by our assumption on (E_1, E_2), we have

$$(E_1[V^l], E_2[V^l]) \in (\blacktriangleright\mathcal{E}^{\succeq}[\![\sigma]\!])_{\omega}(\mathcal{V}^{\succeq}[\![A]\!])$$

By our assumption on (E_2, E_3) we have

$$(E_2[V^l], E_3[V^r]) \in (\blacktriangleright\mathcal{E}^{\succeq}[\![d_\sigma]\!])_k(\mathcal{V}^{\succeq}[\![c]\!]),$$

which finishes the proof. □

Lemma D.67 (heterogeneous transitivity). *Let $c : A_1 \sqsubseteq A_2$ and $e : A_2 \sqsubseteq A_3$. Let $d_\sigma : \sigma \sqsubseteq \sigma'$ and let $d'_\sigma : \sigma' \sqsubseteq \sigma''$.*

If (1) $(M_1, M_2) \in \mathcal{E}^{\succeq}_{\omega}[\![d_\sigma]\!]\mathcal{V}^{\succeq}[\![c]\!]$ and (2) $(M_2, M_3) \in \mathcal{E}^{\succeq}_{j}[\![d'_\sigma]\!]\mathcal{V}^{\succeq}[\![e]\!]$, then $(M_1, M_3) \in \mathcal{E}^{\succeq}_{j}[\![d_\sigma \circ d'_\sigma]\!]\mathcal{V}^{\succeq}[\![c \circ e]\!]$.

Similarly, if $(M_1, M_2) \in \mathcal{E}^{\preceq}_{j}[\![d_\sigma]\!]\mathcal{V}^{\preceq}[\![c]\!]$ and $(M_2, M_3) \in \mathcal{E}^{\preceq}_{\omega}[\![d'_\sigma]\!]\mathcal{V}^{\preceq}[\![e]\!]$, then $(M_1, M_3) \in \mathcal{E}^{\preceq}_{j}[\![d_\sigma \circ d'_\sigma]\!]\mathcal{V}^{\preceq}[\![c \circ e]\!]$.

Proof. Follows from mixed transitivity (Lemma D.64) and the generalized cast lemmas (Lemmas D.48, D.49, D.50, D.51, D.52, D.53, D.54, and D.55).

For example, by EffDnR and ValDnR, we have

$$(M_1,\ \langle \sigma \twoheadleftarrow \sigma' \rangle \langle A_1 \twoheadleftarrow A_2 \rangle M_2) \in \mathcal{E}^{\succeq}_{\omega} [\![\sigma]\!] \mathcal{V}^{\succeq} [\![A_1]\!],$$

and by EffDnL and ValDnL, we have

$$(\langle \sigma \twoheadleftarrow \sigma' \rangle \langle A_1 \twoheadleftarrow A_2 \rangle M_2,\ M_3) \in \mathcal{E}^{\succeq}_{j} [\![d_\sigma \circ d'_\sigma]\!] \mathcal{V}^{\succeq} [\![c \circ e]\!].$$

Then applying mixed transitivity, we have

$$(M_1, M_3) \in \mathcal{E}^{\succeq}_{j} [\![d_\sigma \circ d'_\sigma]\!] \mathcal{V}^{\succeq} [\![c \circ e]\!],$$

as desired.

□