Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

**Master's Thesis**

# Non-Parametric Parametricity

submitted by
Georg Neis
on July 1, 2009

Supervisor
Prof. Dr. Peter Druschel

Advisors
Dr. Derek Dreyer
Dr. Andreas Rossberg

Reviewers
Prof. Dr. Peter Druschel
Dr. Derek Dreyer

**Eidesstattliche Erklärung**
Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

**Statement under Oath**
I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

**Einverständniserklärung**
Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

**Declaration of Consent**
I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

**Acknowledgment**
Many thanks to Derek Dreyer and Andreas Rossberg for support, advice, and all the rest.

**Abstract**

Type abstraction and intensional type analysis are features seemingly at odds—type abstraction is intended to guarantee parametricity and representation independence, while type analysis is inherently non-parametric. Recently, however, several researchers have proposed and implemented "dynamic type generation" as a way to reconcile these features. The idea is that, when one defines an abstract type, one should also be able to generate at run time a fresh type name, which may be used as a dynamic representative of the abstract type for purposes of type analysis. The question remains: in a language with non-parametric polymorphism, does dynamic type generation provide us with the same kinds of abstraction guarantees that we get from parametric polymorphism?

Our goal is to provide a rigorous answer to this question. We define a step-indexed Kripke logical relation for a language with both non-parametric polymorphism and dynamic type generation. Our logical relation enables us to establish parametricity and representation independence results, even in a non-parametric setting, by attaching arbitrary relational interpretations to dynamically-generated type names. In addition, we explore how programs that are provably equivalent in a more traditional parametric logical relation may be "wrapped" systematically to produce terms that are related by our non-parametric relation, and vice versa. This leads us to a novel polarized form of our logical relation, which distinguishes between positive and negative notions of parametricity.

# Contents

# 1 Introduction

## 1.1 Data Abstraction Via Type Abstraction

*Data abstraction* is a key concept in structuring programs. The idea is to make a clear separation between the *interface* of a data structure—*i.e.,* an abstract description of it—and its actual implementation. Clients then should not make any assumptions about the data structure's implementation and only use it according to its interface. The goal is to obtain the following two properties, which go back to Morris [16][1]:

**Privacy.** Clients have no way of learning details of the implementation and hence cannot depend on internals. Consequently, the implementation may be replaced by an improved version without breaking existing code (assuming that the interface and its semantics do not change).

**Integrity.** Clients have no way of affecting details of the implementation. Consequently, the implementation may maintain invariants and can rely on them not being violated.

To this end, statically typed languages like ML provide the concept of an *abstract data type* (ADT), which is a special kind of module. An ADT consists of

1. a *signature*, which declares an abstract type (giving it a name) and lists the operations for processing values of that type,
2. an implementation for the abstract type, *i.e.,* a concrete representation type, and
3. implementations for the operations.

As an example, consider an ADT for integer sets, as shown in Figure 1. Its signature `INT_SET` declares an abstract type `set`, which is represented as a list of integers by the naive implementation `IntSet`. The figure additionally shows a second implementation, `IntSet'`, which is a suitable candidate for replacing the first one. It also represents a set as a list, but maintains the invariant that this list is ordered, which allows for a more efficient implementation of the membership predicate.

Since the representation of the abstract type `set` is not exposed and the language offers no means for a client to inspect it, the only way values of type `set` can be processed is via the provided operations. This enforces privacy and integrity.

## 1.2 Type Abstraction Via Parametric Polymorphism

Technically, the fact that abstract types in ML are truly abstract is a consequence of the *parametric* nature of the language's *polymorphism*. The latter is the mechanism for writing code that works with values of different types. In particular, client code of an ADT is polymorphic in the ADT's concrete representation type. Parametricity roughly means that the run-time behaviour of such code does not depend on what types it is instantiated with. Apart from enforcing abstraction, parametricity enables the erasure of types during compilation and enables one to derive properties of a function solely by looking at its type ("free theorems") [31].

[1] He used the names *secrecy* and *authentication*, though.

```
signature INT_SET =
sig
  type set
  val empty : set
  val add : int * set -> set
  val member : int * set -> bool
end

structure IntSet :> INT_SET =
struct
  type set = int list
  val empty = []
  fun add(x, xs) = x::xs
  fun member(x, []) = false
    | member(x, y::ys) =
      x = y orelse member(x, ys)
end

structure IntSet' :> INT_SET =
struct
  type set = int list
  val empty = []
  fun add(x, []) = [x]
    | add(x, y::ys) =
      if x = y then y::ys
      else if x < y then x::y::ys
      else y::add(x, ys)
  fun member(x, []) = false
    | member(x, y::ys) =
      x = y orelse (x > y andalso member(x, ys))
end
```

Figure 1: An ADT for sets over integers

### 1.3 Dynamic Type Analysis

However, there are a number of modern programming language features that are in direct conflict with parametricity, because they provide some form of type analysis at run time.

Probably the simplest example are *dynamics* [1], an elegant approach for integrating dynamic typing into a statically typed language. In such a setting, any value may be cast to type Dynamic and then exported to the world (*e.g.,* written out to hard disk). Importing data or code (*e.g.,* from hard disk) will again yield a value of type Dynamic. This value can then be dispatched on with the help of a typecase construct that analyses the value's actual type. Clearly, such a language cannot be parametric—types *do* matter at run time. A similar feature that is incompatible with parametricity can be found in languages such as Acute [26] and Alice ML [24, 22]. Since they are designed to support dynamic loading of modules, they require the ability to check dynamically whether a module implements an expected interface, which in turn involves runtime inspection of the module's type components. There have also been a number of more experimental proposals

for languages that employ a typecase construct to facilitate *polytypic* programming (*e.g.,* [33, 30]).

There is a fundamental tension between type analysis and type abstraction. If one can inspect the identity of an unknown type at run time, then the type is not really abstract, so any invariants concerning values of that type may be broken [33]. More concretely, the typecase construct can be (mis-)used to both gain information about the representation of an abstract type and to forge or mutate (in the case of stateful types) values of it. Therefore, languages with a type Dynamic sometimes distinguish between *castable* and *non-castable* types—with types that mention user-defined abstract types belonging to the latter category—and prohibit values with non-castable types from being cast to type Dynamic.

This is, however, an unnecessarily severe restriction, which effectively penalizes programmers for using type abstraction. Given a user-defined abstract type t—implemented internally, say, as int—it is perfectly reasonable to cast a value of type $\mathsf{t} \to \mathsf{t}$ to Dynamic, so long as we can ensure that it will subsequently be cast back only to $\mathsf{t} \to \mathsf{t}$ (not to, say, $\mathsf{int} \to \mathsf{int}$ or $\mathsf{int} \to \mathsf{t}$), *i.e.,* so long as the cast is *abstraction-safe*. Moreover, such casts are useful when marshalling (or "pickling") a modular component whose interface refers to abstract types defined in other components [24, 22].

## 1.4 Data Abstraction Via Dynamic Sealing

Type abstraction can be seen as the static approach to data abstraction. A different approach is *dynamic sealing* [15, 12, 28]. It is typically employed in untyped languages, which do not have the ability to place static restrictions on clients. Consequently, data abstraction is enforced on the level of individual values. For that, languages provide means for generating unique names and using them as *keys* for sealing values. A value sealed by a given key can only be inspected by principals that have access to the key [28]. Data abstraction can thus be achieved by writing a module that creates a key in its local scope and then implements the data operations such that they lock values before handing them out to clients, and, analogously, unlock values that are handed in. One drawback of this approach is that crossing abstraction boundaries results in runtime costs. Another big disadvantage is that it is both costly and hard to reconcile with ML-style module abstraction (in particular, references are difficult to handle).

## 1.5 Data Abstraction Via Dynamic Type Generation

The approach we consider in this thesis is based on the generation of types at run time [25, 21, 30, 23]. The basic idea is similar to the separation of castable and non-castable types: one wants to be able to distinguish between an abstract type and its representation. However, it is sufficient that this dinstinction can be made dynamically (when a cast occurs). Therefore, when one defines an abstract type, one should also be able to generate at run time a "fresh" type name, which may be used as a dynamic representative of the abstract type for purposes of type analysis.[2] (We will see a concrete example of this in Section 2.) Intuitively, the freshness of type name generation ensures that user-defined abstract types are viewed dynamically in the same way that they are viewed statically—*i.e.,* as distinct from all other types.

This technique can be seen as a middle ground between the purely static approach based on parametricity and the purely dynamic approach based on sealing values. As in the dynamic approach, we cannot rely on parametricity and instead generate dynamic names to protect abstractions. However, these are type-level names, not term-level names, and they only "seal" type information. In particular, individual values of abstract type are still directly represented by the

[2] In languages with simple module mechanisms, such as Haskell, it is possible to generate unique type names statically. However, this is not sufficient in the presence of functors and local or first-class modules.

underlying representation type, so that crossing abstraction boundaries has no runtime cost. In that sense, we are closer to the static approach.

## 1.6 Our Goal

The question remains: how do we know that dynamic type generation *works*? In a language with some form of dynamic type analysis—*i.e., non-parametric* polymorphism—does dynamic type generation provably provide us with the same kinds of abstraction guarantees that we get from traditional parametric polymorphism?

Our goal is to provide a rigorous answer to this question. We study an extension of System F, supporting (1) a type-safe cast operator, which is essentially a variant of Girard's J operator [9], and (2) a facility for dynamic generation of fresh type names. For brevity, we will call this language **G**. As a practical language mechanism, the cast operator is somewhat crude in comparison to the more expressive typecase-style constructs proposed in the literature,[3] but it nonetheless renders polymorphism *non-parametric*. Our main technical result is that, in a language with non-parametric polymorphism, parametricity may be provably regained via judicious use of dynamic type generation.

## 1.7 Outline

The rest of the thesis is structured as follows. In Section 2, we present our language under consideration, G, and also give an example to illustrate how dynamic type generation is useful. In Section 3, we explain informally the approach that we have developed for reasoning about G. It employs a *step-indexed Kripke logical relation*. This section is intended to be broadly accessible to readers who are generally familiar with the basic idea of relational parametricity but not with the details of (advanced) logical relations techniques.

In Section 4, we formalize our logical relation for G and show how it may be used to reason about parametricity and representation independence. A particularly appealing feature of our formalization is that the *non*-parametricity of G is encapsulated in the notion of what it means for two *types* to be logically related to each other when viewed as *data*. The definition of this type-level logical relation is a one-liner, which can easily be replaced with an alternative "parametric" version.

In Sections 5–8, we explore how terms related by the parametric version of our logical relation may be "wrapped" systematically to produce terms related by the non-parametric version (and vice versa), thus clarifying how dynamic type generation enables parametric reasoning. This leads us to a novel "polarized" form of our logical relation, which distinguishes between positive and negative notions of parametricity.

In Section 9, we extend G with iso-recursive types to form $G^{\mu}$ and adapt the previous development accordingly. Section 10 has detailed proofs of the logical relations' key properties, the wrapping results, and the examples discussed in the preceding sections. In Section 11, we discuss how the wrapping can be seen as a translation from $F^{\mu}$ into $G^{\mu}$, which we conjecture to be fully abstract.

Finally, in Section 12, we discuss related work and in Section 13, we conclude and suggest directions for future work.

[3] That said, the implementation of dynamic modules in Alice ML, for instance, employs a very similar construct [24, 22].

$$
\begin{array}{lll}
\text{Types} & \tau ::= \alpha \mid b \mid \tau \to \tau \mid \tau \times \tau \mid \forall\alpha.\tau \mid \exists\alpha.\tau \\
\text{Values} & v ::= x \mid c \mid \lambda x{:}\tau.e \mid \langle v_1, v_2\rangle \mid \lambda\alpha.e \mid \mathsf{pack}\ \langle \tau, v\rangle\ \mathsf{as}\ \tau \\
\text{Terms} & e ::= v \mid e\ e \mid \langle e_1, e_2\rangle \mid e.1 \mid e.2 \mid e\ \tau \mid \\
 & \qquad \mathsf{pack}\ \langle \tau, e\rangle\ \mathsf{as}\ \tau \mid \mathsf{unpack}\ \langle \alpha, x\rangle{=}e\ \mathsf{in}\ e \mid \\
 & \qquad \mathsf{cast}\ \tau\ \tau \mid \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e \\
\text{Stores} & \sigma ::= \epsilon \mid \sigma, \alpha{\approx}\tau \\
\text{Config's} & \zeta ::= \sigma; e \\
\\
\text{Type Contexts} & \Delta ::= \epsilon \mid \Delta, \alpha \mid \Delta, \alpha{\approx}\tau \\
\text{Value Contexts} & \Gamma ::= \epsilon \mid \Gamma, x{:}\tau
\end{array}
$$

$\boxed{\Delta; \Gamma \vdash e : \tau}$

$$\cdots$$

$$(\textsc{Ecast})\frac{\Delta \vdash \tau_1 \qquad \Delta \vdash \tau_2}{\Delta; \Gamma \vdash \mathsf{cast}\ \tau_1\ \tau_2 : \tau_1 \to \tau_2 \to \tau_2}$$

$$(\textsc{Enew})\frac{\Delta \vdash \tau \qquad \Delta, \alpha{\approx}\tau; \Gamma \vdash e : \tau' \qquad \Delta \vdash \tau'}{\Delta; \Gamma \vdash \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e : \tau'}$$

$$(\textsc{Econv})\frac{\Delta; \Gamma \vdash e : \tau' \qquad \Delta \vdash \tau \approx \tau'}{\Delta; \Gamma \vdash e : \tau}$$

$\boxed{\Delta \vdash \tau}$

$$(\textsc{Tname})\frac{\alpha{\approx}\tau \in \Delta}{\Delta \vdash \alpha} \qquad \cdots$$

$\boxed{\Delta \vdash \tau \approx \tau}$

$$(\textsc{Cname})\frac{\alpha{\approx}\tau \in \Delta}{\Delta \vdash \alpha \approx \tau} \qquad \cdots$$

$\boxed{\vdash \zeta : \tau}$

$$(\textsc{conf})\frac{\vdash \sigma \qquad \sigma; \epsilon \vdash e : \tau \qquad \epsilon \vdash \tau}{\vdash \sigma; e : \tau}$$

$$
\begin{array}{rrcl}
 & \sigma; (\lambda x{:}\tau.e)\ v & \hookrightarrow & \sigma; e[v/x] \\
 & \sigma; \langle v_1, v_2\rangle.i & \hookrightarrow & \sigma; v_i \\
 & \sigma; (\lambda\alpha.e)\ \tau & \hookrightarrow & \sigma; e[\tau/\alpha] \\
 & \sigma; \mathsf{unpack}\ \langle \alpha, x\rangle{=}(\mathsf{pack}\ \langle \tau, v\rangle)\ \mathsf{in}\ e & \hookrightarrow & \sigma; e[\tau/\alpha][v/x] \\
(\tau_1 = \tau_2) & \sigma; \mathsf{cast}\ \tau_1\ \tau_2 & \hookrightarrow & \sigma; \lambda x_1{:}\tau_1.\lambda x_2{:}\tau_2.x_1 \\
(\tau_1 \neq \tau_2) & \sigma; \mathsf{cast}\ \tau_1\ \tau_2 & \hookrightarrow & \sigma; \lambda x_1{:}\tau_1.\lambda x_2{:}\tau_2.x_2 \\
(\alpha \notin \mathrm{dom}(\sigma)) & \sigma; \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e & \hookrightarrow & \sigma, \alpha{\approx}\tau; e
\end{array}
$$

(. . . plus standard "search" rules . . . )

Figure 2: Syntax and Semantics of G (excerpt)

## 2 The Language G

Figure 2 defines our non-parametric language G. For the most part, it is a standard call-by-value $\lambda$-calculus, consisting of the usual types and terms from System F [9], including pairs

and existential types.[4] We also assume an unspecified set of base types $b$, along with suitable constants $c$ of those types.

Two additional, non-standard constructs isolate the essential aspects of the class of languages we are interested in:

- $\mathsf{cast}\ \tau_1\ \tau_2\ v_1\ v_2$ converts $v_1$ from type $\tau_1$ to $\tau_2$. It checks that those two types are the same at the time of evaluation. If so, the operator *succeeds* and returns $v_1$. Otherwise, it *fails* and defaults to $v_2$, which acts as an $\mathsf{else}$ clause of the target type $\tau_2$.

- $\mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e$ generates a fresh abstract type name $\alpha$. Values of type $\alpha$ can be formed using its *representation type* $\tau$. Both types are deemed *compatible*, but not equivalent. That is, they are considered equal as *classifiers*, but not as *data*. In particular, $\mathsf{cast}\ \alpha\ \tau\ v\ v'$ will not succeed (*i.e.,* it will return $v'$).

Our $\mathsf{cast}$ operator is essentially the same as Harper and Mitchell's *TypeCond* operator [11], which was itself a variant of the non-parametric J operator that Girard studied in his thesis [9]. Our $\mathsf{new}$ construct is similar to previously proposed constructs for dynamic type generation [21, 30, 23]. However, we do not require *explicit* term-level type coercions to witness the isomorphism between an abstract type name $\alpha$ and its representation $\tau$. Instead, our type system is simple enough that we perform this conversion *implicitly*.

For convenience, we will occasionally use expressions of the form $\mathsf{let}\ \ x = e_1\ \mathsf{in}\ e_2$, which abbreviate the term $(\lambda x{:}\tau_1.e_2)\ e_1$ (with $\tau_1$ being an appropriate type for $e_1$). We omit the type annotation for existential packages where clear from context. Moreover, we take the liberty to generalize binary tuples to $n$-ary ones where necessary and to use pattern matching notation to decompose tuples in the obvious manner.

## 2.1 Typing Rules

The typing rules for the System F fragment of G are completely standard and thus omitted from Figure 2. We focus on the non-standard rules related to casting and dynamic type generation. Full formal details of the type system are given in the appendix, together with standard structural properties and type safety.

Typing of casts is straightforward (Rule ECAST): $\mathsf{cast}\ \tau_1\ \tau_2$ is simply treated as a function of type $\tau_1 \to \tau_2 \to \tau_2$. Its first argument is the value to be converted, and its second argument is the default value returned in the case of failure. The rule merely requires that the two types be well-formed.

For an expression $\mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e$, which binds $\alpha$ in $e$, Rule ENEW checks that the body $e$ is well-typed under the assumption that $\alpha$ is implemented by the representation type $\tau$. For that purpose, we enrich type contexts $\Delta$ with entries of the form $\alpha{\approx}\tau$ that keep track of the representation types tied to abstract type names. Note that $\tau$ may not mention $\alpha$

Syntactically, type names are just type variables. When viewed as data, (*i.e.,* when inspected by the $\mathsf{cast}$ operator), types are considered equivalent iff they are syntactically equal. In contrast, when viewed as classifiers for terms, knowledge about the representation of type names may be taken into account. Rule ECONV says that if a term $e$ has type $\tau'$, it may be assigned any other type that is *compatible* with $\tau'$. Type compatibility, in turn, is defined by the judgment $\Delta \vdash \tau_1 \approx \tau_2$. We only show the rule CNAME, which discharges a compatibility assumption $\alpha{\approx}\tau$ from the context; the other rules implement the congruence closure of this axiom. The

[4] We could use a Church encoding of existentials through universals, but distinguishing them gives us more leeway later (cf. Section 5).

important point here is that equivalent types are compatible, but compatible types are not necessarily equivalent.

Finally, Rule ENEW also requires that the type $\tau'$ of the body $e$ does not contain $\alpha$ (*i.e.,* $\tau'$ must be well formed in $\Delta$ alone), such that $\alpha$ cannot escape its scope. However, this is not a real restriction—a type of this form can always be derived by applying ECONV to convert $\tau'$ to $\tau'[\tau/\alpha]$.

Note that the typing rules ensure that type contexts are ordered and acyclic. Consequently, any type $\Delta \vdash \tau$ can be normalized to a closed type $\tau'$ that is compatible with $\tau$, *i.e.,* $\epsilon \vdash \tau'$ and $\Delta \vdash \tau \approx \tau'$. This normalization is done using the substitution $\Delta^*$ that is obtained from $\Delta$ in the following way:

$$\epsilon^* \stackrel{\text{def}}{=} \emptyset$$

$$(\Delta, \alpha)^* \stackrel{\text{def}}{=} \Delta^*$$

$$(\Delta, \alpha{\approx}\tau)^* \stackrel{\text{def}}{=} \Delta^*, \alpha{\mapsto}\Delta^*(\tau)$$

Given this normalization, it easy to see that type checking is decidable.

## 2.2 Dynamic Semantics

The operational semantics has to deal with generation of fresh type names. To that end, we introduce a *type store* $\sigma$ to record generated type names. Hence, reduction is defined on *configurations* $(\sigma; e)$ instead of plain terms. Figure 2 shows the main reduction rules. We omit the standard "search" rules for descending into subterms according to call-by-value, left-to-right evaluation order.

The reduction rules for the F fragment are as usual and do not actually touch the store. However, types occurring in F constructs can contain type names bound in the store.

Reducing the expression $\mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e$ creates a new entry for $\alpha$ in the type store. We rely on the usual hygiene convention for bound variables to ensure that $\alpha$ is fresh with respect to the current store (which can always be achieved by $\alpha$-renaming).[5]

The two remaining rules are for casts. A cast takes two types and checks that they are equivalent (*i.e.,* syntactically equal). In either case, the expression reduces to a function that will return the appropriate one of the additional value arguments, *i.e.,* the value to be converted in case of success, and the default value otherwise. In the former case, type preservation is ensured because source and target types are known to be equivalent.

Type preservation can be expressed using the typing rule CONF for configurations. We formulate this rule by treating the type store as a type context, which is possible because type stores are a syntactic subclass of type contexts. (In a similar manner, we can write $\vdash \sigma$ for well-formedness of store $\sigma$, by viewing it as a type context.) It is worth noting that the representation types in the store are actually never inspected by the dynamic semantics. They are only needed for specifying well-formedness of configurations and proving type safety (see appendix).

## 2.3 Motivating Example

Consider the following attempt to write a simple functional "binary semaphore" ADT [17] in G. Following Mitchell and Plotkin [14], we use an existential type, as we would in System F:

$$\begin{aligned} \tau_{\text{sem}} &:= \exists\alpha.\alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool}) \\ e_{\text{sem}} &:= \mathsf{pack}\ \langle \mathsf{int}, \langle 1, \lambda x{:}\,\mathsf{int}\,.(1 - x), \lambda x{:}\,\mathsf{int}\,.(x \neq 0)\rangle\rangle\ \mathsf{as}\ \tau_{\text{sem}} \end{aligned}$$

[5] A well-known alternative approach would omit the type store in favour of using scope extrusion rules for new binders, as in Rossberg [21].

A semaphore essentially is a flag that can be in two states: either *locked* or *unlocked*. The state can be toggled using the first function of the ADT, and it can be polled using the second. Our little module uses an integer value for representing the state, taking 1 for locked and 0 for unlocked. It is an invariant of the implementation that the integer never takes any other value—otherwise, the toggle function would no longer operate correctly.

In System F, the implementation invariant would be protected by the fact that existential types are parametric: there is no way to inspect the witness of $\alpha$ after opening the package, and hence no client could produce values of type $\alpha$ other than those returned by the module (nor could she apply integer operations to them).

Not so in G. The following program uses $\mathsf{cast}$ to forge a value $s$ of the abstract semaphore type $\alpha$:

$$\begin{array}{rcl} e_{\text{client}} & := & \mathsf{unpack}\ \langle \alpha, \langle s_0, \mathit{toggle}, \mathit{poll} \rangle \rangle = e_{\text{sem}}\ \mathsf{in} \\ & & \mathsf{let}\ s = \mathsf{cast}\ \ \mathsf{int}\ \alpha\ 666\ s_0\ \mathsf{in} \\ & & \langle \mathit{poll}\ s, \mathit{poll}\ (\mathit{toggle}\ s) \rangle \end{array}$$

Because reduction of $\mathsf{unpack}$ simply substitutes the representation type $\mathsf{int}$ for $\alpha$, the consecutive cast succeeds, and the whole expression evaluates to $\langle \mathsf{true}, \mathsf{true} \rangle$—although the second component should have toggled $s$ and thus be different from the first.

The way to prevent this in G is to create a fresh type name as witness of the abstract type:

$$\begin{array}{rcl} e_{\text{sem1}} & := & \mathsf{new}\ \alpha' {\approx} \mathsf{int}\ \mathsf{in} \\ & & \mathsf{pack}\ \langle \alpha', \langle 1, \lambda x{:}\,\mathsf{int}\,.(1 - x), \lambda x{:}\,\mathsf{int}\,.(x \neq 0) \rangle \rangle\ \mathsf{as}\ \tau_{\text{sem}} \end{array}$$

After replacing the initial semaphore implementation with this one, $e_{\text{client}}$ will evaluate to $\langle \mathsf{true}, \mathsf{false} \rangle$ as desired—the $\mathsf{cast}$ expression will no longer succeed, because $\alpha$ will be substituted by the dynamic type name $\alpha'$, and $\alpha' \neq \mathsf{int}$. (Moreover, since $\alpha'$ is only visible statically in the scope of the $\mathsf{new}$ expression, the client has no access to $\alpha'$, and thus cannot *convert* from $\mathsf{int}$ to $\alpha'$ either.) The remaining occurrences of $\mathsf{int}$ within the $\mathsf{pack}$ body do no harm, as they are merely annotations, which do not affect the dynamic semantics.

Now, while it is clear that $\mathsf{new}$ ensures proper type abstraction in the client program $e_{\text{client}}$, we want to prove that it does so for *any* client program. A standard way of doing so is by showing a more general property, namely *representation independence* [20]: we show that the module $e_{\text{sem1}}$ is *contextually equivalent* to another module of the same type, meaning that no G program can observe any difference between the two modules. By choosing that other module to be a suitable reference implementation of the ADT in question, we can conclude that the "real" one behaves properly under all circumstances.

The obvious candidate for a reference implementation of the semaphore ADT is the following:

$$\begin{array}{rcl} e_{\text{sem2}} & := & \mathsf{new}\ \alpha' {\approx} \mathsf{bool}\ \mathsf{in} \\ & & \mathsf{pack}\ \langle \alpha', \langle \mathsf{true}, \lambda x{:}\,\mathsf{bool}\,.\neg x, \lambda x{:}\,\mathsf{bool}\,.x \rangle \rangle\ \mathsf{as}\ \tau_{\text{sem}} \end{array}$$

Here, the semaphore state is represented directly by a Boolean flag and does not rely on any additional invariant. If we can show that $e_{\text{sem1}}$ is contextually equivalent to $e_{\text{sem2}}$, then we can conclude that $e_{\text{sem1}}$'s type representation is truly being held abstract.

## 2.4 Contextual Equivalence

In order to be able to reason about representation independence, we need to make precise the notion of contextual equivalence.

A context $C$ is an expression with a single hole $[\_]$, defined in the usual manner. Typing of contexts is defined by a judgment form $\vdash C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau')$, where the triple $(\Delta; \Gamma; \tau)$

indicates the type of the hole. The judgment implies that for any expression $e$ with $\Delta;\Gamma \vdash e : \tau$ we have $\Delta';\Gamma' \vdash C[e] : \tau'$. The rules are straightforward, the key rule being the one for holes:

$$\frac{\Delta' \vdash \Gamma' \qquad \Delta \subseteq \Delta' \qquad \Gamma \subseteq \Gamma'}{\vdash [\_] : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau)}$$

The full definition can be found in the appendix.

We can now define contextual approximation and contextual equivalence as follows:

**Definition 2.1** (Contextual Approximation & Equivalence)**.** Let $\Delta;\Gamma \vdash e_1 : \tau$ and $\Delta;\Gamma \vdash e_2 : \tau$.

$$\begin{array}{lcl}
\Delta;\Gamma \vdash e_1 \preceq e_2 : \tau & \stackrel{\text{def}}{\iff} & \forall C, \tau', \sigma. \\
& & \vdash \sigma \wedge \vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\sigma;\epsilon;\tau') \ \wedge \\
& & \sigma; C[e_1]\downarrow \implies \sigma; C[e_2]\downarrow \\
\Delta;\Gamma \vdash e_1 \simeq e_2 : \tau & \stackrel{\text{def}}{\iff} & \Delta;\Gamma \vdash e_1 \preceq e_2 : \tau \wedge \\
& & \Delta;\Gamma \vdash e_2 \preceq e_1 : \tau
\end{array}$$

That is, contextual approximation $\Delta;\Gamma \vdash e_1 \preceq e_2 : \tau$ means that for any well-typed program context $C$ with a hole of appropriate type, the termination of $C[e_1]$ implies the termination of $C[e_2]$. Contextual equivalence $\Delta;\Gamma \vdash e_1 \simeq e_2 : \tau$ is just approximation in both directions.

Considering that G does not explicitly contain any recursive or looping constructs, the reader may wonder why termination is used as the notion of "distinguishing observation" in our definition of contextual equivalence. The reason is that the cast operator, together with impredicative polymorphism, makes it possible to write well-typed non-terminating programs [11]. (This was Girard's reason for studying the J operator in the first place [9].) Moreover, using cast, one can encode arbitrary recursive function definitions (see appendix). Other forms of observation may then be encoded in terms of (non-)termination.

# 3 A Logical Relation for G: Main Ideas

Following Reynolds [20] and Mitchell [13], our general approach to reasoning about parametricity and representation independence is to define a *logical relation*. Essentially, logical relations give us a tractable way of proving that two terms are contextually equivalent, which in turn gives us a way of proving that abstract types are really abstract. Of course, since polymorphism in G is non-parametric, the definition of our logical relation in the cases of universal and existential types is somewhat unusual. To place our approach in context, we first review the traditional approach to defining logical relations for languages with parametric polymorphism, such as System F.

## 3.1 Logical Relations for Parametric Polymorphism

Although the technical meaning of "logical relation" is rather woolly, the basic idea is to define an equivalence (or approximation) relation on programs inductively, following the structure of their types. To take the canonical example of arrow types, we would say that two functions are logically related at the type $\tau_1 \to \tau_2$ if, when passed arguments that are logically related at $\tau_1$, either they both diverge or they both converge to values that are logically related at $\tau_2$. The *fundamental theorem* of logical relations states that the logical relation is a congruence with respect to the constructs of the language. Together with what Pitts [17] calls *adequacy*—*i.e.,* the fact that logically related terms have equivalent termination behavior—the fundamental theorem implies that logically related terms are contextually equivalent, since contextual equivalence is defined precisely to be the largest adequate congruence.

Traditionally, the parametric nature of polymorphism is made clear by the definition of the logical relation for universal and existential types. Intuitively, two type abstractions, $\lambda\alpha.e_1$ and $\lambda\alpha.e_2$, are logically related at type $\forall\alpha.\tau$ if they map related *type* arguments to related results. But what does it mean for two type arguments to be related? Moreover, once we settle on two related type arguments $\tau_1'$ and $\tau_2'$, at what type do we relate the results $e_1[\tau_1'/\alpha]$ and $e_2[\tau_2'/\alpha]$?

One approach would be to restrict "related type arguments" to be the *same* type $\tau'$. Thus, $\lambda\alpha.e_1$ and $\lambda\alpha.e_2$ would be logically related at $\forall\alpha.\tau$ iff, for any (closed) type $\tau'$, it is the case that $e_1[\tau'/\alpha]$ and $e_2[\tau'/\alpha]$ are logically related at the type $\tau[\tau'/\alpha]$. A key problem with this definition, however, is that, due to the quantification over *any* argument type $\tau'$, the type $\tau[\tau'/\alpha]$ may in fact be larger than the type $\forall\alpha.\tau$, and thus the definition of the logical relation is no longer inductive in the structure of the type. Another problem is that this definition does not tell us anything about the parametric nature of polymorphism.

Reynolds' alternative approach is a generalization of Girard's "candidates" method for proving strong normalization for System F [9]. The idea is simple: instead of defining two type arguments to be related only if they are the same, allow *any* two different type arguments to be related by an (almost) arbitrary relational interpretation (subject to certain *admissibility* constraints). That is, we parameterize the logical relation at type $\tau$ by an interpretation function $\rho$, which maps each free type variable of $\tau$ to a pair of types $\tau_1', \tau_2'$ together with some (admissible) relation between values of those types. Then, we say that $\lambda\alpha.e_1$ and $\lambda\alpha.e_2$ are logically related at type $\forall\alpha.\tau$ under interpretation $\rho$ iff, for any closed types $\tau_1'$ and $\tau_2'$ and any relation $R$ between values of those types, it is the case that $e_1[\tau_1'/\alpha]$ and $e_2[\tau_2'/\alpha]$ are logically related at type $\tau$ under interpretation $\rho, \alpha \mapsto (\tau_1', \tau_2', R)$.

The miracle of Reynolds/Girard's method is that it simultaneously (1) renders the logical relation inductively well-defined in the structure of the type, and (2) demonstrates the parametricity of polymorphism: logically related type abstractions must behave the same even when passed completely different type arguments, so their behavior may not analyze the type argument and behave in different ways for different arguments. Dually, we can show that two ADTs $\mathsf{pack}\ \langle\tau_1, v_1\rangle\ \mathsf{as}\ \exists\alpha.\tau$ and $\mathsf{pack}\ \langle\tau_2, v_2\rangle\ \mathsf{as}\ \exists\alpha.\tau$ are logically related (and thus contextually equivalent) by exhibiting *some* relational interpretation $R$ for the abstract type $\alpha$, even if the underlying type representations $\tau_1$ and $\tau_2$ are different. This is the essence of what is meant by "representation independence".

Unfortunately, in the setting of G, Reynolds/Girard's method is not directly applicable, precisely because polymorphism in G is not parametric! Two ADTs can easily be distinguished if their underlying representation type is not the same. Furthermore, since $\mathsf{cast}$ can be used to forge values of the "abstract" type, it is no longer sufficient that there is some relational interpretation, but it must be the full logical relation at the representation type. This forces us back to the first approach suggested above, namely to only consider type arguments to be logically related if they are equal, and to not allow any freedom in interpreting the abstract type.

The natural questions, then, are: (1) what metric do we use to define the logical relation inductively, since the structure of the type no longer suffices, and (2) how do we establish that dynamic type generation regains a form of parametricity? We address these questions in the next two sections, respectively.

## 3.2 Step-Indexed Logical Relations for Non-Parametricity

First, in order to provide a metric for inductively defining the logical relation, we employ *step-indexing*. Step-indexed logical relations were proposed originally by Appel and McAllester [7] as a way of giving a simple operational-semantics-based model for general recursive types in the context of foundational proof-carrying code. In subsequent work by Ahmed and others [3, 6],

the method has been adapted to support relational reasoning in a variety of settings, including untyped and imperative languages.

The key idea of step-indexed logical relations is to index the definition of the logical relation not only by the type of the programs being related, but also by a natural number $n$ representing (intuitively) "the number of steps left in the computation". That is, if two terms $e_1$ and $e_2$ are logically related at type $\tau$ for $n$ steps, then if we place them in any program context $C$ and run the resulting programs for $n$ steps of computation, we should not be able to produce observably different results (*e.g.,* $C[e_1]$ evaluating to 5 and $C[e_2]$ evaluating to 7). To show that $e_1$ and $e_2$ are contextually equivalent, then, it suffices to show that they are logically related for $n$ steps, for any $n$.

To see how step-indexing helps us, consider how we might define a step-indexed logical relation for G in the case of universal types: two type abstractions $\lambda\alpha.e_1$ and $\lambda\alpha.e_2$ are logically related at $\forall\alpha.\tau$ for $n$ steps iff, for any type argument $\tau'$, it is the case that $e_1[\tau'/\alpha]$ and $e_2[\tau'/\alpha]$ are logically related at $\tau[\tau'/\alpha]$ for $n-1$ steps. This reasoning is sound because the only way a program context can distinguish between $\lambda\alpha.e_1$ and $\lambda\alpha.e_2$ in $n$ steps is by first applying them to a type argument $\tau'$—which incurs a step of computation for the $\beta$-reduction $(\lambda\alpha.e_i)\,\tau' \hookrightarrow e_i[\tau'/\alpha]$—and then distinguishing between $e_1[\tau'/\alpha]$ and $e_2[\tau'/\alpha]$ within the next $n-1$ steps. Moreover, although the type $\tau[\tau'/\alpha]$ may be larger than $\forall\alpha.\tau$, the step index $n-1$ is smaller, so the logical relation is inductively well-defined.

### 3.3 Kripke Logical Relations for Dynamic Parametricity

Second, in order to establish the parametricity properties of dynamic type generation, we employ *Kripke logical relations*, *i.e.,* logical relations that are indexed by *possible worlds*.[6] Kripke logical relations are appropriate when reasoning about properties that are true only under certain conditions, such as equivalence of modules with local mutable state. For instance, an imperative ADT might only behave according to its specification if its local data structures obey certain invariants. Possible worlds allow one to codify such *local invariants* on the machine store [18].

In our setting, the local invariant we want to establish is what a dynamically generated type name *means*. That is, we will use possible worlds to assign relational interpretations to dynamically generated type names. For example, consider the programs $e_{\text{sem1}}$ and $e_{\text{sem2}}$ from Section 2. We want to show they are logically related at $\exists\alpha.\ \alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool})$ in an empty initial world $w_0$ (*i.e.,* under empty type stores). The proof proceeds roughly as follows. First, we evaluate the two programs. This will have the effect of generating a fresh type name $\alpha'$, with $\alpha' \approx \mathsf{int}$ extending the type store of the first program and $\alpha' \approx \mathsf{bool}$ extending the type store of the second program. At this point, we correspondingly extend the initial world $w_0$ with a mapping from $\alpha'$ to the relation $R = \{(1, \mathsf{true}), (0, \mathsf{false})\}$, thus forming a new world $w$ that specifies the semantic meaning of $\alpha'$.

We now must show that the values

$$\mathsf{pack}\,\langle\alpha', \langle 1, \lambda x{:}\,\mathsf{int}\,.(1-x), \lambda x{:}\,\mathsf{int}\,.(x \neq 0)\rangle\rangle\ \mathsf{as}\ \tau_{\text{sem}}$$

and

$$\mathsf{pack}\,\langle\alpha', \langle \mathsf{true}, \lambda x{:}\,\mathsf{bool}\,.\neg x, \lambda x{:}\,\mathsf{bool}\,.x\rangle\rangle\ \mathsf{as}\ \tau_{\text{sem}}$$

are logically related in the world $w$. Since G's logical relation for existential types is non-parametric, the two packages must have the *same* type representation, but of course the whole point of using new was to ensure that they do (namely, it is $\alpha'$). The remainder of the proof is

[6] In fact, step-indexed logical relations may already be understood as a special case of Kripke logical relations, in which the step index serves as the notion of possible world, and where $n$ is a future world of $m$ iff $n \leq m$.

showing that the value components of the packages are related at the type $\alpha' \times (\alpha' \to \alpha') \times (\alpha' \to \mathsf{bool})$ under the interpretation $\rho = \alpha' \mapsto (\mathsf{int}, \mathsf{bool}, R)$ derived from the world $w$. This last part is completely analogous to what one would show in a standard representation independence proof.

In short, the possible worlds in our Kripke logical relations bring back the ability to assign arbitrary relational interpretations $R$ to abstract types, an ability that was seemingly lost when we moved to a non-parametric logical relation. The only catch is that we can only assign arbitrary interpretations to *dynamic* type names, not to *static*, universally/existentially quantified type variables.

There is one minor technical matter that we glossed over in the above proof sketch but is worth mentioning. Due to nondeterminism of type name allocation, the evaluation of $e_{\mathrm{sem1}}$ and $e_{\mathrm{sem2}}$ may result in $\alpha'$ being replaced by $\alpha'_1$ in the former and $\alpha'_2$ in the latter (for some fresh $\alpha'_1 \neq \alpha'_2$). Moreover, we are also interested in proving equivalence of programs that do not necessarily allocate exactly the same number of type names in the same order.

Consequently, we also include in our possible worlds a partial bijection $\eta$ between the type names of the first program and the type names of the second program, which specifies how each dynamically generated abstract type is concretely represented in the stores of the two programs. We require them to be in 1-1 correspondence because the $\mathsf{cast}$ construct permits the program context to observe equality on type names, as follows:

$$\begin{array}{l}\mathsf{equal?} : \forall\alpha_1.\forall\alpha_2.\,\mathsf{bool} \stackrel{\mathrm{def}}{=} \\ \quad \Lambda\alpha_1.\Lambda\alpha_2.\,\mathsf{cast}\ ((\alpha_1 \to \alpha_1) \to \mathsf{bool})\ ((\alpha_2 \to \alpha_2) \to \mathsf{bool}) \\ \qquad\qquad\qquad (\lambda x{:}(\alpha_1 \to \alpha_1).\,\mathsf{true})\ (\lambda x{:}(\alpha_2 \to \alpha_2).\,\mathsf{false})\ (\lambda x{:}\alpha_2.x)\end{array}$$

We then consider types to be logically related if they are the same *up to* this bijection. For instance, in our running example, when extending $w_0$ to $w$, we would not only extend its relational interpretation with $\alpha' \mapsto (\mathsf{int}, \mathsf{bool}, R)$ but also extend its $\eta$ with $\alpha' \mapsto (\alpha'_1, \alpha'_2)$. Thus, the type representations of the two existential packages, $\alpha'_1$ and $\alpha'_2$, though syntactically distinct, would still be logically related under $w$.

# 4 A Logical Relation for G: Formal Details

Figures 3 and 4 display our step-indexed Kripke logical relation for G in full gory detail. It is easiest to understand this definition by making two passes over it. As the step indices have a way of infecting the whole definition in a superficially complex—but really very straightforward—way, we will first walk through the whole definition *ignoring* all occurrences of $n$'s and $k$'s (as well as auxiliary functions like the $\lfloor\cdot\rfloor_n$ operator). Second, we will pinpoint the few places where step indices actually play an important role in ensuring that the logical relation is inductively well-founded.

## 4.1 Highlights of the Logical Relation

The first section of Figure 3 defines the kinds of semantic objects that the construction of the logical relation is based on (prefixed by Cand for "candidate", but not to be confused with Girard's candidates from Section 3). Following [6], those are then refined by imposing various constraints to yield the objects we are actually interested in. Relations $R$ are sets of *atoms*, which are pairs of terms, $e_1$ and $e_2$, indexed by a possible world $w$. The definition of $\mathrm{Atom}[\tau_1, \tau_2]$ requires that $e_1$ and $e_2$ have the types $\tau_1$ and $\tau_2$ under the type stores $w.\sigma_1$ and $w.\sigma_2$, respectively. (We use the dot notation $w.\sigma_i$ to denote the $i$-th type store component of $w$, and analogous notation for projecting out the other components of worlds.)

$$
\begin{array}{lcl}
R^{\mathrm{val}} & \stackrel{\mathrm{def}}{=} & \{(k, w, v_1, v_2) \mid (k, w, v_1, v_2) \in R\} \\
\\
\mathrm{CandAtom}_n & \stackrel{\mathrm{def}}{=} & \{(k, w, e_1, e_2) \mid k < n \wedge w \in \mathrm{CandWorld}_k\} \\
\mathrm{CandRel}_n & \stackrel{\mathrm{def}}{=} & \{R \subseteq \mathrm{CandAtom}_n^{\mathrm{val}}\} \\
\mathrm{CandSomeRel}_n & \stackrel{\mathrm{def}}{=} & \{(\tau_1, \tau_2, R) \mid R \in \mathrm{CandRel}_n\} \\
\mathrm{CandInterp}_n & \stackrel{\mathrm{def}}{=} & \{\rho \in \mathrm{TVar} \xrightarrow{\mathrm{fin}} \mathrm{CandSomeRel}_n\} \\
\mathrm{CandConc} & \stackrel{\mathrm{def}}{=} & \mathrm{TVar} \xrightarrow{\mathrm{fin}} \mathrm{TVar} \times \mathrm{TVar} \\
\mathrm{CandWorld}_n & \stackrel{\mathrm{def}}{=} & \{(\sigma_1, \sigma_2, \eta, \rho) \mid \eta \in \mathrm{CandConc} \wedge \rho \in \mathrm{CandInterp}_n\} \\
\mathrm{CandInterp} & \stackrel{\mathrm{def}}{=} & \bigcup_{n \geq 0} \mathrm{CandInterp}_n \\
\mathrm{CandWorld} & \stackrel{\mathrm{def}}{=} & \bigcup_{n \geq 0} \mathrm{CandWorld}_n \\
\\
\mathrm{Atom}_n[\tau_1, \tau_2] & \stackrel{\mathrm{def}}{=} & \{(k, w, e_1, e_2) \mid k < n \wedge w \in \mathrm{World}_k \wedge \vdash w.\sigma_1; e_1 : \tau_1 \wedge \vdash w.\sigma_2; e_2 : \tau_2\} \\
\mathrm{Rel}_n[\tau_1, \tau_2] & \stackrel{\mathrm{def}}{=} & \{R \subseteq \mathrm{Atom}_n^{\mathsf{val}}[\tau_1, \tau_2] \mid \\
 & & \quad \forall (k, w, v_1, v_2) \in R.\ \forall (k', w') \sqsupseteq (k, w).\ (k', w', v_1, v_2) \in R\} \\
\mathrm{SomeRel}_n & \stackrel{\mathrm{def}}{=} & \{r = (\tau_1, \tau_2, R) \mid \mathrm{ftv}(\tau_1) = \mathrm{ftv}(\tau_2) = \emptyset \wedge R \in \mathrm{Rel}_n[\tau_1, \tau_2]\} \\
\mathrm{Interp}_n & \stackrel{\mathrm{def}}{=} & \{\rho \in \mathrm{TVar} \xrightarrow{\mathrm{fin}} \mathrm{SomeRel}_n\} \\
\mathrm{Conc} & \stackrel{\mathrm{def}}{=} & \{\eta \in \mathrm{TVar} \xrightarrow{\mathrm{fin}} \mathrm{TVar} \times \mathrm{TVar} \mid \\
 & & \quad \forall \alpha, \alpha' \in \mathrm{dom}(\eta).\ \alpha \neq \alpha' \Rightarrow \eta^1(\alpha) \neq \eta^1(\alpha') \wedge \eta^2(\alpha) \neq \eta^2(\alpha')\} \\
\mathrm{World}_n & \stackrel{\mathrm{def}}{=} & \{w = (\sigma_1, \sigma_2, \eta, \rho) \mid \vdash \sigma_1 \wedge \vdash \sigma_2 \wedge \eta \in \mathrm{Conc} \wedge \rho \in \mathrm{Interp}_n \wedge \\
 & & \quad \mathrm{dom}(\eta) = \mathrm{dom}(\rho) \wedge \rho^1 = \sigma_1^* \cdot \eta^1 \wedge \rho^2 = \sigma_2^* \cdot \eta^2\} \\
\mathrm{Interp} & \stackrel{\mathrm{def}}{=} & \bigcup_{n \geq 0} \mathrm{Interp}_n \\
\mathrm{World} & \stackrel{\mathrm{def}}{=} & \bigcup_{n \geq 0} \mathrm{World}_n
\end{array}
$$

$$
\begin{array}{rcl}
\lfloor (\sigma_1, \sigma_2, \eta, \rho) \rfloor_n & \stackrel{\mathrm{def}}{=} & (\sigma_1, \sigma_2, \eta, \lfloor \rho \rfloor_n) \\
\lfloor \rho \rfloor_n & \stackrel{\mathrm{def}}{=} & \{\alpha \mapsto \lfloor r \rfloor_n \mid \rho(\alpha) = r\} \\
\lfloor (\tau_1, \tau_2, R) \rfloor_n & \stackrel{\mathrm{def}}{=} & (\tau_1, \tau_2, \lfloor R \rfloor_n) \\
\lfloor R \rfloor_n & \stackrel{\mathrm{def}}{=} & \{(k, w, e_1, e_2) \in R \mid k < n\} \\
\\
\triangleright R & \stackrel{\mathrm{def}}{=} & \{(k, w, e_1, e_2) \mid \forall (k', w') \sqsupset (k, w).\ (k', w', e_1, e_2) \in R\} \\
\\
(k', w') \sqsupseteq (k, w) & \stackrel{\mathrm{def}}{\Longleftrightarrow} & k' \leq k \wedge w' \in \mathrm{World}_{k'} \wedge w'.\eta \sqsupseteq w.\eta \wedge w'.\rho \sqsupseteq \lfloor w.\rho \rfloor_{k'} \wedge \forall i \in \{1, 2\}. \\
 & & w'.\sigma_i \supseteq w.\sigma_i \wedge \mathrm{rng}(w'.\eta^i) - \mathrm{rng}(w.\eta^i) \subseteq \mathrm{dom}(w'.\sigma_i) - \mathrm{dom}(w.\sigma_i) \\
\eta' \sqsupseteq \eta & \stackrel{\mathrm{def}}{\Longleftrightarrow} & \forall \alpha \in \mathrm{dom}(\eta).\ \eta'(\alpha) = \eta(\alpha) \\
\rho' \sqsupseteq \rho & \stackrel{\mathrm{def}}{\Longleftrightarrow} & \forall \alpha \in \mathrm{dom}(\rho).\ \rho'(\alpha) = \rho(\alpha) \\
(k', w') \sqsupset (k, w) & \stackrel{\mathrm{def}}{\Longleftrightarrow} & k' < k \wedge (k', w') \sqsupseteq (k, w)
\end{array}
$$

Figure 3: Logical Relation for G

$$
\begin{array}{lcl}
V_n[\![\alpha]\!]\rho & \stackrel{\text{def}}{=} & \lfloor \rho(\alpha).R \rfloor_n \\
V_n[\![b]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,c,c) \in \mathrm{Atom}_n[b,b]\} \\
V_n[\![\tau \times \tau']\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\langle v_1,v_1'\rangle,\langle v_2,v_2'\rangle) \in \mathrm{Atom}_n[\rho^1(\tau\times\tau'),\rho^2(\tau\times\tau')] \mid \\
 & & \quad (k,w,v_1,v_2) \in V_n[\![\tau]\!]\rho \wedge (k,w,v_1',v_2') \in V_n[\![\tau']\!]\rho\} \\
V_n[\![\tau' \to \tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\lambda x{:}\tau_1.e_1,\lambda x{:}\tau_2.e_2) \in \mathrm{Atom}_n[\rho^1(\tau'\to\tau),\rho^2(\tau'\to\tau)] \mid \\
 & & \quad \forall (k',w',v_1,v_2) \in V_n[\![\tau']\!]\rho.\ (k',w') \sqsupseteq (k,w) \Rightarrow \\
 & & \quad (k',w',e_1[v_1/x],e_2[v_2/x]) \in E_n[\![\tau]\!]\rho\} \\
V_n[\![\forall\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\Lambda\alpha.e_1,\Lambda\alpha.e_2) \in \mathrm{Atom}_n[\rho^1(\forall\alpha.\tau),\rho^2(\forall\alpha.\tau)] \mid \\
 & & \quad \forall (k',w') \sqsupseteq (k,w).\ \forall(\tau_1,\tau_2,r) \in T_{k'}[\![\Omega]\!]w'. \\
 & & \quad (k',w',e_1[\tau_1/\alpha],e_2[\tau_2/\alpha]) \in \triangleright E_n[\![\tau]\!]\rho, \alpha{\mapsto}r\} \\
V_n[\![\exists\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\mathsf{pack}\ \langle\tau_1,v_1\rangle\ \mathsf{as}\ \tau_1', \mathsf{pack}\ \langle\tau_2,v_2\rangle\ \mathsf{as}\ \tau_2') \in \mathrm{Atom}_n[\rho^1(\exists\alpha.\tau),\rho^2(\exists\alpha.\tau)] \mid \\
 & & \quad \exists r.\ (\tau_1,\tau_2,r) \in T_k[\![\Omega]\!]w \wedge (k,w,v_1,v_2) \in \triangleright V_n[\![\tau]\!]\rho, \alpha{\mapsto}r\} \\
E_n[\![\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,e_1,e_2) \in \mathrm{Atom}_n[\rho^1(\tau),\rho^2(\tau)] \mid \\
 & & \quad \forall j<k.\ \forall\sigma_1,v_1.\ (w.\sigma_1; e_1 \hookrightarrow^j \sigma_1; v_1) \Rightarrow \\
 & & \quad \exists w', v_2.\ (k-j,w') \sqsupseteq (k,w) \wedge w'.\sigma_1 = \sigma_1 \wedge \\
 & & \quad (w.\sigma_2; e_2 \hookrightarrow^* w'.\sigma_2; v_2) \wedge (k-j,w',v_1,v_2) \in V_n[\![\tau]\!]\rho\} \\
 & & \\
T_n[\![\Omega]\!]w & \stackrel{\text{def}}{=} & \{(w.\eta^1(\tau), w.\eta^2(\tau), (w.\rho^1(\tau), w.\rho^2(\tau), V_n[\![\tau]\!]w.\rho)) \mid \mathrm{ftv}(\tau) \subseteq \mathrm{dom}(w.\rho)\} \\
 & & \\
G_n[\![\epsilon]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\emptyset,\emptyset) \mid k<n \wedge w \in \mathrm{World}_k\} \\
G_n[\![\Gamma, x{:}\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,(\gamma_1, x{\mapsto}v_1),(\gamma_2,x{\mapsto}v_2)) \mid \\
 & & \quad (k,w,\gamma_1,\gamma_2) \in G_n[\![\Gamma]\!]\rho \wedge (k,w,v_1,v_2) \in V_n[\![\tau]\!]\rho\} \\
D_n[\![\epsilon]\!]w & \stackrel{\text{def}}{=} & \{(\emptyset,\emptyset,\emptyset)\} \\
D_n[\![\Delta,\alpha]\!]w & \stackrel{\text{def}}{=} & \{((\delta_1,\alpha{\mapsto}\tau_1),(\delta_2,\alpha{\mapsto}\tau_2),(\rho,\alpha{\mapsto}r)) \mid \\
 & & \quad (\delta_1,\delta_2,\rho) \in D_n[\![\Delta]\!]w \wedge (\tau_1,\tau_2,r) \in T_n[\![\Omega]\!]w\} \\
D_n[\![\Delta,\alpha{\approx}\tau]\!]w & \stackrel{\text{def}}{=} & \{((\delta_1,\alpha{\mapsto}\alpha_1),(\delta_2,\alpha{\mapsto}\alpha_2),(\rho,\alpha{\mapsto}(\rho^1(\tau),\rho^2(\tau),V_n[\![\tau]\!]\rho))) \mid \\
 & & \quad (\delta_1,\delta_2,\rho) \in D_n[\![\Delta]\!]w \wedge w.\sigma_i(\alpha_i) = \delta_i(\tau) \wedge \\
 & & \quad \exists\alpha'.\ \alpha_i = w.\eta^i(\alpha') \wedge w.\rho(\alpha').R = V_n[\![\tau]\!]\rho\}
\end{array}
$$

$$
\Delta;\Gamma \vdash e_1 \precsim e_2 : \tau \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad
\begin{array}{l}
\Delta;\Gamma \vdash e_1 : \tau \wedge \Delta;\Gamma \vdash e_2 : \tau \wedge \forall n \geq 0.\ \forall w_0 \in \mathrm{World}_n\,. \\
\forall(\delta_1,\delta_2,\rho) \in D_n[\![\Delta]\!]w_0.\ \forall(k,w,\gamma_1,\gamma_2) \in G_n[\![\Gamma]\!]\rho. \\
(k,w) \sqsupset (n,w_0) \Rightarrow (k,w,\delta_1\gamma_1(e_1),\delta_2\gamma_2(e_2)) \in E_n[\![\tau]\!]\rho
\end{array}
$$

Figure 4: Logical Relation for G (contd.)

$\mathrm{Rel}[\tau_1,\tau_2]$ defines the set of *admissible* relations, which are permitted to be used as the semantic interpretations of abstract types. For our purposes, admissibile simply means containing only value atoms and being *monotonic*, *i.e.,* closed under world extension: If a relation in $\mathrm{Rel}[\tau_1,\tau_2]$ relates two values $v_1$ and $v_2$ under a world $w$, then the relation must relate those values in any future world of $w$. (We discuss the definition of world extension below.) Monotonicity is needed in order to ensure that we can extend worlds with interpretations of new dynamic type names, without interfering somehow with the interpretations of the old ones.

Worlds $w$ are 4-tuples $(\sigma_1,\sigma_2,\eta,\rho)$, which describe a set of assumptions under which two

terms are related. Here, $\sigma_1$ and $\sigma_2$ are the type stores under which the terms are typechecked and evaluated. The finite mappings $\eta$ and $\rho$ share a common domain, which can be understood as the set of abstract type names that have been generated dynamically (TVar denotes the infinite set of type variables; recall that, syntactically, type names are just variables). These "semantic" type names do not exist in either store $\sigma_1$ or $\sigma_2$.[7] Rather, they provide a way of referring to an abstract type that is represented by *some* type name $\alpha_1$ in $\sigma_1$ and *some* type name $\alpha_2$ in $\sigma_2$. Thus, for each name $\alpha \in \mathrm{dom}(\eta) = \mathrm{dom}(\rho)$, the *concretization* $\eta$ maps the "semantic" name $\alpha$ to a pair of "concrete" names from the stores $\sigma_1$ and $\sigma_2$, respectively. (See the end of Section 3.3 for an example of such an $\eta$.) As the definition of Conc makes clear, distinct semantic type names must have distinct concretizations; consequently, $\eta$ represents a *partial bijection* between $\sigma_1$ and $\sigma_2$.

The last component of the world $w$ is $\rho$, which assigns relational interpretations to the aforementioned semantic type names. Formally, $\rho$ maps each $\alpha$ to a triple $r = (\tau_1, \tau_2, R)$, where $R$ is a monotone relation between values of types $\tau_1$ and $\tau_2$. (Again, see the end of Section 3.3 for an example of such a $\rho$.) The final condition in the definition of World ensures that its components actually fit together, by stipulating that applying $\rho$ as a syntactic type substitution is the same as first applying the corresponding concretizations and then normalizing the result relative to the corresponding type store. As a matter of notation, we write $\cdot$ for function composition and $\eta^i$ and $\rho^i$ to denote the type substitutions $\{\alpha \mapsto \alpha_i \mid \eta(\alpha) = (\alpha_1, \alpha_2)\}$ and $\{\alpha \mapsto \tau_i \mid \rho(\alpha) = (\tau_1, \tau_2, R)\}$, respectively.

The second section of Figure 3 displays the definition of world extension. In order for $w'$ to extend $w$ (written $w' \sqsupseteq w$), it must be the case that (1) $w'$ specifies semantic interpretations for a superset of the type names that $w$ interprets, (2) for the names that $w$ interprets, $w'$ must interpret them in the same way, and (3) any new semantic type names that $w'$ interprets may only correspond to *new* concrete type names that did not exist in the stores of $w$. Although the third condition is not strictly necessary, we have found it to be useful when proving certain examples (*e.g.,* the "order independence" example in Section 4.4).

Figure 4 shows the logical relation itself:

- Given $n \geq 0$, $\tau$ and $\rho \in \mathrm{CandInterp}$ with $\mathrm{ftv}(\tau) \subseteq \mathrm{dom}(\rho)$, we define the logical relation for values, $V_n[\![\tau]\!]\rho \in \mathrm{CandRel}_n$, and the one for terms, $E_n[\![\tau]\!]\rho \subseteq \mathrm{CandAtom}_n$.
- Given $n \geq 0$ and $w \in \mathrm{CandWorld}$, we define the logical relation for *types as data*, $T_n[\![\Omega]\!]w$, as described in Section 3 (here, $\Omega$ represents the *kind* of types).

The candidate sets are useful for making it clear that the logical relation is well-defined. However, we will only ever work with interpretations out of Interp and worlds out of World. One of the first properties we show in Section 10 is that in this case the value relation is not just in CandRel, but in Rel, *i.e.,* admissible.

$V[\![\tau]\!]\rho$ relates values at the type $\tau$, where the free type variables of $\tau$ are given relational interpretations by $\rho$. Ignoring the step indices, $V[\![\tau]\!]\rho$ is mostly very standard. For instance, at certain points (namely, in the $\rightarrow$ and $\forall$ cases), when we quantify over logically related (value or type) arguments, we must allow them to come from an arbitrary future world $w'$ in order to ensure monotonicity. This kind of quantification over future worlds is commonplace in Kripke logical relations.

The only really interesting bit in the definition of $V[\![\tau]\!]\rho$ is the use of $T[\![\Omega]\!]w$ to characterize when the two *type* arguments (resp. components) of a universal (resp. existential) are logically related. As explained in Section 3.3, we consider two types to be logically related in world $w$ iff they are the same up to the partial bijection $w.\eta$. Formally, we define $T[\![\Omega]\!]w$ as a relation on

[7] In fact, technically speaking, we consider $\mathrm{dom}(\eta) = \mathrm{dom}(\rho)$ to be bound variables of the world $w$.

triples $(\tau_1, \tau_2, r)$, where $\tau_1$ and $\tau_2$ are the two logically related types and $r$ is a relation telling us how to relate values of those types. To be logically related means that $\tau_1$ and $\tau_2$ are the concretizations (according to $w.\eta$) of some "semantic" type $\tau'$. Correspondingly, $r$ is the logical relation $V[\![\tau']\!]w.\rho$ at that semantic type. Thus, when we write $E[\![\tau]\!]\rho, \alpha \mapsto r$ in the definition of $V[\![\forall\alpha.\tau]\!]\rho$, this is roughly equivalent to writing $E[\![\tau[\tau'/\alpha]]\!]\rho$ (which our discussion in Section 3.2 might have led the reader to expect to see here instead). The reason for our present formulation is that $E[\![\tau[\tau'/\alpha]]\!]\rho$ is not quite right: the free variables of $\tau$ are interpreted by $\rho$, but the free variables of $\tau'$ are *dynamic* type names whose interpretations are given by $w.\rho$. It is possible to merge $\rho$ and $w.\rho$ into a unified interpretation $\rho'$, but we feel our present approach is cleaner.

Another point of note: since $r$ is uniquely determined from $\tau_1$ and $\tau_2$, it is not really necessary to include it in the $T[\![\Omega]\!]w$ relation. However, as we shall see in Section 6, formulating the logical relation in this way has the benefit of isolating all of the non-parametricity of our logical relation in the definition of $T[\![\Omega]\!]w$.

The term relation $E[\![\tau]\!]\rho$ is very similar to that in previous step-indexed Kripke logical relations [6]. Briefly, it says that two terms are related in an initial world $w$ if whenever the first evaluates to a value under $w.\sigma_1$, the second evaluates to a value under $w.\sigma_2$, and the resulting stores and values are related in some future world $w'$.

The remainder of the definitions in Figure 4 serve to formalize a logical relation for *open* terms. $G[\![\Gamma]\!]\rho$ is the logical relation on value substitutions $\gamma$, which asserts that related $\gamma$'s must map variables in $\mathrm{dom}(\Gamma)$ to related values. $D[\![\Delta]\!]w$ is the logical relation on type substitutions. It asserts that related $\delta$'s must map variables in $\mathrm{dom}(\Delta)$ to types that are related in $w$. For type variables $\alpha$ bound as $\alpha \approx \tau$, the $\delta$'s must map $\alpha$ to a type name whose semantic interpretation in $w$ is precisely the logical relation at $\tau$. Analogously to $T[\![\Omega]\!]w$, the relation $D[\![\Delta]\!]w$ also includes a relational interpretation $\rho$, which may be uniquely determined from the $\delta$'s.

Finally, the open logical relation $\Delta; \Gamma \vdash e_1 \precsim e_2 : \tau$ is defined in a fairly standard way. It says that for any starting world $w_0$, and any type substitutions $\delta_1$ and $\delta_2$ related in that world, if we are given related value substitutions $\gamma_1$ and $\gamma_2$ in any future world $w$, then $\delta_1\gamma_1 e_1$ and $\delta_2\gamma_2 e_2$ are related in $w$ as well.

## 4.2 Why and Where the Steps Matter

As we explained in Section 3.2, step indices play a critical role in making the logical relation well-founded. Essentially, whenever we run into an apparent circularity, we "go down a step" by defining an $n$-level property in terms of an $(n-1)$-level one. Of course, this trick only works if, at all such "stepping points", the only way that an adversarial program context could possibly tell whether the $n$-level property holds or not is by taking one step of computation and then checking whether the underlying $(n-1)$-level property holds. Fortunately, this is the case.

Since worlds contain relations, and relations contain sets of tuples that include worlds, a naïve construction of these objects would have an inconsistent cardinality. We thus stratify both worlds and relations by a step index: $n$-level worlds $w \in \mathrm{World}_n$ contain $n$-level interpretations $\rho \in \mathrm{Interp}_n$, which map type variables to $n$-level relations; $n$-level relations $R \in \mathrm{Rel}_n[\tau_1, \tau_2]$ only contain atoms indexed by a step level $k < n$ and a world $w \in \mathrm{World}_k$. Although our possible worlds have a different structure than in previous work, the technique of mutual world and relation stratification is similar to that used in Ahmed's thesis [2], as well as recent work by Ahmed, Dreyer and Rossberg [6].

Intuitively, the reason this works in our setting is as follows. Viewed as a judgment, our logical relation asserts that two terms $e_1$ and $e_2$ are logically related for $k$ steps in a world $w$ at a type $\tau$ under an interpretation $\rho$ (whose domain contains the free type variables of $\tau$). Clearly, in order to handle the case where $\tau$ is just a type variable $\alpha$, the relations $r$ in the range of $\rho$

must include atoms at step index $k$ (*i.e.,* the $r$'s must be in $\text{SomeRel}_{k+1}$).

But what about the relations in the range of $w.\rho$? Those relations only come into play in the universal and existential cases of the logical relation for values. Consider the existential case (the universal one is analogous). There, $w.\rho$ pops up in the definition of the relation $r$ that comes from $T_k[\![\Omega]\!]w$. However, that $r$ is only needed in defining the relatedness of the values $v_1$ and $v_2$ at a step level less than $k$ (note the definition of $\triangleright R$ in the second section of Figure 3). Consequently, we only need $r$ to include atoms at step $k-1$ and lower (*i.e.,* $r$ must be in $\text{SomeRel}_k$), so the world $w$ from which $r$ is derived need only be in $\text{World}_k$.

As this discussion suggests, it is *imperative* that we "go down a step" in the universal and existential cases of the logical relation. For the other cases, it is not necessary to go down a step, although we have the option of doing so. For example, we could define $k$-level relatedness at pair type $\tau_1 \times \tau_2$ in terms of $(k-1)$-level relatedness at $\tau_1$ and $\tau_2$. But since the type gets smaller, there is no need to. For clarity, we have only gone down a step in the logical relation at the points where it is absolutely necessary, and we have used the $\triangleright$ notation to underscore those points.

## 4.3 Key Properties

The main results concerning our logical relation are as follows:

**Theorem 4.1** (Fundamental Property for $\precsim$)**.** If $\Delta;\Gamma \vdash e : \tau$, then $\Delta;\Gamma \vdash e \precsim e : \tau$.

**Theorem 4.2** (Soundness of $\precsim$ wrt. Contextual Approximation)**.** If $\Delta;\Gamma \vdash e_1 \precsim e_2 : \tau$, then $\Delta;\Gamma \vdash e_1 \preceq e_2 : \tau$.

These theorems establish that our logical relation provides a sound technique for proving contextual equivalence of G programs. The proofs of these theorems rely on many technical lemmas. Details are to be found in Section 10.

## 4.4 Examples

**Semaphore.** We now return to our semaphore example from Section 2 and show how to prove representation independence for the two different implementations $e_{\text{sem1}}$ and $e_{\text{sem2}}$. Recall that the former uses $\mathsf{int}$, the latter $\mathsf{bool}$. To show that they are contextually equivalent, it suffices by Soundness to show that each logically approximates the other. We prove only one direction, namely $\vdash e_{\text{sem1}} \precsim e_{\text{sem2}} : \tau_{\text{sem}}$; the other is proven analogously.

Expanding the definitions, we need to show $(k, w, e_{\text{sem1}}, e_{\text{sem2}}) \in E_n[\![\tau_{\text{sem}}]\!]\emptyset$. Note how each term generates a fresh type name $\alpha_i$ in one step, resulting in a package value. Hence all we need to do is come up with a world $w'$ satisfying

- $(k-1, w') \sqsupseteq (k, w)$,
- $w'.\sigma_1 = w.\sigma_1, \alpha_1 \approx \mathsf{int}$ and $w'.\sigma_2 = w.\sigma_2, \alpha_2 \approx \mathsf{bool}$,
- $(k-1, w', \mathsf{pack}\langle\alpha_1, v_1\rangle, \mathsf{pack}\langle\alpha_2, v_2\rangle) \in V_n[\![\tau_{\text{sem}}]\!]\emptyset$.

where $v_i$ is the term component of $e_{\text{sem}i}$'s implementation. We construct $w'$ by extending $w$ with mappings that establish the relation between the new type names:

$$\begin{aligned} R &:= \{(k'', w'', v_{\mathsf{int}}, v_{\mathsf{bool}}) \in \text{Atom}^{\mathsf{val}}_{k-1}[\mathsf{int}, \mathsf{bool}] \mid \\ &\qquad (v_{\mathsf{int}}, v_{\mathsf{bool}}) = (1, \mathsf{true}) \vee (v_{\mathsf{int}}, v_{\mathsf{bool}}) = (0, \mathsf{false})\} \\ r &:= (\mathsf{int}, \mathsf{bool}, R) \\ w' &:= \lfloor w \rfloor_{k-1} \uplus (\alpha_1 \approx \mathsf{int}, \alpha_2 \approx \mathsf{bool}, \alpha \mapsto (\alpha_1, \alpha_2), \alpha \mapsto r) \end{aligned}$$

The notation for extending a world is defined as follows:

$$w \uplus (\alpha_1{\approx}\tau_1, \alpha_2{\approx}\tau_2, \alpha{\mapsto}(\alpha_1,\alpha_2), \alpha{\mapsto}r) \stackrel{\text{def}}{=} ((w.\sigma_1, \alpha_1{\approx}\tau_1), (w.\sigma_2, \alpha_2{\approx}\tau_2), (w.\eta, \alpha{\mapsto}(\alpha_1,\alpha_2)), (w.\rho, \alpha{\mapsto}r))$$

The first two conditions above are satisfied by construction of $w'$. To show that the packages are related we need to show the existence of an $r'$ with $(\alpha_1, \alpha_2, r') \in T_{k-1}[\![\Omega]\!]w'$ such that $(k-2, \lfloor w' \rfloor_{k-2}, v_1, v_2) \in V_n[\![\tau'_{\text{sem}}]\!]\rho, \alpha{\mapsto}r'$, where $\tau'_{\text{sem}} = \alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool})$. Since $\alpha_i = w'.\eta^i(\alpha)$, $r'$ must be $(\mathsf{int}, \mathsf{bool}, V_{k-1}[\![\alpha]\!]w'.\rho)$ by definition of $T[\![\Omega]\!]$. Of course, we defined $w'$ the way we did so that this $r'$ is exactly $r$.

The proof of $(k-2, \lfloor w' \rfloor_{k-2}, v_1, v_2) \in V_n[\![\tau'_{\text{sem}}]\!]\rho, \alpha{\mapsto}r$ decomposes into three parts, following the structure of $\tau'_{\text{sem}}$:

1. $(k-2, \lfloor w' \rfloor_{k-2}, 1, \mathsf{true}) \in V_n[\![\alpha]\!]\rho, \alpha{\mapsto}r$
   This holds because $V_n[\![\alpha]\!]\rho, \alpha{\mapsto}r = R$.

2. $(k-2, \lfloor w' \rfloor_{k-2}, \lambda x{:}\mathsf{int}.(1-x), \lambda x{:}\mathsf{bool}.\neg x) \in V_n[\![\alpha \to \alpha]\!]\rho, \alpha{\mapsto}r$

   - Suppose we are given related arguments in a future world: $(k'', w'', v'_1, v'_2) \in V_n[\![\alpha]\!]\rho, \alpha{\mapsto}r = R$.
   - Hence either $(v'_1, v'_2) = (1, \mathsf{true})$ or $(v'_1, v'_2) = (0, \mathsf{false})$.
   - Consequently, $1 - v'_1$ and $\neg v'_2$ will evaluate in one step, without effects, to values again related by $R$.
   - In other words, $(k'', w'', 1 - v'_1, \neg v'_2) \in E_n[\![\alpha]\!]\rho, \alpha{\mapsto}r$.

3. $(k-2, \lfloor w' \rfloor_{k-2}, \lambda x.(x \neq 0), \lambda x.x) \in V_n[\![\alpha \to \mathsf{bool}]\!]\rho, \alpha{\mapsto}r$
   Like in the previous part, the arguments $v'_1$ and $v'_2$ will be related by $R$ in some future $(k'', w'')$. Therefore $v'_1 \neq 0$ will reduce in one step without effects to $v'_2$, which already is a value. Because of the definition of the logical relation at type $\mathsf{bool}$, this implies $(k'', w'', v'_1 \neq 0, v'_2) \in E_n[\![\mathsf{bool}]\!]\rho, \alpha{\mapsto}r$.

**Partly Benign Effects.** When side effects are introduced into a pure language, they often falsify various equational laws concerning repeatability and order independence of computations. In this section, we offer some evidence that the effect of dynamic type generation is partly *benign* in that it does not invalidate some of these equational laws.

First, consider the following functions:

$$v_1 := \lambda x{:}(\mathsf{unit} \to \tau).\ \mathsf{let}\ x' = x\ ()\ \mathsf{in}\ x\ ()$$
$$v_2 := \lambda x{:}(\mathsf{unit} \to \tau).\ x\ ()$$

The only difference between $v_1$ and $v_2$ is whether the argument $x$ is applied once or twice. Intuitively, either $x\,()$ diverges, in which case both programs diverge, or else the first application of $x$ terminates, in which case so should the second.

Second, consider the following functions:

$$v'_1 := \lambda x{:}(\mathsf{unit} \to \tau).\lambda y{:}(\mathsf{unit} \to \tau').\ \mathsf{let}\ y' = y\ ()\ \mathsf{in}\ \langle x\ (), y' \rangle$$
$$v'_2 := \lambda x{:}(\mathsf{unit} \to \tau).\lambda y{:}(\mathsf{unit} \to \tau').\langle x\ (), y\ () \rangle$$

The only difference between $v'_1$ and $v'_2$ is the order in which they call their argument callbacks $x$ and $y$. Those calls may both result in the generation of fresh type names, but the order in which the names are generated should not matter.

Using our logical relation, we can prove that $v_1$ and $v_2$ are contextually equivalent, and so are $v_1'$ and $v_2'$. Details can be found at the end of Section 10.

However, as we shall see in the example of $e_1'$ and $e_2'$ in the next section, our G language does *not* enjoy referential transparency. This is to be expected, of course, since new is an effectful operation and (in-)equality of type names is observable in the language.

# 5 Wrapping

We have seen that parametricity can be re-established in G by introducing name generation in the right place. But what is the "right place" in general? That is, given an arbitrary expression $e$ with polymorphic type $\tau_e$, how can we *systematically* transform it into an expression $e'$ of the same type $\tau_e$ that is parametric?

One obvious—but unfortunately bogus—idea is the following: transform $e$ such that every existential *introduction* and every universal *elimination* creates a fresh name for the respective witness or instance type. Formally, apply the following rewrite rules to $e$:

$$\begin{array}{lll} \mathsf{pack}\ \langle \tau, e\rangle\ \mathsf{as}\ \tau' & \rightsquigarrow & \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, e\rangle\ \mathsf{as}\ \tau' \\ e\ \tau & \rightsquigarrow & \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e\ \alpha \end{array}$$

Obviously, this would make every quantified type abstract, so that any cast that tries to inspect it would fail.

Or would it? Perhaps surprisingly, the answer is no. To see why, consider the following expressions of type $(\exists\alpha.\tau') \times (\exists\alpha.\tau')$:

$$\begin{array}{lll} e_1 & := & \mathsf{let}\ x = \mathsf{pack}\ \langle \tau, v\rangle\ \mathsf{in}\ \langle x, x\rangle \\ e_2 & := & \langle \mathsf{pack}\ \langle \tau, v\rangle, \mathsf{pack}\ \langle \tau, v\rangle\rangle \end{array}$$

They are clearly equivalent in a parametric language (and in fact they are even equivalent in G). Yet rewriting yields:

$$\begin{array}{lll} e_1' & := & \mathsf{let}\ x = (\mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, v\rangle)\ \mathsf{in}\ \langle x, x\rangle \\ e_2' & := & \langle \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, v\rangle, \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, v\rangle\rangle \end{array}$$

The resulting expressions are *not* equivalent anymore, because they perform different effects. Here is one distinguishing context:

$$\begin{array}{l} \mathsf{let}\ p = [\_]\ \mathsf{in}\ \mathsf{unpack}\ \langle \alpha_1, x_1\rangle = p.1\ \mathsf{in} \\ \qquad\qquad \mathsf{unpack}\ \langle \alpha_2, x_2\rangle = p.2\ \mathsf{in}\ \mathsf{equal?}\ \alpha_1\ \alpha_2 \end{array}$$

Although the representation type $\tau$ is not disclosed as such, *sharing* between the two abstract types in $e_1'$ is. In a parametric language, that would not be possible.

In order to introduce effects uniformly, and to hide internal sharing, the transformation we are looking for needs to be defined on the structure of types, not terms. Roughly, for each quantifier occurring in $\tau_e$ we need to generate one fresh type name. That is, instead of transforming $e$ itself, we simply *wrap* it with some expression that introduces the necessary names at the boundary, by induction on the type $\tau_e$.

In fact, we can refine the problem further. When looking at a G expression $e$, what do we actually mean by "making it parametric"? We can mean two different things: either ensuring that $e$ *behaves* parametrically, or dually, that any context *treats* $e$ parametrically. In the former case, we are protecting the *context* against $e$; in the latter we protect $e$ against malicious contexts, *i.e.,* ensure what is sometimes referred to as *abstraction safety.*

$$
\begin{array}{lcl}
\mathrm{Wr}^{\pm}_{\alpha} & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.x \\
\mathrm{Wr}^{\pm}_{b} & \stackrel{\text{def}}{=} & \lambda x{:}b.x \\
\mathrm{Wr}^{\pm}_{\tau_1\times\tau_2} & \stackrel{\text{def}}{=} & \lambda x{:}(\tau_1\times\tau_2).\langle \mathrm{Wr}^{\pm}_{\tau_1}\,(x.1), \mathrm{Wr}^{\pm}_{\tau_2}\,(x.2)\rangle \\
\mathrm{Wr}^{\pm}_{\tau_1\to\tau_2} & \stackrel{\text{def}}{=} & \lambda x{:}(\tau_1\to\tau_2).\lambda x_1{:}\tau_1.\,\mathrm{Wr}^{\pm}_{\tau_2}\,(x\,(\mathrm{Wr}^{\mp}_{\tau_1}\,x_1)) \\
\mathrm{Wr}^{\pm}_{\forall\alpha.\tau} & \stackrel{\text{def}}{=} & \lambda x{:}(\forall\alpha.\tau).\Lambda\alpha.\,\mathsf{new}^{\mp}\,\alpha\ \mathsf{in}\ \mathrm{Wr}^{\pm}_{\tau}\,(x\,\alpha) \\
\mathrm{Wr}^{\pm}_{\exists\alpha.\tau} & \stackrel{\text{def}}{=} & \lambda x{:}(\exists\alpha.\tau).\,\mathsf{unpack}\,\langle\alpha,x'\rangle{=}x\ \mathsf{in} \\
 & & \quad \mathsf{new}^{\pm}\,\alpha\ \mathsf{in}\ \mathsf{pack}\,\langle\alpha,\mathrm{Wr}^{\pm}_{\tau}\,x'\rangle\ \mathsf{as}\ \exists\alpha.\tau \\
\mathsf{new}^{+}\,\alpha\ \mathsf{in}\ e & \stackrel{\text{def}}{=} & \mathsf{new}\ \alpha'{\approx}\alpha\ \mathsf{in}\ e[\alpha'/\alpha] \\
\mathsf{new}^{-}\,\alpha\ \mathsf{in}\ e & \stackrel{\text{def}}{=} & e
\end{array}
$$

Figure 5: Wrapping for G

Figure 5 defines a pair of (term-level) wrapping functions that correspond to these two dual requirements: $\mathrm{Wr}^{+}$ protects an expression $e : \tau_e$ from being *used* in a non-parametric way, by inserting fresh names for each existential quantifier. Dually, $\mathrm{Wr}^{-}$ forces $e$ to *behave* parametrically by creating a fresh name for each polymorphic instantiation. The definitions extend to other types in the usual functorial manner. Both definitions are interdependent, because roles switch for function arguments. These operators are similar to the type-directed translation that Sumii and Pierce suggest for establishing type abstraction in an untyped language [28] (they propose the descriptive terms "firewall" for $\mathrm{Wr}^{+}$, and "sandbox" for $\mathrm{Wr}^{-}$). However, their use of dynamic sealing instead of type generation results in the insertion of runtime coercions to seal/unseal each individual value of abstract type, while our wrapping leaves such values alone.

Given these operators, we can go back to our semaphore example: $e_{\mathrm{sem1}}$ can now be obtained as $\mathrm{Wr}^{+}_{\tau_{\mathrm{sem}}}\ e_{\mathrm{sem}}$ (modulo some harmless $\beta$-reductions and $\eta$-expansions). This generalises to any ADT: wrapping its implementation positively will guarantee abstraction by making it parametric. We prove that in the next section.

Positive wrapping is reminiscent of *module sealing* (or opaque signature ascription) in ML-style module languages. If we view $e$ as a module and its type $\tau_e$ as a signature, then $\mathrm{Wr}^{+}_{\tau_e}\ e$ corresponds to the sealing operation $e :> \tau_e$. While module sealing typically only performs static abstraction, wrapping describes the dynamic equivalent [23]. In fact, positive wrapping is precisely how sealing is implemented in Alice ML [24, 22], where the module language is non-parametric otherwise.

The correspondence to module sealing motivates our treatment of existential types. Notice that $\mathrm{Wr}^{+}$ causes a fresh type name to be created only once for each existentially quantified type—that is, corresponding to each existential *introduction*. Another option would be to generate type names with each existential *elimination*. In fact, such a semantics would arise naturally were we to use a Church encoding of existentials in conjunction with our wrapping for universals. However, in such a semantics, unpacking an existential value twice would have the effect of producing two distinct abstract types. While this corresponds intuitively to the "generativity" of unpack in System F, it is undesirable in the context of dynamic, first-class modules. In particular, in order for an abstract type `t` defined by some dynamic module M to have some permanent identity (so that it can be referenced by other dynamic modules), it is important that each unpacking of M yields a handle to the same name for `t`. Moreover, as we show in the next section, our approach to defining wrapping is sufficient to ensure abstraction safety.

# 6 Parametric Reasoning

$$T^\circ_n[\![\Omega]\!]w \stackrel{\text{def}}{=} \{(\tau_1, \tau_2, (w.\sigma^*_1(\tau_1), w.\sigma^*_2(\tau_2), R)) \mid \mathrm{ftv}(\tau_i) \subseteq \mathrm{dom}(w.\sigma_i) \wedge R \in \mathrm{Rel}_n[w.\sigma^*_1(\tau_1), w.\sigma^*_2(\tau_2)]\}$$

(everything else as in Figures 3 and 4)

Figure 6: Parametric Logical Relation

The logical relation developed in Section 4 enables us to do *non-parametric* reasoning about equivalence of G programs. It also enables us to do *parametric* reasoning, but only indirectly: we have to explicitly deal with the effects of new and to define worlds containing relations between type names. It would be preferable if we were able to do parametric reasoning directly. For example, given two expressions $e_1$, $e_2$ that do not use casts, and assuming that the context does not do so either, we should be able to reason about equivalence of $e_1$ and $e_2$ in a manner similar to what we do when reasoning about System F.

## 6.1 A Parametric Logical Relation

Thanks to the modular formulation of our logical relation in Figure 4, it is easy to modify it so that it becomes parametric. All we need to do is swap out the definition of $T[\![\Omega]\!]w$, which relates types as data. Figure 6 gives an alternative definition that allows choosing an arbitrary relation between arbitrary types. Everything else stays exactly the same. We decorate the set of *parametric logical relations* thus obtained with $^\circ$ (*i.e.,* $V^\circ$, $E^\circ$, etc.) to distinguish them from the original ones. Likewise, we write $\precsim^\circ$ for the notion of *parametric logical approximation* defined as in Figure 4 but in terms of the parametric relations. For clarity, we will refer to the original definition as the non-parametric relation.

This modification gives us a seemingly parametric definition of logical approximation for G terms. But what does that actually *mean*? What is the relation between parametric and non-parametric logical approximation and, ultimately, *contextual* approximation? Since the language is not parametric, clearly, parametrically equivalent terms generally are not contextually equivalent.

The answer is given by the wrapping functions we defined in the previous section. The following theorem connects the two notions of logical relation and approximation that we have introduced:

**Theorem 6.1** (Wrapping for $\precsim^\circ$)**.**

1. If $\vdash e_1 \precsim^\circ e_2 : \tau$, then $\vdash \mathrm{Wr}^+_\tau\ e_1 \precsim \mathrm{Wr}^+_\tau\ e_2 : \tau$.
2. If $\vdash e_1 \precsim e_2 : \tau$, then $\vdash \mathrm{Wr}^-_\tau\ e_1 \precsim^\circ \mathrm{Wr}^-_\tau\ e_2 : \tau$.

This theorem justifies the definition of the parametric logical relation. At the same time it can be read as a correctness result for the wrapping operators: it says that whenever we can relate two terms using parametric reasoning, then the positive wrappings of the first term contextually approximates the positive wrapping of the second. Dually, once any properly related terms are wrapped negatively, they can safely be passed to any term that depends on its context behaving parametrically.

The alert reader may wonder why the Wrapping Theorem only talks about closed terms. First of all, simply allowing open terms would not be correct. For instance, it is easy to see that we have

$$\epsilon; x{:}(\forall\alpha.\mathsf{bool}) \vdash x\ \mathsf{bool} \precsim^\circ x\ \mathsf{unit} : \mathsf{bool}$$

because the instantiations of $x$ will be parametric by definition. For $\precsim$ they may of course be non-parametric (consider equal? unit being plugged in for $x$), hence

$$\epsilon; x{:}(\forall\alpha.\mathsf{bool}) \vdash x\ \mathsf{bool} \precsim x\ \mathsf{unit} : \mathsf{bool}$$

does *not* hold. However, since $\mathrm{Wr}^+_{\mathsf{bool}}$ is just the identity function, this is essentially what the naive extension of the Wrapping theorem to open terms would tell us.

The solution to this is to wrap all free value variables at the inverse polarity, so that the theorem would look as follows:

1. If $\Delta;\Gamma \vdash e_1 \precsim^\circ e_2 : \tau$, then $\Delta;\Gamma \vdash \mathrm{Wr}^+_\tau(\gamma^-_\Gamma(e_1)) \precsim \mathrm{Wr}^+_\tau(\gamma^-_\Gamma(e_2)) : \tau$.
2. If $\Delta;\Gamma \vdash e_1 \precsim e_2 : \tau$, then $\Delta;\Gamma \vdash \mathrm{Wr}^-_\tau(\gamma^+_\Gamma(e_1)) \precsim^\circ \mathrm{Wr}^-_\tau(\gamma^+_\Gamma(e_2)) : \tau$.

Here the substitution $\gamma^\pm_\Gamma$ replaces each free variable $x{:}\tau$ by its wrapping $\mathrm{Wr}^\pm_\tau\ x$ and could be defined as follows:

$$\gamma^\pm_\epsilon \stackrel{\mathrm{def}}{=} \emptyset \qquad\qquad \gamma^\pm_{\Gamma,x:\tau} \stackrel{\mathrm{def}}{=} \gamma^\pm_\Gamma, x{\mapsto}(\mathrm{Wr}^\pm_\tau\ x)$$

There are other issues, however. One is that in an attempt to prove the above statement, after unfolding the definition of logical approximation in part 1, we are given some $(\delta_1,\delta_2,\rho) \in D[\![\Delta]\!]$. To instantiate the assumption appropriately, $(\delta_1,\delta_2,\rho)$ needs to be in $D^\circ[\![\Delta]\!]$. In part 2, the situation is the other way around. However, $D[\![\Delta]\!]$ and $D^\circ[\![\Delta]\!]$ are only equal if $\Delta$ does not contain components of the form $\alpha{\approx}\tau'$. Another problem is that wrapped value substitutions no longer are *value* substitutions and hence cannot be used directly for instantiating the assumption. All in all, we believe these problems can be solved, but we leave the solution to future work.

What can we say about the content of the parametric relation? Obviously, it cannot contain arbitrary G terms—*e.g.,* cast $\tau_1\ \tau_2$ will generally not be related to anything (including itself) in $E^\circ$. However, for all constructs besides cast, we obtain the following:

**Theorem 6.2** (Fundamental Property for $\precsim^\circ$)**.** If $\Delta;\Gamma \vdash e : \tau$ and $e$ does not use cast, then $\Delta;\Gamma \vdash e \precsim^\circ e : \tau$.

In particular, this implies that any well-typed System F term is parametrically related to itself. The relation will also contain terms with cast, but only if the use of cast does not violate parametricity. (We discuss this further in Section 7.)

Along the same lines, we can show that our parametric logical relation is sound w.r.t. contextual approximation, *if* the definition of the latter is limited to quantifying only over cast-free contexts.

## 6.2 Examples

**Semaphore.** Consider our running example of the semaphore module again. Using the parametric relation, we can prove that the two implementations are related without actually reasoning about type generation. That aspect is covered once and for all by the Wrapping Theorem.

Recall the two implementations, here given in unwrapped form:

$$e'_{\mathrm{sem1}} := \mathsf{pack}\ \langle \mathsf{int}, \langle 1, \lambda x{:}\,\mathsf{int}\,.(1-x), \lambda x{:}\,\mathsf{int}\,.(x \neq 0)\rangle\rangle\ \mathsf{as}\ \tau_{\mathrm{sem}}$$
$$e'_{\mathrm{sem2}} := \mathsf{pack}\ \langle \mathsf{bool}, \langle \mathsf{true}, \lambda x{:}\,\mathsf{bool}\,.\neg x, \lambda x{:}\,\mathsf{bool}\,.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathrm{sem}}$$

We can prove $\vdash e'_{\text{sem1}} \precsim^{\circ} e'_{\text{sem2}} : \tau_{\text{sem}}$ using conventional parametric reasoning about polymorphic terms. Now define $e_{\text{sem1}} := \text{Wr}^{+}_{\tau_{\text{sem}}}\ e'_{\text{sem1}}$ and $e_{\text{sem2}} := \text{Wr}^{+}_{\tau_{\text{sem}}}\ e'_{\text{sem2}}$, which is effectively equivalent to the original definitions. The Wrapping Theorem then immediately tells us that $\vdash e_{\text{sem1}} \precsim e_{\text{sem2}} : \tau_{\text{sem}}$.

**A Free Theorem.** We can use the parametric relation for proving free theorems [31] in G. For example, for any $\vdash g : \forall\alpha.\alpha \to \alpha$ in G it holds that $\text{Wr}^{-}_{\forall\alpha.\alpha\to\alpha}\ g$ either diverges for all possible arguments $\tau$ and $\vdash v : \tau$, or it returns $v$ in all cases. We first apply the Fundamental Property for $\precsim$ to relate $g$ to itself in $E$, then transfer this to $E^{\circ}$ for $\text{Wr}^{-}_{\forall\alpha.\alpha\to\alpha}\ g$ using the Wrapping Theorem. From there the proof proceeds in the usual way.

# 7 Syntactic vs. Semantic Parametricity

The primary motivation for our parametric relation in the previous section was to enable more direct parametric reasoning about the result of (positively) wrapping System F terms. However, it is also possible to use our parametric relation to reason about terms that are *syntactically*, or *intensionally*, non-parametric (*i.e.,* that use $\mathsf{cast}$'s), so long as they are *semantically*, or *extensionally*, parametric (*i.e.,* the use of $\mathsf{cast}$ is not externally observable).

For example, consider the following two polymorphic functions of type $\forall\alpha.\tau_\alpha$ (here, let $b2i = \lambda x{:}\mathsf{bool}.\ \mathsf{if}\ x\ \mathsf{then}\ 1\ \mathsf{else}\ 0$):

$$
\begin{aligned}
\tau_\alpha &:= \exists\alpha'.\,(\alpha \times \alpha \to \alpha') \times (\alpha' \to \alpha) \times (\alpha' \to \alpha)\\
g_1 &:= \Lambda\alpha.\,\mathsf{pack}\,\langle \alpha \times \alpha, \langle \lambda p.p,\ \lambda x.(x.1),\ \lambda x.(x.2)\rangle\rangle\ \mathsf{as}\ \tau_\alpha\\
g_2 &:= \Lambda\alpha.\,\mathsf{cast}\,\tau_{\mathsf{bool}}\,\tau_\alpha\\
&\qquad (\mathsf{pack}\,\langle \mathsf{int}, \langle \lambda p{:}(\mathsf{bool}\times\mathsf{bool}).\ b2i(p.1) + 2{\times}b2i(p.2),\\
&\qquad\qquad \lambda x{:}\mathsf{int}.\ x\ \mathsf{mod}\ 2 \neq 0,\\
&\qquad\qquad \lambda x{:}\mathsf{int}.\ x\ \mathsf{div}\ 2 \neq 0\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{bool}})\\
&\qquad (g_1\,\alpha)
\end{aligned}
$$

These two functions take a type argument $\alpha$ and return a simple generic ADT for pairs over $\alpha$. But $g_2$ is more clever about it and specializes the representation for $\alpha = \mathsf{bool}$. In that case, it packs both components into the two least significant bits of a single integer. For all other types, $g_2$ falls back to the generic implementation from $g_1$.

Using the parametric relation, we will be able to show that $\vdash \text{Wr}^{+}\ g_1 \preceq \text{Wr}^{+}\ g_2 : \forall\alpha.\tau_\alpha$. One might find this surprising, since $g_2$ is syntactically non-parametric, returning different implementations for different instantiations of its type argument. However, since the two possible implementations $g_2$ returns are extensionally equivalent to each other, $g_2$ is semantically indistinguishable from the syntactically parametric $g_1$.

Formally: Assume that $\tau_1$, $\tau_2$ are the types and $R_\alpha \in \text{Rel}[\tau_1, \tau_2]$ is the relation the context picks, parametrically, for $\alpha$. If $\tau_2 \neq \mathsf{bool}$, the rest of the proof is straightforward. Otherwise, we do not know anything about $\tau_1$ and $R_\alpha$, because $\tau_1$ and $\tau_2$ are related in $T^{\circ}$. Nevertheless, we can construct a suitable relational interpretation $R_{\alpha'} \in \text{Rel}[\tau_1 \times \tau_1, \mathsf{int}]$ for the type $\alpha'$:

$$
\begin{aligned}
R_{\alpha'} := {} & \{(k, w, \langle v, v'\rangle, 0) \mid (k, w, v, \mathsf{false}), (k, w, v', \mathsf{false}) \in R_\alpha\}\\
\cup\ & \{(k, w, \langle v, v'\rangle, 1) \mid (k, w, v, \mathsf{true}), (k, w, v', \mathsf{false}) \in R_\alpha\}\\
\cup\ & \{(k, w, \langle v, v'\rangle, 2) \mid (k, w, v, \mathsf{false}), (k, w, v', \mathsf{true}) \in R_\alpha\}\\
\cup\ & \{(k, w, \langle v, v'\rangle, 3) \mid (k, w, v, \mathsf{true}), (k, w, v', \mathsf{true}) \in R_\alpha\}
\end{aligned}
$$

As it turns out, we do not need to know much about the structure of $R_\alpha$ to define $R_{\alpha'}$. What we are relying on here is only the knowledge that all values in $R_\alpha$ are well-typed, which is built into

our definition of Rel. From that we know that there can never be any other value than true or false on the right side of the relation $R_\alpha$. Hence we can still enumerate all possible cases to define $R_{\alpha'}$, and do a respective case distinction when proving equivalence of the projection operations.

Interestingly, it seems that our proof relies critically on the fact that our logical relations are restricted to syntactically well-typed terms. Were we to lift this restriction, we would be forced (it seems) to extend the definition of $R_{\alpha'}$ with a "junk" case, but the calls to $b2i$ in $g_2$ would get stuck if applied to non-boolean values. We leave further investigation of this observation to future work.

# 8 Polarized Logical Relations

$$
\begin{array}{lcl}
V_n^\pm[\![\alpha]\!]\rho & \stackrel{\text{def}}{=} & \lfloor \rho(\alpha).R \rfloor_n \\
V_n^\pm[\![b]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,c,c) \in \mathrm{Atom}_n[b,b]\} \\
V_n^\pm[\![\tau \times \tau']\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\langle v_1,v_1'\rangle,\langle v_2,v_2'\rangle) \in \mathrm{Atom}_n[\rho^1(\tau\times\tau'),\rho^2(\tau\times\tau')] \mid \\
 & & \quad (k,w,v_1,v_2) \in V_n^\pm[\![\tau]\!]\rho \wedge (k,w,v_1',v_2') \in V_n^\pm[\![\tau']\!]\rho\} \\
V_n^\pm[\![\tau' \to \tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\lambda x{:}\tau_1.e_1,\lambda x{:}\tau_2.e_2) \in \mathrm{Atom}_n[\rho^1(\tau'\to\tau),\rho^2(\tau'\to\tau)] \mid \\
 & & \quad \forall(k',w',v_1,v_2) \in V_n^\mp[\![\tau']\!]\rho.\,(k',w') \sqsupseteq (k,w) \Rightarrow \\
 & & \quad (k',w',e_1[v_1/x],e_2[v_2/x]) \in E_n^\pm[\![\tau]\!]\rho\} \\
V_n^\pm[\![\forall\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\Lambda\alpha.e_1,\Lambda\alpha.e_2) \in \mathrm{Atom}_n[\rho^1(\forall\alpha.\tau),\rho^2(\forall\alpha.\tau)] \mid \\
 & & \quad \forall(k',w') \sqsupseteq (k,w).\ \forall(\tau_1,\tau_2,r) \in T_{k'}^\mp[\![\Omega]\!]w'. \\
 & & \quad (k',w',e_1[\tau_1/\alpha],e_2[\tau_2/\alpha]) \in \triangleright E_n^\pm[\![\tau]\!]\rho,\alpha{\mapsto}r\} \\
V_n^\pm[\![\exists\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\mathsf{pack}\ \langle\tau_1,v_1\rangle,\mathsf{pack}\ \langle\tau_2,v_2\rangle) \in \mathrm{Atom}_n[\rho^1(\exists\alpha.\tau),\rho^2(\exists\alpha.\tau)] \mid \\
 & & \quad \exists r.\,(\tau_1,\tau_2,r) \in T_k^\pm[\![\Omega]\!]w \wedge (k,w,v_1,v_2) \in \triangleright V_n^\pm[\![\tau]\!]\rho,\alpha{\mapsto}r\} \\
E_n^\pm[\![\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,e_1,e_2) \in \mathrm{Atom}_n[\rho^1(\tau),\rho^2(\tau)] \mid \\
 & & \quad \forall j<k.\ \forall\sigma_1,v_1.\,(w.\sigma_1;e_1 \hookrightarrow^j \sigma_1;v_1) \Rightarrow \\
 & & \quad \exists w',v_2.\,(k-j,w') \sqsupseteq (k,w) \wedge w'.\sigma_1 = \sigma_1 \wedge \\
 & & \quad (w.\sigma_2;e_2 \hookrightarrow^* w'.\sigma_2;v_2) \wedge (k-j,w',v_1,v_2) \in V_n^\pm[\![\tau]\!]\rho\}
\end{array}
$$

$$
\begin{array}{lcllcl}
T_n^+[\![\Omega]\!]w & \stackrel{\text{def}}{=} & T_n^\circ[\![\Omega]\!]w & D_n^+[\![\Delta]\!]w & \stackrel{\text{def}}{=} & D_n^\circ[\![\Delta]\!]w \\
T_n^-[\![\Omega]\!]w & \stackrel{\text{def}}{=} & T_n[\![\Omega]\!]w & D_n^-[\![\Delta]\!]w & \stackrel{\text{def}}{=} & D_n[\![\Delta]\!]w
\end{array}
$$

$$
\begin{array}{lcl}
\Delta;\Gamma \vdash e_1 \precsim^\pm e_2 : \tau & \stackrel{\text{def}}{\iff} & \Delta;\Gamma \vdash e_1 : \tau \wedge \Delta;\Gamma \vdash e_2 : \tau \wedge \forall n \geq 0.\forall w_0 \in \mathrm{World}_n\,. \\
 & & \forall(\delta_1,\delta_2,\rho) \in D_n^\mp[\![\Delta]\!]w_0.\,\forall(k,w,\gamma_1,\gamma_2) \in G_n^\mp[\![\Gamma]\!]\rho. \\
 & & (k,w) \sqsupset (n,w_0) \Rightarrow (k,w,\delta_1\gamma_1(e_1),\delta_2\gamma_2(e_2)) \in E_n^\pm[\![\tau]\!]\rho
\end{array}
$$

Figure 7: Polarized Logical Relations

The parametric relation is useful for proving parametricity properties about (the positive wrappings of) G terms. However, it is all-or-nothing: it can only be used to prove parametricity for terms that are *treated* parametrically *and* also *behave* parametrically—cf. the two dual aspects of parametricity described in Section 5. We might also be interested in proving representation independence for terms that do *not* behave parametrically themselves (in either the syntactic or semantic sense considered in the previous section). One situation where this might show up is if we want to show representation independence for generic ADTs that (like the ones in Section 7) return different results for different instantiations of their type arguments, but where (unlike in

Section 7) the difference is not only syntactic but also semantic.

Here is a somewhat contrived example to illustrate the point. Consider the following two polymorphic functions of type $\forall\alpha.\tau_\alpha$:

$$\begin{aligned}
\tau_\alpha &:= \exists\alpha'.\,(\alpha \to \alpha') \times (\alpha' \to \alpha)\\
f_1 &:= \lambda\alpha.\,\mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_\alpha\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x{+}1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\\
&\qquad\qquad (\mathsf{pack}\ \langle \alpha, \langle \lambda x{:}\alpha.x, \lambda x{:}\alpha.x\rangle\rangle\ \mathsf{as}\ \tau_\alpha)\\
f_2 &:= \lambda\alpha.\,\mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_\alpha\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x{+}1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\\
&\qquad\qquad (\mathsf{pack}\ \langle \alpha, \langle \lambda x{:}\alpha.x, \lambda x{:}\alpha.x\rangle\rangle\ \mathsf{as}\ \tau_\alpha)
\end{aligned}$$

These functions can be understood as simplistic functors with a type argument $\alpha$. Both functors implement a simple ADT $\alpha'$. Values of type $\alpha$ can be injected into $\alpha'$, and projected out again. However, both functors specialize the behavior of this ADT for type $\mathsf{int}$—for integers, injecting $n$ and projecting again will not give back $n$, but $n+1$. This is true for both functors, but they implement it in a different way.

We want to prove that both implementations are equivalent under wrapping using a form of parametric reasoning. However, we cannot do that using the parametric relation from the previous section—since the functors do not *behave* parametrically (they return observably different packages for different instantiations of their type argument), they will not be related in $E^\circ$.

To support that kind of reasoning, we need a more refined treatment of parametricity in the logical relation. The idea is to separate the two aforementioned aspects of parametricity. Consequently, we are going to have a pair of separate relations, $E^+$ and $E^-$. The former enforces parametric usage, the latter parametric behavior.

Figure 7 gives the definition of these relations. We call them *polarized*, because they are mutually dependent and the polarity ($+$ or $-$) switches for contravariant positions, *i.e.,* for function arguments and for universal quantifiers. Intuitively, in these places, term and context switch roles.

Except for the consistent addition of polarities, the definition of the polarized relations again only represents a minor modification of the original one.[8] We merely refine the definition of the type relation $T[\![\Omega]\!]w$ to distinguish polarity: in the positive case it behaves parametrically (*i.e.,* allowing an arbitrary relation) and in the negative case non-parametrically (*i.e.,* demanding $r$ be the *logical* relation at some type). Thus, existential types behave parametrically in $E^+$ but non-parametrically in $E^-$, and vice versa for universals.

## 8.1 Key Properties

The way in which polarities switch in the polarized relations mirrors what is going on in the definition of wrapping. That of course is no accident, and we can show the following theorem that relates the polarized relations with the proper and the parametric ones through uses of wrapping:

**Theorem 8.1** (Wrapping for $\precsim^\pm$)**.**

1. If $\vdash e_1 \precsim^+ e_2 : \tau$, then $\vdash \mathrm{Wr}^+_\tau\ e_1 \precsim \mathrm{Wr}^+_\tau\ e_2 : \tau$.
2. If $\vdash e_1 \precsim e_2 : \tau$, then $\vdash \mathrm{Wr}^-_\tau\ e_1 \precsim^- \mathrm{Wr}^-_\tau\ e_2 : \tau$.
3. If $\vdash e_1 \precsim^+ e_2 : \tau$, then $\vdash \mathrm{Wr}^-_\tau\ e_1 \precsim^\circ \mathrm{Wr}^-_\tau\ e_2 : \tau$.
4. If $\vdash e_1 \precsim^\circ e_2 : \tau$, then $\vdash \mathrm{Wr}^+_\tau\ e_1 \precsim^- \mathrm{Wr}^+_\tau\ e_2 : \tau$.

[8] In fact, all four relations can easily be formulated in a single unified definition indexed by $\iota ::= \epsilon \mid \circ \mid + \mid -$. We refrained from doing so here for the sake of clarity; see section 10 for details.

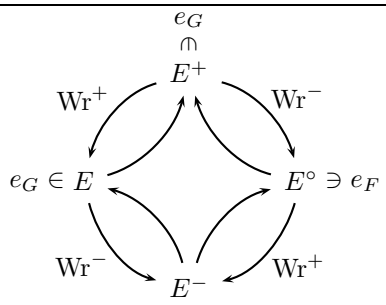


Figure 8: Relating the Relations

Moreover, we can show that the inverse directions of these implications require no wrapping at all:

**Theorem 8.2** (Inclusion for $\precsim^{\pm}$)**.**

1. If $\vdash e_1 \precsim e_2 : \tau$ or $\vdash e_1 \precsim^\circ e_2 : \tau$, then $\vdash e_1 \precsim^+ e_2 : \tau$.
2. If $\vdash e_1 \precsim^- e_2 : \tau$, then $\vdash e_1 \precsim e_2 : \tau$ and $\vdash e_1 \precsim^\circ e_2 : \tau$.

This theorem can equivalently be stated as the chains $E^- \subseteq E \subseteq E^+$ and $E^- \subseteq E^\circ \subseteq E^+$.

Note that Theorem 6.1 follows directly from Theorems 8.1 and 8.2. Similarly, the following property follows from Theorem 8.2 together with Theorem 4.1:

**Theorem 8.3** (Fundamental Property for $\precsim^+$)**.** If $\Delta; \Gamma \vdash e : \tau$, then $\Delta; \Gamma \vdash e \precsim^+ e : \tau$.

Interestingly, compatibility does not hold for $\precsim^{\pm}$ (consider the polarities in the rule for application), which has the consequence that we cannot show Theorem 8.3 directly. For a similar reason, we cannot show any such property for $\precsim^-$ at all.

Figure 8 depicts all of the above properties in a single diagram. Plain arrows denote simple inclusion, while annotated arrows require respective wrapping to go from one relation to the other. The $\in$-operators show the fundamental properties for the respective relations, *i.e.,* which class of terms are included (G terms or F terms).

## 8.2 Example

Getting back to our motivating example from the beginning of the section, it is essentially straightforward to prove that $\vdash f_1 \precsim^+ f_2 : \forall\alpha.\tau_\alpha$. The proof proceeds as usual, except that we have to make a case distinction where we want to show that the functor bodies are related in $E^+$. At that point, we are given a triple $(\tau_1, \tau_2, r) \in T^-[\![\Omega]\!]w$.

If $\tau_1 = \mathsf{int}$, then we know from the definition of $T^-$ that $\tau_2 = \mathsf{int}$, too. We hence know that both sides will evaluate to the specialized version of the ADT. Since we are in $E^+$, we get to pick some $(\tau_1', \tau_2', r') \in T^+[\![\Omega]\!]w$ as the interpretation of $\alpha'$, where the choice of $r'$ is up to us. The natural choice is to use $\tau_1' = \tau_2' = \mathsf{int}$ with the relation $r' = (\mathsf{int}, \mathsf{int}, \{(k, w, n+1, n) \mid n \in \mathbb{N}\})$. The rest of the proof is then straightforward.

If $\tau_1 \neq \mathsf{int}$ we similarly know that $\tau_2 \neq \mathsf{int}$ from the definition of $T^-$. Hence, both sides use the default implementations, which are trivially related in $E^+$, thanks to Theorem 8.3.

Finally, applying the Wrapping Theorem 8.1, we can conclude that $\vdash \mathrm{Wr}^+\ f_1 \precsim \mathrm{Wr}^+\ f_2 : \forall\alpha.\tau_\alpha$, and hence by Soundness, $\vdash \mathrm{Wr}^+\ f_1 \preceq \mathrm{Wr}^+\ f_2 : \forall\alpha.\tau_\alpha$.

Note how we relied on the knowledge that $\tau_1$ and $\tau_2$ can only be $\mathsf{int}$ at the same time. This holds for types related in $T^-$ but not in $T^+$ or $T^\circ$. If we had tried to do this proof in $E^\circ$, the types would have been related by $T^\circ$ only, which would give us too little information to proceed with the necessary case distinction.

# 9 Recursive Types

We now add iso-recursive types to G (calling the resulting language $\mathrm{G}^\mu$):

$$\begin{array}{lrcl} \text{(types)} & \tau & ::= & \ldots \mid \mu\alpha.\tau \\ \text{(values)} & v & ::= & \ldots \mid \mathsf{roll}\ v\ \mathsf{as}\ \tau \\ \text{(expressions)} & e & ::= & \ldots \mid \mathsf{roll}\ e\ \mathsf{as}\ \tau \mid \mathsf{unroll}\ e \end{array}$$

The extensions to the semantics are standard and given in the appendix—they do not affect the type store. The definition of contextual equivalence does not change (except there are more contexts).

## 9.1 Extending the Logical Relations

The step-indexing that we used in defininig our logical relations makes it very easy to adapt them to $\mathrm{G}^\mu$. There are two natural ways in which we could define the value relation at a recursive type:

1. $V^\iota_n[\![\mu\alpha.\tau]\!]\rho \stackrel{\text{def}}{=} \{(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in \mathrm{Atom}_n[\rho^1(\mu\alpha.\tau), \rho^2(\mu\alpha.\tau)] \mid (k, w, v_1, v_2) \in \triangleright V^\iota_k[\![\tau]\!]\rho, \alpha \mapsto V^\iota_k[\![\mu\alpha.\tau/\alpha]\!]\rho\}$
2. $V^\iota_n[\![\mu\alpha.\tau]\!]\rho \stackrel{\text{def}}{=} \{(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in \mathrm{Atom}_n[\rho^1(\mu\alpha.\tau), \rho^2(\mu\alpha.\tau)] \mid (k, w, v_1, v_2) \in \triangleright V^\iota_k[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho\}$

For $\iota \in \{\epsilon, \circ\}$, they are ultimately equivalent due to the validity of a standard substitution property (Lemma 10.11). We do not have this property for the polarized relation, however (see Section 10). In fact, for $\iota \in \{+, -\}$, the first definition wrongly records a fixed polarity for $\alpha$. It is thus crucial that we choose the second one. Only then do all the key properties and the inclusion properties (and thus the diagram) continue to hold in $\mathrm{G}^\mu$. Details can be found in Section 10.

## 9.2 Extending the Wrapping

How can we upgrade the wrapping to account for recursive types? Given an argument of type $\mu\alpha.\tau$, the basic idea is to first unfold it to type $\tau[\mu\alpha.\tau/\alpha]$, then wrap it at that type, and finally fold the result back to type $\mu\alpha.\tau$. Of course, since $\tau[\mu\alpha.\tau/\alpha]$ may be larger than $\mu\alpha.\tau$, a direct implementation of this idea will not result in a well-founded definition. What we could do is recursively wrap at type $\tau$, which definitely is smaller than $\mu\alpha.\tau$. This alone will not have the desired effect, as the recursive wrapping does not know that any $\alpha$ that it comes across actually represents $\mu\alpha.\tau$, which needs to be wrapped using the very function we are defining. The solution is as follows: We first index the wrapping by an environment $\varphi$ that maps type variables to terms. The wrapping at type $\alpha$ then is defined as either the identity (like before) or, if $\alpha \in \mathrm{dom}(\varphi)$, as $\varphi(\alpha)$. Second, we make the wrapping at type $\mu\alpha.\tau$ a function that, after unfolding, wraps at type $\tau$ where $\varphi$ has been extended by a mapping from $\alpha$ to the function itself. That is, $\mathrm{Wr}^\pm_{\mu\alpha.\tau}\,\varphi$ will be a recursive function, which we can encode as shown in the appendix.

$$
\begin{array}{lcl}
F^{\varphi}_{\mu\alpha.\tau} & \stackrel{\text{def}}{=} & \mathsf{fix}\, f(x').\langle \lambda x{:}(\mu\alpha.\tau).\, \mathsf{roll}\ \mathrm{Wr}^{+}_{\tau}(\varphi, \alpha{\mapsto}f)[\mu\alpha.\tau/\alpha]\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \mu\alpha.\tau, \\
& & \qquad \lambda x{:}(\mu\alpha.\tau).\, \mathsf{roll}\ \mathrm{Wr}^{-}_{\tau}(\varphi, \alpha{\mapsto}f)[\mu\alpha.\tau/\alpha]\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \mu\alpha.\tau\rangle \\
& & \quad : \mathsf{unit} \to ((\mu\alpha.\tau) \to (\mu\alpha.\tau)) \times ((\mu\alpha.\tau) \to (\mu\alpha.\tau)) \\
\\
\mathrm{Wr}^{+}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.(\varphi(\alpha)\ ()).1\ x \quad (\text{if } \alpha \in \mathrm{dom}(\varphi)) \\
\mathrm{Wr}^{-}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.(\varphi(\alpha)\ ()).2\ x \quad (\text{if } \alpha \in \mathrm{dom}(\varphi)) \\
\mathrm{Wr}^{\pm}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.x \quad (\text{if } \alpha \notin \mathrm{dom}(\varphi)) \\
\mathrm{Wr}^{\pm}_{b}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}b.x \\
\mathrm{Wr}^{\pm}_{\tau_1\times\tau_2}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\tau_1 \times \tau_2).\langle \mathrm{Wr}^{\pm}_{\tau_1}\,\varphi\ (x.1)), \mathrm{Wr}^{\pm}_{\tau_2}\,\varphi\ (x.2)\rangle \\
\mathrm{Wr}^{\pm}_{\tau_1\to\tau_2}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\tau_1 \to \tau_2).\lambda x'{:}\tau_1.\, \mathrm{Wr}^{\pm}_{\tau_2}\,\varphi\ (x\ (\mathrm{Wr}^{\mp}_{\tau_1}\,\varphi\ x')) \\
\mathrm{Wr}^{\pm}_{\forall\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\forall\alpha.\tau).\Lambda\alpha.\, \mathsf{new}^{\mp}\,\alpha\ \mathsf{in}\ \mathrm{Wr}^{\pm}_{\tau}\,\varphi\ (x\ \alpha) \\
\mathrm{Wr}^{\pm}_{\exists\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\exists\alpha.\tau).\, \mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in}\ \mathsf{new}^{\pm}\,\alpha\ \mathsf{in}\ \mathsf{pack}\ \langle\alpha, \mathrm{Wr}^{\pm}_{\tau}\,\varphi\ x'\rangle\ \mathsf{as}\ \exists\alpha.\tau \\
\mathrm{Wr}^{+}_{\mu\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\mu\alpha.\tau).(F^{\varphi}_{\mu\alpha.\tau}\ ()).1\ x \\
\mathrm{Wr}^{-}_{\mu\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\mu\alpha.\tau).(F^{\varphi}_{\mu\alpha.\tau}\ ()).2\ x \\
\\
\mathrm{Wr}^{\pm}_{\tau} & \stackrel{\text{def}}{=} & \mathrm{Wr}^{\pm}_{\tau}\,\emptyset
\end{array}
$$

Figure 9: Wrapping for $\mathrm{G}^{\mu}$

$$
\begin{array}{lcl}
\mathrm{Wr}^{\pm}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.x \quad (\text{if } \alpha \notin \mathrm{dom}(\varphi)) \\
\mathrm{Wr}^{\pm}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \varphi^{\pm}(\alpha) \quad (\text{if } \alpha \in \mathrm{dom}(\varphi)) \\
\mathrm{Wr}^{\pm}_{\mu\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \mathsf{letrec}\ f^{+} = \lambda x.\, \mathsf{roll}\ (\mathrm{Wr}^{+}_{\tau}\,\varphi'\ (\mathsf{unroll}\, x))[\mu\alpha.\tau/\alpha] \\
& & \quad \mathsf{and}\ f^{-} = \lambda x.\, \mathsf{roll}\ (\mathrm{Wr}^{-}_{\tau}\,\varphi'\ (\mathsf{unroll}\, x))[\mu\alpha.\tau/\alpha] \\
& & \quad \mathsf{in}\ f^{\pm} \qquad (\text{where } \varphi' = \varphi, \alpha{\mapsto}(f^{+}, f^{-}))
\end{array}
$$

Figure 10: Wrapping for $\mathrm{G}^{\mu}$ (informal)

Figure 9 shows the full definition, which, admittedly, looks somewhat unwieldy. This is due to the following reasons:

- Since the bound variable of a recursive type may occur in positions of different polarity, we need to define two mutually recursive functions and then select the right one depending on the polarity.
- To simplify proofs, we want the wrappings to be syntactic values. This requires some $\eta$-expansions.

A less formal but more readable definition of the relevant cases is given in Figure 10.

Note that the environment only plays a role for recursive types, and that for any $\tau$ that does not involve recursive types, $\mathrm{Wr}^{\pm}_{\tau}\,\emptyset$ is the same as our old wrapping $\mathrm{Wr}^{\pm}_{\tau}$ from Section 5. This justifies that we omit an empty $\varphi$ and just write $\mathrm{Wr}^{\pm}_{\tau}$. This notation makes our wrapping theorems for G meaningful statements for $\mathrm{G}^{\mu}$. And indeed they also *hold* for $\mathrm{G}^{\mu}$, as we show in Section 10.

# 10 Detailed Proofs

In this section, we give detailed proofs of the logical relations' key properties, the wrapping theorems, and the examples, that have been given in the previous sections. As a matter of fact, we show them for $\mathrm{G}^{\mu}$, but since recursive types really are an orthogonal feature, the development can be obtained for G by just dropping the extra cases dealing with recursive types from the proofs.

## 10.1 Unified Definition of Our Logical Relations

As briefly mentioned before, it is easy to present our different logical relations in a single definition indexed by:

$$\iota ::= \epsilon \mid - \mid + \mid \circ$$

This unified definition is shown in Figure 11. Notice how we factor out the differences into $T$ and uses of two operators on $\iota$. The $\wr\cdot\wr$ operator occurs in the definition of $D^{\iota}$ and establishes $D^{\circ} = D^{+}$ as well as $D = D^{-}$. The $\neg$ operator is used for switching "polarity", for instance in the definition of $V_n^{\iota}[\![\forall\alpha.\tau]\!]\rho$. They are defined as follows:

$$\begin{array}{ll} \neg\epsilon \stackrel{\text{def}}{=} \epsilon & \wr\epsilon\wr \stackrel{\text{def}}{=} \epsilon \\ \neg- \stackrel{\text{def}}{=} + & \wr-\wr \stackrel{\text{def}}{=} \epsilon \\ \neg+ \stackrel{\text{def}}{=} - & \wr+\wr \stackrel{\text{def}}{=} \circ \\ \neg\circ \stackrel{\text{def}}{=} \circ & \wr\circ\wr \stackrel{\text{def}}{=} \circ \end{array}$$

$\iota$ and its operations are not only useful for making possible the unified definition but also for stating *and* proving almost all the properties in a unified way, avoiding duplicate work. Of course there are properties whose proofs depend on the value of $\iota$, but we have tried to factor out the actual parts that differ and will explain them as we go along.

For convenience, we often omit the step annotation on the restriction operator when it is obvious from context, *e.g.,* we would write $(k-j-1, \lfloor w \rfloor)$ instead of $(k-j-1, \lfloor w \rfloor_{k-j-1})$. Furthermore, at many places we are supposed to establish the well-typedness conditions imposed by the definition of $\mathrm{Atom}[\tau_1, \tau_2]$. Since this is usually easy but somewhat tedious we will do it only in two representative cases, namely Lemma 10.21 and the first wrapping theorem (10.41).

## 10.2 Basic Lemmas

We start with a few very basic lemmas that are needed all the time. In particular, uses of Transitivity of World Extension (10.1) will usually not be mentioned explicitly.

**Lemma 10.1** (Transitivity of World Extension)**.**

1. If $(k'', w'') \sqsupseteq (k', w')$ and $(k', w') \sqsupseteq (k, w)$, then $(k'', w'') \sqsupseteq (k, w)$.
2. If $(k'', w'') \sqsupset (k', w')$ and $(k', w') \sqsupset (k, w)$, then $(k'', w'') \sqsupset (k, w)$.

*Proof:*

1. $k'' \leq k$ is clear, as are $w'' \in \mathrm{World}_{k''}$, $w''.\sigma_i \sqsupseteq w.\sigma_i$, $w''.\eta \sqsupseteq w.\eta$, and $\mathrm{rng}(w''.\eta^i) \setminus \mathrm{rng}(w.\eta^i) \subseteq \mathrm{dom}(w''.\sigma_i) \setminus \mathrm{dom}(w.\sigma_i)$. It remains to show $\rho'' \sqsupseteq \lfloor \rho \rfloor_{k''}$. So suppose $\alpha \in$

$$
\begin{array}{lcl}
V^\iota_n[\![\alpha]\!]\rho & \stackrel{\text{def}}{=} & \lfloor \rho(\alpha).R \rfloor_n \\
V^\iota_n[\![b]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,c,c) \in \mathrm{Atom}_n[b,b]\} \\
V^\iota_n[\![\tau \times \tau']\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\langle v_1, v_1'\rangle, \langle v_2, v_2'\rangle) \in \mathrm{Atom}_n[\rho^1(\tau\times\tau'), \rho^2(\tau\times\tau')] \mid \\
 & & \quad (k,w,v_1,v_2) \in V^\iota_n[\![\tau]\!]\rho \wedge (k,w,v_1',v_2') \in V^\iota_n[\![\tau']\!]\rho\} \\
V^\iota_n[\![\tau' \to \tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in \mathrm{Atom}_n[\rho^1(\tau'\to\tau), \rho^2(\tau'\to\tau)] \mid \\
 & & \quad \forall (k',w',v_1,v_2) \in V^{\neg\iota}_n[\![\tau']\!]\rho.\ (k',w') \sqsupseteq (k,w) \implies \\
 & & \quad (k',w',e_1[v_1/x], e_2[v_2/x]) \in E^\iota_n[\![\tau]\!]\rho\} \\
V^\iota_n[\![\forall\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\Lambda\alpha.e_1, \Lambda\alpha.e_2) \in \mathrm{Atom}_n[\rho^1(\forall\alpha.\tau), \rho^2(\forall\alpha.\tau)] \mid \\
 & & \quad \forall (k',w') \sqsupseteq (k,w).\ \forall(\tau_1,\tau_2,r) \in T^{\neg\iota}_{k'}[\![\Omega]\!]w'. \\
 & & \quad (k',w',e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in \triangleright E^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}r\} \\
V^\iota_n[\![\exists\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\mathsf{pack}\ \langle\tau_1,v_1\rangle\ \mathsf{as}\ \tau_1', \mathsf{pack}\ \langle\tau_2,v_2\rangle\ \mathsf{as}\ \tau_2') \in \mathrm{Atom}_n[\rho^1(\exists\alpha.\tau), \rho^2(\exists\alpha.\tau)] \mid \\
 & & \quad \exists r.\ (\tau_1,\tau_2,r) \in T^\iota_k[\![\Omega]\!]w \wedge (k,w,v_1,v_2) \in \triangleright V^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}r\} \\
V^\iota_n[\![\mu\alpha.\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in \mathrm{Atom}_n[\rho^1(\mu\alpha.\tau), \rho^2(\mu\alpha.\tau)] \mid \\
 & & \quad (k,w,v_1,v_2) \in \triangleright V^\iota_k[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho\} \\
E^\iota_n[\![\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,e_1,e_2) \in \mathrm{Atom}_n[\rho^1(\tau), \rho^2(\tau)] \mid \\
 & & \quad \forall j<k.\ \forall\sigma_1,v_1.\ (w.\sigma_1; e_1 \hookrightarrow^j \sigma_1; v_1) \Rightarrow \\
 & & \quad \exists w', v_2.\ (k-j, w') \sqsupseteq (k,w) \wedge w'.\sigma_1 = \sigma_1 \wedge \\
 & & \quad (w.\sigma_2; e_2 \hookrightarrow^* w'.\sigma_2; v_2) \wedge (k-j, w', v_1, v_2) \in V^\iota_n[\![\tau]\!]\rho\} \\
T^{\epsilon,-}_n[\![\Omega]\!]w & \stackrel{\text{def}}{=} & \{(w.\eta^1(\tau), w.\eta^2(\tau), (w.\rho^1(\tau), w.\rho^2(\tau), V_n[\![\tau]\!]w.\rho)) \mid \mathrm{ftv}(\tau) \subseteq \mathrm{dom}(w.\rho)\} \\
T^{\circ,+}_n[\![\Omega]\!]w & \stackrel{\text{def}}{=} & \{(\tau_1,\tau_2,(w.\sigma_1^*(\tau_1), w.\sigma_2^*(\tau_2), R)) \mid \\
 & & \quad \mathrm{ftv}(\tau_i) \subseteq \mathrm{dom}(w.\sigma_i) \wedge R \in \mathrm{Rel}_n[w.\sigma_1^*(\tau_1), w.\sigma_2^*(\tau_2)]\} \\
G^\iota_n[\![\epsilon]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,\emptyset,\emptyset) \mid k<n \wedge w \in \mathrm{World}_k\} \\
G^\iota_n[\![\Gamma, x{:}\tau]\!]\rho & \stackrel{\text{def}}{=} & \{(k,w,(\gamma_1, x{\mapsto}v_1), (\gamma_2, x{\mapsto}v_2)) \mid \\
 & & \quad (k,w,\gamma_1,\gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho \wedge (k,w,v_1,v_2) \in V^\iota_n[\![\tau]\!]\rho\} \\
D^\iota_n[\![\epsilon]\!]w & \stackrel{\text{def}}{=} & \{(\emptyset,\emptyset,\emptyset)\} \\
D^\iota_n[\![\Delta,\alpha]\!]w & \stackrel{\text{def}}{=} & \{((\delta_1,\alpha{\mapsto}\tau_1), (\delta_2,\alpha{\mapsto}\tau_2), (\rho,\alpha{\mapsto}r)) \mid \\
 & & \quad (\delta_1,\delta_2,\rho) \in D^\iota_n[\![\Delta]\!]w \wedge (\tau_1,\tau_2,r) \in T^\iota_n[\![\Omega]\!]w\} \\
D^\iota_n[\![\Delta,\alpha{\approx}\tau]\!]w & \stackrel{\text{def}}{=} & \{((\delta_1,\alpha{\mapsto}\alpha_1), (\delta_2,\alpha{\mapsto}\alpha_2), (\rho,\alpha{\mapsto}(\rho^1(\tau),\rho^2(\tau),V^{\iota\iota}_n[\![\tau]\!]\rho))) \mid \\
 & & \quad (\delta_1,\delta_2,\rho) \in D^\iota_n[\![\Delta]\!]w \wedge w.\sigma_i(\alpha_i) = \delta_i(\tau) \wedge \\
 & & \quad \exists\alpha'.\ \alpha_i = w.\eta^i(\alpha') \wedge w.\rho(\alpha').R = V^{\iota\iota}_n[\![\tau]\!]\rho\}
\end{array}
$$

$$
\begin{array}{lcl}
\Delta;\Gamma \vdash e_1 \precsim^\iota e_2 : \tau & \stackrel{\text{def}}{\iff} & \Delta;\Gamma \vdash e_1 : \tau \wedge \Delta;\Gamma \vdash e_2 : \tau \wedge \forall n \geq 0.\ \forall w_0 \in \mathrm{World}_n\,. \\
 & & \forall(\delta_1,\delta_2,\rho) \in D^{\neg\iota}_n[\![\Delta]\!]w_0.\ \forall(k,w,\gamma_1,\gamma_2) \in G^{\neg\iota}_n[\![\Gamma]\!]\rho. \\
 & & (k,w) \sqsupset (n,w_0) \Rightarrow (k,w,\delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota_n[\![\tau]\!]\rho
\end{array}
$$

Figure 11: Unified Definition of Our Logical Relations

$\mathrm{dom}(\lfloor\rho\rfloor_{k''})$. We show $\lfloor\rho\rfloor_{k''}(\alpha).R = \rho''(\alpha).R$ (the equality for the other components is shown analogously):

$$
\begin{array}{ll}
\lfloor\rho\rfloor_{k''}(\alpha).R = \lfloor\rho(\alpha).R\rfloor_{k''} = \lfloor\lfloor\rho(\alpha).R\rfloor_{k'}\rfloor_{k''} & \text{since } k'' \leq k' \\
= \lfloor\lfloor\rho\rfloor_{k'}(\alpha).R\rfloor_{k''} = \lfloor\rho'(\alpha).R\rfloor_{k''} & \text{since } \rho' \sqsupseteq \lfloor\rho\rfloor_{k'} \\
= \lfloor\rho'\rfloor_{k''}(\alpha).R = \rho''(\alpha).R & \text{since } \rho'' \sqsupseteq \lfloor\rho'\rfloor_{k''}
\end{array}
$$

2. Similar to (1).

□

**Lemma 10.2** (Restriction)**.**

1. If $k' \leq k$, then $V^\iota_{k'}[\![\tau]\!]\rho = \lfloor V^\iota_k[\![\tau]\!]\rho \rfloor_{k'}$.
2. If $k' \leq k$, then $E^\iota_{k'}[\![\tau]\!]\rho = \lfloor E^\iota_k[\![\tau]\!]\rho \rfloor_{k'}$.

Irrelevance (10.3) states that the logical relation does only depend on $\rho$'s interpretation of those variables that do indeed occur in $\tau$.

**Lemma 10.3** (Irrelevance)**.** If $\lfloor \rho' \rfloor_n \sqsupseteq \lfloor \rho \rfloor_n$ and $\mathrm{ftv}(\tau) \subseteq \mathrm{dom}(\rho)$, then

1. $V^\iota_n[\![\tau]\!]\rho' = V^\iota_n[\![\tau]\!]\rho$,
2. $E^\iota_n[\![\tau]\!]\rho' = E^\iota_n[\![\tau]\!]\rho$, and
3. $G^\iota_n[\![\tau]\!]\rho' = G^\iota_n[\![\tau]\!]\rho$.

*Proof:* (1) and (2) by mutual induction on $\tau$.

1. - Case $\tau = \alpha$:

$$\begin{array}{lll} & (k, w, v_1, v_2) \in V^\iota_n[\![\alpha]\!]\rho' & \\ \iff & (k, w, v_1, v_2) \in \lfloor \rho'(\alpha).R \rfloor_n & \\ \iff & (k, w, v_1, v_2) \in \lfloor \rho' \rfloor_n(\alpha).R & \\ \iff & (k, w, v_1, v_2) \in \lfloor \rho \rfloor_n(\alpha).R & \text{by assumption} \\ \iff & (k, w, v_1, v_2) \in V^\iota_n[\![\alpha]\!]\rho & \end{array}$$

   - Case $\tau = b$: Trivial.
   - The remaining cases all follow easily by induction.
2. Follows from (1).
3. Follows from (1).

□

The next lemma is a combination of Restriction (10.2) and Irrelevance (10.3), but for the type and type substitution relations.

**Lemma 10.4.**

1. If $(\tau_1, \tau_2, r) \in T^\iota_n[\![\Omega]\!]w_0$ and $(k, w) \sqsupseteq (n, w_0)$, then $(\tau_1, \tau_2, \lfloor r \rfloor_k) \in T^\iota_k[\![\Omega]\!]w$.
2. If $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$ and $(k, w) \sqsupseteq (n, w_0)$, then $(\delta_1, \delta_2, \lfloor \rho \rfloor_k) \in D^\iota_k[\![\Delta]\!]w$.

*Proof:*

1. Follows easily by Restriction (10.2) and Irrelevance (10.3).
2. By induction on $\Delta$.
   - Case $\Delta = \epsilon$: Trivial.
   - Case $\Delta = \Delta', \alpha$: Follows by induction and part (1).

- Case $\Delta = \Delta', \alpha{\approx}\tau$:
  - We know $\delta_i = \delta_i', \alpha{\mapsto}\alpha_i$ and $\rho = \rho', \alpha{\mapsto}(\rho^1(\tau), \rho^2(\tau), V_n^{\iota\iota\iota}[\![\tau]\!]\rho')$ where $(\delta_1', \delta_2', \rho') \in D_n^\iota[\![\Delta']\!]w_0$, $w_0.\sigma_i(\alpha_i) = \delta_i(\tau)$, and $w_0.\rho(\alpha').R = V_n^{\iota\iota\iota}[\![\tau]\!]\rho'$ for some $\alpha'$.
  - By induction, $(\delta_1', \delta_2', \lfloor\rho'\rfloor) \in D_k^\iota[\![\Delta']\!]w$.
  - $w.\sigma_i(\alpha_i) = \delta_i(\tau)$ is clear.
  - We need to show $w.\rho(\alpha').R = \lfloor V_n^{\iota\iota\iota}[\![\tau]\!]\lfloor\rho'\rfloor_k\rfloor_k$ and $\lfloor V_n^{\iota\iota\iota}[\![\tau]\!]\rho'\rfloor_k = V_k^{\iota\iota\iota}[\![\tau]\!]\lfloor\rho'\rfloor_k$.
  - Both follow by Restriction (10.2) and Irrelevance (10.3), the former using the fact that $w \in \mathrm{CandWorld}_k$.

□

**Lemma 10.5** (Inclusion)**.** $V_n^\iota[\![\tau]\!]\rho \subseteq E_n^\iota[\![\tau]\!]\rho$

*Proof:* Follows easily from the definition of $E_n^\iota[\![\tau]\!]\rho$, taking the future world to be the current one. □

## 10.3 Validity

The first important property to show is that, under the assumption $\rho \in \mathrm{Interp}$ (instead of just $\rho \in \mathrm{CandInterp}$), the logical value relation itself is a valid relation, *i.e.,* an element of Rel. Since, in the end, we only care about interpretations out of Interp and worlds out of World, this can also be understood as establishing the well-definedness of the logical relation.

For the sake of convenience, whenever we write $V_n^\iota[\![\tau]\!]\rho$, $E_n^\iota[\![\tau]\!]\rho$, $G_n^\iota[\![\Gamma]\!]\rho$, $D_n^\iota[\![\Delta]\!]w$, $T_n^\iota[\![\Omega]\!]w$ from now on, we assume $\rho \in \mathrm{Interp}$ and $w \in \mathrm{World}$.

As a first step, we note that every element of any of the value or term relations is a proper atom.

**Lemma 10.6** (Atomicity)**.**

1. $V_n^\iota[\![\tau]\!]\rho \subseteq \mathrm{Atom}_n^{\mathrm{val}}[\rho^1(\tau), \rho^2(\tau)]$
2. $E_n^\iota[\![\tau]\!]\rho \subseteq \mathrm{Atom}_n[\rho^1(\tau), \rho^2(\tau)]$

*Proof:* By definition. □

The key property of Rel is that its elements must be closed under world extension. Proving this for the value relation is very easy because the property has mostly been built into its definition. In particular, since in the inductive cases $\rho$ is not extended, it is not necessary to prove Closure Under World Extension and LR-Validity (see below) by mutual induction. This would be necessary otherwise since the induction hypothesis requires the extended $\rho$ to be in Interp.

**Lemma 10.7** (Closure Under World Extension)**.**

1. If $(k, w, v_1, v_2) \in V_n^\iota[\![\tau]\!]\rho$ and $(k', w') \sqsupseteq (k, w)$, then $(k', w', v_1, v_2) \in V_n^\iota[\![\tau]\!]\rho$.
2. If $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ and $(k', w') \sqsupseteq (k, w)$, then $(k', w', \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$.

*Proof:*

1. By induction on $\tau$.
   - Case $\tau = \alpha$: Since $\rho \in \mathrm{Interp}$, we know $\rho(\alpha).R \in \mathrm{Rel}_m$ for some $m$ and thus $\rho(\alpha).R$ is closed under world extension.

- Case $\tau = b$: Trivial.
- Case $\tau = \tau_1 \times \tau_2$: Follows easily by induction.
- Case $\tau = \tau_1 \to \tau_2$: $v_i = \lambda x{:}\tau_i'.e_i$.
  - Suppose $(k'', w'', v_3, v_4) \in V_n^{\neg\iota}[\![\tau_1]\!]\rho$ where $(k'', w'') \sqsupseteq (k', w')$.
  - To show: $(k'', w'', e_1[v_3/x], e_2[v_4/x]) \in E_n^{\iota}[\![\tau_2]\!]\rho$.
  - By Transitivity of World Extension (10.1), $(k'', w'') \sqsupseteq (k, w)$.
  - Now instantiate the assumption to get $(k'', w'', e_1[v_3/x], e_2[v_4/x]) \in E_n^{\iota}[\![\tau_2]\!]\rho$.
- Case $\tau = \forall\alpha.\tau'$: $v_i = \Lambda\alpha.e_i$.
  - Suppose $(k''', w''') \sqsupset (k'', w'') \sqsupseteq (k', w')$ and $(\tau_1, \tau_2, r) \in T_{k''}^{\neg\iota}[\![\Omega]\!]w''$.
  - To show: $(k''', w''', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n^{\iota}[\![\tau']\!]\rho, \alpha{\mapsto}r$.
  - By Transitivity of World Extension (10.1), $(k'', w'') \sqsupseteq (k, w)$.
  - Now instantiating the assumption yields the claim.
- Case $\tau = \exists\alpha.\tau'$: $v_i = \mathsf{pack}\ \langle \tau_i, v_i' \rangle\ \mathsf{as}\ \tau_i'$.
  - By assumption there is $r$ such that $(\tau_1, \tau_2, r) \in T_k^{\iota}[\![\Omega]\!]w$ and $(k'', w'', v_1', v_2') \in V_n^{\iota}[\![\tau']\!]\rho, \alpha{\mapsto}r$ for any $(k'', w'') \sqsupset (k, w)$.
  - By Lemma 10.4 we have $(\tau_1, \tau_2, \lfloor r \rfloor_{k'}) \in T_{k'}^{\iota}[\![\Omega]\!]w'$.
  - We show that $(k'', w'', v_1', v_2') \in V_n^{\iota}[\![\tau']\!]\rho, \alpha{\mapsto}\lfloor r \rfloor_{k'}$ for any $(k'', w'') \sqsupset (k', w')$.
  - Suppose $(k'', w'') \sqsupset (k', w')$.
  - By Transitivity of World Extension (10.1), $(k'', w'') \sqsupset (k, w)$.
  - Hence $(k'', w'', v_1', v_2') \in V_n^{\iota}[\![\tau']\!]\rho, \alpha{\mapsto}r$ and thus the claim follows by Restriction (10.2) and Irrelevance (10.3).
- Case $\tau = \mu\alpha.\tau'$: $v_i = \mathsf{roll}\ v_i'\ \mathsf{as}\ \tau_i$.
  - Suppose $(k'', w'') \sqsupset (k', w')$.
  - To show: $(k'', w'', v_1', v_2') \in V_{k'}^{\iota}[\![\tau'[\tau/\alpha]]\!]\rho$
  - By Transitivity of World Extension (10.1), $(k'', w'') \sqsupset (k, w)$.
  - Now instantiating the assumption yields $(k'', w'', v_1', v_2') \in V_k^{\iota}[\![\tau'[\tau/\alpha]]\!]\rho$.
  - The claim then follows by Restriction (10.2).

2. Follows from (1).

□

**Lemma 10.8** (LR-Validity)**.** $V_n^{\iota}[\![\tau]\!]\rho \in \mathrm{Rel}_n[\rho^1(\tau), \rho^2(\tau)]$

*Proof:* Follows from Atomicity (10.6) and Closure Under World Extension (10.7). □

Also notice that we did not need induction on the step index to show validity—even in the presence of recursive types.

## 10.4 Compatibility

The next big goal is to show the fundamental properties and soundness of $\precsim$ with respect to contextual approximation. The basic building blocks for this are what Pitts calls *compatibility* lemmas [17], which state that the logical relations are closed under the constructs of the language. Since we do not have a fundamental property for the negative relation and can only show the one

for the positive relation indirectly, we do not even bother here to try showing some compatibility properties for them. Instead, we focus on $\iota \in \{\epsilon, \circ\}$.

We first have two properties about syntactic type substitutions, which will be needed for proving well-formedness of different syntactic elements.

**Lemma 10.9.** If $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w$, then

1. $\rho^i = w.\sigma_i^* \cdot \delta_i$
2. $w.\sigma_i \vdash \delta_i : \Delta$
3. $\epsilon \vdash \rho^i : \Delta$

*Proof:* Follows easily from the definitions of $D^\iota$ and World. □

**Lemma 10.10.** If $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$, then $w.\sigma_i; \epsilon \vdash \gamma_i : \rho^i(\Gamma)$.

*Proof:* Follows easily from the definition of $G^\iota$. □

The following is a standard substitution lemma for the logical relations, which describes the connection between syntactic and semantic type substitutions. It is mainly needed for showing compatibility with the constructs associated with quantified types. The induction on $n$ is needed in the case of a recursive type.

**Lemma 10.11** (LR-Substitution)**.** For $\iota \in \{\epsilon, \circ\}$:

1. $V^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho) = V^\iota_n[\![\tau[\tau'/\alpha]]\!]\rho$.
2. $E^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho) = E^\iota_n[\![\tau[\tau'/\alpha]]\!]\rho$.

*Proof:* By primary induction on $n$ and secondary induction on $\tau$.

1. 
   - Case $\tau = \alpha$: Trivial.
   - Case $\tau = \alpha' \neq \alpha$ or $\tau = b$: Follows by Irrelevance (10.3).
   - Case $\tau = \tau_0 \times \tau_0'$: Follows easily by induction.
   - Case $\tau = \tau_0' \to \tau_0$: Follows easily by induction.
   - Case $\tau = \forall\alpha'.\tau''$:
     - We show $V^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho) \subseteq V^\iota_n[\![\tau[\tau'/\alpha]]\!]\rho$. The other direction is symmetric.
     - Suppose $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho)$.
     - Suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T^{\neg\iota}_{k'}[\![\Omega]\!]w'$.
     - To show: $(k'', w'', e_1[\tau_1/\alpha'], e_2[\tau_2/\alpha']) \in E^\iota_n[\![\tau''[\tau'/\alpha]]\!]\rho, \alpha'{\mapsto}r$
     - Since $\neg\iota = \iota$ and thus $T^{\neg\iota}_{k'}[\![\Omega]\!]w' = T^\iota_{k'}[\![\Omega]\!]w'$, we know: $(k'', w'', e_1[\tau_1/\alpha'], e_2[\tau_2/\alpha']) \in E^\iota_n[\![\tau'']\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho), \alpha'{\mapsto}r$
     - By Irrelevance (10.3), $(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho) = ((\rho, \alpha'{\mapsto}r)^1(\tau'), (\rho, \alpha'{\mapsto}r)^2(\tau'), V^\iota_n[\![\tau']\!]\rho, \alpha'{\mapsto}r)$.
     - Hence by induction, $(k'', w'', e_1[\tau_1/\alpha'], e_2[\tau_2/\alpha']) \in E^\iota_n[\![\tau''[\tau'/\alpha]]\!]\rho, \alpha'{\mapsto}r$.
   - Case $\tau = \exists\alpha'.\tau''$: Analogously to the previous case.
   - Case $\tau = \mu\alpha'.\tau''$:
     - We show $V^\iota_n[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V^\iota_n[\![\tau']\!]\rho) \subseteq V^\iota_n[\![\tau[\tau'/\alpha]]\!]\rho$. The other direction is symmetric.

- Suppose $(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V_n^\iota[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V_n^\iota[\![\tau']\!]\rho)$.
- Suppose $(k', w') \sqsupset (k, w)$.
- To show: $(k', w', v_1, v_2) \in V_k^\iota[\![\tau''[\tau'/\alpha][\tau[\tau'/\alpha]/\alpha']]\!]\rho$
- By assumption, $(k', w', v_1, v_2) \in V_k^\iota[\![\tau''[\tau/\alpha']]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V_n^\iota[\![\tau']\!]\rho)$.
- By Irrelevance (10.3) and Restriction (10.2),
  $(k', w', v_1, v_2) \in V_k^\iota[\![\tau''[\tau/\alpha']]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V_k^\iota[\![\tau']\!]\rho)$.
- Hence by induction, $(k', w', v_1, v_2) \in V_k^\iota[\![\tau''[\tau/\alpha'][\tau'/\alpha]]\!]\rho$.
- Note that $\tau''[\tau/\alpha'][\tau'/\alpha] = \tau''[\tau'/\alpha][\tau[\tau'/\alpha]/\alpha']$.

2. Follows from part (1).

□

LR-Substitution does not hold for the polarized relations. Consider the case where $\tau = \alpha \to \alpha$. Then, for instance, $V_n^+[\![\tau]\!]\rho, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V_n^+[\![\tau']\!]\rho)$ tells us something about how its elements behave when applied to arguments out of $V_n^+[\![\tau']\!]\rho$. However, $V_n^+[\![\tau[\tau'/\alpha]]\!]\rho$ only tells us something about how its elements behave when applied to arguments out of $V_n^-[\![\tau']\!]\rho$.

We still conjecture a weaker property to hold for the polarized relations, namely where the equality is replaced by inclusion ($\subseteq$ for $\iota = +$ and $\supseteq$ for $\iota = -$). We did not prove it, though, as it does not seem to be very useful.

The following two lemmas are needed for dealing with the particularities of the non-parametric logical relation. We know by the definition of $T$ and $D$ that for any $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$ and any $\alpha$ bound in $\Delta$ there is some $\tau_\alpha$ such that $\delta_1(\alpha)$ and $\delta_2(\alpha)$ are the concretizations of $\tau_\alpha$ w.r.t. $w_0$, *i.e.,* $\delta_1(\alpha) = w_0.\eta^1(\tau)$ and $\delta_2(\alpha) = w_0.\eta^2(\tau)$. We define an operation, au, that yields the substitution $\delta$ that maps each $\alpha$ to its corresponding $\tau_\alpha$ (see Lemma 10.13.1):

**Definition 10.12** (Anti-Unifier)**.** Assume that $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w$. The anti-unifying substitution of $\delta_1$ and $\delta_2$ with respect to $w.\eta$, written $\mathrm{au}(\delta_1, \delta_2, w.\eta)$, is defined as follows.

$$\mathrm{au}(\epsilon, \epsilon, \eta) \stackrel{\text{def}}{=} \epsilon$$

$$\mathrm{au}((\delta_1, \alpha{\mapsto}\tau_1), (\delta_2, \alpha{\mapsto}\tau_2), \eta) \stackrel{\text{def}}{=} \mathrm{au}(\delta_1, \delta_2, \eta), \alpha{\mapsto}\tau \qquad \text{where } \tau = \eta^{-1}(\tau_1) = \eta^{-2}(\tau_2)$$

Here, $\eta^{-i}$ is short for $(\eta^i)^{-1}$, the inverse of $\eta^i$. The latter exists, because the definition of Conc ensures that $\eta^i$ is injective. Furthermore, since $\eta$ is a partial bijection on the generated type names, $\eta^{-1}(\tau_1)$ and $\eta^{-2}(\tau_2)$ are guaranteed to be equal.

**Lemma 10.13.**

1. If $\delta = \mathrm{au}(\delta_1, \delta_2, \eta)$, then $\delta_1 = \eta^1 \cdot \delta$ and $\delta_2 = \eta^2 \cdot \delta$.
2. If $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$, $\delta = \mathrm{au}(\delta_1, \delta_2, w_0.\eta)$, and $(k, w) \sqsupseteq (n, w_0)$, then $\delta = \mathrm{au}(\delta_1, \delta_2, w.\eta)$.

*Proof:*

1. Follows easily from the definition.
2. First, note that $(\delta_1, \delta_2, \lfloor\rho\rfloor_k) \in D_k[\![\Delta]\!]w$ by Lemma 10.4. Furthermore, since $w.\eta$ is an extension of $w_0.\eta$, the former agrees with the latter on $\mathrm{dom}(w_0.\eta)$. As we know $\mathrm{ftv}(\delta_i(\alpha)) \subseteq \mathrm{rng}(w_0.\eta)$ for any $\alpha$, it is clear that $\mathrm{au}(\delta_1, \delta_2, w_0.\eta) = \mathrm{au}(\delta_1, \delta_2, w.\eta)$.

□

The motivation for defining au is the following property, which is crucial for proving compatibility of $\precsim$ for the rules INST, PACK, and CAST, in which its non-parametricity becomes manifest.

**Lemma 10.14.** If $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$ and $\delta = \mathrm{au}(\delta_1, \delta_2, w_0.\eta)$ and $\Delta \vdash \tau$, then:

1. $V_n[\![\tau]\!]\rho = V_n[\![\delta(\tau)]\!]w_0.\rho$
2. $E_n[\![\tau]\!]\rho = E_n[\![\delta(\tau)]\!]w_0.\rho$

*Proof:* By primary induction on $n$ and secondary induction on the derivation of $\Delta \vdash \tau$.

1. 
   - Case $\tau = \alpha$ were $\alpha \in \Delta$:
     - Then it is easy to see that $\Delta = \Delta_1, \alpha, \Delta_2$, $\delta_i = \delta_{i1}, \alpha{\mapsto}\tau_i, \delta_{i2}$, and $\rho = \rho_1, \alpha{\mapsto}r, \rho_2$ where $(\tau_1, \tau_2, r) \in T_n[\![\Omega]\!]w_0$.
     - The latter implies $\tau_i = w_0.\eta^i(\tau')$ and $r = (\_, \_, V_n[\![\tau']\!]w_0.\rho)$.
     - Hence $V_n[\![\alpha]\!]\rho = V_n[\![\tau']\!]w_0.\rho = V_n[\![\delta(\alpha)]\!]w_0.\rho$.
   - Case $\tau = \alpha$ where $\alpha{\approx}\tau' \in \Delta$:
     - Then it is easy to see that $\Delta = \Delta_1, \alpha{\approx}\tau', \Delta_2$, $\delta_i = \delta_{i1}, \alpha{\mapsto}\alpha_i, \delta_{i2}$, and $\rho = \rho_1, \alpha{\mapsto}(\rho_1^1(\tau'), \rho_1^2(\tau'), V_n[\![\tau']\!]\rho_1), \rho_2$ with $\alpha_i = w_0.\eta^i(\alpha')$ and $w_0.\rho(\alpha').R = V_n[\![\tau']\!]\rho_1$ for some $\alpha'$.
     - Because of the injectivity of $w_0.\eta^i$, $w_0.\eta^i(\alpha') = \alpha_i = \delta_i(\alpha) = w_0.\eta^i\delta(\alpha)$ implies $\alpha' = \delta(\alpha)$.
     - Hence $V_n[\![\alpha]\!]\rho = V_n[\![\tau']\!]\rho_1 = V_n[\![\alpha']\!]w_0.\rho = V_n[\![\delta(\alpha)]\!]w_0.\rho$.
   - Case $\tau = b$: Trivial.
   - Case $\tau = \tau_1 \times \tau_2$ with $\Delta \vdash \tau_1$ and $\Delta \vdash \tau_2$: Follows easily by induction.
   - Case $\tau = \tau_1 \to \tau_2$ with $\Delta \vdash \tau_1$ and $\Delta \vdash \tau_2$: Follows easily by induction.
   - Case $\tau = \forall\alpha.\tau'$ with $\Delta, \alpha \vdash \tau'$:
     - We show $V_n[\![\tau]\!]\rho \subseteq V_n[\![\delta(\tau)]\!]w_0.\rho$; the other direction is symmetric.
     - Suppose $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n[\![\forall\alpha.\tau']\!]\rho$.
     - Suppose further $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}[\![\Omega]\!]w'$.
     - We know $(k'', w'', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
     - To show: $(k'', w'', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n[\![\delta(\tau')]\!]w_0.\rho, \alpha{\mapsto}r$
     - By Restriction (10.2) and Irrelevance (10.3), this reduces to showing $E_{k'}[\![\tau']\!]\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r = E_{k'}[\![\delta(\tau')]\!]w'.\rho, \alpha{\mapsto}r$.
     - By assumption and Lemma 10.4, $(\delta_1, \delta_2, \lfloor\rho\rfloor_{k'}) \in D_{k'}[\![\Delta]\!]w'$.
     - Let $(\delta_1', \delta_2', \rho') := ((\delta_1, \alpha{\mapsto}\tau_1), (\delta_2, \alpha{\mapsto}\tau_2), (\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r))$, so $(\delta_1', \delta_2', \rho') \in D_{k'}[\![\Delta, \alpha]\!]w'$.
     - By Lemma 10.13, $\delta = \mathrm{au}(\delta_1, \delta_2, w'.\eta)$.
     - Since $(\tau_1, \tau_2, r) \in T_{k'}[\![\Omega]\!]w'$ we know $\tau_i = w'.\eta^i(\tau'')$ and $r = (w'.\rho^1(\tau''), w'.\rho^2(\tau''), V_{k'}[\![\tau'']\!]w'.\rho)$.
     - It is easy to see then that $\delta, \alpha{\mapsto}\tau'' = \mathrm{au}(\delta_1', \delta_2', w'.\eta)$.
     - Hence by induction, $E_{k'}[\![\tau']\!]\rho' = E_{k'}[\![\delta(\tau')[\tau''/\alpha]]\!]w'.\rho$.
     - By LR-Substitution (10.11), $E_{k'}[\![\delta(\tau')[\tau''/\alpha]]\!]w'.\rho = E_{k'}[\![\delta(\tau')]\!]w'.\rho, \alpha{\mapsto}(w'.\rho^1(\tau''), w'.\rho^2(\tau''), V_{k'}[\![\tau'']\!]w'.\rho) = E_{k'}[\![\delta(\tau')]\!]w'.\rho, \alpha{\mapsto}r$.

- Case $\tau = \exists\alpha.\tau'$ with $\Delta, \alpha \vdash \tau'$: analogously to previous case
- Case $\tau = \mu\alpha.\tau'$ with $\Delta, \alpha \vdash \tau'$:
  - We show $V_n[\![\tau]\!]\rho \subseteq V_n[\![\delta(\tau)]\!]w_0.\rho$; the other direction is symmetric.
  - Suppose $(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V_n[\![\mu\alpha.\tau']\!]\rho$.
  - Suppose further $(k', w') \sqsupset (k, w)$.
  - We know $(k', w', v_1, v_2) \in V_k[\![\tau'[\tau/\alpha]]\!]\rho$.
  - To show: $(k', w', v_1, v_2) \in V_k[\![\delta(\tau')[\delta(\tau)/\alpha]]\!]w_0.\rho$
  - Note that $\delta(\tau')[\delta(\tau)/\alpha] = \delta(\tau'[\tau/\alpha])$.
  - Hence the claim follows by Substitution (A.2.1) and induction.

2. Follows immediately from part (1).

□

Many of the compatibility proofs are straightforward, *i.e.,* they do not deal with the worlds in any interesting way and the non-parametricity does not show up because it is hidden in $T$. Those proofs are thus essentially analogous to their counterparts in an $\mathrm{F}^\mu$ setting [3] and we will not comment on them further.

**Lemma 10.15** (Compatibility: VAR)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta \vdash \Gamma$ and $x{:}\tau \in \Gamma$, then $\Delta; \Gamma \vdash x \precsim^\iota x : \tau$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(x), \delta_2\gamma_2(x)) \in E^\iota_n[\![\tau]\!]\rho$
- By Inclusion (10.5) it suffices to show $(k, w, \delta_1\gamma_1(x), \delta_2\gamma_2(x)) \in V^\iota_n[\![\tau]\!]\rho$.
- By assumption we have $(k, w, \gamma_1(x), \gamma_2(x)) \in V^\iota_n[\![\tau]\!]\rho$.
- Since thereby $\vdash w.\sigma_i; \gamma_i(x) : \rho^1(\tau)$, we know $\gamma_i(x) = \delta_i\gamma_i(x)$.

□

**Lemma 10.16** (Compatibility: CON)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta \vdash \Gamma$, then $\Delta; \Gamma \vdash c \precsim^\iota c : b$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(c), \delta_2\gamma_2(c) \in E^\iota_n[\![b]\!]\rho$, *i.e.,* $(k, w, c, c) \in E^\iota_n[\![b]\!]\rho$.
- By Inclusion (10.5) it suffices to show $(k, w, c, c) \in V^\iota_n[\![b]\!]\rho$, which holds trivially.

□

**Lemma 10.17** (Compatibility: PAIR)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau_1$ and $\Delta; \Gamma \vdash e_3 \precsim^\iota e_4 : \tau_2$, then $\Delta; \Gamma \vdash \langle e_1, e_3\rangle \precsim^\iota \langle e_2, e_4\rangle : \tau_1 \times \tau_2$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\langle e_1, e_3\rangle), \delta_2\gamma_2(\langle e_2, e_4\rangle)) \in E^\iota_n[\![\tau_1 \times \tau_2]\!]\rho$.
- Assume that $w.\sigma_1; \delta_1\gamma_1(\langle e_1, e_3\rangle)$ terminates in $j_1 + j_2 =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1\gamma_1(\langle e_1, e_3\rangle) \\ \hookrightarrow^{j_1} & \sigma_1'; \langle v_1, \delta_1(\gamma_1(e_3))\rangle \\ \hookrightarrow^{j_2} & \sigma_1; \langle v_1, v_3\rangle \end{array}$$

- Instantiate the first assumption to get $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota_n[\![\tau_1]\!]\rho$.
- Consequently there exists $(k - j_1, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(\langle e_2, e_4\rangle) \hookrightarrow^* w'.\sigma_2; \langle v_2, \delta_2\gamma_2(e_4)\rangle$ with $w'.\sigma_1 = \sigma_1'$ and $(k - j_1, w', v_1, v_2) \in V^\iota_n[\![\tau_1]\!]\rho$.
- Instantiate the second assumption and apply Closure Under World Extension (10.7) to get $(k - j_1, w', \delta_1\gamma_1(e_3), \delta_2\gamma_2(e_4)) \in E^\iota_n[\![\tau_2]\!]\rho$.
- Consequently there exists $(k - j, w'') \sqsupseteq (k - j_1, w')$ such that $w'.\sigma_2; \langle v_2, \delta_2\gamma_2(e_4)\rangle \hookrightarrow^* w''.\sigma_2; \langle v_2, v_4\rangle$ with $w''.\sigma_1 = \sigma_1$ and $(k - j, w'', v_3, v_4) \in V^\iota_n[\![\tau_2]\!]\rho$.
- Since, by Closure Under World Extension, also $(k - j, w'', v_1, v_2) \in V^\iota_n[\![\tau_1]\!]\rho$, we get $(k - j, w'', \langle v_1, v_3\rangle, \langle v_2, v_4\rangle) \in V^\iota_n[\![\tau_1 \times \tau_2]\!]\rho$.

□

**Lemma 10.18** (Compatibility: PROJ)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau_1 \times \tau_2$, then $\Delta; \Gamma \vdash e_1.i \precsim^\iota e_2.i : \tau_i$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(e_1.i), \delta_2\gamma_2(e_2.i)) \in E^\iota_n[\![\tau_i]\!]\rho$.
- Assume that $w.\sigma_1; \delta_1\gamma_1(e_1.i)$ terminates in $j' + 1 =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1\gamma_1(e_1.i) \\ \hookrightarrow^{j'} & \sigma_1; \langle v_{11}, v_{12}\rangle.i \\ \hookrightarrow^{1} & \sigma_1; v_{1i} \end{array}$$

- Instantiate the assumption to get $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota_n[\![\tau_1 \times \tau_2]\!]\rho$.
- Consequently there is $(k - j', w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(e_2.i) \hookrightarrow^* w'.\sigma_2; \langle v_{21}, v_{22}\rangle.i$ with $w'.\sigma_1 = \sigma_1$ and $(k - j', w', \langle v_{11}, v_{12}\rangle, \langle v_{21}, v_{22}\rangle) \in V^\iota_n[\![\tau_1 \times \tau_2]\!]\rho$.
- By Closure Under World Extension (10.7), $(k - j, \lfloor w'\rfloor, \langle v_{11}, v_{12}\rangle, \langle v_{21}, v_{22}\rangle) \in V^\iota_n[\![\tau_1 \times \tau_2]\!]\rho$.
- Hence $(k - j, \lfloor w'\rfloor, v_{1i}, v_{2i}) \in V^\iota_n[\![\tau_i]\!]\rho$.

□

**Lemma 10.19** (Compatibility: ABS)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma, x{:}\tau' \vdash e_1 \precsim^\iota e_2 : \tau$, then $\Delta; \Gamma \vdash \lambda x{:}\tau'.e_1 \precsim^\iota \lambda x{:}\tau'.e_2 : \tau' \to \tau$.

*Proof:*

- Suppose $w_0 \in \text{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\lambda x{:}\tau'.e_1), \delta_2\gamma_2(\lambda x{:}\tau'.e_2)) \in E_n^\iota[\![\tau' \to \tau]\!]\rho$
- By Inclusion (10.5) it suffices to show $(k, w, \delta_1\gamma_1(\lambda x{:}\tau'.e_1), \delta_2\gamma_2(\lambda x{:}\tau'.e_2)) \in V_n^\iota[\![\tau' \to \tau]\!]\rho$.
- So suppose $(k', w', v_1, v_2) \in V_n^\iota[\![\tau']\!]\rho$ where $(k', w') \sqsupseteq (k, w)$.
- To show: $(k', w', (\delta_1\gamma_1(e_1))[v_1/x], (\delta_2\gamma(e_2))[v_2/x]) \in E_n^\iota[\![\tau]\!]\rho$
- Let $\gamma_i' := \gamma_i, x{\mapsto}v_i$.
- As $(k', w', \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ by Closure Under World Extension (10.7), this means $(k', w', \gamma_1', \gamma_2') \in G_n^\iota[\![\Gamma, x{:}\tau']\!]\rho$.
- Now instantiate the assumption to get $(k', w', \delta_1\gamma_1'(e_1), \delta_2\gamma_2'(e_2)) \in E_n^\iota[\![\tau]\!]\rho$.
- Note that $\delta_i\gamma_i'(e_i) = \delta_i(\gamma_i(e_i)[v_i/x])$.
- Furthermore, since $(k', w', v_1, v_2) \in V_n^\iota[\![\tau']\!]\rho$ implies $\vdash w'.\sigma_i; v_i : \rho^i(\tau')$, we have $\delta_i(v_i) = v_i$ and thus $\delta_i(\gamma_i(e_i)[v_i/x]) = (\delta_i\gamma_i(e_i))[v_i/x]$.

□

**Lemma 10.20** (Compatibility: APP)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau_1 \to \tau_2$ and $\Delta; \Gamma \vdash e_3 \precsim^\iota e_4 : \tau_1$, then $\Delta; \Gamma \vdash e_1\ e_3 \precsim^\iota e_2\ e_4 : \tau_2$.

*Proof:*

- Suppose $w_0 \in \text{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(e_1\ e_3), \delta_2\gamma_2(e_2\ e_4)) \in E_n^\iota[\![\tau_2]\!]\rho$
- Assume that $w.\sigma_1; \delta_1\gamma_1(e_1\ e_3)$ terminates in $j_1 + j_2 + 1 + j_3 =: j < k$ steps:

$$\begin{array}{ll} & w.\sigma_1; \delta_1\gamma_1(e_1\ e_3) \\ \hookrightarrow^{j_1} & \sigma_1'; (\lambda x{:}\tau_1'.e_1')\ \delta_1(\gamma_1(e_3)) \\ \hookrightarrow^{j_2} & \sigma_1''; (\lambda x{:}\tau_1'.e_1')\ v_3 \\ \hookrightarrow^{1} & \sigma_1''; e_1'[v_3/x] \\ \hookrightarrow^{j_3} & \sigma_1; v_1 \end{array}$$

- Instantiating the first assumption yields the existence of $(k - j_1, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(e_2\ e_4) \hookrightarrow^* w'.\sigma_2; (\lambda x{:}\tau_2'.e_2')\ \delta_2\gamma_2(e_4)$ and $w'.\sigma_1 = \sigma_1'$ and $(k - j_1, w', \lambda x{:}\tau_1'.e_1', \lambda x{:}\tau_2'.e_2') \in V_n^\iota[\![\tau_1 \to \tau_2]\!]\rho$.
- By Closure Under World Extension (10.7) we have $(k - j_1, w', \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$.
- Hence instantiating the second assumption yields the existence of $(k - j_1 - j_2, w'') \sqsupseteq (k - j_1, w')$ such that $w'.\sigma_2; (\lambda x{:}\tau_2'.e_2')\ \delta_2(\gamma_2(e_4)) \hookrightarrow^* w''.\sigma_2; (\lambda x{:}\tau_2'.e_2')\ v_4$ and $w''.\sigma_1 = \sigma_1''$ and $(k - j_1 - j_2, w'', v_3, v_4) \in V_n^\iota[\![\tau_1]\!]\rho$.
- Consequently, $(k - j_1 - j_2, w'', e_1'[v_3/x], e_2'[v_4/x]) \in E_n^\iota[\![\tau_2]\!]\rho$.
- By Closure Under World Extension (10.7), $(k - j_1 - j_2 - 1, \lfloor w'' \rfloor, e_1'[v_3/x], e_2'[v_4/x]) \in E_n^\iota[\![\tau_2]\!]\rho$.

- Therefore there exists $(k-j, w''') \sqsupseteq (k-j_1-j_2-1, \lfloor w'' \rfloor)$ such that $w''.\sigma_2; e_2'[v_4/x] \hookrightarrow^* w'''.\sigma_2; v_2$ and $w'''.\sigma_1 = \sigma_1$ and $(k-j, w''', v_1, v_2) \in V_n^\iota[\![\tau_2]\!]\rho$. □

**Lemma 10.21** (Compatibility: GEN)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta, \alpha; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau$, then $\Delta; \Gamma \vdash \Lambda\alpha.e_1 \precsim^\iota \Lambda\alpha.e_2 : \forall\alpha.\tau$.

*Proof:*

- $\Delta; \Gamma \vdash \Lambda\alpha.e_i : \forall\alpha.\tau$ follows from the assumption and the typing rule EGEN.
- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\Lambda\alpha.e_1), \delta_2\gamma_2(\Lambda\alpha.e_2)) \in E_n^\iota[\![\forall\alpha.\tau]\!]\rho$
- First, we need to show $\vdash w.\sigma_i; \delta_i\gamma_i(\Lambda\alpha.e_i) : \rho^i(\forall\alpha.\tau)$.
  - By Lemma 10.9 we know $w_0.\sigma_i \vdash \delta_i : \Delta$, hence $w_0.\sigma_i, \alpha \vdash \delta_i, \alpha{\mapsto}\alpha : \Delta, \alpha$.
  - So by assumption and Substitution (A.2), $w_0.\sigma_i, \alpha; \delta_i(\Gamma) \vdash \delta_i(e_i) : \delta_i(\tau)$.
  - By Weakening (A.1), $w.\sigma_i, \alpha; \delta_i(\Gamma) \vdash \delta_i(e_i) : \delta_i(\tau)$.
  - By Lemma 10.10 we know $w.\sigma_i; \epsilon \vdash \gamma_i : \rho^i(\Gamma)$.
  - By Lemmas 10.4 and 10.9 we know $\rho^i(\Gamma) = w.\sigma_i^*(\delta_i(\Gamma))$, hence $w.\sigma_i; \epsilon \vdash \gamma_i : \delta_i(\Gamma)$.
  - By Weakening (A.1), $w.\sigma_i, \alpha; \epsilon \vdash \gamma_i : \delta_i(\Gamma)$.
  - By Substitution (A.2), $w.\sigma_i, \alpha; \epsilon \vdash \gamma_i\delta_i(e_i) : \delta_i(\tau)$.
  - Note that $\gamma_i\delta_i(e_i) = \delta_i\gamma_i(e_i)$.
  - By rule EGEN, $w.\sigma_i; \epsilon \vdash \Lambda\alpha.\delta_i\gamma_i(e_i) : \forall\alpha.\delta_i(\tau)$.
  - By Lemma 10.9, $w.\sigma_i; \epsilon \vdash \Lambda\alpha.\delta_i\gamma_i(e_i) : \forall\alpha.\rho^i(\tau)$.
  - $\epsilon \vdash \forall\alpha.\rho^i(\tau)$ follows from Lemma 10.9, assumption, Validity (A.3), Substitution (A.2), and rule TALL.
- Now, by Inclusion (10.5) it suffices to show $(k, w, \delta_1\gamma_1(\Lambda\alpha.e_1), \delta_2\gamma_2(\Lambda\alpha.e_2)) \in V_n^\iota[\![\forall\alpha.\tau]\!]\rho$.
  - So suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}^{\neg\iota}[\![\Omega]\!]w'$.
  - To show: $(k'', w'', \delta_1\gamma_1(e_1)[\tau_1/\alpha], \delta_2\gamma_2(e_2)[\tau_2/\alpha]) \in E_n^\iota[\![\tau]\!]\rho, \alpha{\mapsto}r$
  - $\vdash w''.\sigma_i; \delta_i\gamma_i(e_i)[\tau_i/\alpha] : (\rho, \alpha{\mapsto}r)^i(\tau)$ follows easily by Substitution (A.2) and Weakening (A.1) from what we just showed.
  - By assumption and Lemma 10.4, $(\delta_1, \delta_2, \lfloor\rho\rfloor_{k'}) \in D_{k'}^\iota[\![\Delta]\!]w'$.
  - Since $\neg\iota = \iota$ we have $(\tau_1, \tau_2, r) \in T_{k'}^\iota[\![\Omega]\!]w'$.
  - Let $(\delta_1', \delta_2', \rho') := ((\delta_1, \alpha{\mapsto}\tau_1), (\delta_2, \alpha{\mapsto}\tau_2), (\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r))$, so $(\delta_1', \delta_2', \rho') \in D_{k'}^\iota[\![\Delta, \alpha]\!]w'$.
  - Furthermore, $(k'', w'', \gamma_1, \gamma_2) \in G_{k'}^\iota[\![\Gamma]\!]\rho'$ by Closure Under World Extension (10.7), Restriction (10.2), and Irrelevance (10.3).
  - Now instantiate the assumption with $(\delta_1', \delta_2', \rho')$ and $(k'', w'', \gamma_1, \gamma_2)$ to get $(k'', w'', \delta_1'\gamma_1(e_1), \delta_2'\gamma_2(e_2)) \in E_{k'}^\iota[\![\tau]\!]\rho'$.
  - Note that $\delta_i'\gamma_i(e_i) = \delta_i\gamma_i(e_i)[\tau_i/\alpha]$.
  - The claim then follows by Irrelevance (10.3) and Restriction (10.2). □

The proof of compatibility for INST is one of the few where we have to distinguish between the parametric and non-parametric relation. At the beginning, we want to use the knowledge about $e_1$ and $e_2$ to derive that instantiating them with the according refinements of $\tau_2$ and its logical interpretation results in terms related by $E^\iota[\![\tau_1[\tau_2/\alpha]]\!]$. In order to do that, however, the definition of $V^\iota[\![\forall\alpha.\tau]\!]$ requires us to show that the instantiation triple actually is in $T^{\neg\iota}[\![\Omega]\!]$ (which equals $T^\iota[\![\Omega]\!]$ due to our restriction on $\iota$). In the parametric case, this is obvious because $T^\circ[\![\Omega]\!]$ does not really impose any restrictions. In the non-parametric case, though, we have to show that our triple has the special form that $T[\![\Omega]\!]$ asks for. This is were the anti-unifying substitution and its properties are being exploited. The rest of the proof then goes the usual way.

**Lemma 10.22** (Compatibility: INST)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta;\Gamma \vdash e_1 \precsim^\iota e_2 : \forall\alpha.\tau_1$ and $\Delta \vdash \tau_2$, then $\Delta;\Gamma \vdash e_1\ \tau_2 \precsim^\iota e_2\ \tau_2 : \tau_1[\tau_2/\alpha]$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(e_1\ \tau_2), \delta_2\gamma_2(e_2\ \tau_2)) \in E^\iota_n[\![\tau_1[\tau_2/\alpha]]\!]\rho$
- Assume $w.\sigma_1; \delta_1\gamma_1(e_1\ \tau_2)$ terminates in $j_1 + 1 + j_2 =: j < k$ steps:

$$\begin{array}{ll} & w.\sigma_1; \delta_1\gamma_1(e_1\ \tau_2) \\ \hookrightarrow^{j_1} & \sigma_1'; (\Lambda\alpha.e_1')\ \delta_1(\tau_2) \\ \hookrightarrow^{1} & \sigma_1'; e_1'[\delta_1(\tau_2)/\alpha] \\ \hookrightarrow^{j_2} & \sigma_1; v_1 \end{array}$$

- Instantiate the assumption to get $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota_n[\![\forall\alpha.\tau_1]\!]\rho$.
- Consequently, there is $(k - j_1, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(e_2\ \tau_2) \hookrightarrow^* w'.\sigma_2; (\Lambda\alpha.e_2')\ \delta_2(\tau_2)$ with $w'.\sigma_1 = \sigma_1'$ and $(k - j_1, w', \Lambda\alpha.e_1', \Lambda\alpha.e_2') \in V^\iota_n[\![\forall\alpha.\tau_1]\!]\rho$.
- Let $r := (w'.\sigma_1^*(\delta_1(\tau_2)), w'.\sigma_2^*(\delta_2(\tau_2)), V^\iota_{k-j_1}[\![\tau_2]\!]\rho)$.
- If $\iota = \circ$, then $(\delta_1(\tau_2), \delta_2(\tau_2), r) \in T^\circ_{k-j_1}[\![\Omega]\!]w' = T^{\neg\iota}_{k-j_1}[\![\Omega]\!]w'$ is obvious.
- If $\iota = \epsilon$, then we show $(\delta_1(\tau_2), \delta_2(\tau_2), r) \in T_{k-j_1}[\![\Omega]\!]w' = T^{\neg\iota}_{k-j_1}[\![\Omega]\!]w'$ as follows:
  - Let $\delta := \mathrm{au}(\delta_1, \delta_2, w_0.\eta)$.
  - It suffices to show $(\delta_1(\tau_2), \delta_2(\tau_2), r) = (w'.\eta^1\delta(\tau_2), w'.\eta^2\delta(\tau_2), (w'.\rho^1\delta(\tau_2), w'.\rho^2\delta(\tau_2), V_{k-j_1}[\![\delta(\tau_2)]\!]w'.\rho))$.
  - By Lemma 10.4, $(\delta_1, \delta_2, \lfloor\rho\rfloor_{k-j_1}) \in D_{k-j_1}[\![\Delta]\!]w'$.
  - First, $\delta_i(\tau_2) = w'.\eta^i\delta(\tau_2)$ by Lemma 10.13.
  - Second, $w'.\sigma_i^*(\delta_i(\tau_2)) = w'.\sigma_i^*(w'.\eta^i\delta(\tau_2)) = w'.\rho^i\delta(\tau_2)$ because of Lemma 10.13 and $w' \in \mathrm{World}$.
  - Finally, $V_{k-j_1}[\![\tau_2]\!]\rho = V_{k-j_1}[\![\delta(\tau_2)]\!]w'.\rho$ by Lemma 10.14.
- Instantiating $(k - j_1, w', \Lambda\alpha.e_1', \Lambda\alpha.e_2') \in V^\iota_n[\![\forall\alpha.\tau_1]\!]\rho$ with $(k - j_1 - 1, \lfloor w'\rfloor) \sqsupset (k - j_1, w') \sqsupseteq (k - j_1, w')$ and $(\delta_1(\tau_2), \delta_2(\tau_2), r)$ yields $(k - j_1 - 1, \lfloor w'\rfloor, e_1'[\delta_1(\tau_2)/\alpha], e_2'[\delta_2(\tau_2)])$ $\in E^\iota_n[\![\tau_1]\!]\rho, \alpha{\mapsto}r$.

- Hence there exists $(k-j, w'') \sqsupseteq (k-j_1-1, \lfloor w' \rfloor)$ such that $w'.\sigma_2; e_2'[\delta_2(\tau_2)/\alpha] \hookrightarrow^* w''.\sigma_2; v_2$ with $w''.\sigma_1 = \sigma_1$ and $(k-j, w'', v_1, v_2) \in V_n^\iota[\![\tau_1]\!]\rho, \alpha \mapsto r$.
- It remains to show $(k-j, w'', v_1, v_2) \in V_n^\iota[\![\tau_1[\tau_2/\alpha]]\!]\rho$.
- This follows by Restriction (10.2), LR-Substitution (10.11), and Lemma 10.9.

□

As existential types are dual to universal types, it is not surprising that the proof for PACK requires the same case distinction as the previous one.

**Lemma 10.23** (Compatibility: PACK)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau[\tau'/\alpha]$ and $\Delta \vdash \tau'$, then $\Delta; \Gamma \vdash \mathsf{pack}\ \langle \tau', e_1 \rangle\ \mathsf{as}\ \exists\alpha.\tau \precsim^\iota \mathsf{pack}\ \langle \tau', e_2 \rangle\ \mathsf{as}\ \exists\alpha.\tau : \exists\alpha.\tau$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\mathsf{pack}\ \langle \tau', e_1 \rangle\ \mathsf{as}\ \exists\alpha.\tau), \delta_2\gamma_2(\mathsf{pack}\ \langle \tau', e_2 \rangle\ \mathsf{as}\ \exists\alpha.\tau)) \in E_n^\iota[\![\exists\alpha.\tau]\!]\rho$
- Assume $w.\sigma_1; \delta_1\gamma_1(\mathsf{pack}\ \langle \tau', e_1 \rangle\ \mathsf{as}\ \exists\alpha.\tau) \hookrightarrow^j \sigma_1; \mathsf{pack}\ \langle \delta_1(\tau'), v_1 \rangle\ \mathsf{as}\ \delta_1(\exists\alpha.\tau)$ where $j < k$.
- Instantiating the assumption yields $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E_n^\iota[\![\tau[\tau'/\alpha]]\!]\rho$.
- Consequently, there exists $(k-j, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(\mathsf{pack}\ \langle \tau', e_2 \rangle\ \mathsf{as}\ \exists\alpha.\tau) \hookrightarrow^* w'.\sigma_2; \mathsf{pack}\ \langle \delta_2(\tau'), v_2 \rangle\ \mathsf{as}\ \delta_2(\exists\alpha.\tau)$ with $w'.\sigma_1 = \sigma_1$ and $(k-j, w', v_1, v_2) \in V_n^\iota[\![\tau[\tau'/\alpha]]\!]\rho$.
- It remains to show $(k-j, w', \mathsf{pack}\ \langle \delta_1(\tau'), v_1 \rangle\ \mathsf{as}\ \delta_1(\exists\alpha.\tau), \mathsf{pack}\ \langle \delta_2(\tau'), v_2 \rangle\ \mathsf{as}\ \delta_2(\exists\alpha.\tau)) \in V_n^\iota[\![\exists\alpha.\tau]\!]\rho$.
- Let $r := (w'.\sigma_1^*(\delta_1(\tau')), w'.\sigma_2^*(\delta_2(\tau')), V_{k-j}^\circ[\![\tau']\!]\rho)$.
- If $\iota = \circ$, then $(\delta_1(\tau'), \delta_2(\tau'), r) \in T_{k-j}^\iota[\![\Omega]\!]w'.\rho$ is obvious.
- If $\iota = \epsilon$, then we show $(\delta_1(\tau'), \delta_2(\tau'), r) \in T_{k-j}^\iota[\![\Omega]\!]w'.\rho$ as follows:
  - Let $\delta := \mathrm{au}(\delta_1, \delta_2, w_0.\eta)$.
  - It suffices to show $(\delta_1(\tau'), \delta_2(\tau'), r) = (w'.\eta^1\delta(\tau'), w'.\eta^2(\tau'), (w'.\rho^1\delta(\tau'), w'.\rho^2\delta(\tau'), V_{k-j}[\![\delta(\tau')]\!]w'.\rho))$.
  - By Lemma 10.4, $(\delta_1, \delta_2, \lfloor\rho\rfloor_{k-j}) \in D_{k-j}[\![\Delta]\!]w'$.
  - First, $\delta_i(\tau') = w'.\eta^i\delta(\tau')$ by Lemma 10.13.
  - Second, $w'.\sigma_i^*(\delta_i(\tau')) = w'.\sigma_i^*(w'.\eta^i\delta(\tau')) = w'.\rho^i\delta(\tau')$ because of Lemma 10.13 and $w' \in \mathrm{World}$.
  - Finally, $V_{k-j}[\![\tau']\!]\rho = V_{k-j}[\![\delta(\tau')]\!]w'.\rho$ by Lemma 10.14.
- We claim that $(k'', w'', v_1, v_2) \in V_n^\iota[\![\tau]\!]\rho, \alpha \mapsto r$ for any $(k'', w'') \sqsupset (k-j, w')$.
- By Closure Under World Extension (10.7) we have $(k'', w'', v_1, v_2) \in V_n^\iota[\![\tau[\tau'/\alpha]]\!]\rho$.
- The claim then follows by Restriction (10.2), LR-Substitution (10.11), and Lemma 10.9.

□

**Lemma 10.24** (Compatibility: UNPACK)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta;\Gamma \vdash e_1 \precsim^\iota e_2 : \exists\alpha.\tau'$ and $\Delta,\alpha;\Gamma,x{:}\tau' \vdash e_3 \precsim^\iota e_4 : \tau$ with $\Delta \vdash \tau$, then $\Delta;\Gamma \vdash \mathsf{unpack}\langle\alpha,x\rangle{=}e_1 \mathsf{\ in\ } e_3 \precsim^\iota \mathsf{unpack}\langle\alpha,x\rangle{=}e_2 \mathsf{\ in\ } e_4 : \tau$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1,\delta_2,\rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k,w,\gamma_1,\gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k,w) \sqsupset (n,w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\mathsf{unpack}\ \langle\alpha,x\rangle{=}e_1 \mathsf{\ in\ } e_3), \delta_2\gamma_2(\mathsf{unpack}\ \langle\alpha,x\rangle{=}e_2 \mathsf{\ in\ } e_4)) \mathsf{\ in\ } E^\iota_n[\![\tau]\!]\rho$
- Assume that $w.\sigma_1;\delta_1\gamma_1(\mathsf{unpack}\ \langle\alpha,x\rangle{=}e_1 \mathsf{\ in\ } e_3)$ terminates in $j_1+1+j_2 =: j < k$ steps:

$$\begin{array}{ll} & w.\sigma_1;\delta_1\gamma_1(\mathsf{unpack}\ \langle\alpha,x\rangle{=}e_1 \mathsf{\ in\ } e_3) \\ \hookrightarrow^{j_1} & \sigma_1';\mathsf{unpack}\ \langle\alpha,x\rangle{=}(\mathsf{pack}\ \langle\tau_1,v_1\rangle \mathsf{\ as\ } \tau_1') \mathsf{\ in\ } \delta_1\gamma_1(e_3) \\ \hookrightarrow^{1} & \sigma_1';\delta_1\gamma_1(e_3)[\tau_1/\alpha][v_1/x] \\ \hookrightarrow^{j_2} & \sigma_1;v_3 \end{array}$$

- Instantiating the first assumption yields the existence of $(k-j_1,w') \sqsupseteq (k,w)$ such that $w.\sigma_2;\delta_2\gamma_2(\mathsf{unpack}\ \langle\alpha,x\rangle{=}e_2 \mathsf{\ in\ } e_4) \hookrightarrow^* w'.\sigma_2;\mathsf{unpack}\ \langle\alpha,x\rangle{=}(\mathsf{pack}\ \langle\tau_2,v_2\rangle \mathsf{\ as\ } \tau_2') \mathsf{\ in\ } \delta_2\gamma_2(e_4)$ with
  $w'.\sigma_1 = \sigma_1'$ and $(k-j_1, w', \mathsf{pack}\ \langle\tau_1,v_1\rangle \mathsf{\ as\ } \tau_1', \mathsf{pack}\ \langle\tau_2,v_2\rangle \mathsf{\ as\ } \tau_2') \in V^\iota_n[\![\exists\alpha.\tau']\!]\rho$.
- Hence there is $r$ such that $(\tau_1,\tau_2,r) \in T^\iota_{k-j_1}[\![\Omega]\!]w'$ and
  $(k-j_1-1, \lfloor w'\rfloor, v_1, v_2) \in V^\iota_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- By Lemma 10.4, $(\delta_1,\delta_2,\lfloor\rho\rfloor_{k-j_1}) \in D^\iota_{k-j_1}[\![\Delta]\!]w'$.
- Let $(\delta_1',\delta_2',\rho') := ((\delta_1,\alpha{\mapsto}\tau_1),((\delta_2,\alpha{\mapsto}\tau_2),(\lfloor\rho\rfloor_{k-j_1},\alpha{\mapsto}r)))$, so $(\delta_1',\delta_2',\rho') \in D^\iota_{k-j_1}[\![\Delta,\alpha]\!]w'$.
- By Closure Under World Extension (10.7) we know $(k-j_1-1,\lfloor w'\rfloor,\gamma_1,\gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$.
- By Restriction (10.2) and Irrelevance (10.3), $(k-j_1-1,\lfloor w'\rfloor,\gamma_1,\gamma_2) \in G^\iota_{k-j_1}[\![\Gamma]\!]\rho'$.
- Let $\gamma_i' := \gamma_i, x{\mapsto}v_i$, so by Restriction (10.2) and Irrelevance (10.3) we get
  $(k-j_1-1,\lfloor w'\rfloor,\gamma_1',\gamma_2') \in G^\iota_{k-j_1}[\![\Gamma,x{:}\tau'']\!]\rho'$.
- Now, instantiating the second assumption with $w' \in \mathrm{World}_{k-j_1}$, $(\delta_1',\delta_2',\rho') \in D^\iota_{k-j_1}[\![\Delta,\alpha]\!]w'$ and $(k-j_1-1,\lfloor w'\rfloor,\gamma_1',\gamma_2') \in G^\iota_{k-j_1}[\![\Gamma,x{:}\tau'']\!]\rho'$ yields
  $(k-j_1-1,\lfloor w'\rfloor,\delta_1'\gamma_1'(e_3),\delta_2'\gamma_2'(e_4)) \in E^\iota_{k-j_1}[\![\tau]\!]\rho'$.
- Note that

$$\begin{array}{ll} & \delta_i'\gamma_i'(e_{i+2}) \\ & = \delta_i(\gamma_i(e_{i+2})[v_i/x])[\tau_i/\alpha]) \\ & = \delta_i\gamma_i(e_{i+2})[v_i/x][\tau_i/\alpha]) & \text{since } \vdash w'.\sigma_i;v_i : (\rho,\alpha{\mapsto}V^\iota_{k-j_1}[\![\tau'']\!]w'.\rho)^i(\tau') \\ & = \delta_i\gamma_i(e_{i+2})[\tau_i/\alpha][v_i/x] & \text{since } \vdash w'.\sigma_i;v_i : (\rho,\alpha{\mapsto}V^\iota_{k-j_1}[\![\tau'']\!]w'.\rho)^i(\tau') \end{array}$$

- Therefore, $\sigma_1';\delta_1\gamma_1(e_3)[\tau_1/\alpha][v_1/x] \hookrightarrow^{j_2} \sigma_1;v_3$ implies the existence of $(k-j,w'') \sqsupseteq (k-j_1-1,\lfloor w'\rfloor)$ such that $w'.\sigma_2;\delta_2\gamma_2(e_4)[\tau_2/\alpha][v_2/x] \hookrightarrow^* w''\sigma_2;v_4$ with $w''.\sigma_1 = \sigma_1$ and $(k-j,w'',v_3,v_4) \in V^\iota_{k-j_1}[\![\tau]\!]\rho'$.
- By Restriction (10.2) and, since $\Delta \vdash \tau$, by Irrelevance (10.3), $(k-j,w'',v_3,v_4) \in V^\iota_n[\![\tau]\!]\rho$. □

**Lemma 10.25** (Compatibility: ROLL)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau[\mu\alpha.\tau/\alpha]$, then $\Delta; \Gamma \vdash \mathsf{roll}\ e_1\ \mathsf{as}\ \mu\alpha.\tau \precsim^\iota \mathsf{roll}\ e_2\ \mathsf{as}\ \mu\alpha.\tau : \mu\alpha.\tau$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\mathsf{roll}\ e_1\ \mathsf{as}\ \mu\alpha.\tau), \delta_2\gamma_2(\mathsf{roll}\ e_2\ \mathsf{as}\ \mu\alpha.\tau))$ in $E^\iota_n[\![\mu\alpha.\tau]\!]\rho$
- Assume that $w.\sigma_1; \delta_1\gamma_1(\mathsf{roll}\ e_1\ \mathsf{as}\ \mu\alpha.\tau)$ terminates in $j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1\gamma_1(\mathsf{roll}\ e_1\ \mathsf{as}\ \mu\alpha.\tau) \\ \hookrightarrow^j & \sigma_1; \mathsf{roll}\ v_1\ \mathsf{as}\ \delta_1(\mu\alpha.\tau) \end{array}$$

- Instantiating the assumption yields $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota_n[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho$.
- Consequently, there exists $(k - j, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(\mathsf{roll}\ e_2\ \mathsf{as}\ \mu\alpha.\tau) \hookrightarrow^* w'.\sigma_2; \mathsf{roll}\ v_2\ \mathsf{as}\ \delta_2(\mu\alpha.\tau)$ with $w'.\sigma_1 = \sigma_1$ and $(k - j, w', v_1, v_2) \in V^\iota_n[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho$.
- It remains to show $(k - j, w', \mathsf{roll}\ v_1\ \mathsf{as}\ \delta_1(\mu\alpha.\tau), \mathsf{roll}\ v_2\ \mathsf{as}\ \delta_2(\mu\alpha.\tau)) \in V^\iota_n[\![\mu\alpha.\tau]\!]\rho$.
- This follows from $(k - j, w', v_1, v_2) \in V^\iota_n[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho$ by Closure Under World Expansion (10.7) and Restriction (10.2).

□

**Lemma 10.26** (Compatibility: UNROLL)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \mu\alpha.\tau$, then $\Delta; \Gamma \vdash \mathsf{unroll}\ e_1 \precsim^\iota \mathsf{unroll}\ e_2 : \tau[\mu\alpha.\tau/\alpha]$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^\iota_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G^\iota_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\mathsf{unroll}\ e_1), \delta_2\gamma_2(\mathsf{unroll}\ e_2))$ in $E^\iota_n[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho$
- Assume that $w.\sigma_1; \delta_1\gamma_1(\mathsf{unroll}\ e_1)$ terminates in $j' + 1 =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1\gamma_1(\mathsf{unroll}\ e_1) \\ \hookrightarrow^{j'} & \sigma_1; \mathsf{unroll}\,(\mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1) \\ \hookrightarrow^1 & \sigma_1; v_1 \end{array}$$

- Instantiating the assumption yields $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota_n[\![\mu\alpha.\tau]\!]\rho$.
- Consequently, there exists $(k - j', w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2\gamma_2(\mathsf{unroll}\ e_2) \hookrightarrow^* w'.\sigma_2; \mathsf{unroll}\,(\mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2)$ with $w'.\sigma_1 = \sigma_1$ and $(k - j', w', \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V^\iota_n[\![\mu\alpha.\tau]\!]\rho$.
- It remains to show $(k - j, \lfloor w' \rfloor, v_1, v_2) \in V^\iota_n[\![\tau[\mu\alpha.\tau/\alpha]]\!]\rho$.
- This follows from $(k - j', w', \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V^\iota_n[\![\mu\alpha.\tau]\!]\rho$ by Restriction (10.2).

□

The remaining compatibility properties are those for CAST, NEW, and CONV—rules, which are special to our language and have no counter-part in $F^\mu$.

As mentioned before, compatibility with cast only holds for the non-parametric relation. In the proof, we first have to argue that the argument types on the left-hand side, $\delta_1(\tau_1)$ and $\delta_1(\tau_2)$, are equal if and only if the argument types on the right-hand side, $\delta_2(\tau_1)$ and $\delta_2(\tau_2)$, are, so that we have the same reduction on both sides. This is easy to see with the help of Lemma 10.13, which tells us that $\delta_i = w_0.\eta^i \cdot \delta$ (where $\delta$ is the anti-unifying substitution of $\delta_1$ and $\delta_2$)—meaning that $\delta_1$ and $\delta_2$ map to types that are syntactically identical up to some bijection on type names; Recall that we consider $\mathrm{dom}(w_0.\eta)$ to contain bound variables and thus can assume it to be disjoint from $\mathrm{rng}(w_0.\eta^i)$ without loss of generality. We then have to distinguish two cases. If the type arguments are not equal (the cast fails), there is not much to do, as expected. If the cast succeeds, however, we basically need to show that the argument types are also *semantically* equal, *i.e.,* $V_n[\![\tau_1]\!]\rho = V_n[\![\tau_2]\!]\rho$. Since $\delta(\tau_1) = \delta(\tau_2)$, this follows from Lemma 10.14.

**Lemma 10.27** (Compatibility: CAST)**.** If $\Delta \vdash \Gamma$ and $\Delta \vdash \tau_1$ and $\Delta \vdash \tau_2$, then $\Delta; \Gamma \vdash \mathsf{cast}\ \tau_1\ \tau_2 \precsim \mathsf{cast}\ \tau_1\ \tau_2 : \tau_1 \to \tau_2 \to \tau_2$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \mathsf{cast}\ \delta_1(\tau_1)\ \delta_1(\tau_2), \mathsf{cast}\ \delta_2(\tau_1)\ \delta_2(\tau_2)) \in E_n[\![\tau_1 \to \tau_2 \to \tau_2]\!]\rho$.
- Let $\delta := \mathrm{au}(\delta_1, \delta_2, w_0.\eta)$.
- Then $\delta(\tau_1) = w_0.\eta^{-i}\delta_i(\tau_1) = w_0.\eta^{-i}\delta_i(\tau_2) = \delta(\tau_2)$ by Lemma 10.13.
- Consequently, $\delta_1(\tau_1) = \delta_1(\tau_2)$ iff $\delta_2(\tau_1) = \delta_2(\tau_2)$.
- Case $\delta_i(\tau_1) = \delta_i(\tau_2)$:
  - Then we have the following reductions:
    $w.\sigma_i; \mathsf{cast}\ \delta_i(\tau_1)\ \delta_i(\tau_2) \hookrightarrow^1 w.\sigma_i; \lambda x_1{:}\delta_i(\tau_1).\lambda x_2{:}\delta_i(\tau_2).x_1$
  - Hence it suffices to show
    $(k-1, \lfloor w \rfloor, \lambda x_1{:}\delta_1(\tau_1).\lambda x_2{:}\delta_1(\tau_2).x_1, \lambda x_1{:}\delta_2(\tau_1).\lambda x_2{:}\delta_2(\tau_2).x_1) \in V_n[\![\tau_1 \to \tau_2 \to \tau_2]\!]\rho$.
  - So suppose $(k', w') \sqsupseteq (k-1, \lfloor w \rfloor)$ and $(k', w', v_1, v_2) \in V_n[\![\tau_1]\!]\rho$.
  - To show: $(k', w', \lambda x_2{:}\delta_1(\tau_2).v_1, \lambda x_2{:}\delta_2(\tau_2).v_2) \in E_n[\![\tau_2 \to \tau_2]\!]\rho$.
  - By Inclusion (10.5) it suffices to show
    $(k', w', \lambda x_2{:}\delta_1(\tau_2).v_1, \lambda x_2{:}\delta_2(\tau_2).v_2) \in V_n[\![\tau_2 \to \tau_2]\!]\rho$.
  - So suppose $(k'', w'') \sqsupseteq (k', w')$ and $(k'', w'', v_1', v_2') \in V_n[\![\tau_2]\!]\rho$.
  - To show: $(k'', w'', v_1, v_2) \in E_n[\![\tau_2]\!]$
  - By Inclusion (10.5) it suffices to show $(k'', w'', v_1, v_2) \in V_n[\![\tau_2]\!]\rho$.
  - By Closure Under World Extension (10.7), $(k'', w'', v_1, v_2) \in V_n[\![\tau_1]\!]\rho$.
  - The claim then follows by Lemma 10.14.
- Case $\delta_i(\tau_1) \neq \delta_i(\tau_2)$:
  - Then we have the following reductions:
    $w.\sigma_i; \mathsf{cast}\ \delta_i(\tau_1)\ \delta_i(\tau_2) \hookrightarrow^1 w.\sigma_i; \lambda x_1{:}\delta_i(\tau_1).\lambda x_2{:}\delta_i(\tau_2).x_2$

- Hence it suffices to show
  $(k-1, \lfloor w \rfloor, \lambda x_1{:}\delta_1(\tau_1).\lambda x_2{:}\delta_2(\tau_1).x_2, \lambda x_1{:}\delta_1(\tau_1).\lambda x_2{:}\delta_2(\tau_2).x_2) \in V_n[\![\tau_1 \to \tau_2 \to \tau_2]\!]\rho$.
- So suppose $(k', w') \sqsupseteq (k-1, \lfloor w \rfloor)$ and $(k', w', v_1, v_2) \in V_n[\![\tau_1]\!]\rho$.
- To show: $(k', w', \lambda x_2{:}\delta_2(\tau_1).x_2, \lambda x_2{:}\delta_2(\tau_2).x_2) \in E_n[\![\tau_2 \to \tau_2]\!]\rho$.
- By Inclusion (10.5) it suffices to show
  $(k', w', \lambda x_2{:}\delta_2(\tau_1).x_2, \lambda x_2{:}\delta_2(\tau_2).x_2) \in V_n[\![\tau_2 \to \tau_2]\!]\rho$.
- So suppose $(k'', w'') \sqsupseteq (k', w')$ and $(k'', w'', v_1', v_2') \in V_n[\![\tau_2]\!]\rho$.
- By Inclusion (10.5) it suffices to show $(k'', w'', v_1', v_2') \in V_n[\![\tau_2]\!]\rho$, which is given.

□

Since new is the only construct that modifies the type store, its compatibility proof is also the only one where we actually have to manipulate the world. In particular, we extend the $\eta$ and $\rho$ components of some initial world $w_0$ with bindings for the fresh dynamically-generated type name $\alpha$. The $\eta$ is extended with $\alpha \mapsto (\alpha_1, \alpha_2)$, where $\alpha_1$ and $\alpha_2$ are the concrete fresh names that are chosen when evaluating the left and right new expressions. The $\rho$ is extended so that the relational interpretation of $\alpha$ is simply the logical relation at type $\tau'$. The proof of this lemma is highly reminiscent of the proof of compatibility for reference allocation in a language with mutable references [6].

**Lemma 10.28** (Compatibility: NEW)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta, \alpha{\approx}\tau'; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau$ and $\Delta \vdash \tau$ and $\Delta \vdash \Gamma$, then $\Delta; \Gamma \vdash \mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e_1 \precsim^\iota \mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e_2 : \tau$.

*Proof:*

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$ and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(\mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e_1), \delta_2\gamma_2(\mathsf{new}\,\alpha{\approx}\tau'\ \mathsf{in}\ e_2)) \in E_n^\iota[\![\tau]\!]\rho$.
- Assume $w.\sigma_1; \delta_1\gamma_1(\mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e_1)$ terminates in $1 + j' =: j < k$ steps:

$$\begin{array}{ll} & w.\sigma_1; \delta_1\gamma_1(\mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e_1) \\ \hookrightarrow^1 & w.\sigma_1, \alpha_1{\approx}\delta_1(\tau'); \delta_1\gamma_1(e_1)[\alpha_1/\alpha] \\ \hookrightarrow^{j'} & \sigma_1; v_1 \end{array}$$

- Note that $w.\sigma_2; \delta_2\gamma_2(\mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e_2) \hookrightarrow^1 w.\sigma_2, \alpha_2{\approx}\delta_2(\tau'); \delta_2\gamma_2(e_2)[\alpha_2/\alpha]$.
- Let $w_\alpha := w \uplus (\alpha_1{\approx}\delta_1(\tau'), \alpha_2{\approx}\delta_2(\tau'), \alpha{\mapsto}(\alpha_1, \alpha_2), \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V_k^\iota[\![\tau']\!]\lfloor\rho\rfloor_k))$, so $(k, w_\alpha) \sqsupseteq (k, w)$.
- By Lemma 10.4, $(\delta_1, \delta_2, \lfloor\rho\rfloor_k) \in D_k^\iota[\![\Delta]\!]w_\alpha$.
- Let $(\delta_1', \delta_2', \rho') := ((\delta_1, \alpha{\mapsto}\alpha_1), (\delta_2, \alpha{\mapsto}\alpha_2), (\lfloor\rho\rfloor_k, \alpha{\mapsto}(\rho^1(\tau'), \rho^2(\tau'), V_k^\iota[\![\tau']\!]\lfloor\rho\rfloor_k)))$.
- Note that $w_\alpha.\sigma_i(\alpha_i) = \delta_i(\tau')$, $\alpha_i = w_\alpha.\eta^i(\alpha)$, and $w_\alpha.\rho(\alpha).R = V_k^\iota[\![\tau']\!]\lfloor\rho\rfloor_k$.
- Therefore, $(\delta_1', \delta_2', \rho') \in D_k^\iota[\![\Delta, \alpha{\approx}\tau']\!]w_\alpha$.
- By Closure Under World Extension (10.7) we know $(k-1, \lfloor w_\alpha \rfloor, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$.
- By Restriction (10.2) and Irrelevance (10.3), $(k-1, \lfloor w_\alpha \rfloor, \gamma_1, \gamma_2) \in G_k^\iota[\![\Gamma]\!]\rho'$.
- Now instantiate the assumption with $w_\alpha \in \mathrm{World}_k$, $(\delta_1', \delta_2', \rho') \in D_k^\iota[\![\Delta, \alpha{\approx}\tau']\!]w_\alpha$, $(k-1, \lfloor w_\alpha \rfloor, \gamma_1, \gamma_2) \in G_k^\iota[\![\Gamma]\!]\rho'$ and $(k-1, \lfloor w_\alpha \rfloor) \sqsupset (k, w_\alpha)$ to get $(k-1, \lfloor w \rfloor, \delta_1'\gamma_1(e_1), \delta_2'\gamma_2(e_2)) \in E_k^\iota[\![\tau]\!]\rho'$.

- Note that $\delta_i'\gamma_i(e_i) = \delta_i\gamma_i(e_i)[\alpha_i/\alpha]$.
- Consequently, there exists $(k-j, w'') \sqsupseteq (k-1, w_\alpha)$ such that $w.\sigma_2, \alpha_2 \approx \delta_2(\tau'); \delta_2\gamma_2(e_2) \hookrightarrow^* w''.\sigma_2; v_2$ with $w''.\sigma_1 = \sigma_1$ and $(k-j, w'', v_1, v_2) \in V_k^\iota[\![\tau]\!]\rho'$.
- Because of $\Delta \vdash \tau$, Irrelevance (10.3) and Restriction (10.2) yield $(k-j, w'', v_1, v_2) \in V_n^\iota[\![\tau]\!]\rho$.

□

Compatibility for CONV follows from the fact that compatible types are semantically equal, which we prove separately below. The interesting case is when $\tau_1$ is a variable $\alpha$ bound in $\Delta$ as $\alpha \approx \tau_2$, and the result in this case follows easily from the definition of $D^\iota[\![\Delta, \alpha \approx \tau]\!]w$. This case also makes it obvious that the property does not hold for the polarized relation.

**Lemma 10.29** (Type Compatibility)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta \vdash \tau_1 \approx \tau_2$ and $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w$, then

1. $V_n^\iota[\![\tau_1]\!]\rho = V_n^\iota[\![\tau_2]\!]\rho$ and
2. $E_n^\iota[\![\tau_1]\!]\rho = E_n^\iota[\![\tau_2]\!]\rho$.

*Proof:* By primary induction on $n$ and secondary induction on the derivation of $\Delta \vdash \tau_1 \approx \tau_2$.

1. - Case $\tau_1 = \alpha = \tau_2$: Trivial.
   - Case $\tau_1 = \alpha$ where $\alpha \approx \tau_2 \in \Delta$:
     - Then it is easy to see that $\Delta = \Delta_1, \alpha \approx \tau_2, \Delta_2$, $\delta_i = \delta_{i1}, \alpha \mapsto \alpha_i, \delta_{i2}$, and $\rho = \rho_1, \alpha \mapsto (\rho_1^1(\tau_2), \rho_1^2(\tau_2), V_n^\iota[\![\tau_2]\!]\rho_1), \rho_2$.
     - Hence $V_n^\iota[\![\alpha]\!]\rho = V_n^\iota[\![\tau_2]\!]\rho_1$ and so the claim follows by Irrelevance (10.3).
   - Case $\tau_1 = b = \tau_2$: Trivial.
   - Case $\tau_1 = \tau_{11} \times \tau_{12}$ and $\tau_2 = \tau_{21} \times \tau_{22}$ with $\Delta \vdash \tau_{11} \approx \tau_{21}$ and $\Delta \vdash \tau_{21} \approx \tau_{22}$: Follows easily by induction.
   - Case $\tau_1 = \tau_{11} \to \tau_{12}$ and $\tau_2 = \tau_{21} \to \tau_{22}$ with $\Delta \vdash \tau_{11} \approx \tau_{21}$ and $\Delta \vdash \tau_{12} \approx \tau_{22}$: Follows easily by induction.
   - Case $\tau_1 = \forall\alpha.\tau_1'$ and $\tau_2 = \forall\alpha.\tau_2'$ with $\Delta, \alpha \vdash \tau_1' \approx \tau_2'$:
     - We show $V_n^\iota[\![\tau_1]\!]\rho \subseteq V_n^\iota[\![\tau_2]\!]\rho$; the other direction is symmetric.
     - Suppose $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^\iota[\![\forall\alpha.\tau_1']\!]\rho$.
     - Suppose further $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1'', \tau_2'', r) \in T_{k'}^{\neg\iota}[\![\Omega]\!]w'$.
     - To show: $(k'', w'', e_1[\tau_1''/\alpha], e_2[\tau_2''/\alpha]) \in E_n^\iota[\![\tau_2']\!]\rho, \alpha \mapsto r$.
     - Since $\neg\iota = \iota$ and thus $T_{k'}^{\neg\iota}[\![\Omega]\!]w' = T_{k'}^\iota[\![\Omega]\!]w'$, we have $(k'', w'', e_1[\tau_1''/\alpha], e_2[\tau_2''/\alpha]) \in E_n^\iota[\![\tau_1']\!]\rho, \alpha \mapsto r$.
     - By Restriction (10.2) and Irrelevance (10.3), this reduces to showing $E_{k'}^\iota[\![\tau_1']\!]\lfloor\rho\rfloor_{k'}, \alpha \mapsto r = E_{k'}^\iota[\![\tau_2']\!]\lfloor\rho\rfloor_{k'}, \alpha \mapsto r$.
     - This follows by induction if we can show $((\delta_1, \alpha \mapsto \tau_1''), (\delta_2, \alpha \mapsto \tau_2''), (\lfloor\rho\rfloor_{k'}, \alpha \mapsto r)) \in D_{k'}^\iota[\![\Delta, \alpha]\!]w'$.
     - By Lemma 10.4, $(\delta_1, \delta_2, \lfloor\rho\rfloor_{k'}) \in D_{k'}^\iota[\![\Delta]\!]w'$.
     - Hence indeed $((\delta_1, \alpha \mapsto \tau_1''), (\delta_2, \alpha \mapsto \tau_2''), (\lfloor\rho\rfloor_{k'}, \alpha \mapsto r)) \in D_{k'}^\iota[\![\Delta, \alpha]\!]w'$.
   - Case $\tau_1 = \exists\alpha.\tau_1'$ and $\tau_2 = \exists\alpha.\tau_2'$ with $\Delta, \alpha \vdash \tau_1' \approx \tau_2'$: analogously to the previous case
   - Case $\tau_1 = \mu\alpha.\tau_1'$ and $\tau_2 = \mu\alpha.\tau_2'$ with $\Delta, \alpha \vdash \tau_1' \approx \tau_2'$:
     - We show $V_n^\iota[\![\tau_1]\!]\rho \subseteq V_n^\iota[\![\tau_2]\!]\rho$; the other direction is symmetric.

- Suppose $(k, w, \mathsf{roll}\ e_1\ \mathsf{as}\ \tau_1'', \mathsf{roll}\ e_2\ \mathsf{as}\ \tau_2'') \in V_n^\iota[\![\mu\alpha.\tau_1']\!]\rho$.
- Suppose further $(k', w') \sqsupset (k, w)$.
- To show: $(k', w', v_1, v_2) \in V_k^\iota[\![\tau_2'[\tau_2/\alpha]]\!]\rho$
- We know: $(k', w', v_1, v_2) \in V_k^\iota[\![\tau_1'[\tau_1/\alpha]]\!]\rho$
- The claim then follows by Substitution (A.2.2) and induction.

2. Follows immediately from part (1).

□

**Lemma 10.30** (Compatibility: CONV)**.** For $\iota \in \{\epsilon, \circ\}$: If $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau'$ and $\Delta \vdash \tau \approx \tau'$, then $\Delta; \Gamma \vdash e_1 \precsim^\iota e_2 : \tau$.

*Proof:* Follows from Type Compatibility (10.29). □

## 10.5 Fundamental Properties

Based on all these compatibility lemmas we can show the fundamental properties. Note the restriction in the one for the parametric relation.

**Lemma 10.31** (Fundamental Property for $\precsim$)**.** If $\Delta; \Gamma \vdash e : \tau$, then $\Delta; \Gamma \vdash e \precsim e : \tau$.

*Proof:* By induction on the derivation, in each case using the appropriate compatibility lemma. □

**Lemma 10.32** (Fundamental Property for $\precsim^\circ$)**.** If $\Delta; \Gamma \vdash e : \tau$ and $e$ is $\mathsf{cast}$-free, then $\Delta; \Gamma \vdash e \precsim^\circ e : \tau$.

*Proof:* By induction on the derivation, in each case using the appropriate compatibility lemma. □

**Lemma 10.33** (Fundamental Property for $\precsim^+$)**.** If $\Delta; \Gamma \vdash e : \tau$, then $\Delta; \Gamma \vdash e \precsim^+ e : \tau$.

*Proof:* As already mentioned in Section 8, this can only be shown indirectly, by going through the non-parametric relation:

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^-[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^-[\![\Gamma]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(e), \delta_2\gamma_2(e)) \in E_n^+[\![\tau]\!]\rho$.
- By Fundamental Property (10.31), $\Delta; \Gamma \vdash e \precsim e : \tau$.
- Since $D_n^-[\![\Delta]\!]w_0 = D_n[\![\Delta]\!]w_0$ and $G_n^-[\![\Gamma]\!]\rho \subseteq G_n[\![\Gamma]\!]\rho$ by Chain (10.38), we can instantiate this to get $(k, w, \delta_1\gamma_1(e), \delta_2\gamma_2(e)) \in E_n[\![\tau]\!]\rho$.
- Another application of Chain (10.38) yields $(k, w, \delta_1\gamma_1(e), \delta_2\gamma_2(e)) \in E_n^+[\![\tau]\!]\rho$.

□

## 10.6 Soundness of the Non-Parametric Relation

The full compatibility and the fundamental property of $\precsim$ provide the core of its soundness proof. There is only one step missing, namely to finally establish that $\precsim$ is a precongruence with respect to the constructs of the language. This is in fact the only property the actual soundness proof relies on, but showing it requires one more lemma:

**Lemma 10.34** (LR-Weakening)**.** If $\Delta;\Gamma \vdash e_1 \precsim^\iota e_2 : \tau$, $\Delta' \supseteq \Delta$, $\Gamma' \supseteq \Gamma$, and $\Delta' \vdash \Gamma$, then $\Delta';\Gamma' \vdash e_1 \precsim^\iota e_2 : \tau$.

*Proof:*

- By assumption and Weakening (A.1) we know $\Delta';\Gamma' \vdash e_i : \tau$.
- So suppose $w_0 \in \mathrm{World}_n$, $(\delta_1', \delta_2', \rho') \in D_n^\iota[\![\Delta']\!]w_0$, $(k, w, \gamma_1', \gamma_2') \in G_n^\iota[\![\Gamma']\!]\rho'$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1'\gamma_1'(e_1), \delta_2'\gamma_2'(e_2)) \in E_n^\iota[\![\tau]\!]\rho'$
- Let $\delta_i$, $\rho$, $\gamma_i$ be the restriction of $\delta_i'$, $\rho'$, $\gamma_i'$ to $\mathrm{dom}(\Delta)$, $\mathrm{dom}(\Delta)$, $\mathrm{dom}(\Gamma)$ respectively.
- It is easy to see that then $(\delta_1, \delta_2, \rho) \in D_n^\iota[\![\Delta]\!]w_0$ and, with the help of Irrelevance (10.3), $(k, w, \gamma_1, \gamma_2) \in G_n^\iota[\![\Gamma]\!]\rho$.
- Hence by assumption, $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E^\iota[\![\tau]\!]\rho$.
- Note that $\delta_i'\gamma_i'(e_i) = \delta_i\gamma_i(e_i)$.
- The claim then follows by Irrelevance (10.3).

□

**Lemma 10.35** (Precongruence of $\precsim$)**.** If $\Delta;\Gamma \vdash e_1 : \tau$ and $\Delta;\Gamma \vdash e_2 : \tau$ and $\Delta;\Gamma \vdash e_1 \precsim e_2 : \tau$ and $\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau')$, then $\Delta';\Gamma' \vdash C[e_1] \precsim C[e_2] : \tau'$.

*Proof:* By induction on the derivation of the context typing, in each case using the appropriate compatibility lemma. For a context containing another term we also need the Fundamental Property; for $C = [\_]$ we need LR-Weakening (10.34). □

**Theorem 10.36** (Soundness of $\precsim$ wrt. $\preceq$)**.** If $\Delta;\Gamma \vdash e_1 : \tau$ and $\Delta;\Gamma \vdash e_2 : \tau$ and $\Delta;\Gamma \vdash e_1 \precsim e_2 : \tau$, then $\Delta;\Gamma \vdash e_1 \preceq e_2 : \tau$.

*Proof:*

- Suppose $\vdash \sigma$ and $\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\sigma;\emptyset;\tau')$ and $\sigma; C[e_1]\downarrow$, *i.e.,* $\sigma; C[e_1] \hookrightarrow^j \sigma_1; v_1$.
- To show: $\sigma; C[e_2]\downarrow$
- By Precongruence (10.35) we have $\sigma;\emptyset \vdash C[e_1] \precsim C[e_2] : \tau'$.
- To instantiate this, we first need to create an initial world representing $\sigma$. Say $\sigma = \alpha_1 \approx \tau_1, \ldots, \alpha_n \approx \tau_n$.

- Let

$$\begin{aligned}
\sigma_0 &:= \epsilon \\
\sigma_{i+1} &:= \sigma_i, \alpha_{i+1} \approx \tau_{i+1} \\
\delta_0 &:= \emptyset \\
\delta_{i+1} &:= \delta_i, \alpha_{i+1} \mapsto \alpha_{i+1} \\
\rho_0 &:= \emptyset \\
\rho_{i+1} &:= \rho_i, \alpha_{i+1} \mapsto V_{j+2}[\![\tau_{i+1}]\!]\rho_i \\
w &:= (\sigma, \sigma, \{\alpha_i \mapsto (\alpha_i, \alpha_i) \mid 1 \leq i \leq n\}, \rho_n)
\end{aligned}$$

- Note that $\rho_i \in \mathrm{Interp}_{j+2}$ and $w \in \mathrm{World}_{j+2}$.
- Furthermore, given $0 \leq i < n$, it is easy to see that $(\delta_i, \delta_i, \rho_i) \in D_{j+2}[\![\sigma_i]\!]w$ implies $(\delta_{i+1}, \delta_{i+1}, \rho_{i+1}) \in D_{j+2}[\![\sigma_{i+1}]\!]w$.
- Together with $(\delta_0, \delta_0, \rho_0) \in D_{j+2}[\![\epsilon]\!]w$ this means $(\delta_n, \delta_n, \rho_n) \in D_{j+2}[\![\sigma]\!]w$.
- Now instantiate $\sigma; \emptyset \vdash C[e_1] \precsim C[e_2] : \tau'$ with $w \in \mathrm{World}_{j+2}$, $(\delta_n, \delta_n, \rho_n) \in D_{j+2}[\![\sigma]\!]w$ and $(j, \lfloor w \rfloor, \emptyset, \emptyset) \in G_{j+2}[\![\epsilon]\!]\rho_n$ to get $(j+1, \lfloor w \rfloor, \delta_n\emptyset(C[e_1]), \delta_n\emptyset(C[e_2])) \in E_{j+2}[\![\tau']\!]\rho_n$.
- Note that $\delta_n\emptyset(C[e_i]) = C[e_i]$.
- Consequently, $\sigma; C[e_1] \hookrightarrow^j \sigma_1; v_1$ and thus $\sigma; C[e_2]\downarrow$.

□

## 10.7 Relating the Relations

In the remainder of this section, we give proofs of the properties that relate our logical relations.

There is not much to say about the Chain theorem; its proof is very straightforward and relies on the following lemma:

**Lemma 10.37.** $T_n[\![\Omega]\!]w = T_n^-[\![\Omega]\!]w \subseteq T_n^+[\![\Omega]\!]w = T_n^\circ[\![\Omega]\!]w$

*Proof:* By definition. □

**Theorem 10.38** (Chain)**.**

1. $V_n^-[\![\tau]\!]\rho \subseteq V_n^\iota[\![\tau]\!]\rho \subseteq V_n^+[\![\tau]\!]\rho$
2. $E_n^-[\![\tau]\!]\rho \subseteq E_n^\iota[\![\tau]\!]\rho \subseteq E_n^+[\![\tau]\!]\rho$
3. $G_n^-[\![\tau]\!]\rho \subseteq G_n^\iota[\![\tau]\!]\rho \subseteq G_n^+[\![\tau]\!]\rho$

*Proof:* (1) and (2) mutually by primary induction on $n$ and secondary induction on $\tau$.

1. - Case $\tau = \alpha$ or $\tau = b$: Trivial
   - Case $\tau = \tau' \times \tau''$: Follows easily by induction.
   - Case $\tau = \tau'' \to \tau'$:
     (a) $V_n^-[\![\tau]\!]\rho \subseteq V_n^\iota[\![\tau]\!]\rho$:
     - Suppose $(k, w, \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n^-[\![\tau'' \to \tau']\!]\rho$.
     - Suppose $(k', w', v_1, v_2) \in V_n^{\neg\iota}[\![\tau'']\!]\rho$ with $(k', w') \sqsupseteq (k, w)$.

- By induction, $(k', w', v_1, v_2) \in V_n^+[\![\tau'']\!]\rho$.
- Hence by assumption, $(k', w', e_1[v_1/x], e_2[v_2/x]) \in E_n^-[\![\tau']\!]\rho$.
- So by induction, $(k', w', e_1[v_1/x], e_2[v_2/x]) \in E_n^\iota[\![\tau']\!]\rho$.
- Therefore, $(k, w, \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n^\iota[\![\tau'' \to \tau']\!]\rho$.

(b) $V_n^\iota[\![\tau]\!]\rho \subseteq V_n^+[\![\tau]\!]\rho$:

- Suppose $(k, w, \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n^\iota[\![\tau'' \to \tau']\!]\rho$.
- Suppose $(k', w', v_1, v_2) \in V_n^-[\![\tau'']\!]\rho$ with $(k', w') \sqsupseteq (k, w)$.
- By induction, $(k', w', v_1, v_2) \in V_n^\iota[\![\tau'']\!]\rho$.
- Hence by assumption, $(k', w', e_1[v_1/x], e_2[v_2/x]) \in E_n^\iota[\![\tau']\!]\rho$.
- So by induction, $(k', w', e_1[v_1/x], e_2[v_2/x]) \in E_n^+[\![\tau']\!]\rho$.
- Therefore, $(k, w, \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n^+[\![\tau'' \to \tau']\!]\rho$.

- Case $\tau = \forall\alpha.\tau'$:

(a) $V_n^-[\![\tau]\!]\rho \subseteq V_n^\iota[\![\tau]\!]\rho$:

- Suppose $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^-[\![\forall\alpha.\tau']\!]\rho$.
- Suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}^{\neg\iota}[\![\Omega]\!]w'$.
- By Lemma 10.37, $(\tau_1, \tau_2, r) \in T_{k'}^+[\![\Omega]\!]w'$.
- Hence by assumption, $(k', w', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n^-[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- So by induction, $(k', w', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n^\iota[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- Therefore, $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^\iota[\![\forall\alpha.\tau']\!]\rho$.

(b) $V_n^\iota[\![\tau]\!]\rho \subseteq V_n^+[\![\tau]\!]\rho$:

- Suppose $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^\iota[\![\forall\alpha.\tau']\!]\rho$.
- Suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}^-[\![\Omega]\!]w'$.
- By Lemma 10.37, $(\tau_1, \tau_2, r) \in T_{k'}^\iota[\![\Omega]\!]w'$.
- Hence by assumption, $(k', w', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n^\iota[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- So by induction, $(k', w', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n^+[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- Therefore, $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^+[\![\forall\alpha.\tau']\!]\rho$.

- Case $\tau = \exists\alpha.\tau'$:

(a) $V_n^-[\![\tau]\!]\rho \subseteq V_n^\iota[\![\tau]\!]\rho$:

- Suppose $(k, w, \mathsf{pack}\ \langle\tau_1, v_1\rangle\ \mathsf{as}\ \tau_1', \mathsf{pack}\ \langle\tau_2, v_2\rangle\ \mathsf{as}\ \tau_2') \in V_n^-[\![\exists\alpha.\tau']\!]\rho$.
- So there is $r$ such that $(\tau_1, \tau_2, r) \in T_k^-[\![\Omega]\!]w$ and $(k', w', v_1, v_2) \in V_n^-[\![\tau']\!]\rho, \alpha{\mapsto}r$ for any $(k', w') \sqsupset (k, w)$.
- So by induction, $(k', w', v_1, v_2) \in V_n^\iota[\![\tau']\!]\rho, \alpha{\mapsto}r$ for any $(k', w') \sqsupset (k, w)$.
- By Lemma 10.37, $(\tau_1, \tau_2, r) \in T_k^\iota[\![\Omega]\!]w$.
- Therefore, $(k, w, \mathsf{pack}\ \langle\tau_1, e_1\rangle\ \mathsf{as}\ \tau_1', \mathsf{pack}\ \langle\tau_2, e_2\rangle\ \mathsf{as}\ \tau_2') \in V_n^\iota[\![\exists\alpha.\tau']\!]\rho$.

(b) $V_n^\iota[\![\tau]\!]\rho \subseteq V_n^+[\![\tau]\!]\rho$:

- Suppose $(k, w, \mathsf{pack}\ \langle\tau_1, v_1\rangle\ \mathsf{as}\ \tau_2', \mathsf{pack}\ \langle\tau_2, v_2\rangle\ \mathsf{as}\ \tau_2') \in V_n^\iota[\![\exists\alpha.\tau']\!]\rho$.
- So there is $r$ such that $(\tau_1, \tau_2, r) \in T_k^\iota[\![\Omega]\!]w$ and $(k', w', v_1, v_2) \in V_n^\iota[\![\tau']\!]\rho, \alpha{\mapsto}r$ for any $(k', w') \sqsupset (k, w)$.
- So by induction, $(k', w', v_1, v_2) \in V_n^+[\![\tau']\!]\rho, \alpha{\mapsto}r$ for any $(k', w') \sqsupset (k, w)$.
- By Lemma 10.37, $(\tau_1, \tau_2, r) \in T_k^+[\![\Omega]\!]w$.
- Therefore, $(k, w, \mathsf{pack}\ \langle\tau_1, e_1\rangle\ \mathsf{as}\ \tau_1', \mathsf{pack}\ \langle\tau_2, e_2\rangle\ \mathsf{as}\ \tau_2') \in V_n^+[\![\exists\alpha.\tau']\!]\rho$.

- Case $\tau = \mu\alpha.\tau'$:

(a) $V_n^-[\![\tau]\!]\rho \subseteq V_n^\iota[\![\tau]\!]\rho$:
- Suppose $(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V_n^-[\![\mu\alpha.\tau']\!]\rho$.
- Suppose $(k', w') \sqsupset (k, w)$.
- Then $(k', w', v_1, v_2) \in V_k^-[\![\tau'[\tau/\alpha]]\!]\rho$.
- So by induction, $(k', w', v_1, v_2) \in V_k^\iota[\![\tau'[\tau/\alpha]]\!]\rho$.
- Therefore, $(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V_n^\iota[\![\mu\alpha.\tau']\!]\rho$.

(b) $V_n^\iota[\![\tau]\!]\rho \subseteq V_n^+[\![\tau]\!]\rho$:
- Suppose $(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V_n^\iota[\![\mu\alpha.\tau']\!]\rho$.
- Suppose $(k', w') \sqsupset (k, w)$.
- Then $(k', w', v_1, v_2) \in V_k^\iota[\![\tau'[\tau/\alpha]]\!]\rho$.
- So by induction, $(k', w', v_1, v_2) \in V_k^+[\![\tau'[\tau/\alpha]]\!]\rho$.
- Therefore, $(k, w, \mathsf{roll}\ v_1\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2\ \mathsf{as}\ \tau_2) \in V_n^+[\![\mu\alpha.\tau']\!]\rho$.

2. Follows from (1).

3. Follows from (1).

□

The next is a substitution lemma for the wrapping. Taking $\tau'$ to be $\tau$ (which is how it will be used), it says that wrapping at the unfolding of a recursive type $\mu\alpha.\tau$ relative to some environment (the right-hand side) is *syntactically* the same as "moving the unfolding into the environment" and then wrapping at $\tau$ (and finally applying a substitution to fix the typing annotations inside the wrapping functions, which now may contain $\alpha$ free). This lemma is crucial to manage the recursive type case in the wrapping theorem.

**Lemma 10.39** (WR-Substitution)**.** If $\varphi' = \varphi, \alpha{\mapsto}F^{\varphi}_{\mu\alpha.\tau}$, then $(\mathrm{Wr}^{\pm}_{\tau',}\ \varphi')[\mu\alpha.\tau/\alpha] = \mathrm{Wr}^{\pm}_{\tau'[\mu\alpha.\tau/\alpha]}\ \varphi$.

*Proof:* By induction on $\tau'$.

- Case $\tau' = \alpha$:

$$\begin{aligned}
&(\mathrm{Wr}^{+}_{\tau',}\ \varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\alpha.(\varphi'(\alpha)\ ()).1\ x)[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\alpha.(F^{\varphi}_{\mu\alpha.\tau}\ ()).1\ x)[\mu\alpha.\tau/\alpha] \\
&= \lambda x{:}(\mu\alpha.\tau).(F^{\varphi}_{\mu\alpha.\tau}\ ()).1\ x \\
&= \mathrm{Wr}^{+}_{\mu\alpha.\tau}\ \varphi \\
&= \mathrm{Wr}^{+}_{\tau'[\mu\alpha.\tau/\alpha]}\ \varphi
\end{aligned}$$

  (analogously for $\mathrm{Wr}^-$)

- Case $\tau' = \alpha' \in \mathrm{dom}(\varphi)$:

$$\begin{aligned}
&(\mathrm{Wr}^{+}_{\tau',}\ \varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\alpha'.(\varphi'(\alpha')\ ()).1\ x)[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\alpha'.(\varphi(\alpha')\ ()).1\ x)[\mu\alpha.\tau/\alpha] \\
&= (\mathrm{Wr}^{+}_{\alpha',}\ \varphi)[\mu\alpha.\tau/\alpha] \\
&= \mathrm{Wr}^{+}_{\alpha',}\ \varphi \\
&= \mathrm{Wr}^{+}_{\tau'[\mu\alpha.\tau/\alpha]}\ \varphi
\end{aligned}$$

  (analogously for $\mathrm{Wr}^-$)

- Case $\tau' = \alpha' \notin \mathrm{dom}(\varphi) \cup \{\alpha\}$: Trivial.
- Case $\tau' = \tau_1 \to \tau_2$:

$$\begin{aligned}
&(\mathrm{Wr}^{\pm}_{\tau'}\,\varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\tau'.\lambda x'{:}\tau_1.\,\mathrm{Wr}^{\pm}_{\tau_2}\,\varphi'\;(x\;(\mathrm{Wr}^{\mp}_{\tau_1}\,\varphi'\;x')))[\mu\alpha.\tau/\alpha] \\
&= \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\lambda x'{:}\tau_1[\mu\alpha.\tau/\alpha].(\mathrm{Wr}^{\pm}_{\tau_2}\,\varphi')[\mu\alpha.\tau/\alpha]\;(x\;((\mathrm{Wr}^{\mp}_{\tau_1}\,\varphi')[\mu\alpha.\tau/\alpha]\;x')) \\
&\overset{\text{ih}}{=} \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\lambda x'{:}\tau_1[\mu\alpha.\tau/\alpha].\,\mathrm{Wr}^{\pm}_{\tau_2[\mu\alpha.\tau/\alpha]}\,\varphi\;(x\;(\mathrm{Wr}^{\mp}_{\tau_1[\mu\alpha.\tau/\alpha]}\,\varphi\;x')) \\
&= \mathrm{Wr}^{\pm}_{\tau'[\mu\alpha.\tau/\alpha]}\,\varphi
\end{aligned}$$

- Case $\tau' = \tau_1 \times \tau_2$:

$$\begin{aligned}
&(\mathrm{Wr}^{\pm}_{\tau'}\,\varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\tau'.\langle \mathrm{Wr}^{\pm}_{\tau_1}\,\varphi'\;(x.1), \mathrm{Wr}^{\pm}_{\tau_2}\,\varphi'\;(x.2)\rangle)[\mu\alpha.\tau/\alpha] \\
&= \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\langle(\mathrm{Wr}^{\pm}_{\tau_1}\,\varphi')[\mu\alpha.\tau/\alpha]\;(x.1), (\mathrm{Wr}^{\pm}_{\tau_2}\,\varphi')[\mu\alpha.\tau/\alpha]\;(x.2)\rangle \\
&\overset{\text{ih}}{=} \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\langle \mathrm{Wr}^{\pm}_{\tau_1[\mu\alpha.\tau/\alpha]}\,\varphi\;(x.1), \mathrm{Wr}^{\pm}_{\tau_2[\mu\alpha.\tau/\alpha]}\,\varphi\;(x.2)\rangle \\
&= \mathrm{Wr}^{\pm}_{\tau'[\mu\alpha.\tau/\alpha]}\,\varphi
\end{aligned}$$

- Case $\tau' = \forall\alpha'.\tau''$:

$$\begin{aligned}
&(\mathrm{Wr}^{\pm}_{\tau'}\,\varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\tau'.\Lambda\alpha'.\,\mathsf{new}^{\mp}\,\alpha'\;\mathsf{in}\;\mathrm{Wr}^{\pm}_{\tau''}\,\varphi'\;(x\;\alpha'))[\mu\alpha.\tau/\alpha] \\
&= \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\Lambda\alpha'.\,\mathsf{new}^{\mp}\,\alpha'\;\mathsf{in}\;(\mathrm{Wr}^{\pm}_{\tau''}\,\varphi')[\mu\alpha.\tau]\;(x\;\alpha') \\
&\overset{\text{ih}}{=} \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\Lambda\alpha'.\,\mathsf{new}^{\mp}\,\alpha'\;\mathsf{in}\;\mathrm{Wr}^{\pm}_{\tau''[\mu\alpha.\tau/\alpha]}\,\varphi\;(x\;\alpha') \\
&= \mathrm{Wr}^{\pm}_{\tau'[\mu\alpha.\tau/\alpha]}\,\varphi
\end{aligned}$$

- Case $\tau' = \exists\alpha'.\tau''$:

$$\begin{aligned}
&(\mathrm{Wr}^{\pm}_{\tau'}\,\varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\tau'.\,\mathsf{unpack}\;\langle\alpha', x'\rangle{=}x\;\mathsf{in}\;\mathsf{new}^{\pm}\,\alpha'\;\mathsf{in}\;\mathsf{pack}\;\langle\alpha', \mathrm{Wr}^{\pm}_{\tau''}\,\varphi'\;x'\rangle\;\mathsf{as}\;\tau')[\mu\alpha.\tau/\alpha] \\
&= \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\,\mathsf{unpack}\;\langle\alpha', x'\rangle{=}x\;\mathsf{in}\;\mathsf{new}^{\pm}\,\alpha'\;\mathsf{in}\;\mathsf{pack}\;\langle\alpha', (\mathrm{Wr}^{\pm}_{\tau''}\,\varphi')[\mu\alpha.\tau/\alpha]\;x'\rangle\;\mathsf{as}\;\tau'[\mu\alpha.\tau/\alpha] \\
&\overset{\text{ih}}{=} \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\,\mathsf{unpack}\;\langle\alpha', x'\rangle{=}x\;\mathsf{in}\;\mathsf{new}^{\pm}\,\alpha'\;\mathsf{in}\;\mathsf{pack}\;\langle\alpha', \mathrm{Wr}^{\pm}_{\tau''[\mu\alpha.\tau/\alpha]}\,\varphi\;x'\rangle\;\mathsf{as}\;\tau'[\mu\alpha.\tau/\alpha] \\
&= \mathrm{Wr}^{\pm}_{\tau'[\mu\alpha.\tau/\alpha]}\,\varphi
\end{aligned}$$

- Case $\tau' = \mu\alpha'.\tau''$:

$$
\begin{aligned}
&F^{\varphi'}_{\tau'}[\mu\alpha.\tau/\alpha] \\
&= (\mathsf{fix}\, f(x').\\
&\quad \langle \lambda x{:}\tau'.\, \mathsf{roll}\, (\mathrm{Wr}^{+}_{\tau''}(\varphi, \alpha{\mapsto}f))[\tau'/\alpha']\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau', \\
&\quad \lambda x{:}\tau'.\, \mathsf{roll}\, (\mathrm{Wr}^{-}_{\tau''}(\varphi, \alpha{\mapsto}f))[\tau'/\alpha']\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau' \rangle : \mathsf{unit} \to \hat{\tau'})[\mu\alpha.\tau/\alpha] \\
&= \mathsf{fix}\, f(x').\\
&\quad \langle \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\, \mathsf{roll}\, (\mathrm{Wr}^{+}_{\tau''}(\varphi', \alpha'{\mapsto}x_f))[\tau'/\alpha'][\mu\alpha.\tau/\alpha]\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau'[\mu\alpha.\tau/\alpha], \\
&\quad \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\, \mathsf{roll}\, (\mathrm{Wr}^{-}_{\tau''}(\varphi', \alpha'{\mapsto}x_f))[\tau'/\alpha'][\mu\alpha.\tau/\alpha]\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau'[\mu\alpha.\tau/\alpha] \rangle \\
&\quad : \mathsf{unit} \to \hat{\tau'}[\mu\alpha.\tau/\alpha] \\
&= \mathsf{fix}\, f(x').\\
&\quad \langle \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\, \mathsf{roll}\, (\mathrm{Wr}^{+}_{\tau''}(\varphi', \alpha'{\mapsto}x_f))[\mu\alpha.\tau/\alpha][\tau'[\mu\alpha.\tau/\alpha]/\alpha']\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau'[\mu\alpha.\tau/\alpha], \\
&\quad \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\, \mathsf{roll}\, (\mathrm{Wr}^{-}_{\tau''}(\varphi', \alpha'{\mapsto}x_f))[\mu\alpha.\tau/\alpha][\tau'[\mu\alpha.\tau/\alpha]/\alpha']\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau'[\mu\alpha.\tau/\alpha] \rangle) \\
&\quad : \mathsf{unit} \to \hat{\tau'}[\mu\alpha.\tau/\alpha] \\
&\overset{\text{ih}}{=} \mathsf{fix}\, f(x').\\
&\quad \langle \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\, \mathsf{roll}\, (\mathrm{Wr}^{+}_{\tau''[\mu\alpha.\tau/\alpha]}(\varphi, \alpha'{\mapsto}x_f))[\tau'[\mu\alpha.\tau/\alpha]/\alpha']\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau'[\mu\alpha.\tau/\alpha], \\
&\quad \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].\, \mathsf{roll}\, (\mathrm{Wr}^{-}_{\tau''[\mu\alpha.\tau/\alpha]}(\varphi, \alpha'{\mapsto}x_f))[\tau'[\mu\alpha.\tau/\alpha]/\alpha']\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \tau'[\mu\alpha.\tau/\alpha] \rangle) \\
&\quad : \mathsf{unit} \to \hat{\tau'}[\mu\alpha.\tau/\alpha] \\
&= F^{\varphi}_{\tau'[\mu\alpha.\tau/\alpha]}
\end{aligned}
$$

where $\hat{\tau} = (\tau' \to \tau') \times (\tau' \to \tau')$
Hence:

$$
\begin{aligned}
&(\mathrm{Wr}^{+}_{\tau'}\, \varphi')[\mu\alpha.\tau/\alpha] \\
&= (\lambda x{:}\tau'.(F^{\varphi'}_{\tau'}\ ()).1\ x)[\mu\alpha.\tau/\alpha] \\
&= \lambda x{:}\tau'.(F^{\varphi'}_{\tau'}[\mu\alpha.\tau/\alpha]\ ()).1\ x \\
&= \lambda x{:}\tau'[\mu\alpha.\tau/\alpha].(F^{\varphi}_{\tau'[\mu\alpha.\tau/\alpha]}\ ()).1\ x \\
&= \mathrm{Wr}^{+}_{\tau'[\mu\alpha.\tau/\alpha]}\, \varphi
\end{aligned}
$$

(analogously for $\mathrm{Wr}^{-}$)

□

**Lemma 10.40.** If $\Delta \vdash \tau$, then $\Delta; \epsilon \vdash \mathrm{Wr}^{\pm}_{\tau} : \tau \to \tau$.

*Proof:* Easy induction on $\Delta \vdash \tau$. □

Now we come to the main theorem about the wrapping. Each subitem here actually states two properties, which are obtained by first consistently ignoring the left superscript of the $X^{\iota_1,\iota_2}$ notation in the whole statement, and then the right one. For instance, (1a) states that the positive wrapping transports values from $V^{\circ}$ to $E^{-}$ and, independently, from $V^{+}$ to $E$. Similarly, each proof actually represents two proofs.

The most interesting cases are existential types in the first part and universal types in the second part, because that is where the wrapping actually has to generate a fresh type. Technically, what happens in both cases is that we have some triple $(\tau_1, \tau_2, r) \in T^{\circ,+}[\![\Omega]\!]w'$, but would like it—or something equivalent—to be in $T^{-,\epsilon}[\![\Omega]\!]w''$, where $w''$ must be some extension of $w'$ that incorporates the new names $\alpha_1$ and $\alpha_2$. What we do is choose $w''$ such that it extends $w'$ by a new semantic name $\alpha$ that is connected to the concrete names $\alpha_1$ and $\alpha_2$ as well as their representation types, and is interpreted by the relation $r$. Then we can use $(\alpha_1, \alpha_2, (w''.\rho^1(\alpha), w''.\rho^2(\alpha), V[\![\alpha]\!]w''.\rho))$, which has the form required by $T^{-,\epsilon}[\![\Omega]\!]$ and, since $w''.\rho$ maps $\alpha$ to $r$, carries the same relation as $(\tau_1, \tau_2, r)$.

Another interesting case is a recursive type. Note that the wrapping theorem is stated for an empty environment $\varphi$ (recall that $\mathrm{Wr}^{\pm}_{\tau}$ is just short for $\mathrm{Wr}^{\pm}_{\tau}\,\emptyset$). This may seem not general enough at first, because in the case where $\tau = \mu\alpha.\tau'$ we need an induction hypothesis that talks about wrapping relative to the non-empty environment $\varphi := (\alpha \mapsto F^{\emptyset}_{\tau})$. This is exactly where Lemma 10.39 comes in: it tells us that the terms involving $\mathrm{Wr}^{\pm}_{\tau'[\tau/\alpha]}\,\emptyset$ that are related by use of the induction hypthesis are syntactically equal to the ones involving $\mathrm{Wr}^{\pm}_{\tau'}\,\varphi$ that we are interested in.

**Theorem 10.41** (Wrapping I)**.** Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D^{\circ}_n[\![\Delta]\!]w_0$, $(k, w) \sqsupset (n, w_0)$, and $\Delta \vdash \tau$.

1. (a) If $(k, w, v_1, v_2) \in V^{\circ,+}_n[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}^{+}_{\tau})\ v_1, \delta_2(\mathrm{Wr}^{+}_{\tau})\ v_2) \in E^{-,\epsilon}_n[\![\tau]\!]\rho$.
   (b) If $(k, w, e_1, e_2) \in E^{\circ,+}_n[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}^{+}_{\tau})\ e_1, \delta_2(\mathrm{Wr}^{+}_{\tau})\ e_2) \in E^{-,\epsilon}_n[\![\tau]\!]\rho$.
2. (a) If $(k, w, v_1, v_2) \in V^{+,\epsilon}_n[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}^{-}_{\tau})\ v_1, \delta_2(\mathrm{Wr}^{-}_{\tau})\ v_2) \in E^{\circ,-}_n[\![\tau]\!]\rho$.
   (b) If $(k, w, e_1, e_2) \in E^{+,\epsilon}_n[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}^{-}_{\tau})\ e_1, \delta_2(\mathrm{Wr}^{-}_{\tau})\ e_2) \in E^{\circ,-}_n[\![\tau]\!]\rho$.

*Proof:* By primary induction on $n$ and secondary induction on the derivation of $\Delta \vdash \tau$. Note that $\delta_i$ only affects the type annotations (of function arguments and package types) inside the wrapping function.

We first show that the tuple containg the wrapped terms is a valid atom, which boils down to showing $\vdash w.\sigma_i; \delta_i(\mathrm{Wr}^{\pm}_{\tau})\ e_i : \rho^i(\tau)$.

- By assumption and Atomicity (10.6), $w.\sigma_i; \epsilon \vdash e_i : \rho^i(\tau)$ and $\epsilon \vdash \rho^i(\tau)$.
- By Lemma 10.40, $\Delta; \epsilon \vdash \mathrm{Wr}^{\pm}_{\tau} : \tau \to \tau$.
- By Lemma 10.9 and Substitution (A.2), $w_0.\sigma_1; \epsilon \vdash \delta_i(\mathrm{Wr}^{\pm}_{\tau}) : \delta_i(\tau \to \tau)$.
- By Lemma 10.9 we know $\rho^i(\tau \to \tau) = w_0.\sigma^*_i(\delta^1(\tau \to \tau))$, so $w_0.\sigma_i; \epsilon \vdash \delta_i(\mathrm{Wr}^{\pm}_{\tau}) : \rho^i(\tau \to \tau)$.
- Hence $w.\sigma_i; \epsilon \vdash \delta_i(\mathrm{Wr}^{\pm}_{\tau})\ v_i : \rho^i(\tau)$ by Weakening (A.1) and rule Eapp.

The actual work comes now.

1. (a)
   - Case $\tau = \alpha$ or $\tau = b$: Easy.
   - Case $\tau = \tau' \times \tau''$: $v_i = \langle v_{i1}, v_{i2} \rangle$
     - To show: $(k, w, \delta_1(\lambda x{:}\tau.\langle \mathrm{Wr}^{+}_{\tau'}\ (x.1), \mathrm{Wr}^{-}_{\tau''}\ (x.2)\rangle)\ v_1,$
       $\delta_2(\lambda x{:}\tau.\langle \mathrm{Wr}^{+}_{\tau'}\ (x.1), \mathrm{Wr}^{-}_{\tau''}\ (x.2)\rangle)\ v_2) \in E^{-,\epsilon}_n[\![\tau' \times \tau'']\!]\rho$

– So suppose $w.\sigma_1; \delta_1(\lambda x{:}\tau.\langle \mathrm{Wr}^+_{\tau'}\ (x.1), \mathrm{Wr}^-_{\tau''}\ (x.2)\rangle)\ v_1$ terminates in $1+1+j_1+1+j_2 =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1(\lambda x{:}\tau.\langle \mathrm{Wr}^+_{\tau'}\ (x.1), \mathrm{Wr}^-_{\tau''}\ (x.2)\rangle)\ v_1 \\ \hookrightarrow^1 & w.\sigma_1; \langle \delta_1(\mathrm{Wr}^+_{\tau'})\ (v_1.1), \delta_1(\mathrm{Wr}^-_{\tau''})\ (v_1.2)\rangle \\ \hookrightarrow^1 & w.\sigma_1; \langle \delta_1(\mathrm{Wr}^+_{\tau'})\ v_{11}, \delta_1(\mathrm{Wr}^-_{\tau''})\ (v_1.2)\rangle \\ \hookrightarrow^{j_1} & \sigma_1'; \langle v_{11}', \delta_1(\mathrm{Wr}^-_{\tau''})\ (v_1.2)\rangle \\ \hookrightarrow^1 & \sigma_1'; \langle v_{11}', \delta_1(\mathrm{Wr}^-_{\tau''})\ v_{12}\rangle \\ \hookrightarrow^{j_2} & \sigma_1; \langle v_{11}', v_{12}'\rangle \end{array}$$

– Note that

$$\begin{array}{rl} & w.\sigma_2; \delta_2(\lambda x{:}\tau.\langle \mathrm{Wr}^+_{\tau'}\ (x.1), \mathrm{Wr}^-_{\tau''}\ (x.2)\rangle)\ v_2 \\ \hookrightarrow^1 & w.\sigma_2; \langle \delta_2(\mathrm{Wr}^+_{\tau'})\ (v_2.1), \delta_2(\mathrm{Wr}^-_{\tau''})\ (v_2.2)\rangle \\ \hookrightarrow^1 & w.\sigma_2; \langle \delta_2(\mathrm{Wr}^+_{\tau'})\ v_{21}, \delta_2(\mathrm{Wr}^-_{\tau''})\ (v_2.2)\rangle \end{array}$$

– By assumption and Closure Under World Extension (10.7) we know $(k-2, \lfloor w\rfloor, v_{11}, v_{21}) \in V_n^{\circ,+}[\![\tau']\!]\rho$.
– By induction, $(k-2, \lfloor w\rfloor, \delta_1(\mathrm{Wr}^+_{\tau'})\ v_{11}, \delta_2(\mathrm{Wr}^+_{\tau'})\ v_{21}) \in E_n^{-,\epsilon}[\![\tau']\!]\rho$.
– Consequently, there exists $(k-2-j_1, w') \sqsupseteq (k-2, \lfloor w\rfloor)$ such that $w.\sigma_2; \langle \delta_2(\mathrm{Wr}^+_{\tau'})\ v_{21}, \delta_2(\mathrm{Wr}^-_{\tau''})\ (v_2.2)\rangle \hookrightarrow^* w'.\sigma_2; \langle v_{21}', \delta_2(\mathrm{Wr}^-_{\tau''})\ (v_2.2)\rangle$ with $w'.\sigma_1 = \sigma_1'$ and $(k-2-j_1, w', v_{11}, v_{21}) \in V_n^{-,\epsilon}[\![\tau']\!]\rho$.
– By assumption and Closure Under World Extension (10.7) we know $(k-2-j_1-1, \lfloor w'\rfloor, v_{12}, v_{22}) \in V_n^{\circ,+}[\![\tau'']\!]\rho$.
– By induction, $(k-2-j_1-1, \lfloor w'\rfloor, \delta_1(\mathrm{Wr}^+_{\tau''})\ v_{12}, \delta_2(\mathrm{Wr}^+_{\tau''})\ v_{22}) \in E_n^{-,\epsilon}[\![\tau'']\!]\rho$.
– Consequently, there exists $(k-j, w'') \sqsupseteq (k-2-j_1-1, \lfloor w'\rfloor)$ such that $w'.\sigma_2; \langle v_{21}', \delta_2(\mathrm{Wr}^-_{\tau''})\ v_{22}\rangle \hookrightarrow^* w'.\sigma_2; \langle v_{21}', v_{22}'\rangle$ with $w''.\sigma_1 = \sigma_1$ and $(k-j, w'', v_{12}, v_{22}) \in V_n^{-,\epsilon}[\![\tau'']\!]\rho$.
– By Closure Under World Extension (10.7), $(k-j, w'', \langle v_{11}', v_{12}'\rangle, \langle v_{21}', v_{22}'\rangle) \in V_n^{-,\epsilon}[\![\tau' \times \tau'']\!]\rho$.

• Case $\tau = \tau' \to \tau''$: $v_i = \lambda x{:}\tau_i.e_i$
– To show: $(k, w, \delta_1(\lambda x{:}\tau.\lambda x'{:}\tau'.\, \mathrm{Wr}^+_{\tau''}\ (x\ (\mathrm{Wr}^-_{\tau'}\ x'))))\ v_1$, $\delta_2(\lambda x{:}\tau.\lambda x'{:}\tau'.\, \mathrm{Wr}^+_{\tau''}\ (x\ (\mathrm{Wr}^-_{\tau'}\ x'))))\ v_2) \in E_n^{-,\epsilon}[\![\tau' \to \tau'']\!]\rho$
– Since

$$\begin{array}{rl} & w.\sigma_i; \delta_i(\lambda x{:}\tau.\lambda x'{:}\tau'.\, \mathrm{Wr}^+_{\tau''}\ (x\ (\mathrm{Wr}^-_{\tau'}\ x'))))\ v_i \\ \hookrightarrow^1 & w.\sigma_i; \lambda x'{:}\delta_i(\tau').\delta_i(\mathrm{Wr}^+_{\tau''})\ (v_i\ (\delta_i(\mathrm{Wr}^-_{\tau'})\ x')) \end{array}$$

it suffices to show $(k-1, \lfloor w\rfloor, \lambda x'{:}\delta_1(\tau').\delta_1(\mathrm{Wr}^+_{\tau''})\ (v_1\ (\delta_1(\mathrm{Wr}^-_{\tau'})\ x'))$, $\lambda x'{:}\delta_2(\tau').\delta_2(\mathrm{Wr}^+_{\tau''})\ (v_2\ (\delta_2(\mathrm{Wr}^-_{\tau'})\ x'))) \in V_n^{-,\epsilon}[\![\tau' \to \tau'']\!]\rho$.
– Suppose $(k', w', v_3, v_4) \in V_n^{+,\epsilon}[\![\tau']\!]\rho$ where $(k', w') \sqsupseteq (k-1, \lfloor w\rfloor)$.
– To show: $(k', w', \delta_1(\mathrm{Wr}^+_{\tau''})\ (v_1\ (\delta_1(\mathrm{Wr}^-_{\tau'})\ v_3))$, $\delta_2(\mathrm{Wr}^+_{\tau''})\ (v_2\ (\delta_2(\mathrm{Wr}^-_{\tau'})\ v_4)))) \in E_n^{-,\epsilon}[\![\tau'']\!]\rho$
– So suppose $w'.\sigma_1; \delta_1(\mathrm{Wr}^+_{\tau''})\ (v_1\ (\delta_1(\mathrm{Wr}^-_{\tau'})\ v_3))$ terminates in $j_1+1+j_2 =: j < k'$ steps:

$$\begin{array}{rl} & w'.\sigma_1; \delta_1(\mathrm{Wr}^+_{\tau''})\ (v_1\ (\delta_1(\mathrm{Wr}^-_{\tau'})\ v_3)) \\ \hookrightarrow^{j_1} & \sigma_1''; \delta_1(\mathrm{Wr}^+_{\tau''})\ (v_1\ v_3') \\ \hookrightarrow^1 & \sigma_1''; \delta_1(\mathrm{Wr}^+_{\tau''})\ e_1[v_3'/x] \\ \hookrightarrow^{j_2} & \sigma_1; v_1' \end{array}$$

- By induction, $(k', w', \delta_1(\mathrm{Wr}^-_{\tau'})\ v_3, \delta_2(\mathrm{Wr}^-_{\tau'})\ v_4) \in E^{\circ,-}_n[\![\tau']\!]\rho$.
- This implies the existence of $(k' - j_1, w'') \sqsupseteq (k', w')$ such that $w'.\sigma_2; \delta_2(\mathrm{Wr}^+_{\tau''})\ (v_2\ (\delta_2(\mathrm{Wr}^-_{\tau'})\ v_4)) \hookrightarrow^* w''.\sigma_2; \delta_2(\mathrm{Wr}^+_{\tau''})\ (v_2\ v'_4)$ with $w''.\sigma_1 = \sigma''_1$ and $(k' - j_1, w'', v'_3, v'_4) \in V^{\circ,-}_n[\![\tau']\!]\rho$.
- So by assumption and Closure Under World Extension (10.7), $(k' - j_1 - 1, \lfloor w'' \rfloor, e_1[v'_3/x], e_2[v'_4/x]) \in E^{\circ,+}_n[\![\tau'']\!]\rho$.
- By induction, $(k' - j_1 - 1, \lfloor w'' \rfloor, \delta_1(\mathrm{Wr}^+_{\tau''})\ e_1[v'_3/x], \delta_2(\mathrm{Wr}^+_{\tau''})\ e_2[v'_4/x]) \in E^{-,\epsilon}_n[\![\tau'']\!]\rho$.
- Hence there exists $(k' - j, w''') \sqsupseteq (k' - j_1 - 1, \lfloor w'' \rfloor)$ such that $w''.\sigma_2; \delta_1(\mathrm{Wr}^+_{\tau''})\ e_2[v'_4/x] \hookrightarrow^* w'''.\sigma_2; v'_2$ with $w'''.\sigma_1 = \sigma_1$ and $(k' - j, w''', v'_1, v'_2) \in V^{-,\epsilon}_n[\![\tau'']\!]\rho$.

- Case $\tau = \forall\alpha.\tau'$: $v_i = \Lambda\alpha.e_i$
  - To show: $(k, w, \delta_1(\lambda x{:}\tau.\Lambda\alpha.\,\mathrm{Wr}^+_{\tau'}\ (x\ \alpha))\ v_1,$ $\delta_2(\lambda x{:}\tau.\Lambda\alpha.\,\mathrm{Wr}^+_{\tau'}\ (x\ \alpha))\ v_2) \in V^{-,\epsilon}_n[\![\forall\alpha.\tau']\!]\rho$
  - Since
    $$\begin{array}{rl} & w.\sigma_i; \delta_i(\lambda x{:}\tau.\Lambda\alpha.\,\mathrm{Wr}^+_{\tau'}\ (x\ \alpha))\ v_i \\ \hookrightarrow^1 & w.\sigma_i; \Lambda\alpha.\delta_i(\mathrm{Wr}^+_{\tau'})\ (v_i\ \alpha) \end{array}$$
    it suffices to show $(k - 1, \lfloor w \rfloor, \Lambda\alpha.\delta_1(\mathrm{Wr}^+_{\tau'})\ (v_1\ \alpha), \Lambda\alpha.\delta_2(\mathrm{Wr}^+_{\tau'})\ (v_2\ \alpha)) \in V^{-,\epsilon}_n[\![\forall\alpha.\tau']\!]\rho$.
  - Suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k - 1, \lfloor w \rfloor)$ and $(\tau_1, \tau_2, r) \in T^{+,\epsilon}_{k'}[\![\Omega]\!]w'$.
  - To show: $(k'', w'', \delta'_1(\mathrm{Wr}^+_{\tau'})\ (v_1\ \tau_1), \delta'_2(\mathrm{Wr}^+_{\tau'})\ (v_2\ \tau_2)) \in E^{-,\epsilon}_n[\![\tau']\!]\rho, \alpha{\mapsto}r$, for $\delta'_i := \delta_i, \alpha{\mapsto}\tau_i$.
  - So suppose $w''.\sigma_1; \delta'_1(\mathrm{Wr}^+_{\tau'})\ (v_1\ \tau_1)$ terminates in $1 + j' =: j < k''$ steps:
    $$\begin{array}{rl} & w''.\sigma_1; \delta'_1(\mathrm{Wr}^+_{\tau'})\ (v_1\ \tau_1) \\ \hookrightarrow^1 & w''.\sigma_1; \delta'_1(\mathrm{Wr}^+_{\tau'})\ e_1[\tau_1/\alpha] \\ \hookrightarrow^{j'} & \sigma_1; v'_1 \end{array}$$
  - By Lemma 10.37, $(\tau_1, \tau_2, r) \in T^{\circ,-}_{k'}[\![\Omega]\!]w'$.
  - Instantiate the assumption with $(k'' - 1, \lfloor w'' \rfloor) \sqsupset (k', w') \sqsupseteq (k - 1, \lfloor w \rfloor)$ and $(\tau_1, \tau_2, r)$ to get $(k'' - 1, \lfloor w'' \rfloor, e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E^{\circ,+}_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
  - Since $(\delta_1, \delta_2, \lfloor\rho\rfloor_{k'}) \in D^\circ_{k'}[\![\Delta, \alpha]\!]w'$ by Lemma 10.4, Lemma 10.37 yields $(\delta'_1, \delta'_2, (\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r)) \in D^\circ_{k'}[\![\Delta, \alpha]\!]w'$.
  - Hence by Restriction (10.2), Irrelevance (10.3) and induction, $(k'' - 1, \lfloor w'' \rfloor, \delta'_1(\mathrm{Wr}^+_{\tau'})\ (e_1[\tau_1/\alpha], \delta'_2(\mathrm{Wr}^+_{\tau'})\ e_2[\tau_2/\alpha])) \in E^{-,\epsilon}_n[\![\tau']\!]\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r$.
  - Consequently, there exists $(k'' - j, w''') \sqsupseteq (k'' - 1, \lfloor w'' \rfloor)$ such that $w''.\sigma_2; \delta'_2(\mathrm{Wr}^+_{\tau'})\ e_2[\tau_2/\alpha] \hookrightarrow^* w'''.\sigma_2; v'_2$ with $w'''.\sigma_1 = \sigma_1$ and $(k'' - j, w''', v'_1, v'_2) \in V^{-,\epsilon}_n[\![\tau']\!]\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r$.
  - By Restriction (10.2) and Irrelevance (10.3) the latter means $(k'' - j, w''', v'_1, v'_2) \in V^{-,\epsilon}_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- Case $\tau = \exists\alpha.\tau'$: $v_i = \mathsf{pack}\ \langle\tau_i, v'_i\rangle\ \mathsf{as}\ \tau'_i$
  - To show: $(k, w, \delta_1(\lambda x{:}\tau.\,\mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in}\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \mathsf{pack}\ \langle\alpha, \mathrm{Wr}^+_{\tau'}\ x'\rangle\ \mathsf{as}\ \tau)\ v_1, \delta_2(\lambda x{:}\tau.\,\mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in}\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \mathsf{pack}\ \langle\alpha, \mathrm{Wr}^+_{\tau'}\ x'\rangle\ \mathsf{as}\ \tau)\ v_2) \in E^{-,\epsilon}_n[\![\exists\alpha.\tau']\!]\rho$

– So suppose $w.\sigma_1; \delta_1(\lambda x{:}\tau.\,\mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in\ new}\ \alpha{\approx}\alpha\ \mathsf{in\ pack}\ \langle\alpha, \mathrm{Wr}^+_{\tau'}\ x'\rangle\ \mathsf{as}\ \tau)\ v_1$ terminates in $3 + j' =: j < k$ steps:

$$\begin{array}{ll} & w.\sigma_1; \delta_1(\lambda x{:}\tau.\,\mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in\ new}\ \alpha{\approx}\alpha\ \mathsf{in\ pack}\ \langle\alpha, \mathrm{Wr}^+_{\tau'}\ x'\rangle\ \mathsf{as}\ \tau)\ v_1 \\ \hookrightarrow^1 & w.\sigma_1; \mathsf{unpack}\ \langle\alpha, x'\rangle{=}v_1\ \mathsf{in\ new}\ \alpha{\approx}\alpha\ \mathsf{in\ pack}\ \langle\alpha, \delta_1(\mathrm{Wr}^+_{\tau'})\ x'\rangle\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^1 & w.\sigma_1; \mathsf{new}\ \alpha{\approx}\tau_1\ \mathsf{in\ pack}\ \langle\alpha, \delta_1(\mathrm{Wr}^+_{\tau'})\ v_1'\rangle\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^1 & w.\sigma_1, \alpha_1{\approx}\tau_1; \mathsf{pack}\ \langle\alpha_1, \delta_1'(\mathrm{Wr}^+_{\tau'})\ v_1'\rangle\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^{j'} & \sigma_1; \mathsf{pack}\ \langle\alpha_1, v_1''\rangle\ \mathsf{as}\ \delta_1(\tau) \end{array}$$

where $\delta_1' := \delta_1, \alpha{\mapsto}\alpha_1$

– Note that

$$\begin{array}{ll} & w.\sigma_2; \delta_2(\lambda x{:}\tau.\,\mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in\ new}\ \alpha{\approx}\alpha\ \mathsf{in\ pack}\ \langle\alpha, \mathrm{Wr}^+_{\tau'}\ x'\rangle\ \mathsf{as}\ \tau)\ v_2 \\ \hookrightarrow^1 & w.\sigma_2; \mathsf{unpack}\ \langle\alpha, x'\rangle{=}v_2\ \mathsf{in\ new}\ \alpha{\approx}\alpha\ \mathsf{in\ pack}\ \langle\alpha, \delta_2(\mathrm{Wr}^+_{\tau'})\ x'\rangle\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow^1 & w.\sigma_2; \mathsf{new}\ \alpha{\approx}\tau_2\ \mathsf{in\ pack}\ \langle\alpha, \delta_2(\mathrm{Wr}^+_{\tau'})\ v_2'\rangle\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow^1 & w.\sigma_2, \alpha_2{\approx}\tau_2; \mathsf{pack}\ \langle\alpha_2, \delta_2'(\mathrm{Wr}^+_{\tau'})\ v_2'\rangle\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

where $\delta_2' := \delta_2, \alpha{\mapsto}\alpha_2$

– By assumption we know $(k', w', v_1', v_2') \in V_n^{\circ,+}[\![\tau']\!]\rho, \alpha{\mapsto}r$ for some $r$ with $(\tau_1, \tau_2, r) \in T_k^{\circ,+}[\![\Omega]\!]w$ and any $(k', w') \sqsupset (k, w)$.

– Let $w_\alpha := \lfloor w \rfloor_{k-2} \uplus (\alpha_1{\approx}\tau_1, \alpha_2{\approx}\tau_2, \alpha{\mapsto}(\alpha_1, \alpha_2), \alpha{\mapsto}\lfloor r \rfloor_{k-2})$, so $(k-2, w_\alpha) \sqsupset (k, w)$.

– Hence $(k-2, w_\alpha, v_1', v_2') \in V_n^{\circ,+}[\![\tau']\!]\rho, \alpha{\mapsto}r$.

– By Closure Under World Extension (10.7), $(k-3, \lfloor w_\alpha \rfloor, v_1', v_2') \in V_n^{\circ,+}[\![\tau']\!]\rho, \alpha{\mapsto}r$.

– Let $r' := (w_\alpha.\rho^1(\alpha), w_\alpha.\rho^2(\alpha), V_{k-2}[\![\alpha]\!]w_\alpha) = \lfloor r \rfloor_{k-2}$, so $(\alpha_1, \alpha_2, r') \in T_{k-2}^{-,\epsilon}[\![\Omega]\!]w_\alpha$.

– By Restriction (10.2) and Irrelevance (10.3), $(k-3, \lfloor w_\alpha \rfloor, v_1', v_2') \in V_n^{\circ,+}[\![\tau']\!]\rho'$ for $\rho' := \lfloor \rho \rfloor_{k-2}, \alpha{\mapsto}r'$.

– By Lemma 10.37, $(\alpha_1, \alpha_2, r') \in T_{k-2}^{\circ}[\![\Omega]\!]w_\alpha$.

– Furthermore $(\delta_1, \delta_2, \lfloor \rho \rfloor_{k-2}) \in D_{k-2}^{\circ}[\![\Delta]\!]w_\alpha$ by Lemma 10.4, so $(\delta_1', \delta_2', \rho') \in D_{k-2}^{\circ}[\![\Delta, \alpha]\!]w_\alpha$.

– Hence induction and Restriction (10.2) yields $(k-3, \lfloor w_\alpha \rfloor, \delta_1'(\mathrm{Wr}^+_{\tau'})\ v_1', \delta_2'(\mathrm{Wr}^+_{\tau'})\ v_2') \in E_n^{-,\epsilon}[\![\tau']\!]\rho'$.

– Because $w_\alpha.\sigma_1 = w.\sigma_1, \alpha_1{\approx}\tau_1$, this implies the existence of $(k-j, w') \sqsupseteq (k-3, \lfloor w_\alpha \rfloor)$ such that $w.\sigma_2, \alpha_2{\approx}\tau_2; \mathsf{pack}\ \langle\alpha_2, \delta_2'(\mathrm{Wr}^+_{\tau'})\ v_2'\rangle\ \mathsf{as}\ \delta_2(\tau) \hookrightarrow^* w'.\sigma_2; \mathsf{pack}\ \langle\alpha_2, v_2''\rangle\ \mathsf{as}\ \delta_2(\tau)$ with $w'.\sigma_1 = \sigma_1$ and $(k-j, w', v_1'', v_2'') \in V_n^{-,\epsilon}[\![\tau']\!]\rho'$.

– By Closure Under World Extension (10.7), Restriction (10.2) and Irrelevance (10.3), $(k'', w'', v_1'', v_2'') \in V_n^{-,\epsilon}[\![\tau']\!]\rho, \alpha{\mapsto}\lfloor r' \rfloor_{k-j}$ for any $(k'', w'') \sqsupset (k-j, w')$.

– Since $(\alpha_1, \alpha_2, \lfloor r' \rfloor_{k-j}) \in T_{k-j}^{-,\epsilon}[\![\Omega]\!]w'$ by Lemma 10.4, $(k-j, w', \mathsf{pack}\ \langle\alpha_1, v_1''\rangle\ \mathsf{as}\ \delta_1(\tau), \mathsf{pack}\ \langle\alpha_2, v_2''\rangle\ \mathsf{as}\ \delta_2(\tau)) \in V_n^{-,\epsilon}[\![\exists\alpha.\tau']\!]\rho$.

• Case $\tau = \mu\alpha.\tau'$: $v_i = \mathsf{roll}\ v_i'\ \mathsf{as}\ \tau_i$

– To show: $(k, w, \delta_1(\lambda x{:}\tau.(F_\tau^\emptyset\ ()).1\ x)\ v_1, \delta_2(\lambda x{:}\tau.(F_\tau^\emptyset\ ()).1\ x)\ v_2) \in E_n^{-,\epsilon}[\![\mu\alpha.\tau']\!]\rho$

– So suppose $w.\sigma_1; \delta_1(\lambda x{:}\tau.(F^{\emptyset}_{\tau}\ ()).1\ x)\ v_1$ terminates in $1 + j_c + 1 + j' =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1(\lambda x{:}\tau.(F^{\emptyset}_{\tau}\ ()).1\ x)\ v_1 \\ \hookrightarrow^1 & w.\sigma_1; (\delta_1(F^{\emptyset}_{\tau})\ ()).1\ v_1 \\ \hookrightarrow^{j_c} & w.\sigma_1; \mathsf{roll}\ \delta_1(\mathrm{Wr}^+_{\tau'}(\alpha{\mapsto}F^{\emptyset}_{\tau})[\tau/\alpha])\ (\mathsf{unroll}\ v_1)\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^1 & w.\sigma_1; \mathsf{roll}\ \delta_1(\mathrm{Wr}^+_{\tau'}(\alpha{\mapsto}F^{\emptyset}_{\tau})[\tau/\alpha])\ v_1'\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^{j'} & \sigma_1; \mathsf{roll}\ v_1''\ \mathsf{as}\ \delta_1(\tau) \end{array}$$

– Note that

$$\begin{array}{rl} & w.\sigma_2; \delta_2(\lambda x{:}\tau.(F^{\emptyset}_{\tau}\ ()).1\ x)\ v_2 \\ \hookrightarrow^1 & w.\sigma_2; (\delta_2(F^{\emptyset}_{\tau})\ ()).1\ v_2 \\ \hookrightarrow^{j_c} & w.\sigma_2; \mathsf{roll}\ \delta_2(\mathrm{Wr}^+_{\tau'}(\alpha{\mapsto}F^{\emptyset}_{\tau})[\tau/\alpha])\ (\mathsf{unroll}\ v_2)\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow^1 & w.\sigma_2; \mathsf{roll}\ \delta_2(\mathrm{Wr}^+_{\tau'}(\alpha{\mapsto}F^{\emptyset}_{\tau})[\tau/\alpha])\ v_2'\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

– By assumption we know $(k-j, \lfloor w \rfloor, v_1', v_2') \in V^{\circ,+}_k[\![\tau'[\tau/\alpha]]\!]\rho$.
– By induction,
$(k-j, \lfloor w \rfloor, \delta_1(\mathrm{Wr}^+_{\tau'[\tau/\alpha]})\ v_1', \delta_2(\mathrm{Wr}^+_{\tau'[\tau/\alpha]})\ v_2') \in E^{-,\epsilon}_k[\![\tau'[\tau/\alpha]]\!]\rho$.
– By Lemma 10.39, $\mathrm{Wr}^+_{\tau'[\tau/\alpha]} = \mathrm{Wr}^+_{\tau'}(\alpha{\mapsto}F^{\emptyset}_{\tau})[\tau/\alpha]$.
– Consequently, there exists $(k-j, w') \sqsupset (k-j_c-1, \lfloor w \rfloor)$ such that $w.\sigma_2; \mathsf{roll}\ \delta_2(\mathrm{Wr}^+_{\tau'}(\alpha{\mapsto}F^{\emptyset}_{\tau})[\tau/\alpha])\ v_2'\ \mathsf{as}\ \delta_2(\tau) \hookrightarrow^* w'.\sigma_2; \mathsf{roll}\ v_2''\ \mathsf{as}\ \delta_2(\tau)$ with $w'.\sigma_1 = \sigma_1$ and $(k-j, w', v_1'', v_2'') \in V^{-,\epsilon}_k[\![\tau'[\tau/\alpha]]\!]\rho$.
– By Closure Under World Extension (10.7) the latter implies $(k-j, w', \mathsf{roll}\ v_1''\ \mathsf{as}\ \delta_1(\tau), \mathsf{roll}\ v_2''\ \mathsf{as}\ \delta_2(\tau)) \in V^{-,\epsilon}_k[\![\tau]\!]\rho$.
– The claim then follows by Restriction (10.2).

(b) • Suppose $w.\sigma_1; \delta_1(\mathrm{Wr}^+_{\tau})\ e_1$ terminates in $j_1 + j_2 =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1(\mathrm{Wr}^+_{\tau})\ e_1 \\ \hookrightarrow^{j_1} & \sigma_1'; \delta_1(\mathrm{Wr}^+_{\tau})\ v_1 \\ \hookrightarrow^{j_2} & \sigma_1; v_1' \end{array}$$

• So by assumption there exists $(k-j_1, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2(\mathrm{Wr}^+_{\tau})\ e_2 \hookrightarrow^* w'.\sigma_2; \delta_2(\mathrm{Wr}^+_{\tau})\ v_2$ with $w'.\sigma_1 = \sigma_1'$ and $(k-j_1, w', v_1, v_2) \in V^{\circ,+}_n[\![\tau]\!]\rho$.
• By part (a), $(k-j_1, w', \delta_1(\mathrm{Wr}^+_{\tau})\ v_1, \delta_2(\mathrm{Wr}^+_{\tau})\ v_2) \in E^{-,\epsilon}_n[\![\tau]\!]\rho$.
• Consequently, there exists $(k-j, w'') \sqsupseteq (k-j_1, w')$ such that $w'.\sigma_2; \delta_2(\mathrm{Wr}^+_{\tau})\ v_2 \hookrightarrow^* w''.\sigma_2; v_2'$ with $w''.\sigma_1 = \sigma_1$ and $(k-j, w'', v_1', v_2') \in V^{-,\epsilon}_n[\![\tau]\!]\rho$.

2. (a) • Case $\tau = \alpha$ or $\tau = b$: Trivial.
• Case $\tau = \tau' \to \tau''$: Symmetric to arrow case of (1a).
• Case $\tau = \tau' \times \tau''$: Symmetric to times case of (1a).
• Case $\tau = \forall\alpha.\tau'$: $v_i = \Lambda\alpha.e_i$
– To show: $(k, w, \delta_1(\lambda x{:}\tau.\Lambda\alpha.\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \mathrm{Wr}^-_{\tau'}\ (x\ \alpha))\ v_1$, $\delta_2(\lambda x{:}\tau.\Lambda\alpha.\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \mathrm{Wr}^-_{\tau'}\ (x\ \alpha))\ v_2) \in V^{\circ,-}_n[\![\forall\alpha.\tau']\!]\rho$
– Since

$$\begin{array}{rl} & w.\sigma_i; \delta_i(\lambda x{:}\tau.\Lambda\alpha.\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \mathrm{Wr}^-_{\tau'}\ (x\ \alpha))\ v_i \\ \hookrightarrow^1 & w.\sigma_i; \Lambda\alpha.\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \delta_i(\mathrm{Wr}^-_{\tau'})\ (v_i\ \alpha) \end{array}$$

it suffices to show $(k-1, \lfloor w \rfloor, \Lambda\alpha.\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \delta_1(\mathrm{Wr}^-_{\tau'})\ (v_1\ \alpha), \Lambda\alpha.\ \mathsf{new}\ \alpha{\approx}\alpha\ \mathsf{in}\ \delta_2(\mathrm{Wr}^-_{\tau'})\ (v_2\ \alpha)) \in V^{-,\epsilon}_n[\![\forall\alpha.\tau']\!]\rho$.

– Suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k-1, \lfloor w \rfloor)$ and $(\tau_1, \tau_2, r) \in T^{\circ,+}_{k'}[\![\Omega]\!]w'$.
– To show: $(k'', w'', \mathsf{new}\ \alpha{\approx}\tau_1\ \mathsf{in}\ \delta_1(\mathrm{Wr}^-_{\tau'})\ (v_1\ \alpha),$
$\mathsf{new}\ \alpha{\approx}\tau_2\ \mathsf{in}\ \delta_2(\mathrm{Wr}^-_{\tau'})\ (v_2\ \alpha)) \in E^{\circ,-}_n[\![\tau']\!]\rho, \alpha{\mapsto}r$
– So suppose $w''.\sigma_1; \mathsf{new}\ \alpha{\approx}\tau_1\ \mathsf{in}\ \delta_1(\mathrm{Wr}^-_{\tau'})\ (v_1\ \alpha))$ terminates in $1{+}1{+}j_1{+}j_2 =: j < k''$ steps:

$$\begin{array}{rl} & w''.\sigma_1; \mathsf{new}\ \alpha{\approx}\tau_1\ \mathsf{in}\ \delta_1(\mathrm{Wr}^-_{\tau'})\ (v_1\ \alpha) \\ \hookrightarrow^1 & w''.\sigma_1, \alpha_1{\approx}\tau_1; \delta'_1(\mathrm{Wr}^-_{\tau'})\ (v_1\ \alpha_1) \\ \hookrightarrow^1 & w''.\sigma_1, \alpha_1{\approx}\tau_1; \delta'_1(\mathrm{Wr}^-_{\tau'})\ e_1[\alpha_1/\alpha] \\ \hookrightarrow^{j'} & \sigma_1; v'_1 \end{array}$$

where $\delta'_1 := \delta_1, \alpha{\mapsto}\alpha_1$
– Note that

$$\begin{array}{rl} & w''.\sigma, \mathsf{new}\ \alpha{\approx}\tau_2\ \mathsf{in}\ \delta_2(\mathrm{Wr}^-_{\tau'})\ (v_2\ \alpha) \\ \hookrightarrow^1 & w''.\sigma, \alpha_2{\approx}\tau_2; \delta'_2(\mathrm{Wr}^-_{\tau'})\ (v_2\ \alpha_2) \\ \hookrightarrow^1 & w''.\sigma, \alpha_2{\approx}\tau_2; \delta'_2(\mathrm{Wr}^-_{\tau'})\ e_2[\alpha_2/\alpha] \end{array}$$

where $\delta'_2 := \delta_2, \alpha{\mapsto}\alpha_2$
– Let $w''_\alpha := \lfloor w'' \rfloor_{k''-1} \uplus (\alpha_1{\approx}\tau_1, \alpha_2{\approx}\tau_2, \alpha{\mapsto}(\alpha_1, \alpha_2), \alpha{\mapsto}\lfloor r \rfloor_{k''-1})$, so $(k'' - 1, w''_\alpha) \sqsupset (k'', w'')$.
– Let $r' := (w''_\alpha.\rho^1(\alpha), w''_\alpha.\rho^2(\alpha), V_{k''-1}[\![\alpha]\!]w''_\alpha.\rho) = \lfloor r \rfloor_{k''-1}$, so $(\alpha_1, \alpha_2, r') \in T^{-,\epsilon}_{k''-1}[\![\Omega]\!]w''_\alpha$.
– Instantiating the assumption with $(k''-2, \lfloor w''_\alpha \rfloor) \sqsupset (k''-1, w''_\alpha) \sqsupseteq (k, w)$ and $(\alpha_1, \alpha_2, r')$ yields $(k''-2, \lfloor w''_\alpha \rfloor, e_1[\alpha_1/\alpha], e_2[\alpha_2/\alpha]) \in E^{+,\epsilon}_n[\![\tau']\!]\rho, \alpha{\mapsto}r'$.
– Since $(\delta_1, \delta_2, \lfloor \rho \rfloor_{k''-1}) \in D^\circ_{k''-1}[\![\Delta]\!]w''_\alpha$ by Lemma 10.4, Lemma 10.37 yields $(\delta'_1, \delta'_2, \rho') \in D^\circ_{k''-1}[\![\Delta, \alpha]\!]w''_\alpha$ for $\rho' := \lfloor \rho \rfloor_{k''-1}, \alpha{\mapsto}r'$.
– Hence by Restriction (10.2), Irrelevance (10.3) and induction, $(k''-2, \lfloor w''_\alpha \rfloor, \delta'_1(\mathrm{Wr}^-_{\tau'})\ e_1[\alpha_1/\alpha], \delta'_2(\mathrm{Wr}^-_{\tau'})\ e_2[\alpha_2/\alpha]) \in E^{\circ,-}_n[\![\tau']\!]\rho'$.
– Consequently, there exists $(k''-j, w''') \sqsupseteq (k''-2, \lfloor w''_\alpha \rfloor)$ such that $w''.\sigma_2; \delta'_2(\mathrm{Wr}^-_{\tau'})\ e_2[\alpha_2/\alpha] \hookrightarrow^* w'''.\sigma_2; v'_2$ with $w'''.\sigma_1 = \sigma_1$ and $(k''-j, w''', v'_1, v'_2) \in V^{\circ,-}_n[\![\tau']\!]\rho'$.
– By Restriction (10.2) and Irrelevance (10.3), $(k''-j, w''', v'_1, v'_2) \in V^{\circ,-}_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.

• Case $\tau = \exists\alpha.\tau'$: $v_i = \mathsf{pack}\ \langle \tau_i, v'_i \rangle\ \mathsf{as}\ \tau'_i$
– To show: $(k, w, \delta_1(\lambda x{:}\tau.\ \mathsf{unpack}\ \langle \alpha, x' \rangle{=}x\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \mathrm{Wr}^-_{\tau'}\ x' \rangle\ \mathsf{as}\ \tau)\ v_1,$
$\delta_2(\lambda x{:}\tau.\ \mathsf{unpack}\ \langle \alpha, x' \rangle{=}x\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \mathrm{Wr}^-_{\tau'}\ x' \rangle\ \mathsf{as}\ \tau)\ v_2) \in E^{\circ,-}_n[\![\exists\alpha.\tau']\!]\rho$
– So suppose $w.\sigma_1; \delta_1(\lambda x{:}\tau.\ \mathsf{unpack}\ \langle \alpha, x' \rangle{=}x\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \mathrm{Wr}^-_{\tau'}\ x' \rangle\ \mathsf{as}\ \tau)\ v_1$ terminates in $2 + j' =: j < k$ steps:

$$\begin{array}{rl} & w.\sigma_1; \delta_1(\lambda x{:}\tau.\ \mathsf{unpack}\ \langle \alpha, x' \rangle{=}x\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \mathrm{Wr}^-_{\tau'}\ x' \rangle\ \mathsf{as}\ \tau)\ v_1 \\ \hookrightarrow^1 & w.\sigma_1; \mathsf{unpack}\ \langle \alpha, x' \rangle{=}v_1\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \delta_1(\mathrm{Wr}^-_{\tau'})\ x' \rangle\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^1 & w.\sigma_1; \mathsf{pack}\ \langle \tau_1, \delta'_1(\mathrm{Wr}^-_{\tau'})\ v'_1 \rangle\ \mathsf{as}\ \delta_1(\tau) \\ \hookrightarrow^{j'} & \sigma_1; \mathsf{pack}\ \langle \tau_1, v''_1 \rangle\ \mathsf{as}\ \delta_1(\tau) \end{array}$$

where $\delta'_1 := \delta_1, \alpha{\mapsto}\tau_1$
– Note that

$$\begin{array}{rl} & w.\sigma_2; \delta_2(\lambda x{:}\tau.\ \mathsf{unpack}\ \langle \alpha, x' \rangle{=}x\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \mathrm{Wr}^-_{\tau'}\ x' \rangle\ \mathsf{as}\ \tau)\ v_2 \\ \hookrightarrow^1 & w.\sigma_2; \mathsf{unpack}\ \langle \alpha, x' \rangle{=}v_2\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \delta_2(\mathrm{Wr}^-_{\tau'})\ x' \rangle\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow^1 & w.\sigma_2; \mathsf{pack}\ \langle \tau_2, \delta'_2(\mathrm{Wr}^-_{\tau'})\ v'_2 \rangle\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

where $\delta_2' := \delta_2, \alpha \mapsto \tau_2$
- By assumption we know $(k-2, \lfloor w \rfloor, v_1', v_2') \in V_n^{+,\epsilon}[\![\tau']\!]\rho, \alpha \mapsto r$ for some $r$ with $(\tau_1, \tau_2, r) \in T_k^{+,\epsilon}[\![\Omega]\!]w$.
- By Lemma 10.37, $(\tau_1, \tau_2, r) \in T_k^{\circ}[\![\Omega]\!]w$.
- Furthermore $(\delta_1, \delta_2, \lfloor \rho \rfloor_k) \in D_k^{\circ}[\![\Delta]\!]w$ by Lemma 10.4, so $(\delta_1', \delta_2', (\lfloor \rho \rfloor_k, \alpha \mapsto r)) \in D_k^{\circ}[\![\Delta, \alpha]\!]w$.
- Hence induction and Restriction (10.2) yields $(k-2, \lfloor w \rfloor, \delta_1'(\mathrm{Wr}_{\tau'}^-)\ v_1', \delta_2'(\mathrm{Wr}_{\tau'}^-)\ v_2') \in E_n^{\circ,-}[\![\tau']\!]\lfloor \rho \rfloor_k, \alpha \mapsto r$.
- Consequently, there exists $(k-j, w') \sqsupseteq (k-2, \lfloor w \rfloor)$ such that $w.\sigma_2; \mathsf{pack}\ \langle \tau_2, \delta_2'(\mathrm{Wr}_{\tau'}^-)\ v_2' \rangle\ \mathsf{as}\ \delta_2(\tau) \hookrightarrow^* w'.\sigma_2; \mathsf{pack}\ \langle \tau_2, v_2'' \rangle\ \mathsf{as}\ \delta_2(\tau)$ with $w'.\sigma_1 = \sigma_1$ and $(k-1-j, w', v_1'', v_2'') \in V_n^{\circ,-}[\![\tau']\!]\lfloor \rho \rfloor_k, \alpha \mapsto r$.
- For any $(k'', w'') \sqsupseteq (k-j, w')$, we get $(k'', w'', v_1'', v_2'') \in V_n^{\circ,-}[\![\tau']\!]\rho, \alpha \mapsto \lfloor r \rfloor_{k-j}$ by Closure Under World Extension (10.7), Restriction (10.2), and Irrelevance (10.3).
- Since $(\tau_1, \tau_2, \lfloor r \rfloor_{k-j}) \in T_{k-j}^{\circ,-}[\![\Omega]\!]w'$ by Lemmas 10.37 and 10.4, this implies $(k-j, w', \mathsf{pack}\ \langle \tau_1, v_1'' \rangle\ \mathsf{as}\ \delta_1(\tau), \mathsf{pack}\ \langle \tau_2, v_2'' \rangle\ \mathsf{as}\ \delta_2(\tau)) \in V_n^{\circ,-}[\![\exists \alpha.\tau']\!]\rho$.

• Case $\tau = \mu\alpha.\tau'$: symmetric to respective case of part 1

(b) Symmetric to (1b).

□

**Corollary 10.42.**

1. If $\vdash e_1 \precsim^{\circ,+} e_2 : \tau$, then $\vdash \mathrm{Wr}^+(e_1) \precsim^{-,\epsilon} \mathrm{Wr}^+(e_2) : \tau$.
2. If $\vdash e_1 \precsim^{+,\epsilon} e_2 : \tau$, then $\vdash \mathrm{Wr}^-(e_1) \precsim^{\circ,-} \mathrm{Wr}^-(e_2) : \tau$.

**Theorem 10.43** (Wrapping II)**.** Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^{\circ}[\![\Delta]\!]w_0$, $(k, w) \sqsupseteq (n, w_0)$, and $\Delta \vdash \tau$.

1. (a) If $(k, w, v_1, v_2) \in V_n^{\circ}[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}_\tau^+)\ v_1, \delta_2(\mathrm{Wr}_\tau^+)\ v_2) \in E_n[\![\tau]\!]\rho$.
   (b) If $(k, w, e_1, e_2) \in E_n^{\circ}[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}_\tau^+)\ e_1, \delta_2(\mathrm{Wr}_\tau^+)\ e_2) \in E_n[\![\tau]\!]\rho$.
2. (a) If $(k, w, v_1, v_2) \in V_n[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}_\tau^-)\ v_1, \delta_2(\mathrm{Wr}_\tau^-)\ v_2) \in E_n^{\circ}[\![\tau]\!]\rho$.
   (b) If $(k, w, e_1, e_2) \in E_n[\![\tau]\!]\rho$, then $(k, w, \delta_1(\mathrm{Wr}_\tau^-)\ e_1, \delta_2(\mathrm{Wr}_\tau^-)\ e_2) \in E_n^{\circ}[\![\tau]\!]\rho$.

*Proof:* Follows immediately from Wrapping I (10.41) and Chain (10.38). □

**Corollary 10.44.**

1. If $\vdash e_1 \precsim^{\circ} e_2 : \tau$, then $\vdash \mathrm{Wr}^+(e_1) \precsim \mathrm{Wr}^+(e_2) : \tau$.
2. If $\vdash e_1 \precsim e_2 : \tau$, then $\vdash \mathrm{Wr}^-(e_1) \precsim^{\circ} \mathrm{Wr}^-(e_2) : \tau$.

## 10.8 Examples

### 10.8.1 Semaphore ADT, ∃-version

Given

$$\begin{aligned}\tau &:= \alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool})\\ v_1 &:= \langle \mathsf{true}, \lambda x{:}\alpha.\neg x, \lambda x{:}\alpha.x\rangle\\ v_2 &:= \langle 1, \lambda x{:}\alpha.1 - x, \lambda x{:}\alpha.x \neq 0\rangle\\ e_1 &:= \mathsf{new}\,\alpha{\approx}\mathsf{bool}\;\mathsf{in}\;\mathsf{pack}\,\langle\alpha, v_1\rangle\;\mathsf{as}\;\exists\alpha.\tau\\ e_2 &:= \mathsf{new}\,\alpha{\approx}\mathsf{int}\;\mathsf{in}\;\mathsf{pack}\,\langle\alpha, v_2\rangle\;\mathsf{as}\;\exists\alpha.\tau\end{aligned}$$

we show $\epsilon;\epsilon \vdash e_1 \precsim e_2 : \exists\alpha.\tau$; the other direction is proven analogously.

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\epsilon]\!]w_0$, and $(k, w, \gamma_1, \gamma_2) \in G_n[\![\epsilon]\!]\rho$ with $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(e_1), \delta_2\gamma_2(e_2)) \in E_n[\![\exists\alpha.\tau]\!]\rho$, i.e., $(k, w, e_1, e_2) \in E_n[\![\exists\alpha.\tau]\!]\rho$
- We know:
  - $w.\sigma_1; e_1 \hookrightarrow^1 w.\sigma_1, \alpha_1{\approx}\mathsf{bool}; \mathsf{pack}\,\langle\alpha_1, v_1[\alpha_1/\alpha]\rangle\;\mathsf{as}\;\exists\alpha.\tau$
  - $w.\sigma_2; e_2 \hookrightarrow^* w.\sigma_1, \alpha_2{\approx}\mathsf{int}; \mathsf{pack}\,\langle\alpha_2, v_2[\alpha_2/\alpha]\rangle\;\mathsf{as}\;\exists\alpha.\tau$
- Let

  $$\begin{aligned}R &:= \{(k', w', v_a, v_b) \in \mathrm{Atom}_{k-1} \mid (v_a, v_b) = (\mathsf{true}, 1) \vee (v_a, v_b) = (\mathsf{false}, 0)\}\\ w_\alpha &:= \lfloor w\rfloor_{k-1} \uplus (\alpha_1{\approx}\mathsf{bool}, \alpha_2{\approx}\mathsf{int}, \alpha{\mapsto}(\alpha_1, \alpha_2), \alpha{\mapsto}R)\end{aligned}$$

  such that $(k-1, w_\alpha) \sqsupset (k, w)$.
- We claim $(k-1, w_\alpha, \mathsf{pack}\,\langle\alpha_1, v_1[\alpha_1/\alpha]\rangle\;\mathsf{as}\;\exists\alpha.\tau, \mathsf{pack}\,\langle\alpha_2, v_2[\alpha_2/\alpha]\rangle\;\mathsf{as}\;\exists\alpha.\tau) \in V_n[\![\exists\alpha.\tau]\!]\rho$.
- Note that $\alpha_1 = w^1_\alpha(\alpha)$ and $\alpha_2 = w^2_\alpha(\alpha)$.
- So suppose $(k', w') \sqsupset (k-1, w_\alpha)$.
- To show: $(k', w', v_1[\alpha_1/\alpha], v_2[\alpha_2/\alpha]) \in V_n[\![\alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool})]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$, which decomposes into three parts:
  1. $(k', w', \mathsf{true}, 1) \in V_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$, which holds since $V_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho = V_{k-1}[\![\alpha]\!]w_\alpha.\rho \cap \mathrm{Atom}_n = w_\alpha.\rho(\alpha).R \cap \mathrm{Atom}_{k-1} = R \ni (k', w', \mathsf{true}, 1)$.
  2. $(k', w', \lambda x{:}\alpha_1.\neg x, \lambda x{:}\alpha_2.1 - x) \in V_n[\![\alpha \to \alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$
     - Suppose $(k'', w'', v_3, v_4) \in V_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$ for $(k'', w'') \sqsupseteq (k', w')$, i.e., $(k'', w'', v_3, v_4) \in R$, so either $(v_3, v_4) = (\mathsf{true}, 1)$ or $(v_3, v_4) = (\mathsf{false}, 0)$.
     - To show: $(k'', w'', \neg v_3, 1 - v_4) \in E_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$.
     - If $(v_3, v_4) = (\mathsf{true}, 1)$, then $w''.\sigma_1; \neg v_3 \hookrightarrow^1 w''.\sigma_1; \mathsf{false}$ and $w''.\sigma_2; 1 - v_4 \hookrightarrow^* w''.\sigma_2; 0$.
     - Note that $(k'' - 1, \lfloor w''\rfloor, \mathsf{false}, 0) \in R = V_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$.
     - Similarly for $(v_3, v_4) = (\mathsf{false}, 0)$.
  3. $(k', w', \lambda x{:}\alpha_1.x, \lambda x{:}\alpha_2.x \neq 0) \in V_n[\![\alpha \to \mathsf{bool}]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$

– Suppose $(k'', w'', v_3, v_4) \in V_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$ for $(k'', w'') \sqsupseteq (k', w')$, i.e., $(k'', w'', v_3, v_4) \in R$, so either $(v_3, v_4) = (\mathsf{true}, 1)$ or $(v_3, v_4) = (\mathsf{false}, 0)$.
– To show: $(k'', w'', v_3, v_4 \neq 0) \in E_n[\![\alpha]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$
– If $(v_3, v_4) = (\mathsf{true}, 1)$, then $w''.\sigma_1; v_3 \hookrightarrow^0 w''.\sigma_1; \mathsf{true}$ and $w''.\sigma_2; v_4 \neq 0 \hookrightarrow^* w''.\sigma_2; \mathsf{true}$.
– Note that $(k'' - 1, \lfloor w'' \rfloor, \mathsf{true}, \mathsf{true}) \in V_n[\![\mathsf{bool}]\!]\rho, \alpha{\mapsto}V_{k-1}[\![\alpha]\!]w_\alpha.\rho$.
– Similarly for $(v_3, v_4) = (\mathsf{false}, 0)$.

### 10.8.2 Semaphore ADT, ∀-version

Given

$$\begin{aligned}
\tau &:= \alpha \times (\alpha \to \alpha) \times (\alpha \to \mathsf{bool}) \\
v_1 &:= \langle \mathsf{true}, \lambda x{:}\mathsf{bool}.\neg x, \lambda x{:}\mathsf{bool}.x \rangle \\
v_2 &:= \langle 1, \lambda x{:}\mathsf{int}.x - 1, \lambda x{:}\mathsf{int}.x \neq 0 \rangle \\
e_1 &:= \lambda x{:}(\forall\alpha.\tau \to \mathsf{bool}).(\mathsf{new}\ \alpha{\approx}\mathsf{bool}\ \mathsf{in}\ x\ \alpha)\ v_1 \\
e_2 &:= \lambda x{:}(\forall\alpha.\tau \to \mathsf{bool}).(\mathsf{new}\ \alpha{\approx}\mathsf{int}\ \mathsf{in}\ x\ \alpha)\ v_2
\end{aligned}$$

we show $\epsilon; \epsilon \vdash e_1 \precsim e_2 : (\forall\alpha.\tau \to \mathsf{bool}) \to \mathsf{bool}$; the other direction is proven analogously.

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\epsilon]\!]w_0$, and $(k, w, \gamma_1, \gamma_2) \in G_n[\![\epsilon]\!]\rho$ with $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, w.\eta^1\delta\gamma_1(e_1), w.\eta^2\delta\gamma_2(e_2)) \in E_n[\![(\forall\alpha.\tau \to \mathsf{bool}) \to \mathsf{bool}]\!]\rho$, i.e., $(k, w, e_1, e_2) \in V_n[\![(\forall\alpha.\tau \to \mathsf{bool}) \to \mathsf{bool}]\!]\rho$.
- So suppose $(k', w', \Lambda\alpha.e_3, \Lambda\alpha.e_4) \in V_n[\![\forall\alpha.\tau \to \mathsf{bool}]\!]\rho$ where $(k', w') \sqsupseteq (k, w)$.
- To show: $(k', w', (\mathsf{new}\ \alpha{\approx}\mathsf{bool}\ \mathsf{in}\ (\Lambda\alpha.e_3)\ \alpha)\ v_1, (\mathsf{new}\ \alpha{\approx}\mathsf{int}\ \mathsf{in}\ (\Lambda\alpha.e_4)\ \alpha)\ v_2) \in E_n[\![\mathsf{bool}]\!]\rho$.
- Suppose $w'.\sigma_1; (\mathsf{new}\ \alpha{\approx}\mathsf{bool}\ \mathsf{in}\ (\Lambda\alpha.e_3)\ \alpha)\ v_1$ terminates in $2 + j_1 + 1 + j_2 =: j < k'$ steps:

$$\begin{array}{rl}
 & w'.\sigma_1; (\mathsf{new}\ \alpha{\approx}\mathsf{bool}\ \mathsf{in}\ (\Lambda\alpha.e_3)\ \alpha)\ v_1 \\
\hookrightarrow^2 & w'.\sigma_1, \alpha_1{\approx}\mathsf{bool}; e_3[\alpha_1/\alpha]\ v_1 \\
\hookrightarrow^{j_1} & \sigma_1'; (\lambda x{:}\tau_3.e_3')\ v_1 \\
\hookrightarrow^1 & \sigma_1'; e_3'[v_1/x] \\
\hookrightarrow^{j_2} & \sigma_1; v_3
\end{array}$$

- We need to show the existence of $(k' - j, w'''') \sqsupset (k', w')$ such that $w'.\sigma_2; (\mathsf{new}\ \alpha{\approx}\mathsf{bool}\ \mathsf{in}\ (\Lambda\alpha.e_4)\ \alpha)\ v_2 \hookrightarrow^* w''''.\sigma_2; v_4$ with $w''''.\sigma_1 = \sigma_1$ and $(k' - j, w'''', v_3, v_4) \in V_n[\![\mathsf{bool}]\!]\rho$.
- Let

$$\begin{aligned}
R &:= \{(k'', w'', v_a, v_b) \in \mathrm{Atom}_{k'-1} \mid (v_a, v_b) = (\mathsf{true}, 1) \vee (v_a, v_b) = (\mathsf{false}, 0)\} \\
w'_\alpha &:= \lfloor w' \rfloor_{k'-1} \uplus (\alpha_1{\approx}\mathsf{bool}, \alpha_2{\approx}\mathsf{int}, \alpha{\mapsto}(\alpha_1, \alpha_2), \alpha{\mapsto}R) \\
w'' &:= \lfloor w'_\alpha \rfloor_{k'-2}
\end{aligned}$$

- Since $(k' - 2, w'') \sqsupset (k' - 1, w'_\alpha) \sqsupset (k', w')$ and $\alpha \in \mathrm{dom}(w'_\alpha.\rho)$, we know $(k' - 2, w'', e_3[\alpha_1/\alpha], e_4[\alpha_2/\alpha]) \in E_n[\![\tau \to \mathsf{bool}]\!]\rho, \alpha{\mapsto}V_{k'-1}[\![\alpha]\!]w'_\alpha.\rho$.

- Consequently, there exists $(k' - 2 - j_1, w''') \sqsupseteq (k' - 2, w'')$ such that $w'.\sigma_2, \alpha_2 \approx \mathsf{int}; e_4[\alpha_2/\alpha]\ v_2 \hookrightarrow^* w'''.\sigma_2; (\lambda x{:}\tau_4.e_4')\ v_2$ with $w'''.\sigma_1 = \sigma_1'$ and $(k' - 2 - j_1, w''', \lambda x{:}\tau_3.e_3', \lambda x{:}\tau_4.e_4') \in V_n[\![\tau \to \mathsf{bool}]\!]\rho, \alpha \mapsto V_{k'-1}[\![\alpha]\!]w'_\alpha.\rho$.
- If we can show $(k' - 2 - j_1 - 1, \lfloor w''' \rfloor, v_1, v_2) \in V_n[\![\tau]\!]\rho, \alpha \mapsto V_{k'-1}[\![\alpha]\!]w'_\alpha.\rho$, then the latter yields $(k' - 2 - j_1 - 1, \lfloor w''' \rfloor, e_3'[v_1/x], e_4'[v_2/x]) \in E_n[\![\mathsf{bool}]\!]\rho, \alpha \mapsto V_{k'-1}[\![\alpha]\!]w'_\alpha.\rho$, which in turns implies the existence of the wanted $(k' - j, w'''')$.
- Showing $(k' - 2 - j_1 - 1, \lfloor w''' \rfloor, v_1, v_2) \in V_n[\![\tau]\!]\rho, \alpha \mapsto V_{k'-1}[\![\alpha]\!]w'_\alpha.\rho$ decomposes and is shown as in the previous example.

### 10.8.3 Benign Effects: Double Application

Given

$$\begin{aligned} v_1 &:= \lambda x{:}(\mathsf{unit} \to \tau).\ \mathsf{let}\ x' = x\ ()\ \mathsf{in}\ x\ () \\ v_2 &:= \lambda x{:}(\mathsf{unit} \to \tau).x\ () \end{aligned}$$

we show $\epsilon; \epsilon \vdash v_1 \precsim v_2 : (\mathsf{unit} \to \tau) \to \tau$:

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n[\![\Gamma]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(v_1), \delta_2\gamma_2(v_2)) \in E_n[\![(\mathsf{unit} \to \tau) \to \tau]\!]\rho$, i.e., $(k, w, v_1, v_2) \in V_n[\![(\mathsf{unit} \to \tau) \to \tau]\!]\rho$
- So suppose $(k', w', \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$ where $(k', w') \sqsupseteq (k, w)$.
- To show: $(k', w', \mathsf{let}\ x' = (\lambda x{:}\tau_1.e_1)\ ()\ \mathsf{in}\ (\lambda x{:}\tau_1.e_1)\ (), (\lambda x{:}\tau_2.e_2)\ ()) \in E_n[\![\tau]\!]\rho$
- Suppose $w'.\sigma_1; \mathsf{let}\ x' = (\lambda x{:}\tau_1.e_1)\ ()\ \mathsf{in}\ (\lambda x{:}\tau_1.e_1)\ ()$ terminates in $3 + 2j' =: j < k'$ steps:

$$\begin{array}{ll} & w'.\sigma_1; \mathsf{let}\ x' = (\lambda x{:}\tau_1.e_1)\ ()\ \mathsf{in}\ (\lambda x{:}\tau_1.e_1)\ () \\ \hookrightarrow^1 & w'.\sigma_1; \mathsf{let}\ x' = e_1[()/x]\ \mathsf{in}\ (\lambda x{:}\tau_1.e_1)\ () \\ \hookrightarrow^{j'} & \sigma_1'; \mathsf{let}\ x' = v_1'\ \mathsf{in}\ (\lambda x{:}\tau_1.e_1)\ () \\ \hookrightarrow^1 & \sigma_1'; (\lambda x{:}\tau_1.e_1)\ () \\ \hookrightarrow^1 & \sigma_1'; e_1[()/x] \\ \hookrightarrow^{j'} & \sigma_1; v_1'' \end{array}$$

- Let $w_1' := (\sigma_1', w'.\sigma_2, w'.\eta, w'.\rho)$, so $(k', w_1') \sqsupseteq (k', w')$.
- Since we have $(k' - j' - 2, \lfloor w_1' \rfloor, \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$ by Closure Under World Extension and $(k' - j' - 3, \lfloor w_1' \rfloor, (), ()) \in V_n[\![\mathsf{unit}]\!]\rho$, we get $(k' - j' - 3, \lfloor w_1' \rfloor, e_1[()/x], e_2[()/x]) \in E_n[\![\tau]\!]\rho$.
- Consequently, there exists $(k' - j, w'') \sqsupseteq (k' - j' - 3, \lfloor w_1' \rfloor)$ such that $w'.\sigma_2; e_2[()/x] \hookrightarrow^* w''.\sigma_2; v_2'$ with $w''.\sigma_1 = \sigma_1$ and $(k' - j, w'', v_1'', v_2') \in V_n[\![\tau]\!]\rho$.

We show $\epsilon; \epsilon \vdash v_2 \precsim v_1 : (\mathsf{unit} \to \tau) \to \tau$:

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n[\![\Gamma]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(v_2), \delta_2\gamma_2(v_1)) \in E_n[\![(\mathsf{unit} \to \tau) \to \tau]\!]\rho$, i.e., $(k, w, v_2, v_1) \in V_n[\![(\mathsf{unit} \to \tau) \to \tau]\!]\rho$

- So suppose $(k', w', \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$ where $(k', w') \sqsupseteq (k, w)$.
- To show: $(k', w', (\lambda x{:}\tau_1.e_1)\ (), \mathsf{let}\ x' = (\lambda x{:}\tau_2.e_2)\ ()\ \mathsf{in}\ (\lambda x{:}\tau_2.e_2)\ ()) \in E_n[\![\tau]\!]\rho$
- Suppose $w'.\sigma_1; (\lambda x{:}\tau_1.e_1)\ ()$ terminates in $1 + j' =: j < k'$ steps:

$$\begin{array}{rl} & w'.\sigma_1; (\lambda x{:}\tau_1.e_1)\ () \\ \hookrightarrow^1 & w'.\sigma_1; e_1[()/x] \\ \hookrightarrow^{j'} & \sigma_1; v_1' \end{array}$$

- Instantiating $(k', w', \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$ with $(k'-1, \lfloor w' \rfloor, (), ()) \in V_n[\![\mathsf{unit}]\!]\rho$ yields $(k'-1, \lfloor w' \rfloor, e_1[()/x], e_2[()/x]) \in E_n[\![\tau]\!]\rho$.
- Consequently there exists $(k'-j, w'') \sqsupseteq (k'-1, \lfloor w' \rfloor)$ such that $w'.\sigma_2; \mathsf{let}\ x' = (\lambda x{:}\tau_2.e_2)\ ()\ \mathsf{in}\ (\lambda x{:}\tau_2.e_2)\ () \hookrightarrow^* w''.\sigma_2; \mathsf{let}\ x' = v_2'\ \mathsf{in}\ (\lambda x{:}\tau_2.e_2)\ ()$.
- Let $w_2' = (w'.\sigma_1, w''.\sigma_2, w'.\eta, w'.\rho)$, so $(k', w_2') \sqsupseteq (k', w')$.
- By Closure Under World Extension (10.7) we have $(k', w_2', \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$.
- Instantiating this with $(k'-1, \lfloor w_2' \rfloor, (), ()) \in V_n[\![\mathsf{unit}]\!]\rho$ yields $(k'-1, \lfloor w_2' \rfloor, e_1[()/x], e_2[()/x]) \in E_n[\![\tau]\!]\rho$.
- Consequently there exists $(k'-j, w''') \sqsupseteq (k'-1, \lfloor w_2' \rfloor)$ such that $w''.\sigma_2; e_2[()/x] \hookrightarrow^* w'''.\sigma_2; v_2''$ with $w'''.\sigma_1 = \sigma_1$ and $(k'-j, w''', v_1', v_2'') \in V_n[\![\tau]\!]\rho$.
- Note that

$$\begin{array}{rl} & w'.\sigma_2; \mathsf{let}\ x' = (\lambda x{:}\tau_2.e_2)\ ()\ \mathsf{in}\ (\lambda x{:}\tau_2.e_2)\ () \\ \hookrightarrow^1 & w'.\sigma_2; \mathsf{let}\ x' = e_2[()/x]\ \mathsf{in}\ (\lambda x{:}\tau_2.e_2)\ () \\ \hookrightarrow^* & w''.\sigma_2; \mathsf{let}\ x' = v_2'\ \mathsf{in}\ (\lambda x{:}\tau_2.e_2)\ () \\ \hookrightarrow^1 & w''.\sigma_2; (\lambda x{:}\tau_2.e_2)\ () \\ \hookrightarrow^1 & w''.\sigma_2; e_2[()/x] \\ \hookrightarrow^* & w'''.\sigma_2; v_2'' \end{array}$$

### 10.8.4 Benign Effects: Order Independence

Given

$$\begin{array}{l} v_1 := \lambda x_f{:}(\mathsf{unit} \to \tau).\lambda x_g{:}(\mathsf{unit} \to \tau').\ \mathsf{let}\, x_g' = x_g\ ()\ \mathsf{in}\ \langle x_f\ (), x_g' \rangle \\ v_2 := \lambda x_f{:}(\mathsf{unit} \to \tau).\lambda x_g{:}(\mathsf{unit} \to \tau').\ \langle x_f\ (), x_g\ () \rangle \end{array}$$

we show $\epsilon; \epsilon \vdash v_1 \precsim v_2 : (\mathsf{unit} \to \tau) \to (\mathsf{unit} \to \tau') \to (\tau \times \tau')$.

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n[\![\Gamma]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(v_1), \delta_2\gamma_2(v_2)) \in E_n[\![(\mathsf{unit} \to \tau) \to (\mathsf{unit} \to \tau') \to (\tau \times \tau')]\!]\rho$, i.e., $(k, w, v_1, v_2) \in V_n[\![(\mathsf{unit} \to \tau) \to (\mathsf{unit} \to \tau') \to (\tau \times \tau')]\!]\rho$
- So suppose $(k', w', \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$ where $(k', w') \sqsupseteq (k, w)$.
- To show: $(k', w', \lambda x_g{:}(\mathsf{unit} \to \tau').\ \mathsf{let}\, x_g' = x_g\ ()\ \mathsf{in}\ \langle (\lambda x{:}\tau_1.e_1)\ (), x_g' \rangle$, $\lambda x_g{:}(\mathsf{unit} \to \tau').\ \langle (\lambda x{:}\tau_1.e_1)\ (), x_g\ () \rangle) \in V_n[\![(\mathsf{unit} \to \tau') \to (\tau \times \tau')]\!]\rho$

- So suppose $(k'', w'', \lambda x{:}\tau_3.e_3, \lambda x{:}\tau_4.e_4) \in V_n[\![\mathsf{unit} \to \tau']\!]\rho$ where $(k'', w'') \sqsupseteq (k', w')$.
- To show: $(k'', w'', \mathsf{let}\, x'_g = (\lambda x{:}\tau_3.e_3)\ ()\ \mathsf{in}\ \langle(\lambda x{:}\tau_1.e_1)\ (), x'_g\rangle, \langle(\lambda x{:}\tau_1.e_1)\ (), (\lambda x{:}\tau_4.e_4)\ ()\rangle) \in V_n[\![\tau \times \tau']\!]\rho$
- Suppose $w''.\sigma_1; \mathsf{let}\, x'_g = (\lambda x{:}\tau_3.e_3)\ ()\ \mathsf{in}\ \langle(\lambda x{:}\tau_1.e_1)\ (), x'_g\rangle$ terminates in $1+j_1+1+1+j_2 =: j < k''$ steps:

$$\begin{array}{rl} & w''.\sigma_1; \mathsf{let}\, x'_g = (\lambda x{:}\tau_3.e_3)\ ()\ \mathsf{in}\ \langle(\lambda x{:}\tau_1.e_1)\ (), x'_g\rangle \\ \hookrightarrow^1 & w''.\sigma_1; \mathsf{let}\, x'_g = e_3[()/x]\ \mathsf{in}\ \langle(\lambda x{:}\tau_1.e_1)\ (), x'_g\rangle \\ \hookrightarrow^{j_1} & \sigma'_1; \mathsf{let}\, x'_g = v'_3\ \mathsf{in}\ \langle(\lambda x{:}\tau_1.e_1)\ (), x'_g\rangle \\ \hookrightarrow^1 & \sigma'_1; \langle(\lambda x{:}\tau_1.e_1)\ (), v'_3\rangle \\ \hookrightarrow^1 & \sigma'_1; \langle e_1[()/x], v'_3\rangle \\ \hookrightarrow^{j_2} & \sigma_1; \langle v'_1, v'_3\rangle \end{array}$$

- Let $(k'_2, w'_2) := (k'' - 1 - j_1 - 1 - 1, (\sigma'_1, w''.\sigma_2, w''.\eta, \lfloor w''.\rho \rfloor_{k''-1j_1-1-1}))$, so $(k'_2, w'_2) \sqsupseteq (k'', w'')$.
- Since we have $(k'_2, w'_2, \lambda x{:}\tau_1.e_1, \lambda x{:}\tau_2.e_2) \in V_n[\![\mathsf{unit} \to \tau]\!]\rho$ by Closure Under World Extension and $(k'_2 - 1, \lfloor w'_2 \rfloor, (), ()) \in V_n[\![\mathsf{unit}]\!]\rho$, we get $(k'_2 - 1, \lfloor w'_2 \rfloor, e_1[()/x], e_2[()/x]) \in E_n[\![\tau]\!]\rho$.
- Note that $w'_2.\sigma_1 = \sigma'_1$.
- Consequently, there exists $(k'' - j, w''_2) \sqsupseteq (k'_2 - 1, \lfloor w'_2 \rfloor)$ such that $w''.\sigma_2; e_2[()/x] \hookrightarrow^* w''_2.\sigma_2; v'_2$ with $w''_2.\sigma_1 = \sigma_1$ and $(k'' - j, w''_2, v'_1, v'_2) \in V_n[\![\tau]\!]\rho$.
- Let $(k'_1, w'_1) := (k'' - 1, (w''.\sigma_1, w''_2.\sigma_2, w''.\eta, \lfloor w''.\rho \rfloor_{k''-1}))$, so $(k'_1, w'_1) \sqsupseteq (k'', w'')$.
- Since we have $(k'_1, w'_1, \lambda x{:}\tau_3.e_3, \lambda x{:}\tau_4.e_4) \in V_n[\![\mathsf{unit} \to \tau']\!]\rho$ by Closure Under World Extension and $(k'_1 - 1, \lfloor w'_1 \rfloor, (), ()) \in V_n[\![\mathsf{unit}]\!]\rho$, we get $(k'_1 - 1, \lfloor w'_1 \rfloor, e_3[()/x], e_4[()/x]) \in E_n[\![\tau']\!]\rho$.
- Note that $w'_1.\sigma_1 = w''.\sigma_1$.
- Consequently, there exists $(k'' - 1 - j_1, w''_1) \sqsupseteq (k'_1 - 1, \lfloor w'_1 \rfloor)$ such that $w''_2.\sigma_2; e_4[()/x] \hookrightarrow^* w''_1.\sigma_2; v'_4$ with $w''_1.\sigma_1 = \sigma'_1$ and $(k'' - 1 - j_1, w''_1, v'_3, v'_4) \in V_n[\![\tau']\!]\rho$.
- W.l.o.g. $(\mathrm{dom}(w''_1.\eta) \setminus \mathrm{dom}(w'_1.\eta)) \cap (\mathrm{dom}(w''_2.\eta) \setminus \mathrm{dom}(w'_2.\eta)) = \emptyset$, so $w''_1.\eta \cup w''_2.\eta$ and $w''_1.\rho \cup w''_2.\rho$ are well-defined.
- Let $w_3 := (w''_2.\sigma_1, w''_1.\sigma_2, w''_1.\eta \cup w''_2.\eta, w''_1.\rho \cup w''_2.\rho)$.
- It remains to show the injectivity of $w_3.\eta^i$.
- Note that $\mathrm{rng}(w''_1.\eta^i) \setminus \mathrm{rng}(w'_1.\eta^i) \subseteq \mathrm{dom}(w''_1.\sigma_i) \setminus \mathrm{dom}(w'_1.\sigma_i)$ by definition of world extension.
- Similarly, $\mathrm{rng}(w''_2.\eta^i) \setminus \mathrm{rng}(w'_2.\eta^i) \subseteq \mathrm{dom}(w''_2.\sigma_i) \setminus \mathrm{dom}(w'_2.\sigma_i)$ by definition of world extension.
- Suppose $\alpha, \alpha' \in \mathrm{dom}(w_3.\eta)$.
- Case $\alpha, \alpha' \in \mathrm{dom}(w''.\eta)$: Trivial.
- Case $\alpha \in \mathrm{dom}(w''.\eta)$ and $\alpha' \in \mathrm{dom}(w''_1.\eta) \setminus \mathrm{dom}(w''.\eta)$:
  - Then we know $w_3.\eta^i(\alpha) \in \mathrm{dom}(w''.\sigma_i)$ and $w_3.\eta^i(\alpha') \in \mathrm{dom}(w''_1.\sigma_i) \setminus \mathrm{dom}(w'_1.\sigma_i)$.

- Since $w_1'.\sigma_i \supseteq w''.\sigma_i$, we have $w_3.\eta^i(\alpha) \neq w_3.\eta^i(\alpha')$.

- Case $\alpha \in \mathrm{dom}(w_1''.\eta) \setminus \mathrm{dom}(w''.\eta)$ and $\alpha' \in \mathrm{dom}(w_2''.\eta) \setminus \mathrm{dom}(w''.\eta)$:
  - Then we know $w_3.\eta^i(\alpha) \in \mathrm{dom}(w_1''.\sigma_i) \setminus \mathrm{dom}(w_1'.\sigma_i)$ and $w_3.\eta^i(\alpha') \in \mathrm{dom}(w_2''.\sigma_i) \setminus \mathrm{dom}(w_2'.\sigma_i)$.
  - For $i = 1$ this means $w_3.\eta^1(\alpha) \in \mathrm{dom}(w_1''.\sigma_1) = \mathrm{dom}(\sigma_1') = \mathrm{dom}(w_2'.\sigma_1)$, so it cannot equal $w_3.\eta^1(\alpha')$.
  - For $i = 2$ this means $w_3.\eta^2(\alpha) \in \mathrm{dom}(w_1''.\sigma_2) \setminus \mathrm{dom}(w_2''.\sigma_2)$, so it cannot equal $w_3.\eta^2(\alpha')$.

### 10.8.5 Syntactically Non-Parametric Functor

Given

$$\begin{aligned}
\tau_\alpha &:= \exists\alpha'.(\alpha \times \alpha \to \alpha') \times (\alpha' \to \alpha) \times (\alpha' \to \alpha)\\
b2i &:= \lambda x{:}\mathsf{bool}.\ \mathsf{if}\ x\ \mathsf{then}\ 1\ \mathsf{else}\ 0\\
v_1' &:= \mathsf{pack}\ \langle \alpha \times \alpha, \langle \lambda x{:}(\alpha \times \alpha).x, \lambda x{:}(\alpha \times \alpha).(x.1), \lambda x{:}(\alpha \times \alpha).(x.2)\rangle\rangle\ \mathsf{as}\ \tau_\alpha\\
v_2' &:= \mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}(\mathsf{bool} \times \mathsf{bool}).b2i\ (x.1) + 2 * (b2i\ (x.2)),\\
&\qquad\qquad \lambda x{:}\mathsf{int}.(x\ \mathsf{mod}\ 2) \neq 0, \lambda x{:}\mathsf{int}.(x\ \mathsf{div}\ 2) \neq 0\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{bool}}\\
v_1 &:= \Lambda\alpha.v_1'\\
v_2 &:= \Lambda\alpha.\ \mathsf{cast}\ \tau_{\mathsf{bool}}\ \tau_\alpha\ v_2'\ (v_1\ \alpha)
\end{aligned}$$

we show $\epsilon;\epsilon \vdash v_1 \precsim^\circ v_2 : \forall\alpha.\tau_\alpha$, which implies $\epsilon;\epsilon \vdash \mathrm{Wr}^+_{\forall\alpha.\tau_\alpha}\ v_1 \precsim \mathrm{Wr}^+_{\forall\alpha.\tau_\alpha}\ v_2 : \forall\alpha.\tau_\alpha$ by Corollary 10.44.

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^\circ[\![\epsilon]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^\circ[\![\epsilon]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(v_1), \delta_2\gamma_2(v_2)) \in E_n^\circ[\![\forall\alpha.\tau_\alpha]\!]\rho$, i.e., $(k, w, v_1, v_2) \in V_n^\circ[\![\forall\alpha.\tau_\alpha]\!]\emptyset$
- So suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}^\circ[\![\Omega]\!]w'$.
- To show: $(k'', w'', v_1'[\tau_1/\alpha], \mathsf{cast}\ \tau_{\mathsf{bool}}\ \tau_{\tau_2}\ v_2'\ (v_1\ \tau_2)) \in E_n^\circ[\![\tau_\alpha]\!]\alpha{\mapsto}r$
- Case $\tau_2 \neq \mathsf{bool}$:
  - Then $\tau_{\tau_2} \neq \tau_{\mathsf{bool}}$ and hence $w''.\sigma_2; \mathsf{cast}\ \tau_{\mathsf{bool}}\ \tau_{\tau_2}\ v_2'\ (v_1\ \tau_2) \hookrightarrow^2 v_1'[\tau_2/\alpha]$.
  - Hence it suffices to show $(k'', w'', v_1'[\tau_1/\alpha], v_1'[\tau_2/\alpha]) \in E_n^\circ[\![\tau_\alpha]\!]\alpha{\mapsto}r$ by Inclusion 10.5.
  - Since $v_1$ is cast-free, we know $(k, w, v_1, v_1) \in E_n^\circ[\![\forall\alpha.\tau_\alpha]\!]\emptyset$ by the Fundamental Property (10.32).
  - Assuming $k \neq 0$ (otherwise we are done), this implies $(k, w, v_1, v_1) \in V_n^\circ[\![\forall\alpha.\tau_\alpha]\!]\emptyset$.
  - Instantiating that yields the claim.
- Case $\tau_2 = \mathsf{bool}$:
  - Then $\tau_{\tau_2} = \tau_{\mathsf{bool}}$ and hence $w''.\sigma_1; \mathsf{cast}\ \tau_{\mathsf{bool}}\ \tau_{\tau_2}\ v_2'\ (v_1\ \tau_2) \hookrightarrow^1 w''.\sigma_1; v_2'$.
  - Let
$$\begin{aligned}
R' := {} & \{(k, w, \langle v, v'\rangle, 0) \mid (k, w, v, \mathsf{false}), (k, w, v', \mathsf{false}) \in r.R\}\\
\cup\ & \{(k, w, \langle v, v'\rangle, 1) \mid (k, w, v, \mathsf{true}), (k, w, v', \mathsf{false}) \in r.R\}\\
\cup\ & \{(k, w, \langle v, v'\rangle, 2) \mid (k, w, v, \mathsf{false}), (k, w, v', \mathsf{true}) \in r.R\}\\
\cup\ & \{(k, w, \langle v, v'\rangle, 3) \mid (k, w, v, \mathsf{true}), (k, w, v', \mathsf{true}) \in r.R\}
\end{aligned}$$
  and $r' := (w'.\rho^1(\tau_1 \times \tau_1), \mathsf{int}, R')$, so $(\tau_1 \times \tau_1, \mathsf{int}, r) \in T_{k'}^\circ[\![\Omega]\!]w'$. Also, $\rho := \alpha{\mapsto}r, \alpha'{\mapsto}r'$.

– By Closure Under World Extension (10.7) it remains to show the following three points:
  1. $(k'', w'', \lambda x{:}(\alpha\times\alpha).x, \lambda x{:}(\mathsf{bool}\times\mathsf{bool}).b2i\ (x.1)+2*(b2i\ (x.2))) \in V_n^\circ[\![\alpha\times\alpha \to \alpha']\!]\rho$
     * Suppose $(k''', w''') \sqsupseteq (k'', w'')$ and $(k''', w''', v_3, v_4) \in V_n^\circ[\![\alpha \times \alpha]\!]\rho$.
     * Then $v_3 = \langle v_3', v_3''\rangle$ and $v_4 = \langle v_4', v_4''\rangle$ with $(k''', w''', v_3', v_4') \in r.R$ and $(k''', w''', v_3'', v_4'') \in r.R$.
     * To show: $(k''', w''', v_3, b2i\ (v_4.1) + 2 * (b2i\ (v_4.2))) \in E_n^\circ[\![\alpha']\!]\rho$.
     * Case $v_4' = v_4'' = \mathsf{false}$:
        · Then $w'''.\sigma_2; b2i\ (v_4.1) + 2 * (b2i\ (v_4.2)) \hookrightarrow^* w'''.\sigma_2; 0$.
        · To show: $(k''', w''', v_3, 0) \in V_n^\circ[\![\alpha']\!]\rho = R'$
        · This holds by construction of $R'$.
     * Other cases analogously.
  2. $(k'', w'', \lambda x{:}(\alpha \times \alpha).(x.1), \lambda x{:}\mathsf{int}.(x\ \mathsf{mod}\ 2) \neq 0) \in V_n^\circ[\![\alpha' \to \alpha]\!]\rho$
     * Suppose $(k''', w''') \sqsupseteq (k'', w'')$ and $(k''', w''', v_3, v_4) \in V_n^\circ[\![\alpha']\!]\rho = R'$.
     * To show: $(k''', w''', v_3.1, (v_4\ \mathsf{mod}\ 2) \neq 0) \in E_n^\circ[\![\alpha]\!]\rho$
     * Case $v_4 = 0$:
        · Then $v_3 = \langle v_3', v_3''\rangle$ with $(k''', w''', v_3', \mathsf{false}) \in r.R$ by construction of $R'$.
        · Consequently, $w'''.\sigma_1; v_3.1 \hookrightarrow^1 v_3'$ and $w'''.\sigma_2; (v_4\ \mathsf{mod}\ 2) \neq 0 \hookrightarrow^* \mathsf{false}$.
        · By construction of $R'$, $(k''' - 1, \lfloor w''' \rfloor, v_3', \mathsf{false}) \in r.R = V_n^\circ[\![\alpha']\!]\rho$.
     * Other cases analogously.
  3. $(k'', w'', \lambda x{:}(\alpha \times \alpha).(x.2), \lambda x{:}\mathsf{int}.(x\ \mathsf{div}\ 2) \neq 0) \in V_n^\circ[\![\alpha' \to \alpha]\!]\rho$
     * Analogously to previous case.

### 10.8.6 Semantically Non-Parametric Functor

Given

$$\begin{aligned}
\tau_\alpha &:= \exists\alpha'.(\alpha \to \alpha') \times (\alpha' \to \alpha)\\
v_d &:= \mathsf{pack}\ \langle \alpha, \langle \lambda x{:}\alpha.x, \lambda x{:}\alpha.x\rangle\rangle\ \mathsf{as}\ \tau_\alpha\\
v_1 &:= \Lambda\alpha.\,\mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_\alpha\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x + 1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d\\
v_2 &:= \Lambda\alpha.\,\mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_\alpha\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x + 1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d
\end{aligned}$$

we show $\epsilon;\epsilon \vdash v_1 \precsim^+ v_2 : \forall\alpha.\tau_\alpha$, which, by Corollary 10.42, implies $\epsilon;\epsilon \vdash \mathrm{Wr}^+_{\forall\alpha.\tau_\alpha}\ v_1 \precsim \mathrm{Wr}^+_{\forall\alpha.\tau_\alpha}\ v_2 : \forall\alpha.\tau_\alpha$.

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n^-[\![\epsilon]\!]w_0$, $(k, w, \gamma_1, \gamma_2) \in G_n^-[\![\epsilon]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- To show: $(k, w, \delta_1\gamma_1(v_1), \delta_2\gamma_2(v_2)) \in E_n^+[\![\forall\alpha.\tau_\alpha]\!]\rho$, i.e., $(k, w, v_1, v_2) \in V_n^+[\![\forall\alpha.\tau_\alpha]\!]\emptyset$
- So suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}^-[\![\Omega]\!]w'$.
- To show: $(k'', w'', \mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_{\tau_1}\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x + 1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d[\tau_1/\alpha],$ $\mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_{\tau_2}\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x + 1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d[\tau_2/\alpha]) \in E_n^+[\![\tau_\alpha]\!]\alpha{\mapsto}r$
- Case $\tau_1 \neq \mathsf{int}$:
  – Then $\tau_{\tau_1} \neq \tau_{\mathsf{int}}$ and hence $w''.\sigma_1; \mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_{\tau_1}\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x{+}1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d[\tau_1/\alpha] \hookrightarrow^1 w''.\sigma_1; v_d[\tau_1/\alpha]$.

- Furthermore, by definition of $T^-[\![\Omega]\!]$, also $\tau_2 \neq \mathsf{int}$.
- So $\tau_{\tau_2} \neq \tau_{\mathsf{int}}$ and hence
  $w''.\sigma_2; \mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_{\tau_2}\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x{+}1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d[\tau_2/\alpha] \hookrightarrow^1 w''.\sigma_2; v_d[\tau_2/\alpha]$.
- It remains to show $(k''-1, \lfloor w''\rfloor, v_d[\tau_1/\alpha], v_d[\tau_2/\alpha]) \in V_n^+[\![\tau_\alpha]\!]\alpha{\mapsto}r$.
- By Lemma 10.37, $(\tau_1, \tau_2, r) \in T_{k'}^+[\![\Omega]\!]w'$.
- We claim that $(k_1, w_1, \langle \lambda x{:}\tau_1.x, \lambda x{:}\tau_1.x\rangle, \langle \lambda x{:}\tau_2.x, \lambda x{:}\tau_2.x\rangle) \in V_n^+[\![(\alpha \to \alpha') \times (\alpha' \to \alpha)]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r$ for any $(k_1, w_1) \sqsupset (k''-1, \lfloor w''\rfloor)$.
- It suffices to show
  1. $(k_1, w_1, \lambda x{:}\tau_1.x, \lambda x{:}\tau_2.x) \in V_n^+[\![\alpha \to \alpha']\!]\alpha{\mapsto}r, \alpha'{\mapsto}r$
     * Suppose $(k_2, w_2, v_1', v_2') \in V_n^-[\![\alpha]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r = r.R$ with $(k_2, w_2) \sqsupseteq (k_1, w_1)$.
     * Then $(k_2, w_2, v_1', v_2') \in r.R = V_n^+[\![\alpha']\!]\alpha{\mapsto}r, \alpha'{\mapsto}r$.
  2. $(k_1, w_1, \lambda x{:}\tau_1.x, \lambda x{:}\tau_2.x) \in V_n^+[\![\alpha' \to \alpha]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r'$
     * Suppose $(k_2, w_2, v_1', v_2') \in V_n^-[\![\alpha']\!]\alpha{\mapsto}r, \alpha'{\mapsto}r = r.R$ with $(k_2, w_2) \sqsupseteq (k_1, w_1)$.
     * Then $(k_2, w_2, v_1', v_2') \in r.R = V_n^+[\![\alpha]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r$.

- Case $\tau_1 = \mathsf{int}$:
  - Then $\tau_{\tau_1} = \tau_{\mathsf{int}}$ and hence:
    $$w''.\sigma_1; \mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_{\tau_1}\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x+1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d[\tau_1/\alpha]$$
    $$\hookrightarrow^1 w''.\sigma_1; \mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x+1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}}$$
  - Furthermore, by definition of $T^-[\![\Omega]\!]$, also $\tau_2 = \mathsf{int}$.
  - So $\tau_{\tau_2} = \tau_{\mathsf{int}}$ and hence:
    $$w''.\sigma_2; \mathsf{cast}\ \tau_{\mathsf{int}}\ \tau_{\tau_1}\ (\mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x+1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}})\ v_d[\tau_2/\alpha]$$
    $$\hookrightarrow^1 w''.\sigma_2; \mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x+1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}}$$
  - It remains to show $(k''-1, \lfloor w''\rfloor, \mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x+1, \lambda x{:}\mathsf{int}.x\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}}, \mathsf{pack}\ \langle \mathsf{int}, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x+1\rangle\rangle\ \mathsf{as}\ \tau_{\mathsf{int}}) \in V_n^+[\![\tau_\alpha]\!]\alpha{\mapsto}r$.
  - Let $R' := \{(\hat{k}, \hat{w}, m{+}1, m) \in \mathrm{Atom}_{k''-1}[\mathsf{int}, \mathsf{int}]\}$ and $r' := (\mathsf{int}, \mathsf{int}, R')$, so $(\mathsf{int}, \mathsf{int}, r') \in T_{k''-1}^+[\![\Omega]\!]\lfloor w''\rfloor_{k''-1}$.
  - We claim that $(k_1, w_1, \langle \lambda x{:}\mathsf{int}.x+1, \lambda x{:}\mathsf{int}.x\rangle, \langle \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x+1\rangle) \in V_n^+[\![(\alpha \to \alpha') \times (\alpha' \to \alpha)]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r'$ for any $(k_1, w_1) \sqsupset (k''-1, \lfloor w''\rfloor)$.
  - Note that, again by definition of $T^-[\![\Omega]\!]$, $r.R = V_{k'}[\![\mathsf{int}]\!]w'.\rho$.
  - It suffices to show:
    1. $(k_1, w_1, \lambda x{:}\mathsf{int}.x+1, \lambda x{:}\mathsf{int}.x) \in V_n^+[\![\alpha \to \alpha']\!]\alpha{\mapsto}r, \alpha'{\mapsto}r'$
       * Suppose $(k_2, w_2, v_1', v_2') \in V_n^-[\![\alpha]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r' = r.R = V_{k'}[\![\mathsf{int}]\!]w'.\rho$ with $(k_2, w_2) \sqsupseteq (k_1, w_1)$, i.e. $v_1' = v_2' = m$ for some integer $m$.
       * Consequently, $w_2.\sigma_1; v_1' + 1 \hookrightarrow^1 w_2.\sigma_1; m+1$.
       * Note that $(k_2 - 1, \lfloor w_2\rfloor, m+1, m) \in R' = r'.R = V_n^+[\![\alpha']\!]\alpha{\mapsto}r, \alpha'{\mapsto}r'$.
    2. $(k_1, w_1, \lambda x{:}\mathsf{int}.x, \lambda x{:}\mathsf{int}.x+1) \in V_n^+[\![\alpha' \to \alpha]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r'$
       * Suppose $(k_2, w_2, v_1', v_2') \in V_n^-[\![\alpha']\!]\alpha{\mapsto}r, \alpha'{\mapsto}r' = r'.R = R$ with $(k_2, w_2) \sqsupseteq (k_1, w_1)$, i.e., $v_1' = m+1$ and $v_2' = m$ for some integer $m$.
       * Consequently, $w_2.\sigma_2; v_2' + 1 \hookrightarrow^1 w_2.\sigma_2; m+1$.
       * Note that $(k_2{-}1, \lfloor w_2\rfloor, m{+}1, m{+}1) \in V_{k'}[\![\mathsf{int}]\!]w'.\rho = r.R = V_n^+[\![\alpha]\!]\alpha{\mapsto}r, \alpha'{\mapsto}r'$.

### 10.8.7 A Free Theorem

Suppose $\sigma_0 \vdash v : \forall\alpha.\alpha \to \alpha$. We want to show that either

1. $\sigma; \mathrm{Wr}^-_{\forall\alpha.\alpha\to\alpha}\ v\ \tau\ v' \uparrow$ for all $\sigma$, $\tau$, $v'$ with $\sigma \supseteq \sigma_0$ and $\sigma \vdash v' : \tau$, or
2. $\sigma; \mathrm{Wr}^-_{\forall\alpha.\alpha\to\alpha}\ v\ \tau\ v' \hookrightarrow^* \sigma'; v'$ for all $\sigma$, $\tau$, $v'$ with $\sigma \supseteq \sigma_0$ and $\sigma \vdash v' : \tau$, where $\sigma'$ may be different each time.

Assume (1) does not hold (otherwise we are done). In this case we know that there is at least one appropriate $\sigma_1$, $\tau_1$, $v_1'$ such that $\sigma_1; \mathrm{Wr}^-\ v\ \tau_1\ v_1'$ terminates in $j := j_1 + 1 + j_2 + 1 + j_3$ steps:

$$\begin{array}{ll} & \sigma_1; \mathrm{Wr}^-\ v\ \tau_1\ v_1' \\ \hookrightarrow^{j_1} & \sigma_1'; (\Lambda\alpha.e_1)\ \tau_1\ v_1' \\ \hookrightarrow^{1} & \sigma_1'; e_1[\tau_1/\alpha]\ v_1' \\ \hookrightarrow^{j_2} & \sigma_1''; (\lambda x{:}\tau_1'.e_1')\ v_1' \\ \hookrightarrow^{1} & \sigma_1''; e_1'[v_1'/x] \\ \hookrightarrow^{j_3} & \sigma_1'''; v_1 \end{array}$$

We show that this implies that any $\sigma_2; \mathrm{Wr}^-\ v\ \tau_2\ v_2'$ will indeed evaluate to $\sigma_2'; v_2'$ for some $\sigma_2'$.

- By the Fundamental Property 10.31, $\sigma_0; \epsilon \vdash v \precsim v : \forall\alpha.\alpha \to \alpha$.
- Construct $w_0 \in \mathrm{World}_{j+2}$ and $(\delta_1, \delta_2, \rho) \in D_{j+2}[\![\sigma_0]\!]w_0$ in the same manner as in the proof of Soundness (10.36) except that $w_0.\sigma_1 = \sigma_1$ and $w_0.\sigma_2 = \sigma_2$.
- Instantiating $\sigma_0; \epsilon \vdash v \precsim v : \forall\alpha.\alpha \to \alpha$ then yields $(j+1, \lfloor w_0 \rfloor, v, v) \in V_n[\![\forall\alpha.\alpha \to \alpha]\!]\rho$
- By Wrapping II (10.43) and Inclusion (10.5), $(j+1, \lfloor w_0 \rfloor, \mathrm{Wr}^-\ v, \mathrm{Wr}^-\ v) \in E_n^\circ[\![\forall\alpha.\alpha \to \alpha]\!]\rho$.
- Consequently, there exists $(j+1-j_1, w') \sqsupseteq (j+1, \lfloor w_0 \rfloor)$ such that $\sigma_2; \mathrm{Wr}^-\ v\ \tau_2\ v_2' \hookrightarrow^* w'.\sigma_2; (\Lambda\alpha.e_2)\ \tau_2\ v_2'$ with $w'.\sigma_1 = \sigma_1'$ and $(j+1-j_1, w', \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^\circ[\![\forall\alpha.\alpha \to \alpha]\!]\rho$.
- Let $R := \{(\hat{k}, \hat{w}, v_1', v_2') \in \mathrm{Atom}_{j+1-j_1}[\sigma_1^*(\tau_1), \sigma_2^*(\tau_2)]\}$ and $r := (\sigma_1^*(\tau_1), \sigma_2^*(\tau_2), R)$, so $(\tau_1, \tau_2, r) \in T^\circ_{j+1-j_1}[\![\Omega]\!]w'$.
- Instantiate $(j+1-j_1, w', \Lambda\alpha.e_1, \Lambda\alpha.e_2) \in V_n^\circ[\![\forall\alpha.\alpha \to \alpha]\!]\rho$ to get $(j+1-j_1-1, \lfloor w' \rfloor, e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n^\circ[\![\alpha \to \alpha]\!]\rho, \alpha{\mapsto}r$.
- Consequently, there exists $(j+1-j_1-1-j_2, w'') \sqsupseteq (j+1-j_1-1, \lfloor w' \rfloor)$ such that $w'.\sigma_2; e_2[\tau_2/\alpha]\ v_2' \hookrightarrow^* w''.\sigma_2; (\lambda x{:}\tau_2'.e_2')\ v_2'$ with $w''.\sigma_1 = \sigma_1''$ and $(j+1-j_1-1-j_2, w'', \lambda x{:}\tau_1'.e_1', \lambda x{:}\tau_2'.e_2') \in V_n^\circ[\![\alpha \to \alpha]\!]\rho, \alpha{\mapsto}r$.
- Since $(j+1-j_1-1-j_2-1, \lfloor w'' \rfloor, v_1', v_2') \in R = V_n^\circ[\![\alpha]\!]\rho, \alpha{\mapsto}r$, we get $(j+1-j_1-1-j_2-1, \lfloor w'' \rfloor, e_1'[v_1'/x], e_2'[v_2'/x]) \in E_n^\circ[\![\alpha]\!]\rho, \alpha{\mapsto}r$.
- Consequently, there exists $(1, w''') \sqsupseteq (j+1-j_1-1-j_2-1, \lfloor w'' \rfloor)$ such that $w''.\sigma_2; e_2'[v_2'/x] \hookrightarrow^* w'''.\sigma_2; v_2$ with $w'''.\sigma_1 = \sigma_1'''$ and $(1, w''', v_1, v_2) \in V_n^\circ[\![\alpha]\!]\rho, \alpha{\mapsto}r = R$.
- Hence $v_1 = v_1'$ and $v_2 = v_2'$ by construction of $R$.

# 11 Full Abstraction

The definition of the parametric relation $E^\circ$ (including the extension for recursive types) is largely very similar to that of a typical step-indexed logical relation $E_{\mathrm{F}^\mu}$ for $\mathrm{F}^\mu$, *i.e.,* System F with pairs, existentials and iso-recursive types [3]. The main difference is the presence of worlds, but they are not actually used in a particularly interesting way in $E^\circ$. Therefore, one might expect that any two $\mathrm{F}^\mu$ terms related by the hypothetical $E_{\mathrm{F}^\mu}$ would also be related by $E^\circ$ and vice versa.

However, this is not obvious: $\mathrm{G}^\mu$ is more expressive than $\mathrm{F}^\mu$, *i.e.,* even terms in the parametric relation can contain non-trivial uses of casts (*e.g.,* the generic ADT for pairs from Section 7), and there is no evident way to back-translate these terms into $\mathrm{F}^\mu$, as would be needed for function arguments. That invalidates a proof approach like the one taken by Ahmed and Blume [5].

Ultimately, the property we would like to be able to show is *full abstraction* for the translation of $\mathrm{F}^\mu$ terms into $\mathrm{G}^\mu$ by positive wrapping:

$$\vdash e_1 \simeq_{\mathrm{F}^\mu} e_2 : \tau \iff \vdash \mathrm{Wr}^+_\tau\ e_1 \simeq \mathrm{Wr}^+_\tau\ e_2 : \tau$$

(The semantics of $\mathrm{F}^\mu$ can be obtained from $\mathrm{G}^\mu$ by restricting $\Delta$ to simple variable components, ignoring all the rules related to $\mathsf{cast}$ and $\mathsf{new}$ as well as the conversion rule Econv, and dropping the type store from the reduction relation. Contextual approximation then is defined as for $\mathrm{G}^\mu$except that it does not mention a type store and the universally quantified contexts must have type $(\Delta; \Gamma; \tau) \rightsquigarrow (\epsilon; \epsilon; \tau')$.) This equivalence is even stronger than the one about logical relatedness in $E_{\mathrm{F}^\mu}$ and $E^\circ$ because $\precsim$ is only sound w.r.t. contextual approximation, not complete.

The direction from right to left, called *equivalence reflection*, is not hard to show. We present its proof in the remainder of this section. Unfortunately, it is not known to us whether the other direction, *equivalence preservation*, holds as well. We conjecture that it does, but are not aware of any suitable technique to prove it.

Note that while equivalence reflection also holds for F and G, *i.e.,* in the absence of recursive types, equivalence preservation does not. This is because in G we have non-termination (see Section 2.4) whereas in F we do not. Here is a trivial example exploiting this:

$$e_1 := \lambda f{:}(\mathsf{unit} \to \mathsf{unit}).f\ ()$$
$$e_2 := \lambda f{:}(\mathsf{unit} \to \mathsf{unit}).()$$

Clearly, $e_1$ and $e_2$ are contextually equivalent in F. Wrapping basically leaves them unmodified, because their type is simple. However, $e_1$ and $e_2$ are not contextually equivalent in G, since a G context can apply them to a diverging function.

## 11.1 Equivalence Reflection

We show

$$\vdash \mathrm{Wr}^+_\tau\ e_1 \simeq_{G^\mu} \mathrm{Wr}^+_\tau\ e_2 : \tau \implies \vdash e_1 \simeq_{\mathrm{F}^\mu} e_2 : \tau$$

(where $e_1$ and $e_2$ are $\mathrm{F}^\mu$ terms) by showing its contraposition. Since $\mathrm{F}^\mu$ is a fragment of $\mathrm{G}^\mu$, it suffices to show that any context $C$ that can distinguish $e_1$ and $e_2$ in $\mathrm{F}^\mu$ will also distinguish their positive wrappings in $\mathrm{G}^\mu$. We do this in two steps. First, we prove that $C$ will distinguish their *simple wrappings* (see Lemma 11.4 below), which differ from the proper wrappings in that they don't contain any $\mathsf{new}$. Subsequently, we prove that distinguishing the simple wrappings implies distinguishing the proper wrappings.

$$
\begin{array}{lcl}
G^{\varphi}_{\mu\alpha.\tau} & \stackrel{\text{def}}{=} & \mathsf{fix}\, f(x').\langle \lambda x{:}(\mu\alpha.\tau).\, \mathsf{roll}\ \mathrm{Cr}^{+}_{\tau}(\varphi, \alpha{\mapsto}f)[\mu\alpha.\tau/\alpha]\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \mu\alpha.\tau, \\
 & & \qquad \lambda x{:}(\mu\alpha.\tau).\, \mathsf{roll}\ \mathrm{Cr}^{-}_{\tau}(\varphi, \alpha{\mapsto}f)[\mu\alpha.\tau/\alpha]\ (\mathsf{unroll}\ x)\ \mathsf{as}\ \mu\alpha.\tau\rangle) \\
 & & \qquad : \mathsf{unit} \to ((\mu\alpha.\tau) \to (\mu\alpha.\tau)) \times ((\mu\alpha.\tau) \to (\mu\alpha.\tau)) \\
 & & \\
\mathrm{Cr}^{+}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.(\varphi(\alpha)\ ()).1\ x \quad (\text{if } \alpha \in \mathrm{dom}(\varphi)) \\
\mathrm{Cr}^{-}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.(\varphi(\alpha)\ ()).2\ x \quad (\text{if } \alpha \in \mathrm{dom}(\varphi)) \\
\mathrm{Cr}^{\pm}_{\alpha}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}\alpha.x \quad (\text{if } \alpha \notin \mathrm{dom}(\varphi)) \\
\mathrm{Cr}^{\pm}_{b}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}b.x \\
\mathrm{Cr}^{\pm}_{\tau_1\times\tau_2}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\tau_1 \times \tau_2).\langle \mathrm{Cr}^{\pm}_{\tau_1}\,\varphi\ (x.1)), \mathrm{Cr}^{\pm}_{\tau_2}\,\varphi\ (x.2)\rangle \\
\mathrm{Cr}^{\pm}_{\tau_1\to\tau_2}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\tau_1 \to \tau_2).\lambda x'{:}\tau_1.\, \mathrm{Cr}^{\pm}_{\tau_2}\,\varphi\ (x\ (\mathrm{Cr}^{\mp}_{\tau_1}\,\varphi\ x')) \\
\mathrm{Cr}^{\pm}_{\forall\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\forall\alpha.\tau).\Lambda\alpha.\, \mathrm{Cr}^{\pm}_{\tau}\,\varphi\ (x\ \alpha) \\
\mathrm{Cr}^{\pm}_{\exists\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\exists\alpha.\tau).\, \mathsf{unpack}\ \langle\alpha, x'\rangle{=}x\ \mathsf{in}\ \mathsf{pack}\ \langle\alpha, \mathrm{Cr}^{\pm}_{\tau}\,\varphi\ x'\rangle\ \mathsf{as}\ \exists\alpha.\tau \\
\mathrm{Cr}^{+}_{\mu\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\mu\alpha.\tau).(G^{\varphi}_{\mu\alpha.\tau}\ ()).1\ x \\
\mathrm{Cr}^{-}_{\mu\alpha.\tau}\,\varphi & \stackrel{\text{def}}{=} & \lambda x{:}(\mu\alpha.\tau).(G^{\varphi}_{\mu\alpha.\tau}\ ()).2\ x \\
 & & \\
\mathrm{Cr}^{\pm}_{\tau} & \stackrel{\text{def}}{=} & \mathrm{Cr}^{\pm}_{\tau}\,\emptyset
\end{array}
$$

Figure 12: Simple wrapping (new-erasure of the proper wrapping)

For the first part we actually show something stronger, namely that any term $e$ is contextually equivalent in $\mathrm{G}^{\mu}$ to its simple wrapping. We do this with the help of our non-parametric logical relation, which is sound w.r.t. contextual approximation. The definition of the simple wrapping $\mathrm{Cr}^{\pm}_{\tau}$ is a trivial modification of the proper wrapping's and given in Figure 12. It is essentially a polarized variant of what Birkedal and Harper call the *syntactic projection function* [8].

**Lemma 11.1** (CR-Substitution)**.** If $\varphi' = \varphi, \alpha{\mapsto}G^{\varphi}_{\mu\alpha.\tau}$, then $(\mathrm{Cr}^{\pm}_{\tau'}\,\varphi')[\mu\alpha.\tau/\alpha] = \mathrm{Cr}^{\pm}_{\tau'[\mu\alpha.\tau/\alpha]}\,\varphi$.

*Proof:* Trivial modification of the one for WR-Substitution (10.39). □

**Lemma 11.2.** Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$ and $(k, w) \sqsupset (n, w_0)$ where $\Delta \vdash \tau$.

1. If $(k, w, v_1, v_2) \in V_n[\![\tau]\!]\rho$, then $(k, w, v_1, \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2) \in E_n[\![\tau]\!]\rho$ and $(k, w, \delta_1(\mathrm{Cr}^{\pm}_{\tau})\ v_1, v_2) \in E_n[\![\tau]\!]\rho$.
2. If $(k, w, e_1, e_2) \in E_n[\![\tau]\!]\rho$, then $(k, w, e_1, \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ e_2) \in E_n[\![\tau]\!]\rho$ and $(k, w, \delta_1(\mathrm{Cr}^{\pm}_{\tau})\ e_1, e_2) \in E_n[\![\tau]\!]\rho$.

*Proof:* By primary induction on $n$ and secondary induction on the derivation of $\Delta \vdash \tau$. For both (1) and (2) we show only the first part (where the right hand side is wrapped). The other part is proven analogously.

1. 
   - Case $\tau = \alpha$ or $\tau = b$: Easy.
   - Case $\tau = \tau_1 \times \tau_2$: $v_i = \langle v_{i1}, v_{i2}\rangle$
     - To show: $(k, w, v_1, \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2) \in E_n[\![\tau]\!]\rho$
     - Note that
$$
\begin{array}{rl}
 & w.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2 \\
\hookrightarrow & w.\sigma_2; \langle \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_2.1, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_2.2\rangle \\
\hookrightarrow & w.\sigma_2; \langle \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_{21}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_2.2\rangle
\end{array}
$$

- By assumption, $(k, w, v_{11}, v_{21}) \in V_n[\![\tau_1]\!]\rho$.
- So by induction, $(k, w, v_{11}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_{21}) \in E_n[\![\tau_1]\!]\rho$.
- Hence there is $(k, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_{21} \hookrightarrow^* w'.\sigma_2; v'_{21}$ with $w'.\sigma_1 = w.\sigma_1$ and $(k, w', v_{11}, v'_{21}) \in V_n[\![\tau_1]\!]\rho$.
- Consequently,
$$\begin{array}{rl} & w.\sigma_2; \langle \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_{21}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_2.2\rangle \\ \hookrightarrow^* & w'.\sigma_2; \langle v'_{21}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_2.2\rangle \\ \hookrightarrow & w'.\sigma_2; \langle v'_{21}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_{22}\rangle \end{array}$$
- By assumption and Closure Under World Extension (10.7), $(k, w', v_{12}, v_{22}) \in V_n[\![\tau_2]\!]\rho$.
- By induction, $(k, w', v_{12}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_{22}) \in E_n[\![\tau_2]\!]\rho$.
- Hence there is $(k, w'') \sqsupseteq (k, w')$ such that $w'.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_{22} \hookrightarrow^* w''.\sigma_2; v'_{22}$ with $w''.\sigma_1 = w'.\sigma_1$ and $(k, w'', v_{12}, v'_{22}) \in V_n[\![\tau_2]\!]\rho$.
- Consequently,
$$\begin{array}{rl} & w'.\sigma_2; \langle v'_{21}, \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ v_{22}\rangle \\ \hookrightarrow^* & w''.\sigma_2; \langle v'_{21}, v'_{22}\rangle \end{array}$$
- By Closure Under World Extension (10.7), $(k, w'', v_{11}, v'_{21}) \in V_n[\![\tau_1]\!]\rho$.
- Therefore, $(k, w', \langle v_{11}, v_{12}\rangle, \langle v'_{21}, v'_{22}\rangle) \in V_n[\![\tau_1 \times \tau_2]\!]\rho$.

- Case $\tau = \tau_1 \to \tau_2$: $v_i = \lambda x{:}\tau_{1i}.e_i$
  - To show: $(k, w, v_1, \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2) \in E_n[\![\tau]\!]\rho$
  - Note that $w.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2 \hookrightarrow w.\sigma_2; \lambda x{:}\delta_2(\tau_1).\delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ (v_2\ (\delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ x))$.
  - Hence it suffices to show $(k, w, \lambda x{:}\tau_{11}.e_1, \lambda x{:}\delta_2(\tau_1).\delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ (v_2\ (\delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ x))) \in V_n[\![\tau]\!]\rho$.
  - Suppose $(k', w', v_3, v_4) \in V_n[\![\tau_1]\!]\rho$ where $(k', w') \sqsupseteq (k, w)$.
  - To show: $(k', w', e_1[v_3/x], \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ (v_2\ (\delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_4))) \in E_n[\![\tau_2]\!]\rho$
  - So suppose that $w'.\sigma_1; e_1[v_3/x]$ terminates in $j < k$ steps: $w'.\sigma_1; e_1[v_3/x] \hookrightarrow^j \sigma_1; v'_1$
  - By induction, $(k', w', v_3, \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_4) \in E_n[\![\tau_1]\!]\rho$ by induction.
  - Hence there is $(k', w'') \sqsupseteq (k', w')$ such that $w'.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_4 \hookrightarrow^* w''.\sigma_2; v'_4$ with $w''.\sigma_1 = w'.\sigma_1$ and $(k', w'', v_3, v'_4) \in V_n[\![\tau_1]\!]\rho$.
  - Instantiating the assumption yields $(k', w'', e_1[v_3/x], e_2[v'_4/x]) \in E_n[\![\tau_2]\!]\rho$.
  - By induction, $(k', w'', e_1[v_3/x], \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ e_2[v'_4/x]) \in E_n[\![\tau_2]\!]\rho$.
  - Hence there is $(k' - j, w''') \sqsupseteq (k', w'')$ such that $w''.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ e_2[v'_4/x] \hookrightarrow^* w'''.\sigma_2; v'_2$ with $w'''.\sigma_1 = \sigma_1$ and $(k' - j, w''', v'_1, v'_2) \in V_n[\![\tau_2]\!]\rho$.
  - So we have
$$\begin{array}{rl} & w'.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ (v_2\ (\delta_2(\mathrm{Cr}^{\pm}_{\tau_1})\ v_4)) \\ \hookrightarrow^* & w''.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ (v_2\ v'_4) \\ \hookrightarrow & w''.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ e_2[v'_4/x] \\ \hookrightarrow^* & w'''.\sigma_2; v'_2 \end{array}$$

- Case $\tau = \forall\alpha.\tau'$: $v_i = \Lambda\alpha.e_i$
  - To show: $(k, w, v_1, \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2) \in E_n[\![\tau]\!]\rho$
  - Note that $w.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2 \hookrightarrow w.\sigma_2; \Lambda\alpha.\delta_2(\mathrm{Cr}^{\pm}_{\tau'})\ (v_2\ \alpha)$.
  - Hence it suffices to show $(k, w, \Lambda\alpha.e_1, \Lambda\alpha.\delta_2(\mathrm{Cr}^{\pm}_{\tau'})\ (v_2\ \alpha)) \in V_n[\![\tau]\!]\rho$.

- Suppose $(k'', w'') \sqsupset (k', w') \sqsupseteq (k, w)$ and $(\tau_1, \tau_2, r) \in T_{k'}[\![\Omega]\!]w'$.
- Let $\delta_i' := \delta_i, \alpha{\mapsto}\tau_i$.
- To show: $(k'', w'', e_1[\tau_1/\alpha], \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ (v_2\ \tau_2)) \in E_n[\![\tau]\!]\rho, \alpha{\mapsto}r$
- So suppose that $w''.\sigma_1; e_1[\tau_1/\alpha]$ terminates in $j < k''$ steps: $w''.\sigma_1; e_1[\tau_1/\alpha] \hookrightarrow^j \sigma_1; v_1'$
- Instantiating the assumption yields $(k'', w'', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- Let $\rho' := (\lfloor\rho\rfloor_{k'}, \alpha{\mapsto}r)$, so $(\delta_1', \delta_2', \rho') \in D_{k'}[\![\Delta, \alpha]\!]w'$ by Lemma 10.4.
- By Restriction (10.2) and Irrelevance (10.3), $(k'', w'', e_1[\tau_1/\alpha], e_2[\tau_2/\alpha]) \in E_{k'}[\![\tau']\!]\rho'$.
- By induction, $(k'', w'', e_1[\tau_1/x], \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ e_2[\tau_2/\alpha]) \in E_{k'}[\![\tau']\!]\rho'$.
- By Restriction (10.2) and Irrelevance (10.3), $(k'', w'', e_1[\tau_1/x], \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ e_2[\tau_2/\alpha]) \in E_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- Hence there is $(k'' - j, w''') \sqsupseteq (k'', w'')$ such that $w''.\sigma_2; \delta_2'(\mathrm{Cr}^{\pm}_{\tau_2})\ e_2[\tau_2/\alpha] \hookrightarrow^* w'''.\sigma_2; v_2'$ with $w'''.\sigma_1 = \sigma_1$ and $(k'' - j, w''', v_1', v_2') \in V_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
- So we have

$$\begin{array}{rl} & w'.\sigma_2; \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ (v_2\ \tau_2) \\ \hookrightarrow^* & w''.\sigma_2; \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ (v_2\ \tau_2) \\ \hookrightarrow & w''.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau_2})\ e_2[\tau_2/\alpha] \\ \hookrightarrow^* & w'''.\sigma_2; v_2' \end{array}$$

- Case $\tau = \exists\alpha.\tau'$: $v_i = \mathsf{pack}\ \langle \tau_i, v_i' \rangle\ \mathsf{as}\ \tau_i'$
  - To show: $(k, w, v_1, \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2) \in E_n[\![\tau]\!]\rho$
  - Let $\delta_i' := \delta_i, \alpha{\mapsto}\tau_i$.
  - Note that

$$\begin{array}{rl} & w.\sigma_2; \delta_2(\mathrm{Cr}^{\pm}_{\tau})\ v_2 \\ \hookrightarrow & w.\sigma_2; \mathsf{unpack}\ \langle \alpha, x \rangle{=}v_2\ \mathsf{in}\ \mathsf{pack}\ \langle \alpha, \delta_2(\mathrm{Cr}^{\pm}_{\tau'})\ x \rangle\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow & w.\sigma_2; \mathsf{pack}\ \langle \tau_2, \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ v_2' \rangle\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

  - By assumption there exists $r$ such that $(\tau_1, \tau_2, r) \in T_k[\![\Omega]\!]w$ and $(k{-}1, \lfloor w \rfloor, v_1', v_2') \in V_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
  - Let $\rho' := (\lfloor\rho\rfloor_k, \alpha{\mapsto}r)$, so $(\delta_1', \delta_2', \rho') \in D_k[\![\Delta, \alpha]\!]w$ by Lemma 10.4.
  - By Restriction (10.2) and Irrelevance (10.3), $(k-1, \lfloor w \rfloor, v_1', v_2') \in V_k[\![\tau']\!]\rho'$.
  - So by induction, $(k-1, \lfloor w \rfloor, v_1', \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ v_2') \in E_k[\![\tau']\!]\rho'$.
  - By Restriction (10.2) and Irrelevance (10.3), $(k-1, \lfloor w \rfloor, v_1', \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ v_2') \in E_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
  - Hence there is $(k-1, w') \sqsupseteq (k-1, \lfloor w \rfloor)$ such that $w.\sigma_2; \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ v_2' \hookrightarrow^* w'.\sigma_2; v_2''$ with $w'.\sigma_1 = w.\sigma_1$ and $(k-1, w', v_1', v_2'') \in V_n[\![\tau']\!]\rho, \alpha{\mapsto}r$.
  - Consequently,

$$\begin{array}{rl} & w.\sigma_2; \mathsf{pack}\ \langle \tau_2, \delta_2'(\mathrm{Cr}^{\pm}_{\tau'})\ v_2' \rangle\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow^* & w'.\sigma_2; \mathsf{pack}\ \langle \tau_2, v_2'' \rangle\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

  - Note that $w' \in \mathrm{World}_{k-1}$ implies $w' \in \mathrm{World}_k$ and that for any $(k'', w'') \sqsupset (k, w')$ we have $(k'', w'') \sqsupseteq (k-1, w')$.
  - With the help of Closure Under World Extension (10.7) we get $(k, w', \mathsf{pack}\ \langle \tau_1, v_1' \rangle\ \mathsf{as}\ \tau_1', \mathsf{pack}\ \langle \tau_2, v_2'' \rangle\ \mathsf{as}\ \delta_2(\tau)) \in V_n[\![\exists\alpha.\tau']\!]\rho$.
- Case $\tau = \mu\alpha.\tau'$: $v_i = \mathsf{roll}\ v_i'\ \mathsf{as}\ \tau_i$
  - To show: $(k, w, v_1, \delta_2(\mathrm{Cr}^{+}_{\tau})\ v_2) \in E_n[\![\tau]\!]\rho$ (the proof for $\mathrm{Cr}^{-}$ is symmetric)

– Note that

$$\begin{array}{rl} & w.\sigma_2; \delta_2(\mathrm{Cr}^+_\tau)\ v_2 \\ \hookrightarrow & w.\sigma_2; (\delta_2(G^\emptyset_\tau)\ ()).1\ v_2 \\ \hookrightarrow^* & w.\sigma_2; \mathsf{roll}\ \delta_2(\mathrm{Cr}^+_{\tau'}(\alpha{\mapsto}G^\emptyset_\tau)[\tau/\alpha])\ (\mathsf{unroll}\ v_2)\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow & w.\sigma_2; \mathsf{roll}\ \delta_2(\mathrm{Cr}^+_{\tau'}(\alpha{\mapsto}G^\emptyset_\tau)[\tau/\alpha])\ v_2'\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

– By assumption we know $(k-1, \lfloor w \rfloor, v_1', v_2') \in V_k[\![\tau'[\tau/\alpha]]\!]\rho$.
– By induction, $(k-1, \lfloor w \rfloor, v_1', \delta_2(\mathrm{Cr}^+_{\tau'[\tau/\alpha]})\ v_2') \in E_k[\![\tau'[\tau/\alpha]]\!]\rho$.
– By CR-Substitution (Lemma 11.1), $\mathrm{Cr}^+_{\tau'[\tau/\alpha]} = \mathrm{Cr}^+_{\tau'}(\alpha{\mapsto}G^\emptyset_\tau)[\tau/\alpha]$.
– Hence there is $(k{-}1, w') \sqsupset (k{-}1, \lfloor w \rfloor)$ such that $w.\sigma_2; \delta_2(\mathrm{Cr}^+_{\tau'}(\alpha{\mapsto}G^\emptyset_\tau)[\tau/\alpha])\ v_2' \hookrightarrow^* w'.\sigma_2; v_2''$ with $w'.\sigma_1 = w.\sigma_1$ and $(k-1, w', v_1', v_2'') \in V_k[\![\tau'[\tau/\alpha]]\!]\rho$.
– Consequently,

$$\begin{array}{rl} & w.\sigma_2; \mathsf{roll}\ \delta_2(\mathrm{Cr}^+_{\tau'}(\alpha{\mapsto}G^\emptyset_\tau)[\tau/\alpha])\ v_2'\ \mathsf{as}\ \delta_2(\tau) \\ \hookrightarrow^* & w'.\sigma_2; \mathsf{roll}\ v_2''\ \mathsf{as}\ \delta_2(\tau) \end{array}$$

– Note that $w' \in \mathrm{World}_{k-1}$ implies $w' \in \mathrm{World}_k$ and that for any $(k'', w'') \sqsupset (k, w')$ we have $(k'', w'') \sqsupseteq (k-1, w')$.
– With the help of Closure Under World Extension (10.7) we get $(k, w', \mathsf{roll}\ v_1'\ \mathsf{as}\ \tau_1, \mathsf{roll}\ v_2''\ \mathsf{as}\ \delta_2(\tau)) \in V_n[\![\mu\alpha.\tau']\!]\rho$.

2. • To show: $(k, w, e_1, \delta_2(\mathrm{Cr}^\pm_\tau)\ e_2) \in E_n[\![\tau]\!]\rho$
   • Suppose $w.\sigma_1; e_1$ terminates in $j < k$ steps: $w.\sigma_1; e_1 \hookrightarrow^j \sigma_1; v_1$.
   • By assumption there is $(k-j, w') \sqsupseteq (k, w)$ such that $w.\sigma_2; e_2 \hookrightarrow^* w'.\sigma_2; v_2$ with $w'.\sigma_1 = \sigma_1$ and $(k-j, w', v_1, v_2) \in V_n[\![\tau]\!]\rho$.
   • By part (1), $(k-j, w', v_1, \delta_2(\mathrm{Cr}^\pm_\tau)\ v_2) \in E_n[\![\tau]\!]\rho$.
   • Hence there is $(k-j, w'') \sqsupseteq (k-j, w')$ such that $w'.\sigma_2; \delta_2(\mathrm{Cr}^\pm_\tau)\ v_2 \hookrightarrow^* w''.\sigma_2; v_2'$ with $w''.\sigma_1 = w'.\sigma_1$ and $(k-j, w'', v_1, v_2') \in V_n[\![\tau]\!]\rho$.
   • Note that $(k-j, w'') \sqsupseteq (k, w)$ and that

$$\begin{array}{rl} & w.\sigma_2; \delta_2(\mathrm{Cr}^\pm_\tau)\ e_2 \\ \hookrightarrow^* & w'.\sigma_2; \delta_2(\mathrm{Cr}^\pm_\tau)\ v_2 \\ \hookrightarrow^* & w''.\sigma_2; v_2' \end{array}$$

□

**Lemma 11.3.** If $\Delta; \Gamma \vdash e : \tau$, then $\Delta; \Gamma \vdash e \simeq \mathrm{Cr}^\pm_\tau\ e : \tau$.

*Proof:* We show $\Delta; \Gamma \vdash e \precsim \mathrm{Cr}^\pm_\tau\ e : \tau$. The proof of $\Delta; \Gamma \vdash \mathrm{Cr}^\pm_\tau\ e \precsim e : \tau$ is symmetric. The claim then follows by Soundness (10.36).

- Suppose $w_0 \in \mathrm{World}_n$, $(\delta_1, \delta_2, \rho) \in D_n[\![\Delta]\!]w_0$, $(k, \gamma_1, \gamma_2) \in G_n[\![\Gamma]\!]\rho$, and $(k, w) \sqsupset (n, w_0)$.
- By the Fundamental Property (10.31) we know $\Delta; \Gamma \vdash e \precsim e : \tau$.
- Instantiating this yields $(k, w, \delta_1\gamma_1(e), \delta_2\gamma_2(e)) \in E_n[\![\tau]\!]\rho$.
- By Lemma 11.2, $(k, w, \delta_1\gamma_1(e), \delta_2(\mathrm{Cr}^\pm_\tau)\ \delta_2\gamma_2(e)) \in E_n[\![\tau]\!]\rho$.

- Note that $\delta_2(\mathrm{Cr}_\tau^\pm)\ \delta_2\gamma_2(e) = \delta_2\gamma_2(\mathrm{Cr}_\tau^\pm\ e)$.

□

**Lemma 11.4.**

1. If $\Delta;\Gamma \vdash e : \tau$, $\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\epsilon;\epsilon;\tau')$, and $\epsilon; C[e]\downarrow$, then $\epsilon; C[\mathrm{Cr}_\tau^\pm\ e]\downarrow$.
2. If $\Delta;\Gamma \vdash e : \tau$, $\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\epsilon;\epsilon;\tau')$, and $\epsilon; C[e]\uparrow$, then $\epsilon; C[\mathrm{Cr}_\tau^\pm\ e]\uparrow$[9].

*Proof:* Follows from Lemma 11.3. □

The second part (Lemma 11.10) can be proven in a more direct way. Intuitively, the property holds because the only difference between the reduction of $C[\mathrm{Cr}_\tau^\pm\ e]$ and the reduction of $C[\mathrm{Wr}_\tau^\pm\ e]$ is that during the latter fresh type names are being generated and substituted. Since we assume $C$ to be cast-free, there's no way for these type names to affect the reduction and thus the termination behaviour. We will only sketch the proof and not give formal details, as this would be a very tedious job here and not reveal any insights.

The idea is to use a simulation that relates a term $e_1$ to a term $e_2$ iff $e_1$ is the new-erasure of $e_2$, *i.e.,* $e_1$ is obtained from $e_2$ by dropping all occurences of new. Note that thus the simulation relates the simple wrapping of a term to its proper wrapping.

The definition of Erase, the new-erasure, is trivial. Its only interesting case is

$$\mathrm{Erase}(\mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e) \stackrel{\mathrm{def}}{=} \mathrm{Erase}(e[\tau/\alpha]).$$

For all the other language constructs, the definition just recurses on the subterms. It is easy to see that Erase satisfies standard congruence and substitution properties:

**Lemma 11.5.** If $e_1 = \mathrm{Erase}(e_2)$ and $C$ is new-free, then $C[e_1] = \mathrm{Erase}(C[e_2])$.

**Lemma 11.6.**

1. If $e_1 = \mathrm{Erase}(e_2)$ and $e_1' = \mathrm{Erase}(e_1')$, then $e_1[e_1'/x] = \mathrm{Erase}(e_2[e_2'/x])$.
2. If $e_1 = \mathrm{Erase}(e_2)$, then $e_1[\tau/\alpha] = \mathrm{Erase}(e_2[\tau/\alpha])$.

The simulation argument is the following (where $\hookrightarrow^+$ denotes a non-empty reduction sequence):

**Lemma 11.7.** If $e_1$ is cast-free and $e_1 = \sigma_2^*(\mathrm{Erase}(e_2))$ and $\sigma_1; e_1 \hookrightarrow \sigma_1; e_1'$, then there are $\sigma_2'$ and $e_2'$ with $e_1' = \sigma_2'^*(\mathrm{Erase}(e_2'))$ such that $\sigma_2; e_2 \hookrightarrow^+ \sigma_2'; e_2'$.

This already yields the second part of Lemma 11.10. For the first part we need one more lemma and a trivial induction.

**Lemma 11.8.** If $v = \sigma_2^*(\mathrm{Erase}(e))$, then $\sigma_2; e\downarrow$.

**Lemma 11.9.** If $e_1$ is cast-free and $e_1 = \sigma_2^*(\mathrm{Erase}(e_2))$ and $\sigma_1; e_1\downarrow$, then $\sigma_2; e_2\downarrow$.

*Proof:* By induction on the length of the reduction sequence, using Lemmas 11.8 and 11.7. □

**Lemma 11.10.**

1. If $e$ is cast-free, $C$ is new-free, $\Delta;\Gamma \vdash e : \tau$, $\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\epsilon;\epsilon;\tau')$, and $\epsilon; C[\mathrm{Cr}_\tau^\pm\ e]\downarrow$, then $\epsilon; C[\mathrm{Wr}_\tau^\pm\ e]\downarrow$.

[9] where $\sigma; e\uparrow$ is the negation of $\sigma; e\downarrow$.

2. If $e$ is cast-free, $C$ is new-free, $\Delta;\Gamma \vdash e : \tau$, $\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\epsilon;\epsilon;\tau')$, and $\epsilon; C[\mathrm{Cr}^{\pm}_{\tau}\ e]\uparrow$, then $\epsilon; C[\mathrm{Wr}^{\pm}_{\tau}\ e]\uparrow$.

*Proof:* Since $C[\mathrm{Cr}^{\pm}_{\tau}\ e] = \mathrm{Erase}(C[\mathrm{Wr}^{\pm}_{\tau}\ e])$, part 1 follows from Lemma 11.9 and part 2 from Lemma 11.7. □

Finally, we can prove the actual theorem:

**Theorem 11.11** (Equivalence Reflection)**.** If $\Delta;\Gamma \vdash_{\mathrm{F}^{\mu}} e_1 : \tau$, $\Delta;\Gamma \vdash_{\mathrm{F}^{\mu}} e_2 : \tau$ and $\Delta;\Gamma \vdash \mathrm{Wr}^{\pm}_{\tau}\ e_1 \simeq \mathrm{Wr}^{\pm}_{\tau}\ e_2 : \tau$, then $\Delta;\Gamma \vdash e_1 \simeq_{\mathrm{F}^{\mu}} e_2 : \tau$.

*Proof:* Assume that $\Delta;\Gamma \vdash e_1 \simeq_{\mathrm{F}^{\mu}} e_2 : \tau$ does not hold, *i.e.,* that $e_1$ and $e_2$ are not contextually equivalent in $\mathrm{F}^{\mu}$. Then there is an $\mathrm{F}^{\mu}$-context $C$ that can tell them apart: $C[e_1]\downarrow$ and $C[e_2]\uparrow$ (or vice versa). Note that $C$ also is a valid G context. It is easy to see that $C$ will distinguish $e_1$ and $e_2$ in G, too: $\epsilon; C[e_1]\downarrow$ and $\epsilon; C[e_2]\uparrow$. Using Lemma 11.4 and then Lemma 11.10, this implies that $C$ also distinguishes their wrappings: $\epsilon; C[\mathrm{Wr}^{\pm}_{\tau}\ e_1]\downarrow$ and $\epsilon; C[\mathrm{Wr}^{\pm}_{\tau}\ e_2]\uparrow$. Consequently, $\Delta;\Gamma \vdash \mathrm{Wr}^{\pm}_{\tau}\ e_1 \simeq \mathrm{Wr}^{\pm}_{\tau}\ e_2 : \tau$ does not hold either. □

# 12 Related Work

**Type Generation vs. Other Forms of Data Abstraction.** Traditionally, authors have distinguished between two complementary forms of data abstraction, sometimes dubbed the *static* and the *dynamic* approach [12]. We discussed those in Sections 1.1-1.4. The former is tied to the type system and relies on parametricity (especially for existential types) to hide an ADT's representation from clients [14]. The latter approach is typically employed in untyped languages, which do not have the ability to place static restrictions on clients. Consequently, data hiding has to be enforced on the level of individual values, which is done by generating unique names and using them as *keys* for *dynamically sealing* values [28].

Dynamic type generation as we employ it [21, 30, 23] can be seen as a middle ground, because it bears resemblance to both approaches. As in the dynamic approach, we cannot rely on parametricity and instead generate dynamic names to protect abstractions. However, these are type-level names, not term-level names, and they only 'seal' type information. In particular, individual values of abstract type are still directly represented by the underlying representation type, so that crossing abstraction boundaries has no runtime cost. In that sense, we are closer to the static approach.

Another approach to reconciling type abstraction and type analysis has been proposed by Washburn and Weirich [32]. They introduce a type system that tracks information flow for terms and types-as-data. By distinguishing security levels, the type system can statically prevent unauthorized inspection of types by clients.

**Multi-Language Interoperation.** The closest work to ours probably is that of Matthews and Ahmed [12]. They describe a pair of mutually recursive logical relations that deal with the interoperation between a typed language ("ML") and an untyped language ("Scheme"). Unlike in G, parametric behavior is hard-wired into their ML side: polymorphic instantiation unconditionally performs a form of dynamic sealing to protect against the non-parametric Scheme side. (In contrast, we treat new as its own language construct, orthogonal to universal types.) Dynamic sealing can then be defined in terms of the primitive coercion operators that bridge between the ML and Scheme sides. These coercions are similar to our (meta-level) wrapping operators, but ours perform type-level sealing, not term-level sealing.

The logical relations in Matthews and Ahmed's formalism are somewhat reminiscent of $E^\circ$ and $E$, although theirs are distinct logical relations for two languages, while ours are for a single language and differ only in the definition of $T[\![\Omega]\!]w$. In order to prove the fundamental property for their relations, they prove a "bridge lemma" transferring relatedness in one language to the other via coercions. This is analogous to our Wrapping Theorem for $\precsim^\circ$, but the latter is an independent theorem, not a lemma. Also, they do not propose anything like our polarized logical relations.

A key technical difference is that their formulation of the logical relations does not use possible worlds to capture the type store (the latter is left implicit in their operational semantics). Unfortunately, this resulted in a significant flaw in their paper [4]. They have since reportedly fixed the problem—independently of our work—using a technique similar to ours, but they have yet to write up the details.

**Proof Methods.** Logical relations in various forms are routinely used to reason about program equivalence and type abstraction [20, 13, 17, 3]. In particular, Ahmed, Dreyer and Rossberg recently applied step-indexed logical relations with possible worlds to reason about type abstraction for a language with higher-order state [6]. State in G is comparatively benign, but still requires a circular definition of worlds that we stratify using steps.

Pitts and Stark used logical relations to reason about program equivalence in a language with (term-level) name generation [18] and subsequently generalized their technique to handle mutable references [19]. Sumii and Pierce use them for proving secrecy results for a language with dynamic sealing [27], where generated names are used as keys. Their logical relation uses a form of possible world very similar to ours, but tying relational interpretations to term-level private keys instead of to type names. Their worlds come into play in the interpretation of the type `bits` of encrypted data, whereas in our setup the worlds are important in the interpretation of universal and existential types. In another line of work, Sumii and Pierce have used *bisimulations* to establish abstraction results for both untyped and polymorphic languages [28, 29]. However, none of the languages they investigate mixes the two paradigms.

Grossman, Morrisett and Zdancewic have proposed the use of *abstraction brackets* for syntactically tracing abstraction boundaries [10] during program execution. However, this is a comparatively weak method that does not seem to help in proving parametricity or representation independence results.

# 13 Conclusion and Future Work

In traditional static languages, type abstraction is established by parametric polymorphism. This approach no longer works when dynamic typing features like casts, `typecase`, or reflection are added to the mix. Dynamic type generation addresses this problem.

In this thesis, we have shown that proper use of dynamic type generation succeeds in recovering type abstraction. More specifically:

1. We presented a step-indexed Kripke-style logical relation for reasoning about program equivalence in a non-parametric language with dynamic type generation.

2. We showed that parametricity can be re-established systematically using a simple type-directed wrapping, which then can be reasoned about using a parametric variant of the logical relation.

3. We showed that parametricity can be refined into parametric *behavior* and parametric *usage* and gave a polarized logical relation that distinguishes these dual notions, thereby handling more subtle examples.

The concept of a polarized logical relation seems novel, and it remains to be seen what else it might be useful for. Interestingly, all our logical relations can be defined as a single family differing only in the interpretation $T$ of types-as-data.

An open question is whether the wrapping, when seen as an embedding of $\mathrm{F}^{\mu}$ into $\mathrm{G}^{\mu}$, is fully abstract. We conjecture that it is, but we were only able to show equivalence reflection, not equivalence preservation. Proving full abstraction remains an interesting challenge for future work.

On the practical side, we would like to scale our logical relation to handle a more realistic language like ML. Unfortunately, wrapping cannot easily be extended to a type of mutable references. However, we believe that our approach still scales to a large class of languages, so long as we instrument it with a distinction between module and core levels. Specifically, note that wrapping only does something "interesting" for universal and existential types, and is the identity (modulo $\eta$-expansion) otherwise. Thus, for a language like Standard ML, which does not support first-class polymorphism—or Alice ML, which supports modules-as-first-class-values, but not existentials—wrapping could be confined to the module level (as part of the implementation of opaque signature ascription). For core-level types it could just be the identity. This is a real advantage of type generation over dynamic sealing since, for the latter, the need to seal/unseal individual values of abstract type precludes any attempt to confine wrapping to modules.

# References


[1] Martín Abadi, Luca Cardelli, Benjamin Pierce, and Didier Rémy. Dynamic typing in polymorphic languages. *JFP*, 5(1):111–130, 1995.

[2] Amal Ahmed. *Semantics of Types for Mutable State*. PhD thesis, Princeton University, 2004.

[3] Amal Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In *ESOP*, 2006.

[4] Amal Ahmed. Personal communication, 2009.

[5] Amal Ahmed and Matthias Blume. Typed closure conversion preserves observational equivalence. In *ICFP*, 2008.

[6] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. State-dependent representation independence. In *POPL*, 2009.

[7] Andrew W. Appel and David McAllester. An indexed model of recursive types for foundational proof-carrying code. *TOPLAS*, 23(5):657–683, 2001.

[8] Lars Birkedal and Robert Harper. Relational interpretations of recursive types in an operational setting (summary). In *Theoretical Aspects of Computer Software*, pages 458–490, 1997.

[9] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

[10] Dan Grossman, Greg Morrisett, and Steve Zdancewic. Syntactic type abstraction. *TOPLAS*, 22(6):1037–1080, 2000.

[11] Robert Harper and John C. Mitchell. Parametricity and variants of Girard's J operator. *Information Processing Letters*, 1999.

[12] Jacob Matthews and Amal Ahmed. Parametric polymorphism through run-time sealing, or, theorems for low, low prices! In *ESOP*, 2008.

[13] John C. Mitchell. Representation independence and data abstraction. In *POPL*, 1986.

[14] John C. Mitchell and Gordon D. Plotkin. Abstract types have existential type. *TOPLAS*, 10(3):470–502, 1988.

[15] James H. Morris, Jr. Protection in programming languages. *Commun. ACM*, 16(1):15–21, 1973.

[16] James H. Morris, Jr. Types are not sets. In *POPL '73: Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 120–124, New York, NY, USA, 1973. ACM.

[17] Andrew Pitts. Typed operational reasoning. In Benjamin C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7. MIT Press, 2005.

[18] Andrew Pitts and Ian Stark. Observable properties of higher order functions that dynamically create local names, or: What's new? In *MFCS*, volume 711 of *LNCS*, 1993.

[19] Andrew Pitts and Ian Stark. Operational reasoning for functions with local state. In *HOOTS*, 1998.

[20] John C. Reynolds. Types, abstraction and parametric polymorphism. In *Information Processing*, 1983.

[21] Andreas Rossberg. Generativity and dynamic opacity for abstract types. In *PPDP*, 2003.

[22] Andreas Rossberg. The missing link - dynamic components for ML. In *ICFP*, 2006.

[23] Andreas Rossberg. Dynamic translucency with abstraction kinds and higher-order coercions. In *MFPS*, 2008.

[24] Andreas Rossberg, Didier Le Botlan, Guido Tack, Thorsten Brunklaus, and Gert Smolka. Alice ML through the looking glass. In *TFP*, volume 5, 2004.

[25] Peter Sewell. Modules, abstract types, and distributed versioning. In *POPL*, 2001.

[26] Peter Sewell, James Leifer, Keith Wansbrough, Francesco Zappa Nardelli, Mair Allen-Williams, Pierre Habouzit, and Viktor Vafeiadis. Acute: High-level programming language design for distributed computation. *JFP*, 17(4&5):547–612, 2007.

[27] Eijiro Sumii and Benjamin C. Pierce. Logical relations for encryption. *JCS*, 11(4):521–554, 2003.

[28] Eijiro Sumii and Benjamin C. Pierce. A bisimulation for dynamic sealing. *TCS*, 375(1–3):161–192, 2007.

[29] Eijiro Sumii and Benjamin C. Pierce. A bisimulation for type abstraction and recursion. *JACM*, 54(5):1–43, 2007.

[30] Dimitrios Vytiniotis, Geoffrey Washburn, and Stephanie Weirich. An open and shut typecase. In *TLDI*, 2005.

[31] Philip Wadler. Theorems for free! In *FPCA*, 1989.

[32] Geoffrey Washburn and Stephanie Weirich. Generalizing parametricity using information flow. In *LICS*, 2005.

[33] Stephanie Weirich. Type-safe cast. *JFP*, 14(6):681–695, 2004.

# A The Languages G and $G^{\mu}$

The differences between G and $G^{\mu}$, *i.e.,* everything related to recursive types, are underlined.

## A.1 Syntax and Semantics

### Syntax

$$
\begin{array}{lrcl}
\text{(types)} & \tau & ::= & \alpha \mid b \mid \tau \times \tau \mid \tau \to \tau \mid \forall\alpha.\tau \mid \exists\alpha.\tau \mid \mu\alpha.\tau \\
\text{(values)} & v & ::= & x \mid c \mid \langle v, v\rangle \mid \lambda x{:}\tau.e \mid \Lambda\alpha.e \mid \mathsf{pack}\ \langle\tau, v\rangle\ \mathsf{as}\ \tau \mid \underline{\mathsf{roll}\ v\ \mathsf{as}\ \tau} \\
\text{(expressions)} & e & ::= & v \mid \langle e, e\rangle \mid e.1 \mid e.2 \mid e\ e \mid e\ \tau \mid \mathsf{pack}\ \langle\tau, v\rangle\ \mathsf{as}\ \tau \mid \\
 & & & \mathsf{unpack}\ \langle\alpha, x\rangle{=}e\ \mathsf{in}\ e \mid \underline{\mathsf{roll}\ e\ \mathsf{as}\ \tau} \mid \underline{\mathsf{unroll}\ e} \mid \\
 & & & \mathsf{cast}\ \tau\ \tau \mid \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e
\end{array}
$$

$$
\begin{array}{lrcl}
\text{(type environments)} & \Delta & ::= & \epsilon \mid \Delta, \alpha \mid \Delta, \alpha{\approx}\tau \\
\text{(value environments)} & \Gamma & ::= & \epsilon \mid \Gamma, x{:}\tau
\end{array}
$$

### Reduction

$\boxed{\sigma; e \hookrightarrow \sigma; e}$

$$(\textsc{Rproj1.1})\frac{}{\sigma; \langle v_1, v_2\rangle.1 \hookrightarrow \sigma; v_1} \qquad (\textsc{Rproj2.1})\frac{}{\sigma; \langle v_1, v_2\rangle.2 \hookrightarrow \sigma; v_2}$$

$$(\textsc{Rapp})\frac{}{\sigma; (\lambda x{:}\tau.e)\ v \hookrightarrow \sigma; e[v/x]} \qquad (\textsc{Rinst})\frac{}{\sigma; (\Lambda\alpha.e)\ \tau \hookrightarrow \sigma; e[\tau/\alpha]}$$

$$(\textsc{Runpack})\frac{}{\sigma; \mathsf{unpack}\ \langle\alpha, x\rangle{=}(\mathsf{pack}\langle\tau, v\rangle\ \mathsf{as}\ \tau')\ \mathsf{in}\ e \hookrightarrow \sigma; e[\tau/\alpha][v/x]}$$

$$(\underline{\textsc{Runroll.1}})\frac{}{\sigma; \mathsf{unroll}\,(\mathsf{roll}\ v\ \mathsf{as}\ \tau) \hookrightarrow \sigma; v}$$

$$(\textsc{Rpair.1})\frac{\sigma; e_1 \hookrightarrow \sigma'; e_1'}{\sigma; \langle e_1, e_2\rangle \hookrightarrow \sigma'; \langle e_1', e_2\rangle} \qquad (\textsc{Rpair.2})\frac{\sigma; e_2 \hookrightarrow \sigma'; e_2'}{\sigma; \langle e_1, e_2\rangle \hookrightarrow \sigma'; \langle e_1, e_2'\rangle}$$

$$(\textsc{Rproj1.2})\frac{\sigma; e \hookrightarrow \sigma'; e'}{\sigma; e.1 \hookrightarrow \sigma'; e'.1} \qquad (\textsc{Rproj2.2})\frac{\sigma; e \hookrightarrow \sigma'; e'}{\sigma; e.2 \hookrightarrow \sigma'; e'.2}$$

$$(\textsc{Rapp.1})\frac{\sigma; e_1 \hookrightarrow \sigma'; e_1'}{\sigma; e_1\ e_2 \hookrightarrow \sigma'; e_1'\ e_2} \qquad (\textsc{Rapp.2})\frac{\sigma; e_2 \hookrightarrow \sigma'; e_2'}{\sigma; v\ e_2 \hookrightarrow \sigma'; v\ e_2'}$$

$$(\textsc{Rinst.1})\frac{\sigma; e \hookrightarrow \sigma'; e'}{\sigma; e\ \tau \hookrightarrow \sigma'; e'\ \tau} \qquad (\textsc{Rpack.1})\frac{\sigma; e \hookrightarrow \sigma'; e'}{\sigma; \mathsf{pack}\ \langle\tau, e\rangle\ \mathsf{as}\ \tau' \hookrightarrow \sigma'; \mathsf{pack}\ \langle\tau, e'\rangle\ \mathsf{as}\ \tau'}$$

$$(\textsc{Runpack.1})\frac{\sigma; e_1 \hookrightarrow \sigma'; e_1'}{\sigma; \mathsf{unpack}\ \langle\alpha, x\rangle{=}e_1\ \mathsf{in}\ e_2 \hookrightarrow \sigma'; \mathsf{unpack}\ \langle\alpha, x\rangle{=}e_1'\ \mathsf{in}\ e_2}$$

$$(\underline{\textsc{Rroll}})\frac{\sigma; e \hookrightarrow \sigma'; e'}{\sigma; \mathsf{roll}\ e\ \mathsf{as}\ \tau \hookrightarrow \sigma'; \mathsf{roll}\ e'\ \mathsf{as}\ \tau} \qquad (\underline{\textsc{Runroll.2}})\frac{\sigma; e \hookrightarrow \sigma'; e'}{\sigma; \mathsf{unroll}\ e \hookrightarrow \sigma'; \mathsf{unroll}\ e'}$$

$$(\textsc{Rcast.1})\frac{}{\sigma; \mathsf{cast}\ \tau\ \tau \hookrightarrow \sigma; \lambda x_1{:}\tau.\lambda x_2{:}\tau.x_1}$$

$$(\textsc{Rcast}.2)\frac{\tau_1 \neq \tau_2}{\sigma; \mathsf{cast}\ \tau_1\ \tau_2 \hookrightarrow \lambda x_1{:}\tau_1.\lambda x_2{:}\tau_2.x_2}$$

$$(\textsc{Rnew})\frac{\alpha' \notin \mathrm{dom}(\sigma)}{\sigma; \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ e \hookrightarrow \sigma, \alpha'{\approx}\tau; e[\alpha'/\alpha]}$$

## Type Environments $\boxed{\vdash \Delta}$

$$\frac{}{\vdash \epsilon} \qquad \frac{\vdash \Delta \qquad \alpha \notin \mathrm{dom}(\Delta)}{\vdash \Delta, \alpha} \qquad \frac{\Delta \vdash \tau \qquad \alpha \notin \mathrm{dom}(\Delta)}{\vdash \Delta, \alpha{\approx}\tau}$$

## Value Environments $\boxed{\Delta \vdash \Gamma}$

$$\frac{\vdash \Delta}{\Delta \vdash \epsilon} \qquad \frac{\Delta \vdash \Gamma \qquad \Delta \vdash \tau \qquad x \notin \mathrm{dom}(\Gamma)}{\Delta \vdash \Gamma, x{:}\tau}$$

## Types $\boxed{\Delta \vdash \tau}$

$$(\textsc{Tvar})\frac{\vdash \Delta \qquad \alpha \in \Delta}{\Delta \vdash \alpha} \qquad (\textsc{Tname})\frac{\vdash \Delta \qquad \alpha{\approx}\tau \in \Delta}{\Delta \vdash \alpha}$$

$$(\textsc{Tbase})\frac{\vdash \Delta}{\Delta \vdash b} \qquad (\textsc{Ttimes})\frac{\Delta \vdash \tau_1 \qquad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \times \tau_2} \qquad (\textsc{Tarr})\frac{\Delta \vdash \tau_1 \qquad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \to \tau_2}$$

$$(\textsc{Tall})\frac{\Delta, \alpha \vdash \tau}{\Delta \vdash \forall\alpha.\tau} \qquad (\textsc{Texists})\frac{\Delta, \alpha \vdash \tau}{\Delta \vdash \exists\alpha.\tau} \qquad (\underline{\textsc{Trec}})\frac{\Delta, \alpha \vdash \tau}{\Delta \vdash \mu\alpha.\tau}$$

## Type Compatibility $\boxed{\Delta \vdash \tau \approx \tau'}$

$$(\textsc{Cvar})\frac{\vdash \Delta \qquad \alpha \in \Delta}{\Delta \vdash \alpha \approx \alpha} \qquad (\textsc{Cname})\frac{\vdash \Delta \qquad \alpha{\approx}\tau \in \Delta}{\Delta \vdash \alpha \approx \tau} \qquad (\textsc{Cbase})\frac{\vdash \Delta}{\Delta \vdash b \approx b}$$

$$(\textsc{Ctimes})\frac{\Delta \vdash \tau_1 \approx \tau_1' \qquad \Delta \vdash \tau_2 \approx \tau_2'}{\Delta \vdash \tau_1 \times \tau_2 \approx \tau_1' \times \tau_2'} \qquad (\textsc{Carr})\frac{\Delta \vdash \tau_1 \approx \tau_1' \qquad \Delta \vdash \tau_2 \approx \tau_2'}{\Delta \vdash \tau_1 \to \tau_2 \approx \tau_1' \to \tau_2'}$$

$$(\textsc{Call})\frac{\Delta, \alpha \vdash \tau \approx \tau'}{\Delta \vdash \forall\alpha.\tau \approx \forall\alpha.\tau'} \qquad (\textsc{Cexists})\frac{\Delta, \alpha \vdash \tau \approx \tau'}{\Delta \vdash \exists\alpha.\tau \approx \exists\alpha.\tau'} \qquad (\underline{\textsc{Crec}})\frac{\Delta, \alpha \vdash \tau \approx \tau'}{\Delta \vdash \mu\alpha.\tau \approx \mu\alpha.\tau'}$$

$$(\textsc{Csym})\frac{\Delta \vdash \tau' \approx \tau}{\Delta \vdash \tau \approx \tau'} \qquad (\textsc{Ctrans})\frac{\Delta \vdash \tau \approx \tau'' \qquad \Delta \vdash \tau'' \approx \tau'}{\Delta \vdash \tau \approx \tau'}$$

**Expressions** $\boxed{\Delta;\Gamma \vdash e : \tau}$

$$(\textsc{Evar})\frac{\Delta \vdash \Gamma \qquad x{:}\tau \in \Gamma}{\Delta;\Gamma \vdash x : \tau} \qquad (\textsc{Econ})\frac{\Delta \vdash \Gamma}{\Delta;\Gamma \vdash c : b}$$

$$(\textsc{Epair})\frac{\Delta;\Gamma \vdash e_1 : \tau_1 \qquad \Delta;\Gamma \vdash e_2 : \tau_2}{\Delta;\Gamma \vdash \langle e_1, e_2\rangle : \tau_1 \times \tau_2} \qquad (\textsc{Eproj})\frac{\Delta;\Gamma \vdash e : \tau_1 \times \tau_2}{\Delta;\Gamma \vdash e.i : \tau_i}$$

$$(\textsc{Eabs})\frac{\Delta;\Gamma, x{:}\tau_1 \vdash e : \tau_2}{\Delta;\Gamma \vdash \lambda x{:}\tau_1.e : \tau_1 \to \tau_2} \qquad (\textsc{Eapp})\frac{\Delta;\Gamma \vdash e_1 : \tau_2 \to \tau \qquad \Delta;\Gamma \vdash e_2 : \tau_2}{\Delta;\Gamma \vdash e_1\ e_2 : \tau}$$

$$(\textsc{Egen})\frac{\Delta, \alpha;\Gamma \vdash e : \tau}{\Delta;\Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau} \qquad (\textsc{Einst})\frac{\Delta;\Gamma \vdash e : \forall\alpha.\tau \qquad \Delta \vdash \tau_2}{\Delta;\Gamma \vdash e\ \tau_2 : \tau[\tau_2/\alpha]}$$

$$(\textsc{Epack})\frac{\Delta;\Gamma \vdash e : \tau[\tau_1/\alpha] \qquad \Delta \vdash \tau_1}{\Delta;\Gamma \vdash \mathsf{pack}\ \langle \tau_1, e\rangle\ \mathsf{as}\ \exists\alpha.\tau : \exists\alpha.\tau}$$

$$(\textsc{Eunpack})\frac{\Delta;\Gamma \vdash e_1 : \exists\alpha.\tau_1 \qquad \Delta, \alpha;\Gamma, x{:}\tau_1 \vdash e_2 : \tau \qquad \Delta \vdash \tau}{\Delta;\Gamma \vdash \mathsf{unpack}\ \langle \alpha, x\rangle{=}e_1\ \mathsf{in}\ e_2 : \tau}$$

$$(\underline{\textsc{Eroll}})\frac{\Delta;\Gamma \vdash e : \tau[\mu\alpha.\tau/\alpha]}{\Delta;\Gamma \vdash \mathsf{roll}\ e\ \mathsf{as}\ \mu\alpha.\tau : \mu\alpha.\tau} \qquad (\underline{\textsc{Eunroll}})\frac{\Delta;\Gamma \vdash e : \mu\alpha.\tau}{\Delta;\Gamma \vdash \mathsf{unroll}\ e : \tau[\mu\alpha.\tau/\alpha]}$$

$$(\textsc{Ecast})\frac{\Delta \vdash \Gamma \qquad \Delta \vdash \tau_1 \qquad \Delta \vdash \tau_2}{\Delta;\Gamma \vdash \mathsf{cast}\ \ \tau_1\ \tau_2 : \tau_1 \to \tau_2 \to \tau_2} \qquad (\textsc{Enew})\frac{\Delta, \alpha{\approx}\tau';\Gamma \vdash e : \tau \qquad \Delta \vdash \tau \qquad \Delta \vdash \Gamma}{\Delta;\Gamma \vdash \mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ e : \tau}$$

$$(\textsc{Econv})\frac{\Delta;\Gamma \vdash e : \tau' \qquad \Delta \vdash \tau \approx \tau'}{\Delta;\Gamma \vdash e : \tau}$$

## A.2 Structural Properties

| | | | |
|---|---|---|---|
| (stores) | $\sigma$ | $::=$ | $\epsilon \mid \sigma, \alpha{\approx}\tau$ |
| (type substitutions) | $\delta$ | $::=$ | $\emptyset \mid \delta, \alpha{\mapsto}\tau$ |
| (value substitutions) | $\gamma$ | $::=$ | $\emptyset \mid \gamma, x{\mapsto}v$ |

**Configurations** $\boxed{\vdash \sigma; e : \tau}$

$$\frac{\Delta = \sigma \qquad \Delta;\epsilon \vdash e : \tau \qquad \epsilon \vdash \tau}{\vdash \sigma; e : \tau}$$

**Type Substitutions** $\boxed{\Delta \vdash \delta : \Delta}$

$$\frac{\vdash \Delta'}{\Delta' \vdash \emptyset : \epsilon} \qquad \frac{\Delta' \vdash \delta : \Delta \qquad \Delta' \vdash \tau}{\Delta' \vdash \delta, \alpha{\mapsto}\tau : \Delta, \alpha} \qquad \frac{\Delta' \vdash \delta : \Delta \qquad \alpha'{\approx}\delta(\tau) \in \Delta'}{\Delta' \vdash \delta, \alpha{\mapsto}\alpha' : \Delta, \alpha{\approx}\tau}$$

### Type Substitution Compatibility

$\boxed{\Delta \vdash \delta \approx \delta' : \Delta}$

$$\frac{\vdash \Delta'}{\Delta' \vdash \emptyset \approx \emptyset : \epsilon} \qquad \frac{\Delta' \vdash \delta \approx \delta' : \Delta \qquad \Delta' \vdash \tau \approx \tau'}{\Delta' \vdash \delta, \alpha{\mapsto}\tau \approx \delta', \alpha{\mapsto}\tau' : \Delta, \alpha}$$

$$\frac{\Delta' \vdash \delta \approx \delta' : \Delta \qquad \alpha_1 {\approx} \delta(\tau) \in \Delta' \qquad \alpha_2 {\approx} \delta'(\tau) \in \Delta'}{\Delta' \vdash \delta, \alpha{\mapsto}\alpha_1 \approx \delta', \alpha{\mapsto}\alpha_2 : \Delta, \alpha{\approx}\tau}$$

### Value Substitutions

$\boxed{\Delta; \Gamma \vdash \gamma : \Gamma}$

$$\frac{\Delta \vdash \Gamma'}{\Delta; \Gamma' \vdash \emptyset : \epsilon} \qquad \frac{\Delta; \Gamma' \vdash \gamma : \Gamma \qquad \Delta; \Gamma' \vdash v : \tau}{\Delta; \Gamma' \vdash \gamma, x{\mapsto}v : \Gamma, x{:}\tau}$$

**Lemma A.1** (Weakening)**.**

1. If $\Delta \vdash \tau$ and $\Delta' \supseteq \Delta$ and $\vdash \Delta'$, then $\Delta' \vdash \tau$.
2. If $\Delta \vdash \tau \approx \tau'$ and $\Delta' \supseteq \Delta$ and $\vdash \Delta'$, then $\Delta' \vdash \tau \approx \tau'$.
3. If $\Delta \vdash \Gamma$ and $\Delta' \supseteq \Delta$ and $\vdash \Delta'$, then $\Delta' \vdash \Gamma$.
4. If $\Delta; \Gamma \vdash e : \tau$ and $\Delta' \supseteq \Delta$ and $\vdash \Delta'$, then $\Delta'; \Gamma \vdash e : \tau$.
5. If $\Delta; \Gamma \vdash e : \tau$ and $\Gamma' \supseteq \Gamma$ and $\Delta \vdash \Gamma'$, then $\Delta; \Gamma' \vdash e : \tau$.
6. If $\Delta; \Gamma \vdash \gamma : \Gamma$ and $\Delta' \supseteq \Delta$ and $\vdash \Delta'$, then $\Delta'; \Gamma \vdash \gamma : \Gamma$.

*Proof:* Each by induction on the first derivation. □

**Lemma A.2** (Substitution)**.**

1. If $\Delta \vdash \tau$ and $\Delta' \vdash \delta : \Delta$, then $\Delta' \vdash \delta(\tau)$.
2. If $\Delta \vdash \tau \approx \tau'$ and $\Delta' \vdash \delta \approx \delta' : \Delta$, then $\Delta' \vdash \delta(\tau) \approx \delta'(\tau')$.
3. If $\Delta \vdash \Gamma$ and $\Delta' \vdash \delta : \Delta$, then $\Delta' \vdash \delta(\Gamma)$.
4. If $\Delta; \Gamma \vdash e : \tau$ and $\Delta' \vdash \delta : \Delta$, then $\Delta'; \delta(\Gamma) \vdash \delta(e) : \delta(\tau)$.
5. If $\Delta; \Gamma \vdash e : \tau$ and $\Delta; \Gamma' \vdash \gamma : \Gamma$, then $\Delta; \Gamma' \vdash \gamma(e) : \tau$.

*Proof:* Each by induction on the first derivation. □

**Lemma A.3** (Validity)**.**

1. If $\Delta \vdash \tau$, then $\vdash \Delta$.
2. If $\Delta \vdash \tau \approx \tau'$, then $\vdash \Delta$.
3. If $\Delta \vdash \Gamma$, then $\vdash \Delta$.
4. If $\Delta; \Gamma \vdash e : \tau$, then $\vdash \Delta$ and $\Delta \vdash \Gamma$ and $\Delta \vdash \tau$.

*Proof:* Each by induction on the derivation. □

**Lemma A.4** (Variable Containment)**.**

1. If $\Delta \vdash \tau$ and $\alpha \in \mathrm{ftv}(\tau)$, then $\alpha \in \mathrm{dom}(\Delta)$.

2. If $\Delta \vdash \tau \approx \tau'$ and $\alpha \in \mathrm{ftv}(\tau) \cup \mathrm{ftv}(\tau')$, then $\alpha \in \mathrm{dom}(\Delta)$.
3. If $\Delta \vdash \Gamma$ and $\alpha \in \mathrm{ftv}(\Gamma)$, then $\alpha \in \mathrm{dom}(\Delta)$.
4. If $\Delta; \Gamma \vdash e : \tau$ and $\alpha \in \mathrm{ftv}(\Gamma) \cup \mathrm{ftv}(e) \cup \mathrm{ftv}(\tau)$, then $\alpha \in \mathrm{dom}(\Delta)$.
5. If $\Delta; \Gamma \vdash e : \tau$ and $x \in \mathrm{fvv}(e)$, then $x \in \mathrm{dom}(\Gamma)$.

*Proof:* Each by induction on the derivation. □

## A.3 Type Safety

**Theorem A.5** (Preservation)**.** If $\sigma; e \hookrightarrow \sigma'; e'$ and $\vdash \sigma; e : \tau$, then $\vdash \sigma'; e' : \tau$.

*Proof:* By induction on the first derivation. □

**Lemma A.6** (Canonical Values)**.** Assume $\vdash \sigma; v : \tau$. Then:

1. If $\tau = \tau_1 \times \tau_2$, then $v = \langle v_1, v_2 \rangle$.
2. If $\tau = \tau_1 \to \tau_2$, then $v = \lambda x{:}\tau_1'.e$.
3. If $\tau = \forall \alpha.\tau_1$, then $v = \Lambda\alpha.e$.
4. If $\tau = \exists \alpha.\tau_1$, then $v = \mathsf{pack}\ \langle \tau_2, v_1 \rangle\ \mathsf{as}\ \tau'$.
5. If $\tau = \mu\alpha.\tau_1$, then $v = \mathsf{roll}\ v'\ \mathsf{as}\ \tau'$.

*Proof:* By induction on the derivation. □

**Theorem A.7** (Progress)**.** If $\vdash \sigma; e : \tau$ and $e \neq v$, then $\sigma; e \hookrightarrow \sigma'; e'$.

*Proof:* By induction on the derivation. □

## A.4 Contextual Approximation and Equivalence

$$\begin{array}{lrcl} \text{(contexts)} & C & ::= & [\_] \mid \langle C, e\rangle \mid \langle e, C\rangle \mid C.1 \mid C.2 \mid \lambda x{:}\tau.C \mid C\ e \mid e\ C \mid \\ & & & \Lambda\alpha.C \mid C\ \tau \mid \mathsf{pack}\langle \tau, C\rangle \mid \mathsf{unpack}\langle \alpha, x\rangle{=}C\ \mathsf{in}\ e \mid \\ & & & \mathsf{unpack}\ \langle \alpha, x\rangle{=}e\ \mathsf{in}\ C \mid \mathsf{roll}\ C\ \mathsf{as}\ \tau \mid \mathsf{unroll}\ C \mid \mathsf{new}\ \alpha{\approx}\tau\ \mathsf{in}\ C \end{array}$$

### Contexts

$$\boxed{\vdash C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta; \Gamma; \tau)}$$

$$(\textsc{Cempty})\ \frac{\Delta \subseteq \Delta' \quad \Gamma \subseteq \Gamma' \quad \Delta' \vdash \Gamma'}{\vdash [\_] : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau)}$$

$$(\textsc{Cabs})\ \frac{\vdash C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma', x{:}\tau_1; \tau_2)}{\vdash \lambda x{:}\tau_1.C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_1 \to \tau_2)}$$

$$(\textsc{Cpair}.1)\ \frac{\vdash C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_1) \qquad \Delta'; \Gamma' \vdash e : \tau_2}{\vdash \langle C, e\rangle : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_1 \times \tau_2)}$$

$$(\textsc{Cpair}.2)\ \frac{\vdash C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_2) \qquad \Delta'; \Gamma' \vdash e : \tau_1}{\vdash \langle e, C\rangle : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_1 \times \tau_2)}$$

$$(\textsc{Cproj})\ \frac{\vdash C : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_1 \times \tau_2)}{\vdash C.i : (\Delta; \Gamma; \tau) \rightsquigarrow (\Delta'; \Gamma'; \tau_i)}$$

$$(\textsc{App}.1)\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_1 \to \tau_2) \qquad \Delta';\Gamma' \vdash e : \tau_1}{\vdash C\ e : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_2)}$$

$$(\textsc{App}.2)\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_1) \qquad \Delta';\Gamma' \vdash e : \tau_1 \to \tau_2}{\vdash e\ C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_2)}$$

$$(\textsc{Gen})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta',\alpha;\Gamma';\tau')}{\vdash \Lambda\alpha.C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\forall\alpha.\tau')}$$

$$(\textsc{Inst})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\forall\alpha.\tau_1) \qquad \Delta' \vdash \tau_2}{\vdash C\ \tau_2 : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_1[\tau_2/\alpha])}$$

$$(\textsc{Pack})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_1[\tau_2/\alpha]) \qquad \Delta' \vdash \tau_2}{\vdash \mathsf{pack}\langle\tau_2, C\rangle : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\exists\alpha.\tau_1)}$$

$$(\textsc{Unpack}.1)\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\exists\alpha.\tau_1) \qquad \Delta',\alpha;\Gamma',x{:}\tau_1 \vdash e_2 : \tau_2 \qquad \Delta' \vdash \tau_2}{\vdash \mathsf{unpack}\ \langle\alpha,x\rangle{=}C\ \mathsf{in}\ e : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_2)}$$

$$(\textsc{Unpack}.2)\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta',\alpha;\Gamma',x{:}\tau_1;\tau_2) \qquad \Delta';\Gamma' \vdash e : \exists\alpha.\tau_1 \qquad \Delta' \vdash \tau_2}{\vdash \mathsf{unpack}\ \langle\alpha,x\rangle{=}e\ \mathsf{in}\ C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_2)}$$

$$(\underline{\textsc{Roll}})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau'[\mu\alpha.\tau'/\alpha])}{\vdash \mathsf{roll}\ C\ \mathsf{as}\ \mu\alpha.\tau' : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\mu\alpha.\tau')}$$

$$(\underline{\textsc{Unroll}})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\mu\alpha.\tau')}{\vdash \mathsf{unroll}\ C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau'[\mu\alpha.\tau'])}$$

$$(\textsc{New})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta',\alpha{\approx}\tau_1;\Gamma';\tau_2) \qquad \Delta' \vdash \tau_2 \qquad \Delta' \vdash \Gamma'}{\vdash \mathsf{new}\ \alpha{\approx}\tau'\ \mathsf{in}\ C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau_2)}$$

$$(\textsc{Conv})\frac{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau') \qquad \Delta' \vdash \tau' \approx \tau''}{\vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\Delta';\Gamma';\tau'')}$$

## Termination

$\boxed{\sigma; e\downarrow}$

$$\sigma; e\downarrow \stackrel{\text{def}}{\iff} \nexists\sigma', v.\ \sigma; e \hookrightarrow^* \sigma'; v$$

## Contextual Approximation

$\boxed{\Delta;\Gamma \vdash e_1 \preceq e_2 : \tau}$

$$\begin{aligned}\Delta;\Gamma \vdash e_1 \preceq e_2 : \tau \stackrel{\text{def}}{\iff}\ & \Delta;\Gamma \vdash e_1 : \tau \wedge \Delta;\Gamma \vdash e_2 : \tau \wedge \forall\sigma, C, \tau'.\\ & \vdash \sigma \wedge \vdash C : (\Delta;\Gamma;\tau) \rightsquigarrow (\sigma;\epsilon;\tau') \wedge \sigma; C[e_1]\downarrow \implies \sigma; C[e_2]\downarrow\end{aligned}$$

## Contextual Equivalence

$\boxed{\Delta;\Gamma \vdash e_1 \simeq e_2 : \tau}$

$$\Delta;\Gamma \vdash e_1 \simeq e_2 : \tau \stackrel{\text{def}}{\iff} \Delta;\Gamma \vdash e_1 \preceq e_2 : \tau \wedge \Delta;\Gamma \vdash e_2 \preceq e_1 : \tau$$

## A.5 Encoding Recursive Functions

### A.5.1 Using cast

$$\begin{array}{rcl} \mathsf{fix}'\, f(x).e : \tau_1 \to \tau_2 \text{ with } v_d & := & \lambda x_a{:}\tau_1.v\ (\forall\tau.\tau \to \tau_1 \to \tau_2)\ v\ x_a \\ \text{where } v & = & \Lambda\alpha.\lambda x_s{:}\tau.(\lambda f{:}(\tau_1 \to \tau_2).\lambda x{:}\tau_1.e)\ v' \\ \text{and } v' & = & \lambda x_a{:}\tau_1.(\mathsf{cast}\ \tau\ (\forall\alpha.\tau \to \tau_1 \to \tau_2)\ x_s\ v_d)\ x_a \end{array}$$

Due to cast's required default argument, $\mathsf{fix}'$ also needs to take a default value. Consequently, a fixed-point operator only exists for inhabited types. It is easy to verify the following two properties:

- $\sigma; (\mathsf{fix}'\, f(x).e : \tau_1 \to \tau_2 \text{ with } v_d)\ v \hookrightarrow^* \sigma; e[\mathsf{fix}'\, f(x).e : \tau_1 \to \tau_2 \text{ with } v_d/f][v/x]$, for any $\sigma$.
- If $\Delta; \Gamma, f{:}\tau_1 \to \tau_2, x{:}\tau_1 \vdash e : \tau_2$ and $\Delta; \Gamma \vdash v_d : \forall\alpha.\alpha \to \tau_1 \to \tau_2$, then $\Delta; \Gamma \vdash (\mathsf{fix}'\, f(x).e : \tau_1 \to \tau_2 \text{ with } v_d) : \tau_1 \to \tau_2$.

### A.5.2 Using Recursive Types

$$\begin{array}{rcl} \mathsf{fix}\, f(x).e : \tau_1 \to \tau_2 & := & \lambda x_a{:}\tau_1.v\ (\mathsf{roll}\ v\ \mathsf{as}\ \mu\tau.\tau \to \tau_1 \to \tau_2)\ x_a \\ \text{where } v & = & \lambda x_s{:}(\mu\tau.\tau \to \tau_1 \to \tau_2).(\lambda f{:}(\tau_1 \to \tau_2).\lambda x{:}\tau_1.e)\ (\lambda x_a{:}\tau_1.(\mathsf{unroll}\ x_s)\ x_s\ x_a) \end{array}$$

It is easy to verify the following two properties:

- $\sigma; (\mathsf{fix}\, f(x).e : \tau_1 \to \tau_2)\ v \hookrightarrow^* \sigma; e[\mathsf{fix}\, f(x).e : \tau_1 \to \tau_2/f][v/x]$, for any $\sigma$.
- If $\Delta; \Gamma, f{:}\tau_1 \to \tau_2, x{:}\tau_1 \vdash e : \tau_2$, then $\Delta; \Gamma \vdash (\mathsf{fix}\, f(x).e : \tau_1 \to \tau_2) : \tau_1 \to \tau_2$.