Under consideration for publication in J. Functional Programming

Parametric Polymorphism Through Run-time Sealing

or, Theorems for Low, Low Prices!

AMAL AHMED
Indiana University

LINDSEY KUPER
Indiana University

JACOB MATTHEWS
Google
(*e-mail:* amal@cs.indiana.edu, lkuper@cs.indiana.edu, jacobm@cs.uchicago.edu)

Abstract

We show how to extend System F's parametricity guarantee to a multi-language system in the style of Matthews and Findler that combines System F with an untyped language by use of dynamic sealing. While the use of sealing for this purpose has been suggested before, it has never been proven to preserve parametricity. In this paper we prove that it does using step-indexed logical relations.

Note: This is a draft from Fall 2011. It was updated in February 2017 to elide unfinished sections that did not pertain to the parametricity result.

1 Introduction

There have been two major strategies for hiding the implementation details of one part of a program from its other parts: the static approach and the dynamic approach.

The static approach can be summarized by the slogan "information hiding = parametric polymorphism." In it, the language's type system is equipped with a facility such as existential types so that it can reject programs in which one module makes unwarranted assumptions about the internal details of another, even if those assumptions happen to be true. This approach rests on Reynolds' notion of abstraction (Reynolds, 1983), later redubbed the "parametricity" theorem by Wadler (1989).

The dynamic approach, which goes back to Morris (1973), can be summarized by the alternate slogan "information hiding = local scope + generativity." Rather than statically rejecting programs that make unwarranted assumptions, the dynamic approach simply takes away programs' ability to see if those assumptions are correct. It allows a programmer to *dynamically seal* values by creating unique keys ($create\text{-}key : \rightarrow key$) and using those keys with locking and unlocking operations ($seal : v \times key \rightarrow opaque$ and $unseal : opaque \times key \rightarrow v$ respectively). A value locked with a particular key is opaque

to third parties: nothing can be done but unlock it with the same key. Here is a simple implementation written in Scheme, where **gensym** is a function that generates a new, completely unique symbol every time it is called:

```
(define (create-key) (gensym))
(define (seal v k1) (λ (k2) (if (eq? k1 k2) v (error))))
(define (unseal sealed-v k) (sealed-v k))
```

Using this facility a module can hand out a particular value while hiding its representation by creating a fresh key in its private lexical scope, sealing the value and handing the result to clients, and then unsealing it again whenever it returns. This is the primary information-hiding mechanism in many untyped languages. For instance PLT Scheme (Flatt, 1997) uses generative `structs`, essentially a (much) more sophisticated version of sealing, to build abstractions for a great variety of programming constructs such as an object system. Furthermore, the idea has seen some use recently even in languages whose primary information-hiding mechanism is static, as recounted by Sumii and Pierce (2004).

Both of these strategies seem to match an intuitive understanding of what information-hiding ought to entail. So it is surprising that a fundamental question — what is the relationship between the guarantee provided by the static approach and the dynamic approach? — has not been answered in the literature. In this paper we take a new perspective on the problem, posing it as a question of parametricity in a multi-language system (Matthews & Findler, 2007). After reviewing our previous work on multi-language systems and giving a multi-language system that combines System F (henceforth "ML") and an untyped call-by-value lambda calculus (henceforth "Scheme") (Section 2), we use this vantage point to show two results.

First, in Section 4 we show that dynamic sealing preserves ML's parametricity guarantee even when interoperating with Scheme. For the proof, we define two mutually dependent logical relations, one for ML and one for Scheme. Instead of presenting the logical relations outright, we first explain (in Section 3) some of the technical subtleties involved by considering a series of naive attempts at a definition. In particular, the relation for Scheme is indexed *not by types*, but by a natural number that, intuitively, records the number of steps available for future evaluation; this stratification is essential for modeling the recursive functions (available via encoding of fixed-point combinators) present in the language. Furthermore, one might expect the ML relation to be indexed only by types, especially since our ML language — *when considered alone* — does not contain any features that permit unbounded computation. However, indexing by types alone does not suffice in the multi-language setting since recursive functions are now available in ML via interaction with Scheme. Thus, the ML relation is indexed by steps as well as types, much like the relation for recursive types given by Ahmed (2006).

The more significant complication in the definition of the logical relation, however, arises due to dynamically sealed values which, from a semantic modeling perspective, bear a resemblance to dynamically allocated memory locations in a language with mutable references. Roughly, in order to decide whether two sealed values are related, we need to consult a world w that keeps track of information about all the keys that have been generated up to that point. Thus, we use a Kripke logical relation — i.e., a logical relation that is indexed by the aforementioned *possible worlds* w. Kripke logical relations are

e	::=	x | v | (e e) | (*op* e e) | (if0 e e e) | (*pd* e) | (cons e e) | (fst e) | (rst e) | (SM^{τ} e)	**e**	::=	**x** | **v** | (**e e**) | (*op* **e e**) | (**if0 e e e**) | (**cons e e**) | (**fst e**) | (**rst e**) | ($^{\tau}MS$ e)
v	::=	(λx. e) | $\overline{n}$ | nil | (cons v_1 v_2)	**v**	::=	(λ**x** : τ. **e**) | $\overline{n}$ | **nil** | (**cons** $\mathbf{v}_1$ $\mathbf{v}_2$)
op	::=	+ | −	*op*	::=	+ | −
pd	::=	proc? | nat? | nil? | pair?	**x**	::=	ML variables
x	::=	Scheme variables	**E**	::=	$[\,]_M$ | (**E e**) | (**v E**) | (*op* **E e**) | (*op* **v E**) | (**if0 E e e**) | (**cons E e**) | (**cons v E**) | (**fst E**) | (**rst E**) | ($^{\tau}MS$ E)
E	::=	$[\,]_S$ | (E e) | (v E) | (*op* E e) | (*op* v E) | (if0 E e e) | (*pd* E) | (cons E e) | (cons v E) | (fst E) | (rst E) | (SM^{τ} **E**)	τ	::=	**Nat** | τ^* | $\tau_1 \to \tau_2$

$\mathcal{E}$ ::= E | **E**

Fig. 1. Natural embedding of Scheme (left) and ML (right): grammar

essentially needed when reasoning about properties that only hold under certain conditions; the possible worlds keep track of those conditions, which in our setting have to do with relatedness of values protected by dynamic sealing.

Once the logical relations are in place, we prove the Fundamental Property for both relations (Section 4). The Fundamental Property of a logical relation — also known as the Basic Lemma — says that if a term is well-typed then it is logically related to itself. One novelty of our proof is its use of what we call the "bridge lemma," which states that if two terms are related in one language, then wrapping those terms in boundaries results in terms that are related in the other. The overall proof structure is otherwise essentially standard.

2 A brief introduction to multi-language systems

To make the present work self-contained, in this section we summarize some relevant material from earlier work (Matthews & Findler, 2007).

2.1 The natural embedding

The natural embedding multi-language system, presented in Figures 1–3, is a method of modeling the semantics of a minimal "ML" (simply-typed, call-by-value lambda calculus) with a minimal "Scheme" (untyped, call-by-value lambda calculus) such that both languages have natural access to foreign values. They receive foreign numbers as native numbers, and they can call foreign functions as native functions. Note that throughout this article we have typeset the terms, types, and contexts of our ML language using a **bold red font with serifs**, and those of our Scheme language with a blue sans-serif font. These font differences are semantically meaningful.

To the core languages we add new syntax, evaluation contexts, and reduction rules that define syntactic boundaries, written $^{\tau}MS$ (which may be read as "ML outside, Scheme inside") and SM^{τ} ("Scheme outside, ML inside"), to allow cross-language communication. The metavariable $\mathcal{E}$ denotes the top-level evaluation context: top-level programs may be ML programs that optionally call into Scheme (which means we would choose $\mathcal{E} = \mathbf{E}$), or

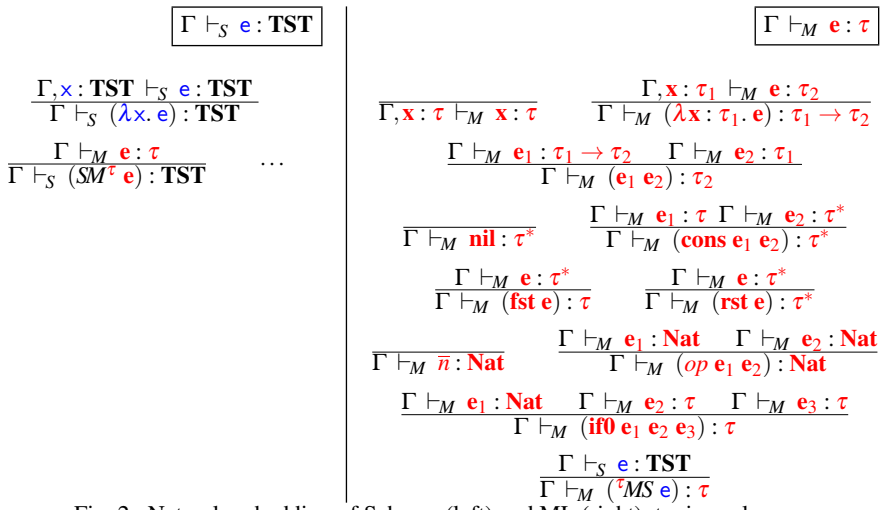

Fig. 2. Natural embedding of Scheme (left) and ML (right): typing rules

they may be Scheme programs that optionally call into ML (in which case we would let $\mathcal{E} = \mathsf{E}$).

We assume we can translate numbers from one language to the other, and give reduction rules for boundary-crossing numbers based on that assumption:

$$\mathcal{E}[(SM^{\mathbf{Nat}}\ \overline{n})]_S \longmapsto \mathcal{E}[\overline{n}] \qquad \mathcal{E}[({}^{\mathbf{Nat}}MS\ \overline{n})]_M \longmapsto \mathcal{E}[\overline{n}]$$

To convert procedures across languages, we use native proxy procedures. We represent a Scheme procedure in ML at type $\tau_1 \to \tau_2$ by a new procedure that takes an argument of type τ_1, converts it to a Scheme equivalent, runs the original Scheme procedure on that value, and then converts the result back to ML at type τ_2. For example, $({}^{\tau_1 \to \tau_2}MS\ (\lambda \mathsf{x}.\, \mathsf{e}))$ becomes $(\lambda \mathbf{x} : \tau_1.\, ({}^{\tau_2}MS\ (\lambda \mathsf{x}.\, \mathsf{e})\ (SM^{\tau_1}\ \mathbf{x})))$ and vice versa for Scheme to ML. Note that the boundary that converts the argument is an SM^{τ_1} boundary, not an ${}^{\tau_1}MS$ boundary—*i.e.*, the direction of conversion reverses for function arguments.

Whenever a Scheme value is converted to ML, we also check that value's first-order properties: we check to see if a Scheme value is a number before converting it to an ML value of type **Nat** and that it is a procedure value before converting it to an ML value of arrow type (and signal an error if either check fails).

Theorem 2.1 (*Natural embedding type safety (Matthews & Findler, 2007)*)
If $\vdash_M \mathbf{e} : \tau$, then either $\mathbf{e} \longmapsto^* \mathbf{v}$, $\mathbf{e} \longmapsto^* \mathbf{Error}$: *str*, or $\mathbf{e}$ diverges.

Prior work has shown that the dynamic checks in this system naturally give rise to higher-order contracts (Findler & Felleisen, 2002; Findler & Blume, 2006); in Section **??** we show another way of arriving at the same conclusion, this time equating a contract enforcing that an untyped term e behave as a (closed) type specification τ (which we write e^{τ}) by converting it to and from ML at that type: to a first approximation,

$$\mathsf{e}^{\tau} = (SM^{\tau}\ ({}^{\tau}MS\ \mathsf{e})).$$

$$
\begin{array}{lll}
\mathcal{E}[(\lambda \mathsf{x}.\ \mathsf{e})\ \mathsf{v}]_S & \longmapsto \mathcal{E}[\mathsf{e}[\mathsf{x} := \mathsf{v}]] & \\
\mathcal{E}[(\mathsf{v}_1\ \mathsf{v}_2)]_S & \longmapsto \textbf{Error}\text{: non-proc} & (\mathsf{v}_1 \neq (\lambda \mathsf{x}.\ \mathsf{e}) \text{ for any } \mathsf{x}, \mathsf{e}) \\
\mathcal{E}[(+\ \overline{n_1}\ \overline{n_2})]_S & \longmapsto \mathcal{E}[\overline{n_1 + n_2}] & \\
\mathcal{E}[(-\ \overline{n_1}\ \overline{n_2})]_S & \longmapsto \mathcal{E}[\overline{max(n_1 - n_2, 0)}] & \\
\mathcal{E}[(op\ \mathsf{v}_1\ \mathsf{v}_2)]_S & \longmapsto \textbf{Error}\text{: non-num} & (\mathsf{v}_1 \neq \overline{n} \text{ or } \mathsf{v}_2 \neq \overline{n}) \\
\mathcal{E}[(\mathsf{if0}\ \overline{0}\ \mathsf{e}_1\ \mathsf{e}_2)]_S & \longmapsto \mathcal{E}[\mathsf{e}_1] & \\
\mathcal{E}[(\mathsf{if0}\ \mathsf{v}\ \mathsf{e}_1\ \mathsf{e}_2)]_S & \longmapsto \mathcal{E}[\mathsf{e}_2] & (\mathsf{v} \neq \overline{0}) \\
\mathcal{E}[(\mathsf{proc?}\ (\lambda \mathsf{x}.\ \mathsf{e}))]_S & \longmapsto \mathcal{E}[\overline{0}] & \\
\mathcal{E}[(\mathsf{proc?}\ \mathsf{v})]_S & \longmapsto \mathcal{E}[\overline{1}] & (\mathsf{v} \neq (\lambda \mathsf{x}.\ \mathsf{e}) \text{ for any } \mathsf{x}, \mathsf{e}) \\
\mathcal{E}[(\mathsf{nat?}\ \overline{n})]_S & \longmapsto \mathcal{E}[\overline{0}] & \\
\mathcal{E}[(\mathsf{nat?}\ \mathsf{v})]_S & \longmapsto \mathcal{E}[\overline{1}] & (\mathsf{v} \neq \overline{n} \text{ for any } n) \\
\mathcal{E}[(\mathsf{nil?}\ \mathsf{nil})]_S & \longmapsto \mathcal{E}[\overline{0}] & \\
\mathcal{E}[(\mathsf{nil?}\ \mathsf{v})]_S & \longmapsto \mathcal{E}[\overline{1}] & (\mathsf{v} \neq \mathsf{nil}) \\
\mathcal{E}[(\mathsf{pair?}\ (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2))]_S & \longmapsto \mathcal{E}[\overline{0}] & \\
\mathcal{E}[(\mathsf{pair?}\ \mathsf{v})]_S & \longmapsto \mathcal{E}[\overline{1}] & (\mathsf{v} \neq (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2) \text{ for any } \mathsf{v}_1, \mathsf{v}_2) \\
\mathcal{E}[(\mathsf{fst}\ (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2))]_S & \longmapsto \mathcal{E}[\mathsf{v}_1] & \\
\mathcal{E}[(\mathsf{fst}\ \mathsf{v})]_S & \longmapsto \textbf{Error}\text{: non-pair} & (\mathsf{v} \neq (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2) \text{ for any } \mathsf{v}_1, \mathsf{v}_2) \\
\mathcal{E}[(\mathsf{rst}\ (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2))]_S & \longmapsto \mathcal{E}[\mathsf{v}_2] & \\
\mathcal{E}[(\mathsf{rst}\ \mathsf{v})]_S & \longmapsto \textbf{Error}\text{: non-pair} & (\mathsf{v} \neq (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2) \text{ for any } \mathsf{v}_1, \mathsf{v}_2) \\
\mathcal{E}[(SM^{\mathbf{Nat}}\ \overline{n})]_S & \longmapsto \mathcal{E}[\overline{n}] & \\
\mathcal{E}[(SM^{\tau_1 \to \tau_2}\ \mathbf{v})]_S & \longmapsto \mathcal{E}[(\lambda \mathsf{x}.\ (SM^{\tau_2}\ \mathbf{v}\ ({}^{\tau_1}MS\ \mathsf{x})))] & \\
\mathcal{E}[(SM^{\tau^*}\ \mathbf{nil})]_S & \longmapsto \mathcal{E}[\mathsf{nil}] & \\
\mathcal{E}[(SM^{\tau^*}\ (\mathbf{cons}\ \mathbf{v}_1\ \mathbf{v}_2))]_S & \longmapsto \mathcal{E}[(\mathsf{cons}\ (SM^{\tau}\ \mathbf{v}_1)\ (SM^{\tau^*}\ \mathbf{v}_2))] &
\end{array}
$$

$$
\begin{array}{lll}
\mathcal{E}[(\lambda \mathbf{x} : \tau.\ \mathbf{e})\ \mathbf{v}]_M & \longmapsto \mathcal{E}[\mathbf{e}[\mathbf{x} := \mathbf{v}]] & \\
\mathcal{E}[(+\ \overline{n_1}\ \overline{n_2})]_M & \longmapsto \mathcal{E}[\overline{n_1 + n_2}] & \\
\mathcal{E}[(-\ \overline{n_1}\ \overline{n_2})]_M & \longmapsto \mathcal{E}[\overline{max(n_1 - n_2, 0)}] & \\
\mathcal{E}[(\mathbf{if0}\ \overline{0}\ \mathbf{e}_1\ \mathbf{e}_2)]_M & \longmapsto \mathcal{E}[\mathbf{e}_1] & \\
\mathcal{E}[(\mathbf{if0}\ \overline{n}\ \mathbf{e}_1\ \mathbf{e}_2)]_M & \longmapsto \mathcal{E}[\mathbf{e}_2] & (\overline{n} \neq 0) \\
\mathcal{E}[(\mathbf{fst}\ (\mathbf{cons}\ \mathbf{v}_1\ \mathbf{v}_2))]_M & \longmapsto \mathcal{E}[\mathbf{v}_1] & \\
\mathcal{E}[(\mathbf{fst}\ \mathbf{nil})]_M & \longmapsto \textbf{Error}\text{: nil} & \\
\mathcal{E}[(\mathbf{rst}\ (\mathbf{cons}\ \mathbf{v}_1\ \mathbf{v}_2))]_M & \longmapsto \mathcal{E}[\mathbf{v}_2] & \\
\mathcal{E}[(\mathbf{rst}\ \mathbf{nil})]_M & \longmapsto \textbf{Error}\text{: nil} & \\
\mathcal{E}[({}^{\mathbf{Nat}}MS\ \overline{n})]_M & \longmapsto \mathcal{E}[\overline{n}] & \\
\mathcal{E}[({}^{\mathbf{Nat}}MS\ \mathsf{v})]_M & \longmapsto \textbf{Error}\text{: non-num} & (\mathsf{v} \neq \overline{n} \text{ for any } \overline{n}) \\
\mathcal{E}[({}^{\tau_1 \to \tau_2}MS\ (\lambda \mathsf{x}.\ \mathsf{e}))]_M & \longmapsto \mathcal{E}[(\lambda \mathbf{x} : \tau_1.\ ({}^{\tau_2}MS\ \lambda \mathsf{x}.\ \mathsf{e}\ (SM^{\tau_1}\ \mathbf{x})))] & \\
\mathcal{E}[({}^{\tau_1 \to \tau_2}MS\ \mathsf{v})]_M & \longmapsto \textbf{Error}\text{: non-proc} & (\mathsf{v} \neq (\lambda \mathsf{x}.\ \mathsf{e}) \text{ for any } \mathsf{x}, \mathsf{e}) \\
\mathcal{E}[({}^{\tau^*}MS\ \mathsf{nil})]_M & \longmapsto \mathcal{E}[\mathbf{nil}] & \\
\mathcal{E}[({}^{\tau^*}MS\ (\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}_2))]_M & \longmapsto \mathcal{E}[(\mathbf{cons}\ ({}^{\tau}MS\ \mathsf{v}_1)\ ({}^{\tau^*}MS\ \mathsf{v}_2))] & \\
\mathcal{E}[({}^{\tau^*}MS\ \mathsf{v})]_M & \longmapsto \textbf{Error}\text{: non-list} & (\mathsf{v} \text{ is not a pair or } \mathsf{nil})
\end{array}
$$

Fig. 3. Natural embedding of Scheme (top) and ML (bottom): operational semantics

2.2 Polymorphism, attempt one

An omission from the "ML" side of the natural embedding to this point is that it contains no polymorphism. We now extend it to support polymorphism by replacing the simply-typed lambda calculus with System F. When we do so, we immediately hit the question of how to properly handle boundaries. In this subsection, we make what we consider the most straightforward decision of how to handle boundaries and show that it results in a system

Syntax

$$\begin{array}{rcl} \mathbf{e} & ::= & \cdots \mid \Lambda\alpha.\,\mathbf{e} \mid \mathbf{e}\ \tau \\ \mathbf{v} & ::= & \cdots \mid \Lambda\alpha.\,\mathbf{e} \mid ({}^{\mathbf{L}}MS\ \mathsf{v}) \\ \tau & ::= & \cdots \mid \forall\alpha.\,\tau \mid \alpha \mid \mathbf{L} \\ \Delta & ::= & \bullet \mid \Delta, \tau \\ \mathbf{E} & ::= & \cdots \mid \mathbf{E}\ \tau \end{array}$$

Typing rules $\boxed{\Delta;\Gamma \vdash_M \mathbf{e} : \tau}$

$$\frac{ftv(\tau) \in \Delta}{\Delta \vdash \tau} \qquad \frac{\Delta, \alpha; \Gamma \vdash_M \mathbf{e} : \tau}{\Delta;\Gamma \vdash_M \Lambda\alpha.\,\mathbf{e} : \forall\alpha.\,\tau} \qquad \frac{\Delta;\Gamma \vdash_M \mathbf{e} : \forall\alpha.\,\tau' \quad \Delta \vdash \tau}{\Delta;\Gamma \vdash_M \mathbf{e}\ \tau : \tau'[\alpha := \tau]}$$

Operational semantics $\boxed{e \longmapsto e'}$

$$\begin{array}{rcl} \mathcal{E}[(\Lambda\alpha.\,\mathbf{e})\ \tau]_M & \longmapsto & \mathcal{E}[\mathbf{e}[\alpha := \tau]] \\ \mathcal{E}[({}^{\forall\alpha.\,\tau}MS\ \mathsf{v})]_M & \longmapsto & \mathcal{E}[(\Lambda\alpha.\,({}^{\tau}MS\ \mathsf{v}))] \\ \mathcal{E}[(SM^{\forall\alpha.\,\tau}\ \mathbf{v})]_S & \longmapsto & \mathcal{E}[(SM^{\tau[\alpha:=\mathbf{L}]}\ \mathbf{v}\ \mathbf{L})] \\ \mathcal{E}[(SM^{\mathbf{L}}\ ({}^{\mathbf{L}}MS\ \mathsf{v}))]_S & \longmapsto & \mathcal{E}[\mathsf{v}] \end{array}$$

Fig. 4. Extensions to Figures 1–3 for non-parametric polymorphism

that does not preserve System F's parametricity property; in the next subsection we refine our strategy using dynamic sealing techniques.

Figure 4 shows the extensions we need to make to Figures 1–3 to support non-parametric polymorphism. To ML's syntax we add type abstractions ($\Lambda\alpha.\,\mathbf{e}$) and type application ($\mathbf{e}\ \tau$); to its types we add $\forall\alpha.\,\tau$ and α. A new evaluation context $\mathbf{E}\ \tau$ governs the evaluation of type applications, and a type environment Δ keeps track of the scope of type variables. We write $ftv(\tau)$ to denote the free type variables of τ.

Our embedding converts Scheme functions that work polymorphically into polymorphic ML values, and converts ML type abstractions directly into plain Scheme functions that behave polymorphically. For example, ML might receive the Scheme function $(\lambda\mathsf{x}.\,\mathsf{x})$ from a boundary with type $\forall\alpha.\,\alpha \to \alpha$ and use it successfully as an identity function, and Scheme might receive the ML type abstraction $(\Lambda\alpha.\,\lambda\mathbf{x} : \alpha.\,\mathbf{x})$ as a regular function that behaves as the identity function for any value Scheme gives it.

To support this behavior, the model must create a type abstraction from a regular Scheme value when converting from Scheme to ML, and must drop a type abstraction when converting from ML to Scheme. The former is straightforward: we reduce a redex of the form $({}^{\forall\alpha.\,\tau}MS\ \mathsf{v})$ by dropping the universal quantifier on the type in the boundary and binding the now-free type variable in τ by wrapping the entire expression in a Λ form, yielding $(\Lambda\alpha.\,({}^{\tau}MS\ \mathsf{v}))$.

This works for ML, but making a dual of it in Scheme would be somewhat silly, since every Scheme value inhabits the same type so type abstraction and application forms would be useless. Instead, we would like to allow Scheme to use an ML value of type, say, $\forall\alpha.\,\alpha \to \alpha$ directly as a function. To make boundaries with universally-quantified types behave that way, when we convert a polymorphic ML value to a Scheme value we need to remove its initial type-abstraction by applying it to some type and then convert the resulting value according to the resulting type. As for which type to apply it to, we need

a type to which we can reliably convert any Scheme value, though it must not expose any of those values' properties. In prior work, we used the "lump" type to represent arbitrary, opaque Scheme values in ML; we reuse it here as the argument to the ML type abstraction. More specifically, we add **L** as a new base type in ML and we add the cancellation rule for lumps to the set of reductions. These changes, along with all the other additions required to support polymorphism, are summarized in Figure 4.

Theorem 2.2
The polymorphic natural embedding is type-sound.

2.3 Polymorphism, attempt two

Although the embedding shown in Figure 4 is type safe, the polymorphism is not parametric in the sense of Reynolds (1983). We can see this with an example: it is well-known that in System F, for which parametricity holds, the only value with type $\forall\alpha.\ \alpha \to \alpha$ is the polymorphic identity function. In the system we have built so far, though, the term

$$({}^{\forall\alpha.\ \alpha\to\alpha}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x})))$$

has type $\forall\alpha.\ \alpha \to \alpha$, but it evaluates to a type abstraction which when applied to the type **Nat** evaluates as follows:

$$\begin{array}{ll} & (\Lambda\alpha.\ ({}^{\alpha\to\alpha}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x})))\ \mathbf{Nat} \\ \longmapsto & ({}^{\mathbf{Nat}\to\mathbf{Nat}}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x}))) \\ \longmapsto & (\lambda \mathbf{y} : \mathbf{Nat}.\ ({}^{\mathbf{Nat}}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x}))\ (SM^{\mathbf{Nat}}\ \mathbf{y}))) \end{array}$$

Since ML requires that the argument to this function be a number, this is equivalent to

$$(\lambda \mathbf{y} : \mathbf{Nat}.\ ({}^{\mathbf{Nat}}MS\ (\lambda \mathsf{x}.\ (+\ \mathsf{x}\ \overline{1}))\ (SM^{\mathbf{Nat}}\ \mathbf{y})))$$

which is well-typed but is not the identity function.

The problem with the misbehaving $\forall\alpha.\ \alpha \to \alpha$ function above is that while the type system rules out ML fragments that try to treat values of type α non-generically, embedded Scheme programs are still able to observe the concrete choice made for α and act accordingly. To restore parametricity, we use *dynamic sealing* to protect ML values whose implementation should be hidden from Scheme. When we apply a Scheme function to an SM-wrapped ML value whose type is the instantiation of what was originally a type variable, such as in the application of $(\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x}))$ to $(SM^{\mathbf{Nat}}\ \mathbf{y})$ in our example, we generate a new key and provide Scheme with an opaque sealed value about which Scheme cannot make any observations. When Scheme returns a value to ML at a type that was originally α, ML unseals it or signals an error if it is not a sealed value with the appropriate key.

We formalize this notion in Figure 5. To do so, we need to make several changes to our framework. Rather than directly substituting types for free type variables on boundary annotations, we introduce type-like annotations of the form $\langle k; \tau\rangle$ that indicate on a boundary's type annotation that a particular type τ is the instantiation of what was originally a type variable. These *key annotations* allow us to remember when a type was originally abstract. The *key* k appearing inside a key annotation is a unique value generated during the evaluation of type applications.

Syntax

$$
\begin{array}{rcl}
\mathbf{e} & ::= & \cdots \mid \Lambda\alpha.\,\mathbf{e} \mid \mathbf{e}\ \tau \mid ({}^{\kappa}MS\ \mathsf{e}) \\
\mathsf{e} & ::= & \cdots \mid (SM^{\kappa}\ \mathbf{e}) \\
\mathbf{v} & ::= & \cdots \mid \Lambda\alpha.\,\mathbf{e} \mid ({}^{\mathbf{L}}MS\ \mathsf{v}) \\
\mathsf{v} & ::= & \cdots \mid (SM^{\langle k;\tau\rangle}\ \mathbf{v}) \\
\tau & ::= & \cdots \mid \forall\alpha.\,\tau \mid \alpha \mid \mathbf{L} \\
\kappa & ::= & \mathbf{Nat} \mid \kappa_1 \to \kappa_2 \mid \kappa^* \mid \forall\alpha.\,\kappa \mid \alpha \mid \mathbf{L} \mid \langle k;\tau\rangle \\
\Delta & ::= & \bullet \mid \Delta, \tau \\
\mathbf{E} & ::= & \cdots \mid \mathbf{E}\ \tau \mid ({}^{\kappa}MS\ \mathsf{E}) \\
\mathsf{E} & ::= & \cdots \mid (SM^{\kappa}\ \mathbf{E})
\end{array}
$$

Key erasure $\boxed{|\kappa| = \tau}$

$$
\begin{array}{rcl}
|\mathbf{Nat}| & = & \mathbf{Nat} \\
|\kappa_1 \to \kappa_2| & = & |\kappa_1| \to |\kappa_2| \\
|\kappa^*| & = & |\kappa|^* \\
|\forall\alpha.\,\kappa| & = & \forall\alpha.\,|\kappa| \\
|\alpha| & = & \alpha \\
|\mathbf{L}| & = & \mathbf{L} \\
|\langle k;\tau\rangle| & = & \tau
\end{array}
$$

Typing rules $\boxed{\Delta;\Gamma \vdash_M \mathbf{e} : \tau}$ $\boxed{\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}}$

$$\frac{\Delta,\alpha;\Gamma \vdash_M \mathbf{e} : \tau}{\Delta;\Gamma \vdash_M (\Lambda\alpha.\,\mathbf{e}) : \forall\alpha.\,\tau} \qquad \frac{\Delta;\Gamma \vdash_M \mathbf{e} : \forall\alpha.\,\tau' \quad \Delta \vdash \tau}{\Delta;\Gamma \vdash_M \mathbf{e}\ \tau : \tau'[\alpha := \tau]}$$

$$\frac{\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST} \quad \Delta \vdash |\kappa|}{\Delta;\Gamma \vdash_M ({}^{\kappa}MS\ \mathsf{e}) : |\kappa|} \qquad \frac{\Delta;\Gamma \vdash_M \mathbf{e} : |\kappa| \quad \Delta \vdash |\kappa|}{\Delta;\Gamma \vdash_S (SM^{\kappa}\ \mathbf{e}) : \mathbf{TST}}$$

Well-typedness of configurations $\boxed{\Delta;\Gamma \vdash_M K \triangleright \mathbf{e} : \tau}$ $\boxed{\Delta;\Gamma \vdash_S K \triangleright \mathsf{e} : \mathbf{TST}}$

$$\frac{keys(\mathbf{e}) \in K \quad \Delta \cap K = \emptyset \quad \Delta;\Gamma \vdash_M \mathbf{e} : \tau}{\Delta;\Gamma \vdash_M K \triangleright \mathbf{e} : \tau} \qquad \frac{keys(\mathsf{e}) \in K \quad \Delta \cap K = \emptyset \quad \Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}}{\Delta;\Gamma \vdash_S K \triangleright \mathsf{e} : \mathbf{TST}}$$

Operational semantics $\boxed{K \triangleright e \longmapsto K' \triangleright e'}$ $\boxed{K \triangleright e \longmapsto \mathbf{Error}\text{: str}}$

$$
\begin{array}{lcll}
K \triangleright (\Lambda\alpha.\,\mathbf{e})\ \tau & \longmapsto & K, k \triangleright \mathbf{e}[\alpha := \langle k;\tau\rangle] & \text{where } k \notin K \\
K \triangleright ({}^{\forall\alpha.\,\kappa}MS\ \mathsf{v}) & \longmapsto & K \triangleright \Lambda\alpha.\,({}^{\kappa}MS\ \mathsf{v}) & \\
K \triangleright ({}^{\langle k;\tau\rangle}MS\ (SM^{\langle k;\tau\rangle}\ \mathbf{v})) & \longmapsto & K \triangleright \mathbf{v} & \\
K \triangleright ({}^{\langle k;\tau\rangle}MS\ \mathsf{v}) & \longmapsto & \mathbf{Error}\text{: bad value} & \text{where } \mathsf{v} \neq (SM^{\langle k;\tau\rangle}\ \mathbf{v}) \text{ for any } \mathbf{v} \\
K \triangleright (SM^{\forall\alpha.\,\kappa}\ \mathbf{v}) & \longmapsto & K \triangleright (SM^{\kappa[\alpha:=\mathbf{L}]}\ \mathbf{v}\ \mathbf{L}) & \\
K \triangleright (SM^{\mathbf{L}}\ ({}^{\mathbf{L}}MS\ \mathsf{v})) & \longmapsto & K \triangleright \mathsf{v} &
\end{array}
$$

Fig. 5. Extensions to Figures 1–3 to support parametric polymorphism

To accommodate key annotations, we generalize our system to use *conversion schemes* κ rather than type annotations on boundary expressions. Conversion schemes are similar to type annotations, but may also contain key annotations. In the typing rules for both *MS* and *SM* boundary expressions, we must stipulate that conversion schemes on boundaries are

well-formed. However, since the type environment Δ knows nothing of key annotations, the premise $\Delta \vdash |\kappa|$ uses the *key erasure* metafunction $|.|$ that takes conversion schemes to types. Key annotations $\langle k; \tau \rangle$ are "erased" to τ, and $|.|$ recurs structurally on other types.

We want the operational semantics to keep track of dynamically generated keys k. Hence, rather than having our reduction rules operate on expressions as in Figures 1–3 and 4, we adopt reduction rules that relate *configurations* $K \triangleright e$. A configuration $K \triangleright e$ comprises an expression e and a *key store* K which is the set of keys k that have been generated thus far during evaluation. When we evaluate a type application $(\Lambda\alpha.\ \mathbf{e})\ \tau$, we generate a unique key k, add k to K, and perform a *sealing substitution* $\mathbf{e}[\alpha := \langle k; \tau \rangle]$. Sealing substitution is defined on conversion schemes κ and on expressions e (of either language). Figure A 1 in the Appendix spells out the definition of sealing substitution in full detail; the following two paragraphs cover the interesting cases.

A sealing substitution $\kappa[\alpha := \langle k; \tau \rangle]$ replaces occurrences of α in κ with the key annotation $\langle k; \tau \rangle$. The most interesting cases are when κ is a type abstraction, a type variable, or itself a key annotation; we summarize these cases below:

$$\begin{array}{rcll}
(\forall\alpha'.\ \kappa)[\alpha := \langle k; \tau \rangle] & = & \forall\alpha'.\ \kappa[\alpha := \langle k; \tau \rangle] & (\text{if } \alpha' \neq \alpha) \\
 & = & \forall\alpha.\ \kappa & (\text{if } \alpha' = \alpha) \\
\alpha'[\alpha := \langle k; \tau \rangle] & = & \alpha' & (\text{if } \alpha' \neq \alpha) \\
 & = & \langle k; \tau \rangle & (\text{if } \alpha' = \alpha) \\
\langle k'; \tau' \rangle[\alpha := \langle k; \tau \rangle] & = & \langle k'; \tau' \rangle &
\end{array}$$

A sealing substitution $e[\alpha := \langle k; \tau \rangle]$ replaces occurrences of α in e (where e can be an expression of either language, since α may occur behind an SM boundary in Scheme) with the key annotation $\langle k; \tau \rangle$. The interesting cases are as follows:

$$\begin{array}{rcll}
(\lambda\mathbf{x} : \tau_1.\ \mathbf{e})[\alpha := \langle k; \tau \rangle] & = & \lambda\mathbf{x} : \tau_1[\alpha := \tau].\ \mathbf{e}[\alpha := \langle k; \tau \rangle] & \\
(\Lambda\alpha'.\ \mathbf{e})[\alpha := \langle k; \tau \rangle] & = & \Lambda\alpha'.\ \mathbf{e}[\alpha := \langle k; \tau \rangle] & (\text{if } \alpha' \neq \alpha) \\
 & = & \Lambda\alpha'.\ \mathbf{e} & (\text{if } \alpha' = \alpha) \\
(\mathbf{e}\ \tau_1)[\alpha := \langle k; \tau \rangle] & = & \mathbf{e}[\alpha := \langle k; \tau \rangle]\ \tau_1[\alpha := \tau] & \\
({}^{\kappa}MS\ \mathsf{e})[\alpha := \langle k; \tau \rangle] & = & ({}^{\kappa[\alpha := \langle k; \tau \rangle]}MS\ \mathsf{e}[\alpha := \langle k; \tau \rangle]) & \\
(SM^{\kappa}\ \mathbf{e})[\alpha := \langle k; \tau \rangle] & = & (SM^{\kappa[\alpha := \langle k; \tau \rangle]}\ \mathbf{e}[\alpha := \langle k; \tau \rangle]) &
\end{array}$$

Note that when we perform sealing substitution on an ML term $\lambda\mathbf{x} : \tau_1.\ \mathbf{e}$, we only substitute the type τ (rather than the whole key annotation $\langle k; \tau \rangle$) for occurrences of α appearing in $\mathbf{x}$'s type annotation τ_1. Similarly, when substituting into a type application $\mathbf{e}\ \tau_1$, we only substitute τ for α in τ_1.

Theorem 2.3

The embedding presented in Figure 5 is type-sound.

Returning to our earlier $\forall\alpha.\ \alpha \to \alpha$ example under our newly defined model of polymorphism, we see that the term

$$({}^{\forall\alpha.\ \alpha \to \alpha}MS\ (\lambda\mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x})))$$

is not rejected statically; rather, it *becomes* parametric at run-time. When applied to **Nat** as before, evaluation proceeds as follows:[1]

$$\begin{array}{ll} & (\Lambda\alpha.\ ({}^{\alpha\to\alpha}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x})))) \ \mathbf{Nat} \\ \longmapsto & ({}^{\langle k;\mathbf{Nat}\rangle\to\langle k;\mathbf{Nat}\rangle}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x}))) \\ \longmapsto & (\lambda \mathbf{y} : \mathbf{Nat}.\ ({}^{\langle k;\mathbf{Nat}\rangle}MS\ (\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x}))\ (SM^{\langle k;\mathbf{Nat}\rangle}\ \mathbf{y}))) \end{array}$$

When $(\lambda \mathsf{x}.\ (\mathsf{if0}\ (\mathsf{nat?}\ \mathsf{x})\ (+\ \mathsf{x}\ \overline{1})\ \mathsf{x}))$ is applied to $(SM^{\langle k;\mathbf{Nat}\rangle}\ \mathbf{y})$, Scheme's nat? predicate will not recognize the opaque, sealed value as a **Nat** and so must take the second branch of the if0 expression. Therefore the entire expression evaluates to

$$(\lambda \mathbf{y} : \mathbf{Nat}.\ ({}^{\langle k;\mathbf{Nat}\rangle}MS\ (SM^{\langle k;\mathbf{Nat}\rangle}\ \mathbf{y})))$$

which under our operational semantics evaluates to $(\lambda \mathbf{y} : \mathbf{Nat}.\ \mathbf{y})$, the identity function for **Nat**.

In the above example, we were lucky to have encountered an x in the second branch of the if0 expression, allowing us to return the sealed value that the ML context was expecting. If we had some other expression in place of x, say, the constant $\overline{5}$, evaluation would reach

$$(\lambda \mathbf{y} : \mathbf{Nat}.\ ({}^{\langle k;\mathbf{Nat}\rangle}MS\ \overline{5}))$$

which steps to a "bad value" error under our operational semantics, since $\overline{5}$ is not even an *SM* boundary expression, let alone one annotated with the necessary $\langle k; \mathbf{Nat}\rangle$. Therefore our system enforces parametricity *dynamically*, in this case by raising a run-time error rather than permitting a term of type $\forall\alpha.\ \alpha\to\alpha$ to behave differently from the identity function.

3 Proving parametricity: a naive approach

The examples of the preceding section suggest that the language of Figure 5 is parametric — in the sense that all terms in the language map related environments to related results — but they do not *prove* that that is the case. Our next step will be to attempt to rigorously establish parametricity. Our approach will be to define two logical relations, one for ML and one for Scheme, and prove the fundamental property for each relation.

Our goal will not be to establish the exact same equivalences that would hold for terms in plain System F, but only because the embedding we are considering gives terms the power to diverge and to signal errors, as we've seen from the examples considered so far. For instance, we cannot show that any ML value of type $\forall\alpha.\ \alpha\to\alpha$ must be the identity function, but we *can* show that it must be either the identity function, the function that always diverges, or the function that always signals an error.

The reader who is acquainted with logical relations for System F will find our first attempt at a logical relation for ML to be almost entirely familiar. The only differences will be for the lump type **L** and the universal type $\forall\alpha.\ \tau$, as we explain in section 3.2.

[1] Although we've redefined the reduction relation $\longmapsto$ to operate on configurations rather than just expressions, we will disregard the key store for the moment and still speak of expressions evaluating to expressions. The key store will come into play when we *prove* parametricity in the following section.

3.1 Interpreting types as relations

We start by establishing a notion of relatedness for terms in our ML language. For each type τ in the language, we define its interpretation $\mathcal{V}_M[\![\tau]\!]$ as a relation on values inhabiting that type. In particular, $\mathcal{V}_M[\![\tau]\!]$ is defined by induction on the structure of ML types τ. Our definition of $\mathcal{V}_M[\![.]\!]$ is also parameterized by a type interpretation δ, which we defer discussion of until the next subsection.

The simplest of our interpretations will be for our base type, **Nat**. As we might expect, two values of type **Nat** are related if they are the same natural number:

$$\mathcal{V}_M[\![\mathbf{Nat}]\!]\delta \stackrel{\text{def}}{=} \{(\overline{n},\overline{n}) \mid \overline{n} \in \mathbb{N}\}$$

For values of list type, **nil** is related to itself, and two lists of values are related if their elements are related pairwise:

$$\begin{array}{rcl} \mathcal{V}_M[\![\tau^*]\!]\delta & \stackrel{\text{def}}{=} & \{(\mathbf{nil},\mathbf{nil})\} \\ & \cup & \{((\mathbf{cons}\ \mathbf{v}_1\ \mathbf{v'}_1),(\mathbf{cons}\ \mathbf{v}_2\ \mathbf{v'}_2)) \mid \\ & & \quad (\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta \wedge (\mathbf{v'}_1,\mathbf{v'}_2) \in \mathcal{V}_M[\![\tau^*]\!]\delta\} \end{array}$$

Next we define relatedness for values of function type. Intuitively, we consider two function values to be related if, when applied to related arguments, they produce related results. The following definition captures this intuition:

$$\begin{array}{rcl} \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta & \stackrel{\text{def}}{=} & \{(\lambda\mathbf{x}:\tau_1.\,\mathbf{e}_1, \lambda\mathbf{x}:\tau_1.\,\mathbf{e}_2) \mid \\ & & \quad \forall(\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1]\!]\delta. \\ & & \quad\quad (\mathbf{e}_1[\mathbf{x}:=\mathbf{v}_1],\mathbf{e}_2[\mathbf{x}:=\mathbf{v}_2]) \in \mathcal{E}_M[\![\tau_2]\!]\delta\} \end{array}$$

Here, since $\mathbf{e}_1[\mathbf{x}:=\mathbf{v}_1]$ and $\mathbf{e}_2[\mathbf{x}:=\mathbf{v}_2]$ are not generally values, we must say that they are expressions related at the type τ_2 — that is, that they belong to $\mathcal{E}_M[\![\tau_2]\!]\delta$ which is a relation on expressions. The relation $\mathcal{E}_M[\![.]\!]$ is readily defined in terms of $\mathcal{V}_M[\![.]\!]$: we say that two expressions of type τ are related if they evaluate to related values of type τ under the configuration-based operational semantics we defined in Section 2.3.

$$\begin{array}{rcl} \mathcal{E}_M[\![\tau]\!]\delta & \stackrel{\text{def}}{=} & \{(\mathbf{e}_1,\mathbf{e}_2) \mid \forall\psi_1,\psi'_1,\mathbf{v}_1.\ \psi_1 \triangleright \mathbf{e}_1 \longmapsto^* \psi'_1 \triangleright \mathbf{v}_1 \Longrightarrow \\ & & \quad \exists\psi_2,\psi'_2,\mathbf{v}_2.\ \psi_2 \triangleright \mathbf{e}_2 \longmapsto^* \psi'_2 \triangleright \mathbf{v}_2 \wedge (\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta\} \end{array}$$

Note that the definition of $\mathcal{V}_M[\![\tau_1 \to \tau_2]\!]$ is well founded: the definition refers to $\mathcal{V}_M[\![.]\!]$ at the types τ_1 and τ_2, both of which are smaller than the type $\tau_1 \to \tau_2$.

3.2 Handling polymorphic types

So far, we have defined relatedness for ML values of base type, list type, and function type. We have not yet handled the types that make polymorphism possible in our multi-language system: type variables α, the type $\forall\alpha.\ \tau$ of type abstractions, and the lump type $\mathbf{L}$.

Let us consider the α case first: when are two values related at the type α? If we encountered two values $\mathbf{v}_1$ and $\mathbf{v}_2$ "of type α", it would happen while evaluating the respective bodies of two type abstractions that had already been applied to concrete types, say, τ_1 and τ_2. Then α would have already been instantiated with τ_1 and τ_2, respectively. So the interpretation of the type α must be some binary relation $\mathbf{R}$ containing pairs of

closed values $(\mathbf{v}_1, \mathbf{v}_2)$ where $\mathbf{v}_1$ is of type τ_1 and $\mathbf{v}_2$ is of type τ_2. We will use the notation $\mathbf{R} : \tau_1 \leftrightarrow \tau_2$ to indicate that a relation $\mathbf{R}$ has this property:

$$\mathbf{R} : \tau_1 \leftrightarrow \tau_2 \quad \stackrel{\text{def}}{\Leftrightarrow} \quad \forall (\mathbf{v}_1, \mathbf{v}_2) \in \mathbf{R}.\ \emptyset \vdash_M \mathbf{v}_1 : \tau_1 \ \wedge\ \emptyset \vdash_M \mathbf{v}_2 : \tau_2$$

The particular $\mathbf{R}$ we want for $\mathcal{V}_M[\![\alpha]\!]$ will partly depend on the concrete types τ_1 and τ_2 that will have been chosen. Here is where the type interpretation δ comes into play: it maps type variables α to triples $(\tau_1, \tau_2, \mathbf{R})$, where τ_1 and τ_2 are the types used to instantiate α, and $\mathbf{R}$ relates values of types τ_1 and τ_2 as described above. (More generally, whenever we write $\mathcal{V}_M[\![\tau]\!]\delta$, δ must provide a mapping for all type variables that might appear free in τ.)

We can then concisely define the interpretation $\mathcal{V}_M[\![\alpha]\!]$ as the result of looking up α in δ and then projecting out the third component of the resulting $(\tau_1, \tau_2, \mathbf{R})$ triple. We use the notation $\delta(\alpha).\mathbf{R}$ as shorthand for "look up α in δ, then extract $\mathbf{R}$":

$$\mathcal{V}_M[\![\alpha]\!]\delta \quad \stackrel{\text{def}}{=} \quad \delta(\alpha).\mathbf{R}$$

Now that we have described δ, it is straightforward to define an interpretation for $\forall\alpha.\ \tau$ types. When two expressions $\Lambda\alpha.\ \mathbf{e}_1$ and $\Lambda\alpha.\ \mathbf{e}_2$ are related at the type $\forall\alpha.\ \tau$, intuitively it means that *if we provide* an interpretation for α—which, as we have already said, is simply a relation $\mathbf{R} : \tau_1 \leftrightarrow \tau_2$ for some τ_1 and τ_2—then applying $\Lambda\alpha.\ \mathbf{e}_1$ and $\Lambda\alpha.\ \mathbf{e}_2$ to τ_1 and τ_2 respectively will result in two expressions that are related with respect to the $\mathbf{R}$ we provided. We formalize this notion with the following definition:

$$\begin{aligned}\mathcal{V}_M[\![\forall\alpha.\ \tau]\!]\delta \quad \stackrel{\text{def}}{=} \quad \{&(\Lambda\alpha.\ \mathbf{e}_1, \Lambda\alpha.\ \mathbf{e}_2) \mid \\ &\forall\tau_1, \tau_2, \mathbf{R}.\ \mathbf{R} : \tau_1 \leftrightarrow \tau_2 \implies \\ &(\mathbf{e}_1[\alpha := \langle k_1; \tau_1\rangle], \mathbf{e}_2[\alpha := \langle k_2; \tau_2\rangle]) \in \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]\}\end{aligned}$$

Note that here we perform sealing substitution, as described in section 2.3, since that is what happens upon type application according to our operational semantics. The resulting expressions are related at the type τ, but under an extended δ that now contains a binding for the α that may appear free in τ.

Our use of sealing substitution rather than ordinary type substitution in the above definition is the first point at which our logical relation has differed from a standard logical relation for System F. We now address the other non-standard aspect of our system: relatedness at the lump type $\mathbf{L}$. In our system, the only values of type $\mathbf{L}$ are *MS* boundary expressions that have been annotated with $\mathbf{L}$ and that wrap a Scheme value. When should we say that two such values, say, $({}^{\mathbf{L}}MS\ \mathsf{v}_1)$ and $({}^{\mathbf{L}}MS\ \mathsf{v}_2)$, are related? We have little choice but to define their relatedness in terms of the relatedness of v_1 and v_2, as follows:

$$\mathcal{V}_M[\![\mathbf{L}]\!]\delta \quad \stackrel{\text{def}}{=} \quad \{(({}^{\mathbf{L}}MS\ \mathsf{v}_1), ({}^{\mathbf{L}}MS\ \mathsf{v}_2)) \mid (\mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S\}$$

but this merely puts off the problem, as it requires us to define a relation $\mathcal{V}_S$ on Scheme values. We handle the definition of $\mathcal{V}_S$ in the next subsection.

So far, in the interest of simplicity, our definitions have ignored the possibility of terms that raise errors. For instance, a precise definition would say that two expressions are related at the type $\tau_1 \to \tau_2$ if, when passed arguments related at τ_1, they both diverge *or* both raise the same error *or* both evaluate to values related at τ_2. Figure 6 summarizes

$$
\begin{array}{lcl}
\mathbf{R} : \tau_1 \leftrightarrow \tau_2 & \stackrel{\text{def}}{\Leftrightarrow} & \forall(\mathbf{v}_1, \mathbf{v}_2) \in \mathbf{R}.\ \emptyset \vdash_M \mathbf{v}_1 : \tau_1 \ \wedge\ \emptyset \vdash_M \mathbf{v}_2 : \tau_2 \\
\mathcal{V}_M[\![\mathbf{Nat}]\!]\delta & \stackrel{\text{def}}{=} & \{(\overline{n}, \overline{n})\} \\
\mathcal{V}_M[\![\tau^*]\!]\delta & \stackrel{\text{def}}{=} & \{(\mathbf{nil}, \mathbf{nil})\} \\
& \cup & \{((\mathbf{cons}\ \mathbf{v}_1\ \mathbf{v'}_1), (\mathbf{cons}\ \mathbf{v}_2\ \mathbf{v'}_2)) \mid \\
& & \quad (\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta \ \wedge\ (\mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau^*]\!]\delta\} \\
\mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta & \stackrel{\text{def}}{=} & \{(\lambda \mathbf{x} : \tau_1.\ \mathbf{e}_1, \lambda \mathbf{x} : \tau_1.\ \mathbf{e}_2) \mid \\
& & \quad \forall(\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1]\!]\delta. \\
& & \quad\quad (\mathbf{e}_1[\mathbf{x} := \mathbf{v}_1], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_2]) \in \mathcal{E}_M[\![\tau_2]\!]\delta\} \\
\mathcal{V}_M[\![\alpha]\!]\delta & \stackrel{\text{def}}{=} & \delta(\alpha).\mathbf{R} \\
\mathcal{V}_M[\![\forall \alpha.\ \tau]\!]\delta & \stackrel{\text{def}}{=} & \{(\Lambda\alpha.\ \mathbf{e}_1, \Lambda\alpha.\ \mathbf{e}_2) \mid \\
& & \quad \forall \tau_1, \tau_2, \mathbf{R}.\ \mathbf{R} : \tau_1 \leftrightarrow \tau_2 \Longrightarrow \\
& & \quad\quad (\mathbf{e}_1[\alpha := \langle k_1; \tau_1 \rangle], \mathbf{e}_2[\alpha := \langle k_2; \tau_2 \rangle]) \in \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]\} \\
\mathcal{V}_M[\![\mathbf{L}]\!]\delta & \stackrel{\text{def}}{=} & \{((^{\mathbf{L}}MS\ \mathsf{v}_1), (^{\mathbf{L}}MS\ \mathsf{v}_2)) \mid (\mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S\} \\
\mathcal{E}_M[\![\tau]\!]\delta & \stackrel{\text{def}}{=} & \{(\mathbf{e}_1, \mathbf{e}_2) \mid \\
& & \quad \forall K_1, K_2.\ \forall \text{str}.\ K_1 \triangleright \mathbf{e}_1 \longmapsto^* \mathbf{Error}\text{: str} \Longrightarrow \\
& & \quad\quad K_2 \triangleright \mathbf{e}_2 \longmapsto^* \mathbf{Error}\text{: str} \ \wedge \\
& & \quad \forall K_1, K'_1, \mathbf{v}_1.\ K_1 \triangleright \mathbf{e}_1 \longmapsto^* K'_1 \triangleright \mathbf{v}_1 \Longrightarrow \\
& & \quad\quad \exists K_2, K'_2, \mathbf{v}_2.\ K_2 \triangleright \mathbf{e}_2 \longmapsto^* K'_2 \triangleright \mathbf{v}_2 \ \wedge\ (\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta\}
\end{array}
$$

Fig. 6. Semantic interpretations of ML types: a naive attempt

the definitions presented in the last two sections and adds a clause to the definition of $\mathcal{E}_M[\![.]\!]$ to account for the possibility of an error.

3.3 Relatedness of Scheme values

As we saw above, the relatedness of two lump values at type $\mathbf{L}$ in ML depends on the relatedness of the Scheme values inside them, according to some relation $\mathcal{V}_S$. Let us now attempt to define $\mathcal{V}_S$.

Since Scheme values are untyped — or uni-typed to be precise, since Scheme terms are a special case of typed code where every term has type **TST** (Harper, 2010) — $\mathcal{V}_S$ cannot be defined by induction on Scheme types in the way that $\mathcal{V}_M[\![.]\!]$ was defined by induction on ML types.[2] Instead, we define $\mathcal{V}_S$ as a union of several value relations, one for each of the syntactic value forms in Scheme. For instance, one such relation will relate lambda abstractions:

$$
\mathcal{V}_S \stackrel{\text{def}}{=} \cdots \cup \{(\lambda \mathsf{x}.\ \mathsf{e}_1, \lambda \mathsf{x}.\ \mathsf{e}_2) \mid \forall(\mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S \Longrightarrow (\mathsf{e}_1[\mathsf{x} := \mathsf{v}_1], \mathsf{e}_2[\mathsf{x} := \mathsf{v}_2]) \in \mathcal{E}_S\} \cup \cdots
$$

We see here that just as $\mathcal{V}_M[\![.]\!]$ relies on $\mathcal{E}_M[\![.]\!]$, our definition of $\mathcal{V}_S$ requires us to define a relation $\mathcal{E}_S$ on Scheme expressions. The definition of $\mathcal{E}_S$ is straightforward: we say that two Scheme expressions e_1 and e_2 are related if whenever e_1 evaluates to a value v_1, e_2 evaluates to some value v_2 such that v_1 and v_2 are related under $\mathcal{V}_S$. For the time being, we

[2] Naively, we may think of $\mathcal{V}_S$ as an interpretation of the Scheme type **TST** as a relation on Scheme values, but since Scheme has just one type we will not bother to write $\mathcal{V}_S[\![\mathbf{TST}]\!]$.

again ignore the possibility that an expression will evaluate to an error, and define $\mathcal{E}_S$ as follows:

$$\mathcal{E}_S \stackrel{\text{def}}{=} \{(\mathsf{e}_1, \mathsf{e}_2) \mid \forall \psi_1, \psi_1', \mathsf{v}_1.\ \psi_1 \triangleright \mathsf{e}_1 \longmapsto^* \psi_1' \triangleright \mathsf{v}_1 \Longrightarrow \exists \psi_2, \psi_2', \mathsf{v}_2.\ \psi_2 \triangleright \mathsf{e}_2 \longmapsto^* \psi_2' \triangleright \mathsf{v}_2 \wedge (\mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S\}$$

With $\mathcal{E}_S$ handled, we can turn our attention back to $\mathcal{V}_S$. Unfortunately, our definition of $\mathcal{V}_S$ is not well-founded: it refers to $\mathcal{V}_S$ itself, both to ensure relatedness of arguments and relatedness of the results of function application. Essentially, since all Scheme terms have the same type, **TST**, the type of the function arguments and results are no smaller than the type of the functions themselves, resulting in an ill-founded definition.

To resolve this problem, we redefine $\mathcal{V}_S$ as a three-place relation relating two Scheme values and a natural number j, called the *step index*, that represents the remaining number of steps available for future execution. Membership of a triple $(j, \mathsf{v}_1, \mathsf{v}_2)$ in the relation indicates that values v_1 and v_2 are related for j steps; that is, they cannot be distinguished by any computation running for no more than j steps. Now the Scheme logical relation can be defined by induction on steps, yielding a well-founded relation. We say that two Scheme lambda terms are related for j steps if they take arguments that are related for any strictly *smaller* number of steps j' to results that are also related for j' steps. Put another way, the Scheme expressions $(\lambda \mathsf{x}.\ \mathsf{e}_1)$ and $(\lambda \mathsf{x}.\ \mathsf{e}_2)$ are indistinguishable for j steps if, given two arguments v_1 and v_2 that are indistinguishable for any smaller number of steps j', applying $(\lambda \mathsf{x}.\ \mathsf{e}_1)$ to v_1 and $(\lambda \mathsf{x}.\ \mathsf{e}_2)$ to v_2 will result in two expressions that are also indistinguishable for j' steps. We define $\mathcal{V}_S$ for lambda expressions as follows (assuming a step-indexed version of $\mathcal{E}_S$):

$$\mathcal{V}_S \stackrel{\text{def}}{=} \cdots \cup \{(j, \lambda \mathsf{x}.\ \mathsf{e}_1, \lambda \mathsf{x}.\ \mathsf{e}_2) \mid \forall (j', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S.\ j' < j \Longrightarrow (j', \mathsf{e}_1[\mathsf{x} := \mathsf{v}_1], \mathsf{e}_2[\mathsf{x} := \mathsf{v}_2]) \in \mathcal{E}_S\} \cup \cdots$$

To complete the definition of $\mathcal{V}_S$, we need to consider relatedness for the rest of Scheme's syntactic value forms. Defining relatedness for numbers and lists of values is straightforward, but we also need to consider *sealed values* of the form $(SM^{\langle k;\tau\rangle}\ \mathbf{v})$ for some ML value $\mathbf{v}$.

When are two such sealed values $(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1)$ and $(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)$ related? One possibility is to define their relatedness in terms of the relatedness of the underlying ML values $\mathbf{v}_1$ and $\mathbf{v}_2$ at the type α. We will return to this question at the end of the following section, after we reconsider our definition of relatedness of ML values.

3.4 Relatedness of ML values, revisited

Our new step-indexed definition of $\mathcal{V}_S$ requires us to revisit the definition of the ML relation $\mathcal{V}_M[\![.]\!]$ at the type $\mathbf{L}$. (Recall that $\mathcal{V}_M[\![.]\!]$ relies on $\mathcal{V}_S$ since Scheme values can be embedded in ML as *MS* boundary expressions of type $\mathbf{L}$.) For how many steps must the Scheme values embedded in two ML lumps be related? One option is to say that two ML lumps $({}^{\mathbf{L}}MS\ \mathsf{v}_1)$ and $({}^{\mathbf{L}}MS\ \mathsf{v}_2)$ are related if the underlying Scheme values v_1 and v_2 are related for *some* number of steps n according to $\mathcal{V}_S$:

$$\mathcal{V}_M[\![\mathbf{L}]\!]\delta \stackrel{\text{def?}}{=} \{(({}^{\mathbf{L}}MS\ \mathsf{v}_1), ({}^{\mathbf{L}}MS\ \mathsf{v}_2)) \mid \exists n.\ (n, \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S\}$$

However, recall that we allow *MS* and *SM* boundaries to be nested inside each other and consider the scenario where v_1 and v_2 are of the form $\ldots({}^{\mathbf{L}}MS\ \mathsf{v'}_1)\ldots$ and $\ldots({}^{\mathbf{L}}MS\ \mathsf{v'}_2)\ldots$, respectively. Clearly we want our logical relations to be defined so that we require $\mathsf{v'}_1$ and $\mathsf{v'}_2$ to be related for for fewer than n steps. That, however, is not possible if we adopt the above definition for $\mathcal{V}_M[\![\mathbf{L}]\!]$.

The solution to this ill-foundedness dilemma is to allow step indices to "leak" from the Scheme relation back into the ML relation. Just as we did with $\mathcal{V}_S$ and $\mathcal{E}_S$, we define step-indexed versions of $\mathcal{V}_M[\![.]\!]$ and $\mathcal{E}_M[\![.]\!]$. Adding a step index to the ML relation allows us to define the interpretation of $\mathbf{L}$ as follows:

$$\mathcal{V}_M[\![\mathbf{L}]\!]\delta \stackrel{\text{def}}{=} \{(j,({}^{\mathbf{L}}MS\ \mathsf{v}_1),({}^{\mathbf{L}}MS\ \mathsf{v}_2)) \mid (j-1,\mathsf{v}_1,\mathsf{v}_2) \in \mathcal{V}_S\}$$

Our reasoning for using the step index $j-1$ is that if we claim that the outer lump expressions are related for j steps, then the values they wrap must be related for at least $j-1$ steps, since looking inside the wrapper "uses up" one step of computation.

We will need to add step indices to $\mathcal{V}_M[\![.]\!]$ and $\mathcal{E}_M[\![.]\!]$ for the rest of the type interpretations as well, and well-formed relations $\mathbf{R}$ will need a step-index component. The definition of $\mathbf{R} : \tau_1 \leftrightarrow \tau_2$ also has to be updated to $\mathbf{R} : \tau_1 \overset{n}{\leftrightarrow} \tau_2$, which says that $\mathbf{R}$ is a *downward-closed* relation containing triples $(j,\mathbf{v}_1,\mathbf{v}_2)$ where $j < n$, $\mathbf{v}_1$ is of type τ_1, and $\mathbf{v}_2$ is of type τ_2. Relations are downward closed (or monotonic) if whenever $(j,\mathbf{v}_1,\mathbf{v}_2) \in \mathbf{R}$, then for all $j' \leq j$, $(j',\mathbf{v}_1,\mathbf{v}_2) \in \mathbf{R}$.

Having fixed our definition of relatedness of ML values, we return to the question of relatedness for sealed values in Scheme. We would like to define relatedness for two sealed values $(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1)$ and $(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)$. As a first attempt, we could say that $(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1)$ and $(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)$ are related for j steps if $\mathbf{v}_1$ and $\mathbf{v}_2$ are related by our ML relation at the type α for one fewer step:

$$\mathcal{V}_S \stackrel{\text{def?}}{=} \cdots \cup \{(j,(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1),(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)) \mid (j-1,\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\alpha]\!]\delta\} \cup \cdots$$

but to define $\mathcal{V}_S$ in this way, the type interpretation δ with which we are parameterizing $\mathcal{V}_M[\![.]\!]$ needs to come from somewhere. We can solve this problem by adding δ as a parameter to $\mathcal{V}_S$ and $\mathcal{E}_S$, resulting in the definition:

$$\mathcal{V}_S\,\delta \stackrel{\text{def?}}{=} \cdots \cup \{(j,(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1),(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)) \mid (j-1,\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\alpha]\!]\delta\} \cup \cdots$$

Recall that the definition of $\mathcal{V}_M[\![\alpha]\!]\delta$ is just $\delta(\alpha).\mathbf{R}$, so we can write that into our definition:

$$\mathcal{V}_S\,\delta \stackrel{\text{def?}}{=} \cdots \cup \{(j,(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1),(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)) \mid (j-1,\mathbf{v}_1,\mathbf{v}_2) \in \delta(\alpha).\mathbf{R}\} \cup \cdots$$

Finally, since we know from the boundary annotations on our two *SM* expressions that they contain values of types τ_1 and τ_2 specifically, we would like to stipulate that $\mathbf{R}$ (which, as previously mentioned, is now step-indexed) relates values of those types.

$$\mathcal{V}_S\,\delta \stackrel{\text{def}}{=} \cdots \cup \{(j,(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1),(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)) \mid \delta(\alpha) = (\tau_1,\tau_2,\mathbf{R}) \wedge (j-1,\mathbf{v}_1,\mathbf{v}_2) \in \mathbf{R}\} \cup \cdots$$

Unfortunately, as we will see in the next section, the above definition still turns out to be wrong.

3.5 The case of the missing α

Let $\mathcal{V}[\![.]\!]$ be an interpretation of types as relations on values that is parameterized by a type interpretation δ, like the $\mathcal{V}_S$ and $\mathcal{V}_M[\![.]\!]$ that we have seen in this section. Consider the following equivalence between type interpretations:

$$\mathcal{V}[\![\tau]\!]\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathcal{V}[\![\tau']\!]\delta)] \quad \equiv \quad \mathcal{V}[\![\tau[\alpha := \tau']]\!]\delta$$

On the left side, we are interpreting the type τ, which may contain free occurrences of the type variable α, as a relation parametrized by a type interpretation δ that has been extended with a binding for α. On the right, we are syntactically substituting the type τ' for free occurrences of α in τ before interpreting the resulting type (this time without extending δ with a binding for α).

Both sides represent a "substitution" of τ' for free occurrences of α in the type τ. On the right, we substitute the *syntactic* type τ' for α, whereas on the left we substitute the *semantic* interpretation of τ' for α.

For any logical relation for a language with polymorphism, it is critical that the above equivalence hold. Without it, we cannot prove (the type application case of) the fundamental property of the logical relation. Unfortunately, this equivalence does *not* hold for our relation $\mathcal{V}_M[\![.]\!]$ given our current definitions. Consider the case where τ is the ML lump type $\mathbf{L}$, α is our ML type variable α, and τ' is an ML type τ'. Since $\mathbf{L}$ is opaque, trying to substitute τ' for free occurrences of α in $\mathbf{L}$ just gives us $\mathbf{L}$ again, meaning that we would want the following strange equivalence to hold:

$$\mathcal{V}_M[\![\mathbf{L}]\!]\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathcal{V}_M[\![\tau']\!]\delta)] \quad \stackrel{?}{\equiv} \quad \mathcal{V}_M[\![\mathbf{L}[\alpha := \tau']]\!]\delta \quad \equiv \quad \mathcal{V}_M[\![\mathbf{L}]\!]\delta$$

Here, the interpretation of $\mathbf{L}$ under the δ extended with α on the left is supposed to be equivalent to the interpretation of $\mathbf{L}$ under plain δ with no binding for α — that is, we seem to have lost track of α entirely on the right. Expanding out the definition of $\mathcal{V}_M[\![\mathbf{L}]\!]$ on both the left and right, we see that for the above equivalence to hold, it must be the case that:

$$\mathcal{V}_S\,\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathcal{V}_S\,\delta)] \quad \stackrel{?}{\equiv} \quad \mathcal{V}_S\,\delta$$

We can prove the above in the backward direction, but not in the forward direction. Specifically, in the forward direction, consider the case where we know that $(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1)$ and $(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)$ are related in $\mathcal{V}_S\,\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathcal{V}_S\,\delta)]$; we must show that they are related in $\mathcal{V}_S\,\delta$, which we can no longer do since we have lost the mapping for α!

The problem here arises from the fact that we attempted to define relatedness of *SM* boundary terms by parameterizing our Scheme logical relation with δ. But the purpose of δ is to provide a mapping for type variables that appear free in the type being interpreted. Hence, the Scheme relation should need no such mapping, since type variables never appear in **TST**. But if we do not parametrize $\mathcal{V}_S$ with δ, then where can we find the information stored in δ that we need to define relatedness of *SM* boundary terms? Recall that we discussed in Section 2.3, *Scheme's dynamically generated keys are the corresponding*

notion to ML's α types. Therefore, just as δ provides interpretations for ML's type variables α, we need a different mechanism to keep track of (and provide interpretations for) all the related seals s generated thus far during evaluation. In the next section, we introduce *worlds* to serve exactly that purpose.

4 Parametricity via worlds

Consider two arbitrary expressions e_1 and e_2 of our multi-language system. We wish to know whether e_1 and e_2 behave parametrically, that is, that given related inputs, they produce related results. Since parametricity in our system is enforced dynamically, we need to consider the run-time semantics of our language. In particular, if e_1 and e_2 contain type applications of the form $(\Lambda\alpha.\,\mathbf{e'}_1)\ \tau_1$ and $(\Lambda\alpha.\,\mathbf{e'}_2)\ \tau_2$, the operational semantics will generate fresh keys k_1 and k_2 and add them to the respective key stores K_1 and K_2.

Additionally, at this point, we also get to choose an arbitrary relation $\mathbf{R}$ that relates values of type τ_1 and τ_2; from now on, sealed values of the form $(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1)$ and $(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)$ will only be considered related if $\mathbf{v}_1$ and $\mathbf{v}_2$ are related by $\mathbf{R}$.

In this section, we introduce the concept of *worlds* to keep track of all the keys generated thus far during evaluation (in the form of key stores K_1 and K_2) and to track the relational interpretation $\mathbf{R}$ for related sealed values. With the addition of worlds, we will finally be able to establish parametricity for our multi-language system by setting up a Kripke (or *possible worlds*) logical relation where terms are logically related not only *at a particular type* and *for a particular number of steps*, but also *in a given world*.

4.1 Worlds track information about keys

A *world* $w = (K_1, K_2, \mathcal{C}, \Psi)$ comprises two key stores, K_1 and K_2, a *concretization map* $\mathcal{C}$, and a *type variable interpretation* Ψ. The concretization map $\mathcal{C}$ is a finite mapping from type variables α to pairs of keys (k_1, k_2), where k_1 is contained in the key store K_1 and k_2 is contained in K_2. The type variable interpretation Ψ is a finite mapping from type variables α to triples $(\tau_1, \tau_2, \mathbf{R})$, where τ_1 and τ_2 are closed types and $\mathbf{R}$ relates values of types τ_1 and τ_2. We use the notation $w.K_1$, $w.K_2$, $w.\mathcal{C}$, and $w.\Psi$ for the projections of w's respective components.

Note that $\mathcal{C}$ and Ψ have the same domain and that the the pairs of keys (k_1, k_2) in the range of $w.\mathcal{C}$ correspond one-to-one with pairs of types τ_1 and τ_2 in the range of $w.\Psi$.

When two expressions e_1 and e_2 are related in a world w, the key stores $w.K_1$ and $w.K_2$ respectively contain the keys that have been generated during the evaluation of e_1 and e_2 so far. Suppose that during the evaluation of e_1 a type abstraction is applied to τ_1, causing the key k_1 to be generated; simultaneously, during the evaluation of e_2, a type abstraction is applied to τ_2, causing k_2 to be generated. Then e_1 and e_2 would be related in a world w such that $k_1 \in w.K_1$, $k_2 \in w.K_2$, $w.\mathcal{C}(\alpha) = (k_1, k_2)$, and $w.\Psi(\alpha) = (\tau_1, \tau_2, \mathbf{R})$ for some $\mathbf{R}$.

4.2 Adding worlds to the logical relations

Figure 8 gives the complete (and finally correct) definitions of our Scheme logical relation, which is now indexed both by steps and by worlds, and our ML logical relation, which

$$
\begin{array}{rcll}
\lfloor R \rfloor_n & \stackrel{\text{def}}{=} & \{(j,w,e_1,e_2) \in R \mid j < n\} & n\text{-approximation of relations } R \\
\lfloor \Psi \rfloor_n & \stackrel{\text{def}}{=} & \{\alpha \mapsto (\tau_1,\tau_2,\lfloor R \rfloor_n) \mid \Psi(\alpha) = (\tau_1,\tau_2,R)\} & n\text{-approximation of type var interps} \\
\lfloor (K_1,K_2,\mathcal{C},\Psi) \rfloor_n & \stackrel{\text{def}}{=} & (K_1,K_2,\mathcal{C},\lfloor \Psi \rfloor_n) & n\text{-approximation of worlds} \\
\blacktriangleright R & \stackrel{\text{def}}{=} & \{(j,w,e_1,e_2) \mid w \in \text{World}_j \wedge (j-1,\lfloor w \rfloor_{j-1},e_1,e_2) \in R\} & \text{“Later” operator} \\
\Psi' \sqsupseteq \Psi & \stackrel{\text{def}}{=} & \forall \alpha \in \text{dom}(\Psi).\ \Psi'(\alpha) = \Psi(\alpha) & \text{Type var interp extension} \\
(j',w') \sqsupseteq (j,w) & \stackrel{\text{def}}{\Leftrightarrow} & j' \leq j \wedge w' \in \text{World}_{j'} \wedge w'.\Psi \sqsupseteq \lfloor w.\Psi \rfloor_{j'} \wedge w'.K_1 \supseteq w.K_1 \wedge w'.K_2 \supseteq w.K_2 & \text{World extension} \\
(j',w') \sqsupset (j,w) & \stackrel{\text{def}}{\Leftrightarrow} & j' < j \wedge (j',w') \sqsupseteq (j,w) & \text{Strict world extension} \\
w \boxplus (\alpha \mapsto k_1,k_2,\tau_1,\tau_2,\mathbf{R}) & \stackrel{\text{def}}{=} & ((w.K_1,k_1),(w.K_2,k_2),(w.\mathcal{C}[\alpha \mapsto (k_1,k_2)]) (w.\Psi[\alpha \mapsto (\tau_1,\tau_2,\mathbf{R})]) & \text{Update components of } w
\end{array}
$$

$$
\begin{array}{rcl}
\text{MAtom}_n[\tau_1,\tau_2] & \stackrel{\text{def}}{=} & \{(j,w,\mathbf{e}_1,\mathbf{e}_2) \mid j < n \wedge w \in \text{World}_j \wedge \vdash_M w.K_1 \triangleright \mathbf{e}_1 : \tau_1 \wedge \vdash_M w.K_2 \triangleright \mathbf{e}_2 : \tau_2\} \\
\text{MAtom}[\tau]\delta & \stackrel{\text{def}}{=} & \bigcup_{n \geq 0} \{(j,w,\mathbf{e}_1,\mathbf{e}_2) \in \text{MAtom}_n[\delta_1(\tau),\delta_2(\tau)]\} \\
\text{SAtom}_n & \stackrel{\text{def}}{=} & \{(j,w,\mathbf{e}_1,\mathbf{e}_2) \mid j < n \wedge w \in \text{World}_j \wedge \vdash_S w.K_1 \triangleright \mathbf{e}_1 : \mathbf{TST} \wedge \vdash_S w.K_2 \triangleright \mathbf{e}_2 : \mathbf{TST}\} \\
\text{SAtom} & \stackrel{\text{def}}{=} & \bigcup_{n \geq 0} \{(j,w,\mathbf{e}_1,\mathbf{e}_2) \in \text{SAtom}_n\} \\
\text{Rel}_n[\tau_1,\tau_2] & \stackrel{\text{def}}{=} & \{\mathbf{R} \subseteq \text{MAtom}_n^{\text{val}}[\tau_1,\tau_2] \mid \forall (j,w,\mathbf{v}_1,\mathbf{v}_2) \in \mathbf{R}.\ \forall (j',w') \sqsupseteq (j,w).\ (j',w',\mathbf{v}_1,\mathbf{v}_2) \in \mathbf{R}\} \\
\text{SomeRel}_n & \stackrel{\text{def}}{=} & \{(\tau_1,\tau_2,\mathbf{R}) \mid FV(\tau_1) = FV(\tau_2) = \emptyset \wedge \mathbf{R} \in \text{Rel}_n[\tau_1,\tau_2]\} \\
\text{Conc} & \stackrel{\text{def}}{=} & \{\mathcal{C} \in TVar \xrightarrow{\text{fin}} (k_1,k_2)\} \\
\text{World}_n & \stackrel{\text{def}}{=} & \{w = (K_1,K_2,\mathcal{C},\Psi) \mid \mathcal{C} \in \text{Conc} \wedge \text{rng}(\Psi) \subseteq \text{SomeRel}_n\}
\end{array}
$$

Fig. 7. Worlds and supporting definitions

is now indexed by types, steps, and worlds. In this section, we will compare the world-indexed relations shown in Figure 8 to the step-indexed, but not world-indexed, versions we saw in the last section and summarize the differences. The addition of worlds requires that we define new concepts and notational devices, which we cover informally below and spell out in detail in Figure 7.

The most obvious difference compared to the last section is that $\mathcal{V}_S$ and $\mathcal{V}_M[\![.]\!]$ now contain 4-tuples (j,w,v_1,v_2), where j is a step index as before, w is a world, and v_1 and v_2 are Scheme or ML values, respectively. Likewise, $\mathcal{E}_S$ and $\mathcal{E}_M[\![.]\!]$ contain 4-tuples (j,w,e_1,e_2) where e_1 and e_2 are Scheme or ML expressions, and relations $\mathbf{R}$ contain 4-tuples $(j,w,\mathbf{v}_1,\mathbf{v}_2)$ where $\mathbf{v}_1$ and $\mathbf{v}_2$ are ML values.

$$
\begin{array}{lcl}
\mathcal{V}_S & \stackrel{\text{def}}{=} & \{(j,w,\overline{n},\overline{n}) \in \text{SAtom}\} \\
& \cup & \{(j,w,\mathsf{nil},\mathsf{nil}) \in \text{SAtom}\} \\
& \cup & \{(j,w,(\mathsf{cons}\ \mathsf{v}_1\ \mathsf{v}'_1),(\mathsf{cons}\ \mathsf{v}_2\ \mathsf{v}'_2)) \in \text{SAtom} \mid \\
& & \quad (j,w,\mathsf{v}_1,\mathsf{v}_2) \in \mathcal{V}_S \wedge (j,w,\mathsf{v}'_1,\mathsf{v}'_2) \in \blacktriangleright \mathcal{V}_S\} \\
& \cup & \{(j,w,\lambda\mathsf{x}.\ \mathsf{e}_1,\lambda\mathsf{x}.\ \mathsf{e}_2) \in \text{SAtom} \mid \\
& & \quad \forall (j',w',\mathsf{v}_1,\mathsf{v}_2) \in \mathcal{V}_S.\ (j',w') \sqsupset (j,w) \Longrightarrow \\
& & \quad\quad (j',w',\mathsf{e}_1[\mathsf{x} := \mathsf{v}_1],\mathsf{e}_2[\mathsf{x} := \mathsf{v}_2]) \in \mathcal{E}_S\} \\
& \cup & \{(j,w,(SM^{\langle k_1;\tau_1\rangle}\ \mathbf{v}_1),(SM^{\langle k_2;\tau_2\rangle}\ \mathbf{v}_2)) \in \text{SAtom} \mid \\
& & \quad \exists\alpha.\ w.\mathcal{C}(\alpha) = (k_1,k_2) \wedge w.\Psi(\alpha) = (\tau_1,\tau_2,\mathbf{R}) \wedge (j,w,\mathbf{v}_1,\mathbf{v}_2) \in \blacktriangleright \mathbf{R}\} \\
\mathcal{E}_S & \stackrel{\text{def}}{=} & \{(j,w,\mathsf{e}_1,\mathsf{e}_2) \in \text{SAtom} \mid \\
& & \quad \forall i < j.\ \forall \text{str}.\ w.K_1 \triangleright \mathsf{e}_1 \longmapsto^i \mathbf{Error}\text{: str} \Longrightarrow w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* \mathbf{Error}\text{: str} \wedge \\
& & \quad \forall i < j.\ \forall K'_1,\mathsf{v}_1.\ w.K_1 \triangleright \mathsf{e}_1 \longmapsto^i K'_1 \triangleright \mathsf{v}_1 \Longrightarrow \\
& & \quad\quad \exists K'_2,\mathsf{v}_2,w'.\ w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* K'_2 \triangleright \mathsf{v}_2 \wedge \\
& & \quad\quad (j-i,w') \sqsupseteq (j,w) \wedge w'.K_1 = K'_1 \wedge w'.K_2 = K'_2 \wedge \\
& & \quad\quad (j-i,w',\mathsf{v}_1,\mathsf{v}_2) \in \mathcal{V}_S\} \\
\mathcal{V}_M[\![\mathbf{Nat}]\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,\overline{n},\overline{n}) \in \text{MAtom}[\mathbf{Nat}]\delta\} \\
\mathcal{V}_M[\![\tau^*]\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,\mathbf{nil},\mathbf{nil}) \in \text{MAtom}[\tau^*]\delta\} \\
& \cup & \{(j,w,(\mathbf{cons}\ \mathbf{v}_1\ \mathbf{v}'_1),(\mathbf{cons}\ \mathbf{v}_2\ \mathbf{v}'_2)) \in \text{MAtom}[\tau^*]\delta \mid \\
& & \quad (j,w,\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta \wedge (j,w,\mathbf{v}'_1,\mathbf{v}'_2) \in \blacktriangleright \mathcal{V}_M[\![\tau^*]\!]\delta\} \\
\mathcal{V}_M[\![\tau \to \tau']\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,\lambda\mathbf{x} : \delta_1(\tau).\ \mathbf{e}_1,\lambda\mathbf{x} : \delta_2(\tau).\ \mathbf{e}_2) \in \text{MAtom}[\tau \to \tau']\delta \mid \\
& & \quad \forall (j',w',\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta.\ (j',w') \sqsupseteq (j,w) \Longrightarrow \\
& & \quad\quad (j',w',\mathbf{e}_1[\mathbf{x} := \mathbf{v}_1],\mathbf{e}_2[\mathbf{x} := \mathbf{v}_2]) \in \mathcal{E}_M[\![\tau']\!]\delta\} \\
\mathcal{V}_M[\![\mathbf{L}]\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,(^{\mathbf{L}}MS\ \mathbf{v}_1),(^{\mathbf{L}}MS\ \mathbf{v}_2)) \in \text{MAtom}[\mathbf{L}]\delta \mid (j,w,\mathbf{v}_1,\mathbf{v}_2) \in \blacktriangleright \mathcal{V}_S\} \\
\mathcal{V}_M[\![\alpha]\!]\delta & \stackrel{\text{def}}{=} & \delta(\alpha).R \\
\mathcal{V}_M[\![\forall\alpha.\ \tau]\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,\Lambda\alpha.\ \mathbf{e}_1,\Lambda\alpha.\ \mathbf{e}_2) \in \text{MAtom}[\forall\alpha.\ \tau]\delta \mid \\
& & \quad \forall (j',w') \sqsupseteq (j,w).\ \forall (\tau_1,\tau_2,\mathbf{R}) \in \text{SomeRel}_{j'}.\ \forall k_1 \notin w'.K_1.\ \forall k_2 \notin w'.K_2. \\
& & \quad\quad (j',w' \boxplus (\alpha \mapsto k_1,k_2,\tau_1,\tau_2,\mathbf{R}),\mathbf{e}_1[\alpha := \langle k_1;\tau_1\rangle],\mathbf{e}_2[\alpha := \langle k_2;\tau_2\rangle]) \\
& & \quad\quad\quad \in \blacktriangleright \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1,\tau_2,\mathbf{R})]\} \\
\mathcal{E}_M[\![\tau]\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,\mathbf{e}_1,\mathbf{e}_2) \in \text{MAtom}[\tau]\delta \mid \\
& & \quad \forall i < j.\ \forall \text{str}.\ w.K_1 \triangleright \mathbf{e}_1 \longmapsto^i \mathbf{Error}\text{: str} \Longrightarrow w.K_2 \triangleright \mathbf{e}_2 \longmapsto^* \mathbf{Error}\text{: str} \wedge \\
& & \quad \forall i < j.\ \forall K'_1,\mathbf{v}_1.\ w.K_1 \triangleright \mathbf{e}_1 \longmapsto^i K'_1 \triangleright \mathbf{v}_1 \Longrightarrow \\
& & \quad\quad \exists K'_2,\mathbf{v}_2,w'.\ w.K_2 \triangleright \mathbf{e}_2 \longmapsto^* K'_2 \triangleright \mathbf{v}_2 \wedge \\
& & \quad\quad (j-i,w') \sqsupseteq (j,w) \wedge w'.K_1 = K'_1 \wedge w'.K_2 = K'_2 \wedge \\
& & \quad\quad (j-i,w',\mathbf{v}_1,\mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta\} \\
\mathcal{V}_*[\![T]\!]\delta & \stackrel{\text{def}}{=} & \mathcal{V}_M[\![T]\!]\delta \quad \text{if } T = \tau \\
& & \mathcal{V}_S \quad\quad\ \ \text{if } T = \mathbf{TST} \\
\mathcal{D}[\![\Delta]\!] & \stackrel{\text{def}}{=} & \{(n,\delta) \mid \text{dom}(\delta) = \Delta \wedge \forall\alpha \in \Delta.\ \delta(\alpha) \in \text{SomeRel}_n\} \\
\mathcal{G}[\![\Gamma]\!]\delta & \stackrel{\text{def}}{=} & \{(j,w,\gamma_1,\gamma_2) \mid w \in \text{World}_j \wedge \text{dom}(\Gamma) = \text{dom}(\gamma_1) = \text{dom}(\gamma_2) \wedge \\
& & \quad \forall x \in \text{dom}(\Gamma).\ (j,w,\gamma_1(x),\gamma_2(x)) \in \mathcal{V}_*[\![\Gamma(x)]\!]\delta\} \\
\Delta;\Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_2 : \tau & \stackrel{\text{def}}{=} & \forall n \geq 0.\ \forall j < n.\ \forall\delta,\gamma_1,\gamma_2,w.\ (n,\delta) \in \mathcal{D}[\![\Delta]\!] \wedge \\
& & \quad (j,w,\gamma_1,\gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta \Longrightarrow (j,w,\delta_1(\gamma_1(\mathbf{e}_1)),\delta_2(\gamma_2(\mathbf{e}_2))) \in \mathcal{E}_M[\![\tau]\!]\delta \\
\Delta;\Gamma \vdash_S \mathsf{e}_1 \lesssim_S \mathsf{e}_2 : \mathbf{TST} & \stackrel{\text{def}}{=} & \forall n \geq 0.\ \forall j < n.\ \forall\delta,\gamma_1,\gamma_2,w.\ (n,\delta) \in \mathcal{D}[\![\Delta]\!] \wedge \\
& & \quad (j,w,\gamma_1,\gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta \Longrightarrow (j,w,\delta_1(\gamma_1(\mathsf{e}_1)),\delta_2(\gamma_2(\mathsf{e}_2))) \in \mathcal{E}_S \\
\Delta;\Gamma \vdash_M \mathbf{e}_1 \approx_M \mathbf{e}_2 : \tau & \stackrel{\text{def}}{=} & \Delta;\Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_2 : \tau \wedge \Delta;\Gamma \vdash_M \mathbf{e}_2 \lesssim_M \mathbf{e}_1 : \tau \\
\Delta;\Gamma \vdash_S \mathsf{e}_1 \approx_S \mathsf{e}_2 : \mathbf{TST} & \stackrel{\text{def}}{=} & \Delta;\Gamma \vdash_S \mathsf{e}_1 \lesssim_S \mathsf{e}_2 : \mathbf{TST} \wedge \Delta;\Gamma \vdash_S \mathsf{e}_2 \lesssim_S \mathsf{e}_1 : \mathbf{TST}
\end{array}
$$

Fig. 8. World-indexed semantic interpretations of Scheme and ML expressions and values

In Figure 7, we use the metavariable R to range over sets of 4-tuples of this form.[3] The *n-approximation* of a relation R, written $\lfloor R \rfloor_n$, simply throws out every 4-tuple in R that has a step index of n or greater. We can then define the n-approximation $\lfloor \Psi \rfloor_n$ of a type variable interpretation Ψ as the the type variable interpretation that is identical to Ψ except that the relation component of each triple in its domain has been n-approximated. Likewise, the n-approximation $\lfloor w \rfloor_n$ of a world w is the world identical to w except that its Ψ component has been n-approximated.

The "later" operator $\blacktriangleright$, defined on relations R has the effect of subtracting 1 from the step index j and *approximating* the world w at level $j-1$. For any relation R,

$$(j, w, e_1, e_2) \in \blacktriangleright R \iff (j-1, \lfloor w \rfloor_{j-1}, e_1, e_2) \in R$$

We use $\blacktriangleright$ at three places in the world-indexed relation: the *SM* boundary case for Scheme, and the $\mathbf{L}$ and $\forall\alpha.\ \tau$ cases for ML.

An *extension* w' of a world w is one in which the key stores $w'.K_1$ and $w'.K_2$ are supersets of $w.K_1$ and $w.K_2$ respectively, and in which for every type variable α in the domain of $w.\Psi$, we have that $w'.\Psi(\alpha) = w.\Psi(\alpha)$ and that $w'.\mathcal{C}(\alpha) = w.\mathcal{C}(\alpha)$.

We define an ordering $\sqsupseteq$ (pronounced "extends") on pairs of step indices and worlds. $(j', w') \sqsupseteq (j, w)$ holds when w' is an extension of w and $j' \leq j$. The strict version, $\sqsupset$, requires that j' be strictly less than j. (Overloading terminology, we will refer to these kinds of extensions as "world extension" as well, even though they are defined on pairs of step indices and worlds rather than just on worlds.) Intuitively, (j', w') represents some later stage of evaluation — or a future point in time — when *more* keys have potentially been generated and in which there are potentially *fewer* steps left for evaluation.

Wherever the inequalities $j' < j$ or $j' \leq j$' might appear in the naive step-indexed relation we similarly allow for the possibility of moving to a future world, requiring $(j', w') \sqsupset (j, w)$ or $(j', w') \sqsupseteq (j, w)$, respectively, in the world-indexed relation of Figure 8. This happens in three places: for lambda abstractions in the definition of $\mathcal{V}_S$, and for the types $\tau_1 \to \tau_2$ and $\forall\alpha.\ \tau$ in $\mathcal{V}_M[\![.]\!]$. In the expression relations $\mathcal{E}_S$ and $\mathcal{E}_M[\![.]\!]$, we use $(j-i, w') \sqsupset (j, w)$, which can be thought of as replacing an implicit $(j-i) < j$ in the naive step-indexed version.

Keys are generated when ML type abstractions are applied to concrete types. Recall that $\mathcal{V}_M[\![.]\!]$ says that two ML type abstractions (that is, values of type $\forall\alpha.\ \tau$) are related if, when we apply them to types that are related according to some $\mathbf{R}$, they produce expressions that are also related with respect to $\mathbf{R}$. Our world-indexed version of $\mathcal{V}_M[\![.]\!]$ must account for the keys generated during that type application. We can add newly generated keys k_1 and k_2 to a world w by adding k_1 and k_2 to the key stores $w.K_1$ and $w.K_2$, extending $w.\mathcal{C}$ with a binding from α to (k_1, k_2), and extending $w.\Psi$ with a binding from α to an appropriate triple $(\tau_1, \tau_2, \mathbf{R})$. We use the notation $w' \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R})$ in the definition of $\mathcal{V}_M[\![\forall\alpha.\ \tau]\!]$ to indicate an update to all the components of w'.

We can define the following particularly interesting sets of *atoms* or 4-tuples:

- SAtom_n contains all tuples $(j, w, \mathsf{e}_1, \mathsf{e}_2)$, where $j < n$ for the given subscript n, and $w.K_1 \triangleright \mathsf{e}_1$ and $w.K_2 \triangleright \mathsf{e}_2$ are well-typed configurations of type **TST**.

[3] Note that relations $\mathbf{R}$ are instances of R. In fact, we could replace every occurrence of R in Figure 7 with $\mathbf{R}$ and still have it be correct with respect to the logical relations defined in Figure 8.

- SAtom is the union of the sets SAtom_n for all $n \geq 0$.
- $\text{MAtom}_n[\tau_1, \tau_2]$ contains all atoms $(j, w, \mathbf{e}_1, \mathbf{e}_2)$, where $j < n$ for the given subscript n, and $w.K_1 \triangleright \mathbf{e}_1$ and $w.K_2 \triangleright \mathbf{e}_2$ are well-typed configurations of the types τ_1 and τ_2, respectively.
- $\text{MAtom}[\tau]\delta$ is the union of the sets $\text{MAtom}_n[\delta_1(\tau), \delta_2(\tau)]$ for all $n \geq 0$.

These definitions allow us to specify basic well-formedness and well-foundedness conditions on the tuples that appear in our logical relations. We require that elements of $\mathcal{V}_S$ and of $\mathcal{E}_S$ belong to SAtom, and that elements of each $\mathcal{V}_M[\![\tau]\!]\delta$ and $\mathcal{E}_M[\![\tau]\!]\delta$ belong to $\text{MAtom}[\tau]\delta$ for all types τ and type relations δ. We use the notation $\text{MAtom}_n^{\text{val}}[\tau_1, \tau_2]$ to indicate the restriction of $\text{MAtom}_n[\tau_1, \tau_2]$ to values. Relations $\mathbf{R}$ must be subsets of $\text{MAtom}_n^{\text{val}}[\tau_1, \tau_2]$ for some τ_1 and τ_2.

Now that we have correct definitions of our logical relations at last, we can define relatedness of open expressions (bottom of Figure 8). The relation $\Delta;\Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_2 : \tau$, which is defined on pairs of open ML terms $\mathbf{e}_1$ and $\mathbf{e}_2$ such that $\Delta;\Gamma \vdash_M \mathbf{e}_1 : \tau$ and $\Delta;\Gamma \vdash_M \mathbf{e}_2 : \tau$, says that: given a world w, a type interpretation δ that respects Δ, and two value substitutions γ_1 and γ_2 that are related in w and respect Γ, applying δ_1 and γ_1 to $\mathbf{e}_1$ and δ_2 and γ_2 to $\mathbf{e}_2$ will yield expressions that are related in world w at the type τ under δ. The analogous relation $\Delta;\Gamma \vdash_S \mathtt{e}_1 \lesssim_S \mathtt{e}_2 : \mathbf{TST}$ on open Scheme terms says that given a world w, a type interpretation δ that respects Δ, and two value substitutions γ_1 and γ_2 that are related in world w and respect Γ, applying δ_1 and γ_1 to $\mathtt{e}_1$ and δ_2 and γ_2 to $\mathtt{e}_2$ will expressions that are related in the Scheme relation at world w.

The relations on open terms in Figure 8 capture the essence of what we mean by "parametricity": two terms are related if we can drop them into related environments (represented by the two value substitutions γ_1 and γ_2 and the type interpretations δ and get results that are still related. Put another way, related terms take related arguments to related results. In the next section, we present the bridge lemma that will allow us to prove parametricity for ML terms that contain embedded Scheme terms and vice versa.

4.3 The bridge lemma and fundamental property

The critical piece we will need is a bridge lemma that allows us to carry relatedness in one language to relatedness in the other:

Lemma 4.1 (*bridge lemma*)
For all $j \geq 0$, worlds w such that $w \in \text{World}_j$, type environments Δ, type relations δ such that $\Delta \vdash \delta$, and types τ such that $\Delta \vdash \tau$, both of the following hold:

1. For all $\mathtt{e}_1$ and $\mathtt{e}_2$,
   if $(j, w, \mathtt{e}_1, \mathtt{e}_2) \in \mathcal{E}_S$ then $(j, w, ({}^{\delta_1(\tau)}MS\ \mathtt{e}_1), ({}^{\delta_2(\tau)}MS\ \mathtt{e}_2)) \in \mathcal{E}_M[\![\tau]\!]\delta$.
2. For all $\mathbf{e}_1$ and $\mathbf{e}_2$,
   if $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\tau]\!]\delta$ then $(j, w, (SM^{\delta_1(\tau)}\ \mathbf{e}_1), (SM^{\delta_2(\tau)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

Proof
By induction on τ. All cases are straightforward given the induction hypotheses. □

With the bridge lemma established, the fundamental property (and hence the fact that logical approximation implies contextual approximation) is essentially standard. We restrict the parametricity theorem to key-free terms; otherwise we would have to show that any sealed value is related to itself at type α which is false. (A conversion strategy is key-free if it contains no instances of $\langle k;\tau\rangle$ for any k. A term is key-free if it contains no conversion strategies with keys.) This restriction is purely technical, since the claim applies to open terms where keys may be introduced by closing environments.

Theorem 4.2 (*parametricity / fundamental property*)
For all key-free terms $\mathbf{e}$ and e:

1. If $\Delta;\Gamma \vdash_M \mathbf{e} : \tau$, then $\Delta;\Gamma \vdash_M \mathbf{e} \lesssim_M \mathbf{e} : \tau$.
2. If $\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}$, then $\Delta;\Gamma \vdash_S \mathsf{e} \lesssim_S \mathsf{e} : \mathbf{TST}$.

Proof
By simultaneous induction on the derivations $\Delta;\Gamma \vdash_M \mathbf{e} : \tau$ and $\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}$. The boundary cases both follow from lemma 4.1. □

Theorem 4.2 establishes a very strong reasoning principle for our multi-language system. Here we give an example of how it can be used by proving that the only values of type $\forall\alpha.\ \alpha \to \alpha$ are now either the function that always diverges, or the function that always signals an error, or the identity function.

Theorem 4.3
If $; \vdash_M \mathbf{v} : \forall\alpha.\ \alpha \to \alpha$, then one of the following holds:

1. $\forall\tau$, $\mathbf{v}'$ such that $; \vdash_M \mathbf{v}' : \tau$, $((\mathbf{v}\ \tau)\ \mathbf{v}')$ diverges;
2. $\forall\tau$, $\mathbf{v}'$ such that $; \vdash_M \mathbf{v}' : \tau$, $((\mathbf{v}\ \tau)\ \mathbf{v}') \longmapsto^* \mathbf{Error}$: str for some str; or
3. $\forall\tau$, $\mathbf{v}'$ such that $; \vdash_M \mathbf{v}' : \tau$, $((\mathbf{v}\ \tau)\ \mathbf{v}') \longmapsto^* \mathbf{v}'$.

The proof follows from the Fundamental Property and the definition of the logical relation. As usual, we need to pick a relation $\mathbf{R}$ that relates $\mathbf{v}'$ to itself, but this time under appropriate step indices and worlds.

References

Ahmed, Amal. (2006). Step-indexed syntactic logical relations for recursive and quantified types. *Pages 69–83 of: Esop*. Extended version: Harvard University Technical Report TR-01-06, `http://ttic.uchicago.edu/~amal/papers/lr-recquant-techrpt.pdf`.

Findler, Robert Bruce, & Blume, Matthias. (2006). Contracts as pairs of projections.

Findler, Robert Bruce, & Felleisen, Matthias. (2002). Contracts for higher-order functions. *Icfp*.

Flatt, Matthew. (1997). *PLT MzScheme: Language manual*. Technical Report TR97-280. Rice University. `http://www.plt-scheme.org/software/mzscheme/`.

Harper, Robert. (2010). *Practical foundations for programming languages*. Working Draft.

Matthews, Jacob, & Findler, Robert Bruce. (2007). Operational semantics for multi-language programs. *Popl*. Extended version: University of Chicago Technical Report TR-2007-8, under review.

Morris, Jr., James H. (1973). Types are not sets. *Popl*.

Reynolds, John C. (1983). Types, abstraction and parametric polymorphism. *Pages 513–523 of: Ifip congress*.

Sumii, Ejiro, & Pierce, Benjamin. (2004). A bisimulation for dynamic sealing. *Popl*.

Wadler, Philip. (1989). Theorems for free! *Pages 347–359 of: Functional programming languages and computer architecture (fpca)*.

A Appendix

$\boxed{\kappa[\alpha := \langle k; \tau\rangle]}$

$$
\begin{array}{rcll}
\mathbf{Nat}[\alpha := \langle k;\tau\rangle] & = & \mathbf{Nat} & \\
\kappa_1 \to \kappa_2[\alpha := \langle k;\tau\rangle] & = & \kappa_1[\alpha := \langle k;\tau\rangle] \to \kappa_2[\alpha := \langle k;\tau\rangle] & \\
\kappa^*[\alpha := \langle k;\tau\rangle] & = & \kappa[\alpha := \langle k;\tau\rangle]^* & \\
(\forall\alpha'.\,\kappa)[\alpha := \langle k;\tau\rangle] & = & \forall\alpha'.\,\kappa[\alpha := \langle k;\tau\rangle] & (\text{if } \alpha' \neq \alpha) \\
 & = & \forall\alpha.\,\kappa & (\text{if } \alpha' = \alpha) \\
\alpha'[\alpha := \langle k;\tau\rangle] & = & \alpha' & (\text{if } \alpha' \neq \alpha) \\
 & = & \langle k;\tau\rangle & (\text{if } \alpha' = \alpha) \\
\mathbf{L}[\alpha := \langle k;\tau\rangle] & = & \mathbf{L} & \\
\langle k';\tau'\rangle[\alpha := \langle k;\tau\rangle] & = & \langle k';\tau'\rangle &
\end{array}
$$

$\boxed{e[\alpha := \langle k; \tau\rangle]}$

$$
\begin{array}{rcll}
\mathbf{x}[\alpha := \langle k;\tau\rangle] & = & \mathbf{x} & \\
(\lambda\mathbf{x}:\tau_1.\,\mathbf{e})[\alpha := \langle k;\tau\rangle] & = & \lambda\mathbf{x}:\tau_1[k := \tau].\,\mathbf{e}[\alpha := \langle k;\tau\rangle] & \\
\overline{n}[\alpha := \langle k;\tau\rangle] & = & \overline{n} & \\
\mathbf{nil}[\alpha := \langle k;\tau\rangle] & = & \mathbf{nil} & \\
\mathbf{fst}[\alpha := \langle k;\tau\rangle] & = & \mathbf{fst} & \\
\mathbf{rst}[\alpha := \langle k;\tau\rangle] & = & \mathbf{rst} & \\
(\mathbf{e}_1\ \mathbf{e}_2)[\alpha := \langle k;\tau\rangle] & = & (\mathbf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathbf{e}_2[\alpha := \langle k;\tau\rangle]) & \\
(op\ \mathbf{e}_1\ \mathbf{e}_2)[\alpha := \langle k;\tau\rangle] & = & (op\ \mathbf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathbf{e}_2[\alpha := \langle k;\tau\rangle]) & \\
(\mathbf{if0}\ \mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3)[\alpha := \langle k;\tau\rangle] & = & (\mathbf{if0}\ \mathbf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathbf{e}_2[\alpha := \langle k;\tau\rangle]\ \mathbf{e}_3[\alpha := \langle k;\tau\rangle]) & \\
(\mathbf{cons}\ \mathbf{e}_1\ \mathbf{e}_2)[\alpha := \langle k;\tau\rangle] & = & (\mathbf{cons}\ \mathbf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathbf{e}_2[\alpha := \langle k;\tau\rangle]) & \\
(\Lambda\alpha'.\,\mathbf{e})[\alpha := \langle k;\tau\rangle] & = & \Lambda\alpha'.\,\mathbf{e}[\alpha := \langle k;\tau\rangle] & (\text{if } \alpha' \neq \alpha) \\
 & = & \Lambda\alpha'.\,\mathbf{e} & (\text{if } \alpha' = \alpha) \\
\mathbf{e}\ \tau_1[\alpha := \langle k;\tau\rangle] & = & \mathbf{e}[\alpha := \langle k;\tau\rangle]\ \tau_1[k := \tau] & \\
({}^{\kappa}MS\ \mathsf{e})[\alpha := \langle k;\tau\rangle] & = & ({}^{\kappa[\alpha:=\langle k;\tau\rangle]}MS\ \mathsf{e}[\alpha := \langle k;\tau\rangle]) &
\end{array}
$$

$$
\begin{array}{rcl}
\mathsf{x}[\alpha := \langle k;\tau\rangle] & = & \mathsf{x} \\
(\lambda\mathsf{x}.\,\mathsf{e})[\alpha := \langle k;\tau\rangle] & = & \lambda\mathsf{x}.\,\mathsf{e}[\alpha := \langle k;\tau\rangle] \\
\overline{n}[\alpha := \langle k;\tau\rangle] & = & \overline{n} \\
\mathsf{nil}[\alpha := \langle k;\tau\rangle] & = & \mathsf{nil} \\
\mathsf{fst}[\alpha := \langle k;\tau\rangle] & = & \mathsf{fst} \\
\mathsf{rst}[\alpha := \langle k;\tau\rangle] & = & \mathsf{rst} \\
(\mathsf{e}_1\ \mathsf{e}_2)[\alpha := \langle k;\tau\rangle] & = & (\mathsf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathsf{e}_2[\alpha := \langle k;\tau\rangle]) \\
(op\ \mathsf{e}_1\ \mathsf{e}_2)[\alpha := \langle k;\tau\rangle] & = & (op\ \mathsf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathsf{e}_2[\alpha := \langle k;\tau\rangle]) \\
(\mathsf{if0}\ \mathsf{e}_1\ \mathsf{e}_2\ \mathsf{e}_3)[\alpha := \langle k;\tau\rangle] & = & (\mathsf{if0}\ \mathsf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathsf{e}_2[\alpha := \langle k;\tau\rangle]\ \mathsf{e}_3[\alpha := \langle k;\tau\rangle]) \\
(pd\ \mathsf{e})[\alpha := \langle k;\tau\rangle] & = & (pd\ \mathsf{e}[\alpha := \langle \alpha;\tau\rangle]) \\
(\mathsf{cons}\ \mathsf{e}_1\ \mathsf{e}_2)[\alpha := \langle k;\tau\rangle] & = & (\mathsf{cons}\ \mathsf{e}_1[\alpha := \langle k;\tau\rangle]\ \mathsf{e}_2[\alpha := \langle k;\tau\rangle]) \\
(SM^{\kappa}\ \mathbf{e})[\alpha := \langle k;\tau\rangle] & = & (SM^{\kappa[\alpha:=\langle k;\tau\rangle]}\ \mathbf{e}[\alpha := \langle k;\tau\rangle])
\end{array}
$$

Fig. A 1. Sealing substitution on conversion schemes (top) and expressions (bottom)

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

$$\frac{}{\Delta \vdash \mathbf{Nat}} \qquad \frac{\Delta \vdash \tau_1 \quad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \to \tau_2} \qquad \frac{\Delta \vdash \tau}{\Delta \vdash \tau^*} \qquad \frac{\Delta, \alpha \vdash \tau}{\Delta \vdash \forall \alpha.\, \tau} \qquad \frac{\alpha \in \Delta}{\Delta \vdash \alpha} \qquad \frac{}{\Delta \vdash \mathbf{L}}$$

Fig. A 2. Well-formedness of type environments

Appendix: Proofs

1 Preliminaries

1.1 Transitivity of world extension

Lemma 1.1 (transitivity of world extension)**.** *For all worlds w, w', w'' and step indices j, j', j'': if $(j', w') \sqsupseteq (j, w)$ and $(j'', w'') \sqsupseteq (j', w')$, then $(j'', w'') \sqsupseteq (j, w)$.*

Proof. Straightforward, by unfolding definition of $\sqsupseteq$. □

1.2 Downward closure of $\mathcal{V}_S$

Lemma 1.2 (downward closure of $\mathcal{V}_S$)**.** *For all $j, w, \mathsf{v}_1, \mathsf{v}_2$:*
If $(j, w, \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$, then for all $(j', w') \sqsupseteq (j, w)$,
$(j', w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.

Proof. Straightforward, by unfolding of definition of $\mathcal{V}_S$. □

1.3 Downward closure of $\mathcal{V}_M[\![.]\!]$

Lemma 1.3 (downward closure of $\mathcal{V}_M[\![.]\!]$)**.** *Let $\Delta \vdash \tau$ and $\Delta \vdash \delta$. For all $j, w, \mathbf{v}_1, \mathbf{v}_2$:*
If $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta$, then for all $(j', w') \sqsupseteq (j, w)$,
$(j', w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta$.

Proof. By induction on the derivation $\Delta \vdash \tau$. The $\mathbf{L}$ case relies on downward closure of $\mathcal{V}_S$. □

1.4 Weakening

Lemma 1.4 (weakening)**.** *Let $\Delta \vdash \tau$ and $\Delta \vdash \delta$. For all $j, w, \mathbf{v}_1, \mathbf{v}_2$:*
If $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta$,
then for all $\alpha, \tau_1, \tau_2, \mathbf{R}$ such that $\alpha \notin dom(\delta)$ and $(\tau_1, \tau_2, \mathbf{R}) \in SomeRel_j$,
$(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$.

Proof. Straightforward from the definition of $\mathcal{V}_M[\![.]\!]$ □

1.5 Downward closure of $\mathcal{D}[\![.]\!]$

Lemma 1.5 (downward closure of $\mathcal{D}[\![.]\!]$)**.** *If $(j, \delta) \in \mathcal{D}[\![\Delta]\!]$ and $j' \leq j$, then $(j', \delta) \in \mathcal{D}[\![\Delta]\!]$.*

Proof. Straightforward. □

1.6 Downward closure of $\mathcal{G}[\![.]\!]$

Lemma 1.6 (downward closure of $\mathcal{G}[\![.]\!]$)**.** *If* $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$ *and* $(j', w') \sqsupseteq (j, w)$*, then* $(j', \delta) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Proof. Straightforward from the definition of $\mathcal{G}[\![]\!]$ and downward closure of $\mathcal{V}_S$ and $\mathcal{V}_M[\![]\!]$. □

1.7 World approximation preserves key stores

Lemma 1.7 (world approximation preserves key stores)**.** *For all worlds* w *and approximation indices* n*, for* $i = 1, 2$*:*
If $w.K_i = K_i'$*, then* $\lfloor w \rfloor_n.K_i = K_i'$.

Proof. Immediate from the definition of $\lfloor . \rfloor_n$. □

1.8 Step decrease is valid world extension

Lemma 1.8 (step decrease is valid world extension)**.** *For all* j, w *such that* $w \in \mathit{World}_j$*,* $(j - 1, \lfloor w \rfloor_{j-1}) \sqsupseteq (j, w)$.

Proof. Straightforward from definitions. □

1.9 World component update is valid world extension

Lemma 1.9 (world component update is valid world extension)**.** *For all* $j, w, \alpha, k_1, k_2, \tau_1, \tau_2, \mathbf{R}$ *such that* $w \in \mathit{World}_j$ *and* $(\tau_1, \tau_2, \mathbf{R}) \in \mathit{SomeRel}_j$*:*
$(j, w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R})) \sqsupseteq (j, w)$.

Proof. We're required to show that the following five conjuncts of the definition of $\sqsupseteq$ hold:

- $j \leq j$ (immediate);
- $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}) \in \text{World}_j$, which we show as follows:

  From the premise $w \in \text{World}_j$, we have that w is some tuple $(K_1, K_2, \mathcal{C}, \Psi)$ such that $\mathcal{C} \in \text{Conc}$ and $\text{rng}(\Psi) \subseteq \text{SomeRel}_j$. We need to show that these conditions remain true for the updated world $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R})$:

  - $(w.\mathcal{C}[\alpha \mapsto (k_1, k_2)]) \in \text{Conc}$, which follows from the definition of Conc, since $\mathcal{C} \in \text{Conc}$;
  - $\text{rng}(w.\Psi[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]) = \text{SomeRel}_j$, which follows from $\text{rng}(\Psi) \subseteq \text{SomeRel}_j$ and the premise that $(\tau_1, \tau_2, \mathbf{R}) \in \text{SomeRel}_j$.
- $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).\Psi \sqsupseteq \lfloor w.\Psi \rfloor_j$, which we show as follows:

  By the definition of $\sqsupseteq$ on type variable interpretations, we must show that for all $\alpha' \in \text{dom}(\lfloor w.\Psi \rfloor_j)$, $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).\Psi(\alpha') = \lfloor w.\Psi \rfloor_j(\alpha')$.

  First note that since $w \in \text{World}_j$, $\lfloor w.\Psi \rfloor_j$ is equivalent to $w.\Psi$, so we have only to show that for all $\alpha' \in \text{dom}(w.\Psi)$, $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).\Psi(\alpha') = w.\Psi(\alpha')$.

  This follows from the fact that $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).\Psi = w.\Psi[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$.
- $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).K_1 \supseteq w.K_1$, which follows from the fact that
  $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).K_1 = (w.K_1, k_1)$;

- $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).K_2 \supseteq w.K_2$, which follows from the fact that $w \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}).K_2 = (w.K_2, k_2)$.

□

2 Compositionality

The compositionality lemma tells us that we have two equivalent ways of interpreting a type τ that contains occurrences of a type variable α. We can replace occurrences of α in τ with τ' before interpreting τ under δ, or we can choose to interpret τ as it is, occurrences of α and all, but under an extended δ that binds α to $(\delta_1(\tau'), \delta_2(\tau'), \mathcal{V}_M[\![\tau']\!]\delta)$—that is, one that binds α to an interpreted τ'.

Lemma 2.1 (compositionality)**.** *Let $\Delta \vdash \tau'$ and $\Delta \vdash \delta$ and let $\mathbf{R} = \mathcal{V}_M[\![\tau']\!]\delta$.*
Then, for all $j, w, \mathbf{v}_1, \mathbf{v}_2, \tau, \alpha$ such that $\Delta, \alpha \vdash \tau$:

$$(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau]\!]\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathbf{R})] \iff (j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau[\alpha := \tau']]\!]\delta.$$

Proof. By induction on derivations $\Delta, \alpha \vdash \tau$. Throughout the proof, we use δ^+ as an abbreviation for $\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathbf{R})]$.

- Case: $\dfrac{}{\Delta, \alpha \vdash \mathbf{Nat}}$

  Immediate in both directions since $\mathbf{Nat}$ contains no type variables.

- Case: $\dfrac{\Delta, \alpha \vdash \tau_1 \quad \Delta, \alpha \vdash \tau_2}{\Delta, \alpha \vdash \tau_1 \to \tau_2}$

  – $\Longrightarrow$:
  Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta^+$.
  To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![(\tau_1 \to \tau_2)[\alpha := \tau']]\!]\delta$,
  or, equivalently, that $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1[\alpha := \tau'] \to \tau_2[\alpha := \tau']]\!]\delta$.
  Note that it must be the case that
  $\mathbf{v}_1 \equiv \lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_1$
  and
  $\mathbf{v}_2 \equiv \lambda \mathbf{x} : \delta_2(\tau_1).\ \mathbf{e}_2$
  for some $\mathbf{e}_1, \mathbf{e}_2$.
  Consider arbitrary $j', w', \mathbf{v}_{11}, \mathbf{v}_{21}$ such that $(j', w') \sqsupseteq (j, w)$ and
  $(j', w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1[\alpha := \tau']]\!]\delta$.
  Instantiate $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta^+$ with $j', w', \mathbf{v}_{11}, \mathbf{v}_{21}$. Note that:
  * $(j', w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1]\!]\delta^+$ by the induction hypothesis (in the $\Longleftarrow$ direction), and
  * $(j', w') \sqsupseteq (j, w)$.

  Therefore we have that: $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2]\!]\delta^+$.
  To show: $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2[\alpha := \tau']]\!]\delta$.
  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     * $i < j'$, and
     * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i \mathbf{Error}$: str.

To show: $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^* \textbf{Error}: \text{str}$.
Instantiate the first conjunct of $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2]\!]\delta^+$ with i, str. Note that:

* $i < j'$, and
* $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i \textbf{Error}: \text{str}$.

Therefore we have that $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^* \textbf{Error}: \text{str}$, as we were required to show.

2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v'}_1$ such that:
   * $i < j'$, and
   * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   To show: There exist $K_2', \mathbf{v'}_2, w''$ such that:
   * $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   * $(j' - i, w'') \sqsupseteq (j', w')$,
   * $w''.K_1 = K_1'$,
   * $w''.K_2 = K_2'$, and
   * $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau_2[\alpha := \tau']]\!]\delta$.

   Instantiate the second conjunct of $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2]\!]\delta^+$ with $i, K_1', \mathbf{v'}_1$. Note that:
   * $i < j'$, and
   * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   Therefore there exist $K_2', \mathbf{v'}_2, w''$ such that:
   * $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   * $(j' - i, w'') \sqsupseteq (j', w')$,
   * $w''.K_1 = K_1'$,
   * $w''.K_2 = K_2'$, and
   * $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau_2]\!]\delta^+$,

   which fulfills our proof obligation since, by the induction hypothesis, we have that $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau_2[\alpha := \tau']]\!]\delta$.

– $\Longleftarrow$:

Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![(\tau_1 \to \tau_2)[\alpha := \tau']]\!]\delta$,
or, equivalently, $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1[\alpha := \tau'] \to \tau_2[\alpha := \tau']]\!]\delta$.
To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta^+$.
Note that it must be the case that
$\mathbf{v}_1 \equiv \lambda \mathbf{x} : \delta_1(\tau_1[\alpha := \tau']).\ \mathbf{e}_1$
and
$\mathbf{v}_2 \equiv \lambda \mathbf{x} : \delta_2(\tau_1[\alpha := \tau']).\ \mathbf{e}_2$
for some $\mathbf{e}_1, \mathbf{e}_2$.
Consider arbitrary $j', w', \mathbf{v}_{11}, \mathbf{v}_{21}$ such that $(j', w') \sqsupseteq (j, w)$ and
$(j', w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1]\!]\delta^+$.
Instantiate $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1[\alpha := \tau'] \to \tau_2[\alpha := \tau']]\!]\delta$ with $j', w', \mathbf{v}_{11}, \mathbf{v}_{21}$. Note that:

* $(j', w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1[\alpha := \tau']]\!]\delta$ by the induction hypothesis (in the $\Longrightarrow$ direction), and
* $(j', w') \sqsupseteq (j, w)$.

Therefore we have that: $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2[\alpha := \tau']]\!]\delta$.
To show: $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2]\!]\delta^+$.
The proof is in two parts:

1. (Error conjunct): Consider arbitrary i, str such that:
   * $i < j'$, and
   * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i$ **Error**: str.

   To show: $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^*$ **Error**: str.
   Instantiate the first conjunct of $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2[\alpha := \tau']]\!]\delta$ with i, str. Note that:
   * $i < j'$, and
   * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i$ **Error**: str.

   Therefore we have that $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^*$ **Error**: str, as we were required to show.
2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v'}_1$ such that:
   * $i < j'$, and
   * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   To show: There exist $K_2', \mathbf{v'}_2, w''$ such that:
   * $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   * $(j' - i, w'') \sqsupseteq (j', w')$,
   * $w''.K_1 = K_1'$,
   * $w''.K_2 = K_2'$, and
   * $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau_2]\!]\delta^+$.

   Instantiate the second conjunct of $(j', w', \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}], \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}]) \in \mathcal{E}_M[\![\tau_2[\alpha := \tau']]\!]\delta$ with $i, K_1', \mathbf{v'}_1$. Note that:
   * $i < j'$, and
   * $w'.K_1 \triangleright \mathbf{e}_1[\mathbf{x} := \mathbf{v}_{11}] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   Therefore there exist $K_2', \mathbf{v'}_2, w''$ such that:
   * $w'.K_2 \triangleright \mathbf{e}_2[\mathbf{x} := \mathbf{v}_{21}] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   * $(j' - i, w'') \sqsupseteq (j', w')$,
   * $w''.K_1 = K_1'$,
   * $w''.K_2 = K_2'$, and
   * $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau_2[\alpha := \tau']]\!]\delta$,

   which fulfills our proof obligation since, by the induction hypothesis, we have that $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\tau_2]\!]\delta^+$.

- Case: $\dfrac{\Delta, \alpha \vdash \tau}{\Delta, \alpha \vdash \tau^*}$

  - $\Longrightarrow$:
    Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau^*]\!]\delta^+$.
    To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau^*[\alpha := \tau']]\!]\delta$.
    There are two possibilities to consider:
    * $\mathbf{v}_1 \equiv \mathbf{v}_2 \equiv \mathbf{nil}$.
      In this case, we have only to note that $\tau^*[\alpha := \tau'] \equiv (\tau[\alpha := \tau'])^*$. Since **nil** is related to **nil** at *any* list type, under any δ, it's clearly the case that $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau[\alpha := \tau']^*]\!]\delta$, fulfilling our proof obligation.

* $\mathbf{v}_1 \equiv (\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12})$, and
$\mathbf{v}_2 \equiv (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})$,
for some $\mathbf{v}_{11}$ and $\mathbf{v}_{21}$ and some $\mathbf{v}_{12}$ and $\mathbf{v}_{22}$.
Expanding the definition of $\mathcal{V}_M[\![\tau^*]\!]\delta^+$, we have that
$(j, w, \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau]\!]\delta^+$
and that $(j, w, \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau^*]\!]\delta^+$.
So, by the induction hypothesis (in the $\Longrightarrow$ direction), we have that
$(j, w, \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau[\alpha := \tau']]\!]\delta$
and that $(j, w, \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau^*[\alpha := \tau']]\!]\delta$.
Since $\tau^*[\alpha := \tau'] \equiv (\tau[\alpha := \tau'])^*$, we have that
$(j, w, \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![(\tau[\alpha := \tau'])^*]\!]\delta$.
Therefore,
$(j, w, (\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12}), (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})) \in \mathcal{V}_M[\![(\tau[\alpha := \tau'])^*]\!]\delta$,
which is equivalent to
$(j, w, (\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12}), (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})) \in \mathcal{V}_M[\![\tau^*[\alpha := \tau']]\!]\delta$,
as we were required to show.

– $\Longleftarrow$:
Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau^*[\alpha := \tau']]\!]\delta$.
To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau^*]\!]\delta^+$.
We consider both possibilities:

* $\mathbf{v}_1 \equiv \mathbf{v}_2 \equiv \mathbf{nil}$.
Immediate, since $\mathbf{nil}$ is related to $\mathbf{nil}$ at any list type and under any δ.

* $\mathbf{v}_1 \equiv (\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12})$, and
$\mathbf{v}_2 \equiv (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})$,
for some $\mathbf{v}_{11}$ and $\mathbf{v}_{21}$ and some $\mathbf{v}_{12}$ and $\mathbf{v}_{22}$.
Note that $\tau^*[\alpha := \tau']$ is equivalent to $(\tau[\alpha := \tau'])^*$. So, expanding the definition of $\mathcal{V}_M[\![(\tau[\alpha := \tau'])^*]\!]\delta$, we have that
$(j, w, \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau[\alpha := \tau']]\!]\delta$
and that $(j, w, \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![(\tau[\alpha := \tau'])^*]\!]\delta$.
Therefore, by the induction hypothesis (in the $\Longleftarrow$ direction), we have that:
$(j, w, \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau]\!]\delta^+$
and that $(j, w, \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau^*]\!]\delta^+$.
We therefore have that
$(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau^*]\!]\delta^+$,
as we were required to show.

- Case: $\dfrac{\Delta, \alpha \vdash \sigma}{\Delta, \alpha \vdash \forall\alpha.\ \sigma}$

Immediate in both directions since $\forall\alpha.\ \sigma$ contains no free occurrences of α.

- Case: $\dfrac{\Delta, \alpha, \beta \vdash \sigma}{\Delta, \alpha \vdash \forall\beta.\ \sigma}$

– $\Longrightarrow$:
Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\forall\beta.\ \sigma]\!]\delta^+$.
To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![(\forall\beta.\ \sigma)[\alpha := \tau']]\!]\delta$.
Since $\beta \neq \alpha$, we know that $(\forall\beta.\ \sigma)[\alpha := \tau'] \equiv \forall\beta.\ (\sigma[\alpha := \tau'])$, so it's equivalent to show that
$(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\forall\beta.\ (\sigma[\alpha := \tau'])]\!]\delta$.

Note that it must be the case that

$\mathbf{v}_1 \equiv \Lambda\beta.\ \mathbf{e}_1$

and

$\mathbf{v}_2 \equiv \Lambda\beta.\ \mathbf{e}_2$.

Consider arbitrary $j', w', \tau_1, \tau_2, \mathbf{R}', k_1, k_2$ such that:

* $(j', w') \sqsupseteq (j, w)$,
* $(\tau_1, \tau_2, \mathbf{R}') \in \text{SomeRel}_{j'}$,
* $k_1 \notin w'.K_1$, and
* $k_2 \notin w'.K_2$.

Instantiate $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\forall\beta.\ \sigma]\!]\delta^+$ with $j', w', \tau_1, \tau_2, \mathbf{R}', k_1, k_2$. Note that:

* $(j', w') \sqsupseteq (j, w)$,
* $(\tau_1, \tau_2, \mathbf{R}') \in \text{SomeRel}_{j'}$,
* $k_1 \notin w'.K_1$, and
* $k_2 \notin w'.K_2$.

We therefore have that:

$(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
$\quad \blacktriangleright\ \mathcal{E}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

To show:

$(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
$\quad \blacktriangleright\ \mathcal{E}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

The proof is in two parts.

(Note that the K_1 and K_2 components of $w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}')$ are equivalent to $w'.K_1 \uplus \{k_1\}$ and $w'.K_2 \uplus \{k_2\}$, respectively. Therefore, in the below we write $w'.K_1 \uplus \{k_1\}$ and $w'.K_2 \uplus \{k_2\}$ in place of $(w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}')).K_1$ and $(w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}')).K_2$.)

1. (Error conjunct): Consider arbitrary i, str such that:
   * $i < j'$, and
   * $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i \textbf{Error}$: str.

   To show: $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^* \textbf{Error}$: str.
   Instantiate the first conjunct of
   $(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
   $\quad \blacktriangleright\ \mathcal{E}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$
   with i, str. Note that:
   * $i < j'$, and
   * $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i \textbf{Error}$: str.

   Therefore we have that $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^* \textbf{Error}$: str, as we were required to show.
2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v'}_1$ such that:
   * $i < j'$, and
   * $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   To show: There exist $K_2', \mathbf{v'}_2, w''$ such that:
   * $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   * $(j' - i, w'') \sqsupseteq (j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'))$,

* $w''.K_1 = K'_1$,
* $w''.K_2 = K'_2$, and
* $(j' - i, w'', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

Instantiate the second conjunct of
$(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
$\quad \blacktriangleright\, \mathcal{E}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$
with $i, K'_1, \mathbf{v}'_1$. Note that:

* $i < j'$, and
* $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i K'_1 \triangleright \mathbf{v}'_1$.

Therefore there exist $K'_2, \mathbf{v}'_2, w''$ such that:

* $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^* K'_2 \triangleright \mathbf{v}'_2$,
* $(j' - i, w'') \sqsupseteq (j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'))$,
* $w''.K_1 = K'_1$,
* $w''.K_2 = K'_2$, and
* $(j' - i, w'', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$,

which fulfills our proof obligation, since, by the induction hypothesis, we have that $(j' - i, w'', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

– $\Longleftarrow$:

Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![(\forall\beta.\ \sigma)[\alpha := \tau']]\!]\delta$.
To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\forall\beta.\ \sigma]\!]\delta^+$.
Since $\beta \neq \alpha$, we know that $(\forall\beta.\ \sigma)[\alpha := \tau'] \equiv \forall\beta.\ \sigma[\alpha := \tau']$, so we have that
$(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\forall\beta.\ \sigma[\alpha := \tau']]\!]\delta$.
Note that it must be the case that
$\mathbf{v}_1 \equiv \Lambda\beta.\ \mathbf{e}_1$
and
$\mathbf{v}_2 \equiv \Lambda\beta.\ \mathbf{e}_2$
for some $\mathbf{e}_1, \mathbf{e}_2$.
Consider arbitrary $j', w', \tau_1, \tau_2, \mathbf{R}', k_1, k_2$ such that:

* $(j', w') \sqsupseteq (j, w)$,
* $(\tau_1, \tau_2, \mathbf{R}') \in \text{SomeRel}_{j'}$, and
* $k_1 \notin w'.K_1$, and
* $k_2 \notin w'.K_2$.

Instantiate $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\forall\beta.\ \sigma[\alpha := \tau']]\!]\delta$ with $j', w', \tau_1, \tau_2, \mathbf{R}', k_1, k_2$. Note that:

* $(j', w') \sqsupseteq (j, w)$,
* $(\tau_1, \tau_2, \mathbf{R}') \in \text{SomeRel}_{j'}$, and
* $k_1 \notin w'.K_1$, and
* $k_2 \notin w'.K_2$.

We therefore have that:
$(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
$\quad \blacktriangleright\, \mathcal{E}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.
To show:

$(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
$\quad \blacktriangleright\, \mathcal{E}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

The proof is in two parts.

(As in the $\Longrightarrow$ direction, the K_1 and K_2 components of $w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}')$ are equivalent to $w'.K_1 \uplus \{k_1\}$ and $w'.K_2 \uplus \{k_2\}$, respectively. Therefore, in the below we write $w'.K_1 \uplus \{k_1\}$ and $w'.K_2 \uplus \{k_2\}$ in place of $(w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}')).K_1$ and $(w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}')).K_2$.)

1. (Error conjunct): Consider arbitrary i, str such that:
   - $i < j'$, and
   - $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i$ **Error**: str.

   To show: $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^*$ **Error**: str.
   Instantiate the first conjunct of
   $(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
   $\quad \blacktriangleright\, \mathcal{E}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$
   with i, str. Note that:
   - $i < j'$, and
   - $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i$ **Error**: str.

   Therefore we have that $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^*$ **Error**: str, as we were required to show.
2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v'}_1$ such that:
   - $i < j'$, and
   - $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   To show: There exist $K_2', \mathbf{v'}_2, w''$ such that:
   - $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   - $(j' - i, w'') \sqsupseteq (j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'))$,
   - $w''.K_1 = K_1'$,
   - $w''.K_2 = K_2'$, and
   - $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

   Instantiate the second conjunct of
   $(j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'), \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle], \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle]) \in$
   $\quad \blacktriangleright\, \mathcal{E}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$
   with $i, K_1', \mathbf{v'}_1$. Note that:
   - $i < j'$, and
   - $w'.K_1 \uplus \{k_1\} \triangleright \mathbf{e}_1[\beta := \langle k_1;\ \tau_1\rangle] \longmapsto^i K_1' \triangleright \mathbf{v'}_1$.

   Therefore there exist $K_2', \mathbf{v'}_2, w''$ such that:
   - $w'.K_2 \uplus \{k_2\} \triangleright \mathbf{e}_2[\beta := \langle k_2;\ \tau_2\rangle] \longmapsto^* K_2' \triangleright \mathbf{v'}_2$,
   - $(j' - i, w'') \sqsupseteq (j', w' \boxplus (\beta \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}'))$,
   - $w''.K_1 = K_1'$,
   - $w''.K_2 = K_2'$, and
   - $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\sigma[\alpha := \tau']]\!]\delta[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$,

   which fulfills our proof obligation, since, by the induction hypothesis, we have that $(j' - i, w'', \mathbf{v'}_1, \mathbf{v'}_2) \in \mathcal{V}_M[\![\sigma]\!]\delta^+[\beta \mapsto (\tau_1, \tau_2, \mathbf{R}')]$.

- Case: $\dfrac{\alpha \in \Delta, \alpha}{\Delta, \alpha \vdash \alpha}$

- $\Longrightarrow$:
  Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\alpha]\!]\delta^+$.
  To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\alpha[\alpha := \tau']]\!]\delta$,
  or, equivalently, that $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau']\!]\delta$.
  From the definition of $\mathcal{V}_M[\![\alpha]\!]\delta^+$, we have that
  $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \delta^+(\alpha).R$.
  Since $\delta^+(\alpha) = (\delta_1(\tau'), \delta_2(\tau'), \mathbf{R})$, we have that $\delta^+(\alpha).R = \mathbf{R}$.
  Therefore $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathbf{R}$, so $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau']\!]\delta$, as we were required to show.
- $\Longleftarrow$:
  Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\alpha[\alpha := \tau']]\!]\delta$.
  To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\alpha]\!]\delta^+$.
  Since $\mathcal{V}_M[\![\alpha[\alpha := \tau']]\!]\delta \equiv \mathcal{V}_M[\![\tau']\!]\delta$,
  we have that $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathbf{R}$.
  Since $\delta^+(\alpha).R \equiv \mathbf{R}$, then, we have that $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \delta^+(\alpha).R$.
  Therefore $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\alpha]\!]\delta^+$, as required.

- Case: $\dfrac{\beta \in \Delta, \alpha}{\Delta, \alpha \vdash \beta}$
  - $\Longrightarrow$:
    Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta]\!]\delta^+$.
    To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta[\alpha := \tau']]\!]\delta$,
    or, equivalently, that $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta]\!]\delta$.
    From the definition of $\mathcal{V}_M[\![\beta]\!]\delta^+$, we have that
    $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \delta^+(\beta).R$.
    Since δ^+ is defined as $\delta[\alpha \mapsto (\delta_1(\tau'), \delta_2(\tau'), \mathbf{R})]$—that is, δ^+ is simply δ extended with a binding for α—we know that $\delta^+(\beta).R$ is equivalent to $\delta(\beta).R$.
    So $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \delta(\beta).R$.
    Therefore $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta]\!]\delta$, as we were required to show.
  - $\Longleftarrow$:
    Given: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta[\alpha := \tau']]\!]\delta$.
    To show: $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta]\!]\delta^+$.
    Since β is a type variable, $\beta[\alpha := \tau'] \equiv \beta$, so we have that
    $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta]\!]\delta$.
    Therefore, $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \delta(\beta).R$.
    Since $\delta(\beta).R \equiv \delta^+(\beta).R$, we have that, $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \delta^+(\beta).R$.
    Therefore $(j, w, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\beta]\!]\delta^+$.

- Case: $\dfrac{}{\Delta, \alpha \vdash \mathbf{L}}$

  Immediate in both directions since $\mathbf{L}$ contains no type variables.

□

3 Bridge lemma

Lemma 3.1 (bridge lemma)**.** *For all $j \geq 0$, worlds w such that $w \in \mathit{World}_j$, type environments Δ, type relations δ such that $\Delta \vdash \delta$, and types τ such that $\Delta \vdash \tau$, both of the following hold:*

1. *For all e_1 and e_2,*
   if $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$ then $(j, w, ({}^{\delta_1(\tau)}MS\ \mathsf{e}_1), ({}^{\delta_2(\tau)}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\tau]\!]\delta$.
2. *For all $\mathbf{e}_1$ and $\mathbf{e}_2$,*
   if $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\tau]\!]\delta$ then $(j, w, (SM^{\delta_1(\tau)}\ \mathbf{e}_1), (SM^{\delta_2(\tau)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

Proof. We prove (1) and (2) simultaneously, by induction on derivations $\Delta \vdash \tau$.

- Case (1): $\dfrac{}{\Delta \vdash \mathbf{Nat}}$

  Given: $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$.

  To show: $(j, w, ({}^{\delta_1(\mathbf{Nat})}MS\ \mathsf{e}_1), ({}^{\delta_2(\mathbf{Nat})}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\mathbf{Nat}]\!]\delta$.

  Since $\delta_1(\mathbf{Nat}) \equiv \delta_2(\mathbf{Nat}) \equiv \mathbf{Nat}$, it's equivalent to show that

  $(j, w, ({}^{\mathbf{Nat}}MS\ \mathsf{e}_1), ({}^{\mathbf{Nat}}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\mathbf{Nat}]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct):

     Consider arbitrary i, str such that:

     - $i < j$, and
     - $w.K_1 \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{e}_1) \longmapsto^i \mathbf{Error}$: str.

     To show: $w.K_2 \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{e}_2) \longmapsto^* \mathbf{Error}$: str.

     From the operational semantics, there are two ways that $w.K_1 \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{e}_1)$ might have evaluated to **Error**: str: (a) during the evaluation of e_1, or (b) after e_1 has evaluated to some non-number value v_1, raising **Error**: non-num. We consider both these possibilities:

     (a) $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^i \mathbf{Error}$: str.
     Instantiate the first conjunct of the premise $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$ with i, str. Since $i < j$, we have that $w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* \mathbf{Error}$: str.
     It follows that $w.K_2 \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{e}_2) \longmapsto^* \mathbf{Error}$: str, as we were required to show.

     (b) $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^{i_1} K_1' \triangleright \mathsf{v}_1$ and $K_1' \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{v}_1) \longmapsto^{i_2} \mathbf{Error}$: non-num and $i = i_1 + i_2$.
     Instantiate the second conjunct of the premise $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$ with i_1, K_1', v_1. Note that:

     - $i_1 < j$, and
     - $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^{i_1} K_1' \triangleright \mathsf{v}_1$.

     Therefore, there exist K_2', v_2, w' such that:

     - $w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* K_2' \triangleright \mathsf{v}_2$,
     - $(j - i_1, w') \sqsupseteq (j, w)$,
     - $w.K_1 = K_1'$,
     - $w.K_2 = K_2'$, and
     - $(j - i_1, w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.

Since v_1 and v_2 are related and we know that v_1 is a non-num, we also know that v_2 is a non-num. Therefore, from above and from the operational semantics, we have that:

$$\begin{array}{rcl} w.K_2 \triangleright (^{\mathbf{Nat}}MS\ \mathsf{e}_2) & \longmapsto^* & K_2' \triangleright (^{\mathbf{Nat}}MS\ \mathsf{v}_2) \\ & \longmapsto^* & \textbf{Error}\text{: non-num}, \end{array}$$

as we were required to show.

2. (Value conjunct):

   Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:

   - $i < j$, and
   - $w.K_1 \triangleright (^{\mathbf{Nat}}MS\ \mathsf{e}_1) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

   To show: There exist $K_2', \mathbf{v}_2, w'$ such that:

   - $w.K_2 \triangleright (^{\mathbf{Nat}}MS\ \mathsf{e}_2) \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
   - $(j - i, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_1'$,
   - $w'.K_2 = K_2'$, and
   - $(j - i, w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{Nat}]\!]\delta$.

   Since $w.K_1 \triangleright (^{\mathbf{Nat}}MS\ \mathsf{e}_1) \longmapsto^i K_1' \triangleright \mathbf{v}_1$, from the operational semantics it follows that there exist i_1, $\overline{n}$, and $\overline{n}$ such that

$$\begin{array}{rcl} w.K_1 \triangleright (^{\mathbf{Nat}}MS\ \mathsf{e}_1) & \longmapsto^{i_1} & K_1' \triangleright (^{\mathbf{Nat}}MS\ \overline{n}) \\ & \longmapsto^{1} & K_1' \triangleright \overline{n}. \end{array}$$

   (Note that the key store is already K_1' after i_1 steps, since no keys can be generated during the one-step transition from $(^{\mathbf{Nat}}MS\ \overline{n})$ to $\overline{n}$.)

   Therefore, it follows that

   - $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^{i_1} K_1' \triangleright \overline{n}$,
   - $\mathbf{v}_1 = \overline{n}$, and
   - $i = i_1 + 1$.

   Instantiate the second conjunct of $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$ with $i_1, K_1', \overline{n}$. Note that:

   - $i_1 < j$ (since $i_1 = i - 1$ and $i < j$), and
   - $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^{i_1} K_1' \triangleright \overline{n}$ (from above).

   Therefore, there exist K_2'', v_2, w'' such that:

   - $w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* K_2'' \triangleright \mathsf{v}_2$,
   - $(j - i_1, w'') \sqsupseteq (j, w)$,
   - $w''.K_1 = K_1'$,
   - $w''.K_2 = K_2''$, and
   - $(j - i_1, w'', \overline{n}, \mathsf{v}_2) \in \mathcal{V}_S$.

   Note that by the definition of $\mathcal{V}_S$, v_2 must be $\overline{n}$.

   Choose:

   - $K_2' = K_2''$,
   - $\mathbf{v}_2 = \overline{n}$,
   - $w' = \lfloor w'' \rfloor_{j-i}$.

To show:

- $w.K_2 \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{e}_2) \longmapsto^* K_2'' \triangleright \overline{n}$, which follows from above, since we have that

$$\begin{array}{lll} w.K_2 \triangleright ({}^{\mathbf{Nat}}MS\ \mathsf{e}_2) & \longmapsto^* & K_2'' \triangleright ({}^{\mathbf{Nat}}MS\ \overline{n}) \\ & \longmapsto^1 & K_2'' \triangleright \overline{n}; \end{array}$$

- $(j - i, \lfloor w'' \rfloor_{j-i}) \sqsupseteq (j, w)$, which follows from Lemma 1.1, since:
  - $(j - i_1, w'') \sqsupseteq (j, w)$ (from above), and
  - $(j - i, \lfloor w'' \rfloor_{j-i}) \sqsupseteq (j - i_1, w'')$ (because $(j - i) \leq (j - i_1)$, since $i_1 < i$);
- $\lfloor w'' \rfloor_{j-i}.K_1 = K_1'$, which follows from Lemma 1.7 and from above;
- $\lfloor w'' \rfloor_{j-i}.K_2 = K_2''$, which follows from Lemma 1.7 and from above;
- $(j - i, \lfloor w'' \rfloor_{j-i}, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{Nat}]\!]\delta$ (immediate from the definition of $\mathcal{V}_M[\![\mathbf{Nat}]\!]$, since $\mathbf{v}_1 = \mathbf{v}_2 = \overline{n}$).

- Case (2): $\dfrac{}{\Delta \vdash \mathbf{Nat}}$

  Given: $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\mathbf{Nat}]\!]\delta$.

  To show: $(j, w, (SM^{\delta_1(\mathbf{Nat})}\ \mathbf{e}_1), (SM^{\delta_2(\mathbf{Nat})}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  Since $\delta_1(\mathbf{Nat}) \equiv \delta_2(\mathbf{Nat}) \equiv \mathbf{Nat}$, it's equivalent to show that

  $(j, w, (SM^{\mathbf{Nat}}\ \mathbf{e}_1), (SM^{\mathbf{Nat}}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct:)

     Consider arbitrary i, str such that:

     - $i < j$, and
     - $w.K_1 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_1) \longmapsto^i$ **Error**: str.

     To show: $w.K_2 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_2) \longmapsto^*$ **Error**: str.

     From the operational semantics, for $w.K_1 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_1)$ to have evaluated to **Error**: str, there must have been an error during the evaluation of $\mathbf{e}_1$. (The alternative would be for $\mathbf{e}_1$ to have evaluated to some non-number value $\mathbf{v}_1$, raising **Error**: non-num, but that situation cannot occur since since $\mathbf{e}_1$ has type $\mathbf{Nat}$.) Therefore it must be the case that

     $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^i$ **Error**: str.

     Instantiate the first conjunct of the premise $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\tau]\!]\delta$ with i, str. Since $i < j$, we have that $w.K_2 \triangleright \mathbf{e}_2 \longmapsto^*$ **Error**: str.

     It follows that $w.K_2 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_2) \longmapsto^*$ **Error**: str, as we were required to show.

  2. (Value conjunct:)

     Consider arbitrary i, K_1', v_1 such that:

     - $i < j$, and
     - $w.K_1 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_1) \longmapsto^i K_1' \triangleright \mathsf{v}_1$.

     To show: There exist K_2', v_2, w' such that:

     - $w.K_2 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_2) \longmapsto^* K_2' \triangleright \mathsf{v}_2$,
     - $(j - i, w') \sqsupseteq (j, w)$,
     - $w'.K_1 = K_1'$,

- $w'.K_2 = K_2'$, and
- $(j - i, w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.

Since $w.K_1 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_1) \longmapsto^i K_1' \triangleright \mathsf{v}_1$, from the operational semantics it follows that there exist i_1, $\overline{n}$, and $\overline{n}$ such that

$$\begin{array}{lll} w.K_1 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_1) & \longmapsto^{i_1} & K_1' \triangleright (SM^{\mathbf{Nat}}\ \overline{n}) \\ & \longmapsto^{1} & K_1' \triangleright \overline{n}. \end{array}$$

(Note that the key store is already K_1' after i_1 steps, since no keys can be generated during the one-step transition from $(SM^{\mathbf{Nat}}\ \overline{n})$ to $\overline{n}$.)

Therefore, it follows that

- $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^{i_1} K_1' \triangleright \overline{n}$,
- $\mathsf{v}_1 = \overline{n}$, and
- $i = i_1 + 1$.

Instantiate the second conjunct of $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\mathbf{Nat}]\!]\delta$ with $i_1, K_1', \overline{n}$. Note that:

- $i_1 < j$ (since $i_1 = i - 1$ and $i < j$), and
- $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^{i_1} K_1' \triangleright \overline{n}$ (from above).

Therefore, there exist $K_2'', \mathbf{v}_2, w''$ such that:

- $w.K_2 \triangleright \mathbf{e}_2 \longmapsto^* K_2'' \triangleright \mathbf{v}_2$,
- $(j - i_1, w'') \sqsupseteq (j, w)$,
- $w''.K_1 = K_1'$,
- $w''.K_2 = K_2''$, and
- $(j - i_1, w'', \overline{n}, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{Nat}]\!]\delta$.

Note that by the definition of $\mathcal{V}_M[\![\mathbf{Nat}]\!]$, $\mathbf{v}_2$ must be $\overline{n}$.

Choose:

- $K_2' = K_2''$,
- $\mathsf{v}_2 = \overline{n}$,
- $w' = \lfloor w'' \rfloor_{j-i}$.

To show:

- $w.K_2 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_2) \longmapsto^* K_2'' \triangleright \overline{n}$, which follows from above, since we have that

$$\begin{array}{lll} w.K_2 \triangleright (SM^{\mathbf{Nat}}\ \mathbf{e}_2) & \longmapsto^* & K_2'' \triangleright (SM^{\mathbf{Nat}}\ \overline{n}) \\ & \longmapsto^{1} & K_2'' \triangleright \overline{n}; \end{array}$$

- $(j - i, \lfloor w'' \rfloor_{j-i}) \sqsupseteq (j, w)$, which follows from Lemma 1.1, since:
  - $(j - i_1, w'') \sqsupseteq (j, w)$ (from above), and
  - $(j - i, \lfloor w'' \rfloor_{j-i}) \sqsupseteq (j - i_1, w'')$ (because $(j - i) \leq (j - i_1)$, since $i_1 < i$);
- $\lfloor w'' \rfloor_{j-i}.K_1 = K_1'$, which follows from Lemma 1.7 and from above;
- $\lfloor w'' \rfloor_{j-i}.K_2 = K_2''$, which follows from Lemma 1.7 and from above;
- $(j - i, \lfloor w'' \rfloor_{j-i}, \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$ (immediate from the definition of $\mathcal{V}_S$, since $\mathsf{v}_1 = \mathsf{v}_2 = \overline{n}$).

- Case (1): $\frac{\Delta \vdash \tau_1 \quad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \to \tau_2}$

  Given: $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$.

  To show: $(j, w, ({}^{\delta_1(\tau_1 \to \tau_2)}MS\ \mathsf{e}_1), ({}^{\delta_2(\tau_1 \to \tau_2)}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$.

  Equivalently, show:

  $(j, w, ({}^{\delta_1(\tau_1) \to \delta_1(\tau_2)}MS\ \mathsf{e}_1), ({}^{\delta_2(\tau_1) \to \delta_2(\tau_2)}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct):
     Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright ({}^{\delta_1(\tau_1) \to \delta_1(\tau_2)}MS\ \mathsf{e}_1) \longmapsto^i$ **Error**: str.

     To show: $w.K_2 \triangleright ({}^{\delta_2(\tau_1) \to \delta_2(\tau_2)}MS\ \mathsf{e}_2) \longmapsto^*$ **Error**: str.

     Straightforward from the operational semantics and above assumptions.
  2. (Value conjunct): Straightforward.

- Case (2): $\frac{\Delta \vdash \tau_1 \quad \Delta \vdash \tau_2}{\Delta \vdash \tau_1 \to \tau_2}$

  Given: $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$.

  To show: $(j, w, (SM^{\delta_1(\tau_1 \to \tau_2)}\ \mathbf{e}_1), (SM^{\delta_2(\tau_1 \to \tau_2)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  Equivalently, show: $(j, w, (SM^{\delta_1(\tau_1) \to \delta_1(\tau_2)}\ \mathbf{e}_1), (SM^{\delta_2(\tau_1) \to \delta_2(\tau_2)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct):
     Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright (SM^{\delta_1(\tau_1) \to \delta_1(\tau_2)}\ \mathbf{e}_1) \longmapsto^i$ **Error**: str.

     To show: $w.K_2 \triangleright (SM^{\delta_2(\tau_1) \to \delta_2(\tau_2)}\ \mathbf{e}_2) \longmapsto^*$ **Error**: str.

     Straightforward from the operational semantics and above assumptions.
  2. (Value conjunct): Straightforward.

- Case (1): $\frac{\Delta \vdash \tau}{\Delta \vdash \tau^*}$

  Given: $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$.

  To show: $(j, w, ({}^{\delta_1(\tau^*)}MS\ \mathsf{e}_1), ({}^{\delta_2(\tau^*)}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\tau^*]\!]\delta$.

  Equivalently, show: $(j, w, ({}^{(\delta_1(\tau))^*}MS\ \mathsf{e}_1), ({}^{(\delta_2(\tau))^*}MS\ \mathsf{e}_2)) \in \mathcal{E}_M[\![\tau^*]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct):
     Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright ({}^{(\delta_1(\tau))^*}MS\ \mathsf{e}_1) \longmapsto^i$ **Error**: str.

To show: $w.K_2 \triangleright ({}^{(\delta_2(\tau))^*}MS\ \mathtt{e}_2) \longmapsto^*$ **Error**: str.

Straightforward from the operational semantics and above assumptions.

2. (Value conjunct):
   Straightforward.

- Case (2): $\frac{\Delta \vdash \tau}{\Delta \vdash \tau^*}$

  Given: $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\tau^*]\!]\delta$.

  To show: $(j, w, (SM^{\delta_1(\tau^*)}\ \mathbf{e}_1), (SM^{\delta_2(\tau^*)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  Equivalently, show: $(j, w, (SM^{(\delta_1(\tau))^*}\ \mathbf{e}_1), (SM^{(\delta_2(\tau))^*}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct):
     Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright (SM^{(\delta_1(\tau))^*}\ \mathbf{e}_1) \longmapsto^i$ **Error**: str.

     To show: $w.K_2 \triangleright (SM^{(\delta_2(\tau))^*}\ \mathbf{e}_2) \longmapsto^*$ **Error**: str.

     Straightforward from the operational semantics and above assumptions.
  2. (Value conjunct):
     Straightforward.

- Case (1): $\frac{\Delta, \alpha \vdash \tau}{\Delta \vdash \forall\alpha.\ \tau}$

  Given: $(j, w, \mathtt{e}_1, \mathtt{e}_2) \in \mathcal{E}_S$.

  To show: $(j, w, ({}^{\delta_1(\forall\alpha.\ \tau)}MS\ \mathtt{e}_1), ({}^{\delta_2(\forall\alpha.\ \tau)}MS\ \mathtt{e}_2)) \in \mathcal{E}_M[\![\forall\alpha.\ \tau]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct):
     Straightforward.
  2. (Value conjunct):
     Straightforward.

- Case (2): $\frac{\Delta, \alpha \vdash \tau}{\Delta \vdash \forall\alpha.\ \tau}$

  Given: $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\forall\alpha.\ \tau]\!]\delta$.

  To show: $(j, w, (SM^{\delta_1(\forall\alpha.\ \tau)}\ \mathbf{e}_1), (SM^{\delta_2(\forall\alpha.\ \tau)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct):
     Straightforward.
  2. (Value conjunct):
     Straightforward.

- Case (1): $\frac{\alpha \in \Delta}{\Delta \vdash \alpha}$

  Given: $(j, w, \mathtt{e}_1, \mathtt{e}_2) \in \mathcal{E}_S$.

  To show: $(j, w, ({}^{\delta_1(\alpha)}MS\ \mathtt{e}_1), ({}^{\delta_2(\alpha)}MS\ \mathtt{e}_2)) \in \mathcal{E}_M[\![\alpha]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct):
     Straightforward.
  2. (Value conjunct):
     Straightforward.

- Case (2): $\frac{\alpha \in \Delta}{\Delta \vdash \alpha}$

  Given: $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\alpha]\!]\delta$.

  To show: $(j, w, (SM^{\delta_1(\alpha)}\ \mathbf{e}_1), (SM^{\delta_2(\alpha)}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct):
     Straightforward.
  2. (Value conjunct):
     Straightforward.

- Case (1): $\frac{}{\Delta \vdash \mathbf{L}}$

  Given: $(j, w, \mathtt{e}_1, \mathtt{e}_2) \in \mathcal{E}_S$.

  To show: $(j, w, ({}^{\delta_1(\mathbf{L})}MS\ \mathtt{e}_1), ({}^{\delta_2(\mathbf{L})}MS\ \mathtt{e}_2)) \in \mathcal{E}_M[\![\mathbf{L}]\!]\delta$.

  Since $\delta_1(\mathbf{L}) \equiv \delta_2(\mathbf{L}) \equiv \mathbf{L}$, it's equivalent to show that

  $(j, w, ({}^{\mathbf{L}}MS\ \mathtt{e}_1), ({}^{\mathbf{L}}MS\ \mathtt{e}_2)) \in \mathcal{E}_M[\![\mathbf{L}]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct):
     Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright ({}^{\mathbf{L}}MS\ \mathtt{e}_1) \longmapsto^i$ **Error**: str.

     To show: $w.K_2 \triangleright ({}^{\mathbf{L}}MS\ \mathtt{e}_2) \longmapsto^*$ **Error**: str.

     From the operational semantics, for $w.K_1 \triangleright ({}^{\mathbf{L}}MS\ \mathtt{e}_1)$ to have evaluated to **Error**: str, there must have been an error during the evaluation of $\mathtt{e}_1$. (If $\mathtt{e}_1$ evaluated to a value $\mathtt{v}_1$ without error, then the resulting $({}^{\mathbf{L}}MS\ \mathtt{v}_1)$ would also be a value and hence unable to step to **Error**: str.) Therefore it must be the case that

     $w.K_1 \triangleright \mathtt{e}_1 \longmapsto^i$ **Error**: str.

     Instantiate the first conjunct of the premise $(j, w, \mathtt{e}_1, \mathtt{e}_2) \in \mathcal{E}_S$ with i, str. Since $i < j$, we have that $w.K_2 \triangleright \mathtt{e}_2 \longmapsto^*$ **Error**: str.

     It follows that $w.K_2 \triangleright ({}^{\mathbf{L}}MS\ \mathtt{e}_2) \longmapsto^*$ **Error**: str, as we were required to show.

2. (Value conjunct):

   Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:

   - $i < j$, and
   - $w.K_1 \triangleright ({}^{\mathbf{L}}MS\ \mathsf{e}_1) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

   To show: There exist $K_2', \mathbf{v}_2, w'$ such that:

   - $w.K_2 \triangleright ({}^{\mathbf{L}}MS\ \mathsf{e}_2) \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
   - $(j - i, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_1'$,
   - $w'.K_2 = K_2'$, and
   - $(j - i, w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{L}]\!]\delta$.

   Since $w.K_1 \triangleright ({}^{\mathbf{L}}MS\ \mathsf{e}_1) \longmapsto^i K_1' \triangleright \mathbf{v}_1$, from the operational semantics it follows that there exists v_1 such that

   $$w.K_1 \triangleright ({}^{\mathbf{L}}MS\ \mathsf{e}_1) \quad \longmapsto^i \quad K_1' \triangleright ({}^{\mathbf{L}}MS\ \mathsf{v}_1).$$

   Therefore, it follows that
   $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^i K_1' \triangleright \mathsf{v}_1$.
   Also note that by the operational semantics, $\mathbf{v}_1$ must be $({}^{\mathbf{L}}MS\ \mathsf{v}_1)$, since $({}^{\mathbf{L}}MS\ \mathsf{v}_1)$ is a value.
   Instantiate the second conjunct of $(j, w, \mathsf{e}_1, \mathsf{e}_2) \in \mathcal{E}_S$ with i, K_1', v_1. Note that:

   - $i < j$, and
   - $w.K_1 \triangleright \mathsf{e}_1 \longmapsto^i K_1' \triangleright \mathsf{v}_1$ (from above).

   Therefore, there exist K_2'', v_2, w'' such that:

   - $w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* K_2'' \triangleright \mathsf{v}_2$,
   - $(j - i, w'') \sqsupseteq (j, w)$,
   - $w''.K_1 = K_1'$,
   - $w''.K_2 = K_2''$, and
   - $(j - i, w'', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.

   Choose:

   - $K_2' = K_2''$,
   - $\mathbf{v}_2 = ({}^{\mathbf{L}}MS\ \mathsf{v}_2)$,
   - $w' = w''$.

   To show:

   - $w.K_2 \triangleright ({}^{\mathbf{L}}MS\ \mathsf{e}_2) \longmapsto^* K_2'' \triangleright ({}^{\mathbf{L}}MS\ \mathsf{v}_2)$ (which follows from above, since we have that $w.K_2 \triangleright \mathsf{e}_2 \longmapsto^* K_2'' \triangleright \mathsf{v}_2$);
   - $(j - i, w'') \sqsupseteq (j, w)$ (immediate from above);
   - $w''.K_1 = K_1'$ (immediate from above);
   - $w''.K_2 = K_2''$ (immediate from above);
   - $(j - i, w'', ({}^{\mathbf{L}}MS\ \mathsf{v}_1), ({}^{\mathbf{L}}MS\ \mathsf{v}_2)) \in \mathcal{V}_M[\![\mathbf{L}]\!]\delta$, which we show as follows:
     Since $(j - i, w'', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$ (from above), we have from Lemma 1.2 that $(j - i - 1, \lfloor w'' \rfloor_{j-i-1}, \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.
     Therefore, by definition of $\blacktriangleright$,
     $(j - i, w'', \mathsf{v}_1, \mathsf{v}_2) \in \blacktriangleright \mathcal{V}_S$.
     So, by definition of $\mathcal{V}_M[\![\mathbf{L}]\!]$,
     $(j - i, w'', ({}^{\mathbf{L}}MS\ \mathsf{v}_1), ({}^{\mathbf{L}}MS\ \mathsf{v}_2)) \in \mathcal{V}_M[\![\mathbf{L}]\!]\delta$.

- Case (2): $\frac{}{\Delta \vdash \mathbf{L}}$

  Given: $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\mathbf{L}]\!]\delta$.

  To show: $(j, w, (SM^{\delta_1(\mathbf{L})}\ \mathbf{e}_1), (SM^{\delta_2(\mathbf{L})}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  Since $\delta_1(\mathbf{L}) \equiv \delta_2(\mathbf{L}) \equiv \mathbf{L}$, it's equivalent to show that

  $(j, w, (SM^{\mathbf{L}}\ \mathbf{e}_1), (SM^{\mathbf{L}}\ \mathbf{e}_2)) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_1) \longmapsto^i$ **Error**: str.

     To show: $w.K_2 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_2) \longmapsto^*$ **Error**: str.

     From the operational semantics, there are two ways that $w.K_1 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_1)$ might have evaluated to **Error**: str: (a) during the evaluation of $\mathbf{e}_1$, or (b) after $\mathbf{e}_1$ has evaluated to some value $\mathbf{v}_1$. We consider both these possibilities:

     (a) $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^i$ **Error**: str.
     Instantiate the first conjunct of the premise $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\mathbf{L}]\!]\delta$ with i, str. Since $i < j$, we have that $w.K_2 \triangleright \mathbf{e}_2 \longmapsto^*$ **Error**: str.
     It follows that $w.K_2 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_2) \longmapsto^*$ **Error**: str, as we were required to show.

     (b) $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^{i_1} K_1' \triangleright \mathbf{v}_1$ and $K_1' \triangleright (SM^{\mathbf{L}}\ \mathbf{v}_1) \longmapsto^{i_2}$ **Error**: str and $i = i_1 + i_2$.
     Instantiate the second conjunct of the premise $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\mathbf{L}]\!]\delta$ with $i_1, K_1', \mathbf{v}_1$. Note that:
     - $i_1 < j$, and
     - $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^{i_1} K_1' \triangleright \mathbf{v}_1$.

     Therefore, there exist $K_2', \mathbf{v}_2, w'$ such that:
     - $w.K_2 \triangleright \mathbf{e}_2 \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
     - $(j - i_1, w') \sqsupseteq (j, w)$,
     - $w.K_1 = K_1'$,
     - $w.K_2 = K_2'$, and
     - $(j - i_1, w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{L}]\!]\delta$.

     Therefore, by definition of $\mathcal{V}_M[\![\mathbf{L}]\!]$, $\mathbf{v}_1$ and $\mathbf{v}_2$ must be $({}^{\mathbf{L}}MS\ \mathsf{v}_1)$ and $({}^{\mathbf{L}}MS\ \mathsf{v}_2)$ for some Scheme values v_1 and v_2 respectively. But then, by the operational semantics, we have that

     $$K_1' \triangleright (SM^{\mathbf{L}}\ \mathbf{v}_1) \equiv K_1' \triangleright (SM^{\mathbf{L}}\ ({}^{\mathbf{L}}MS\ \mathsf{v}_1)) \longmapsto^1 K_1' \triangleright \mathsf{v}_1,$$

     contradicting our earlier assumption that $K_1' \triangleright (SM^{\mathbf{L}}\ \mathbf{v}_1) \longmapsto^{i_2}$ **Error**: str. Therefore this case cannot occur.

  2. (Value conjunct):

     Consider arbitrary i, K_1', v_1 such that:
     - $i < j$, and
     - $w.K_1 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_1) \longmapsto^i K_1' \triangleright \mathsf{v}_1$.

     To show: There exist K_2', v_2, w' such that:
     - $w.K_2 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_2) \longmapsto^* K_2' \triangleright \mathsf{v}_2$,

- $(j-i, w') \sqsupseteq (j, w)$,
- $w'.K_1 = K_1'$,
- $w'.K_2 = K_2'$, and
- $(j-i, w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.

Since $w.K_1 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_1) \longmapsto^i K_1' \triangleright \mathsf{v}_1$, from the operational semantics it follows that there exist i_1 and $\mathbf{v}_1$ such that

$$\begin{array}{lll} w.K_1 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_1) & \longmapsto^{i_1} & K_1' \triangleright (SM^{\mathbf{L}}\ \mathbf{v}_1) \\ & \equiv & K_1' \triangleright (SM^{\mathbf{L}}\ ({}^{\mathbf{L}}MS\ \mathsf{v}_1)) \\ & \longmapsto^{1} & K_1' \triangleright \mathsf{v}_1. \end{array}$$

(Note that the key store is already K_1' after i_1 steps, since no keys can be generated during the one-step transition from $(SM^{\mathbf{L}}\ ({}^{\mathbf{L}}MS\ \mathsf{v}_1))$ to v_1.)

Therefore, it follows that

- $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^{i_1} K_1' \triangleright \mathbf{v}_1$,
- $\mathbf{v}_1 = ({}^{\mathbf{L}}MS\ \mathsf{v}_1)$, and
- $i = i_1 + 1$.

Instantiate the second conjunct of $(j, w, \mathbf{e}_1, \mathbf{e}_2) \in \mathcal{E}_M[\![\mathbf{L}]\!]\delta$ with $i_1, K_1', \mathbf{v}_1$. Note that:

- $i_1 < j$ (since $i_1 = i - 1$ and $i < j$), and
- $w.K_1 \triangleright \mathbf{e}_1 \longmapsto^{i_1} K_1' \triangleright \mathbf{v}_1$ (from above).

Therefore, there exist $K_2'', \mathbf{v}_2, w''$ such that:

- $w.K_2 \triangleright \mathbf{e}_2 \longmapsto^* K_2'' \triangleright \mathbf{v}_2$,
- $(j-i_1, w'') \sqsupseteq (j, w)$,
- $w''.K_1 = K_1'$,
- $w''.K_2 = K_2''$, and
- $(j-i_1, w'', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{L}]\!]\delta$.

Since $(j-i_1, w'', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\mathbf{L}]\!]\delta$ and $\mathbf{v}_1 = ({}^{\mathbf{L}}MS\ \mathsf{v}_1)$, it follows from the definition of $\mathcal{V}_M[\![\mathbf{L}]\!]$ that there exists v'_2 such that:

- $\mathbf{v}_2 = ({}^{\mathbf{L}}MS\ \mathsf{v}'_2)$, and
- $(j-i_1, w'', \mathsf{v}_1, \mathsf{v}'_2) \in \blacktriangleright \mathcal{V}_S$.

Choose:

- $K_2' = K_2''$,
- $\mathsf{v}_2 = \mathsf{v}'_2$,
- $w' = \lfloor w'' \rfloor_{j-i}$.

To show:

- $w.K_2 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_2) \longmapsto^* K_2'' \triangleright \mathsf{v}'_2$, which follows from above, since we have that

$$\begin{array}{lll} w.K_2 \triangleright (SM^{\mathbf{L}}\ \mathbf{e}_2) & \longmapsto^{*} & K_2'' \triangleright (SM^{\mathbf{L}}\ \mathbf{v}_2) \\ & \equiv & K_2'' \triangleright (SM^{\mathbf{L}}\ ({}^{\mathbf{L}}MS\ \mathsf{v}'_2)) \\ & \longmapsto^{1} & K_2'' \triangleright \mathsf{v}'_2; \end{array}$$

- $(j-i, \lfloor w'' \rfloor_{j-i}) \sqsupseteq (j, w)$, which follows from Lemma 1.1, since:
  - $(j-i_1, w'') \sqsupseteq (j, w)$ (from above), and

$*$ $(j-i, \lfloor w'' \rfloor_{j-i}) \sqsupseteq (j-i_1, w'')$ (because $(j-i) \leq (j-i_1)$, since $i_1 < i$);

- $\lfloor w'' \rfloor_{j-i}.K_1 = K_1'$, which follows from Lemma 1.7 and from above;
- $\lfloor w'' \rfloor_{j-i}.K_2 = K_2''$, which follows from Lemma 1.7 and from above;
- $(j-i, \lfloor w'' \rfloor_{j-i}, \mathsf{v}_1, \mathsf{v}'_2) \in \mathcal{V}_S$, which we show as follows:
  Since $(j-i_1, w'', \mathsf{v}_1, \mathsf{v}'_2) \in \blacktriangleright \mathcal{V}_S$, we have by definition of $\blacktriangleright$ that
  $(j-i_1-1, \lfloor w'' \rfloor_{j-i_1-1}, \mathsf{v}_1, \mathsf{v}'_2) \in \mathcal{V}_S$.
  Therefore, since $j-i_1-1 = j-(i_1+1) = j-i$,
  $(j-i, \lfloor w'' \rfloor_{j-i}, \mathsf{v}_1, \mathsf{v}'_2) \in \mathcal{V}_S$, as we were required to show.

□

4 Parametricity / fundamental property

Theorem 4.1 (parametricity / fundamental property)**.** *For all key-free terms* $\mathbf{e}$ *and* e*:*

1. *If* $\Delta; \Gamma \vdash_M \mathbf{e} : \tau$*, then* $\Delta; \Gamma \vdash_M \mathbf{e} \lesssim_M \mathbf{e} : \tau$.
2. *If* $\Delta; \Gamma \vdash_S \mathsf{e} :$ ***TST****, then* $\Delta; \Gamma \vdash_S \mathsf{e} \lesssim_S \mathsf{e} :$ ***TST***.

Proof. By simultaneous induction on the derivations $\Delta; \Gamma \vdash_M \mathbf{e} : \tau$ and $\Delta; \Gamma \vdash_S \mathsf{e} :$ **TST**.

- Case (MVar): $\dfrac{}{\Delta; \Gamma, \mathbf{x} : \tau \vdash_M \mathbf{x} : \tau}$

  To show: $\Delta; \Gamma, \mathbf{x} : \tau \vdash_M \mathbf{x} \lesssim_M \mathbf{x} : \tau$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(\mathbf{x})), \delta_2(\gamma_2(\mathbf{x}))) \in \mathcal{E}_M[\![\tau]\!]\delta$, or, equivalently (since there are no type variables in $\mathbf{x}$), that $(j, w, \gamma_1(\mathbf{x}), \gamma_2(\mathbf{x})) \in \mathcal{E}_M[\![\tau]\!]\delta$

  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright \gamma_1(\mathbf{x}) \longmapsto^i$ **Error**: str.

     But $\gamma_1(\mathbf{x})$ is a value (not **Error**: str), so this case cannot occur.
  2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
     - $i < j$, and
     - $w.K_1 \triangleright \gamma_1(\mathbf{x}) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

     Since $\gamma_1(\mathbf{x})$ is a value, it must be the case that $i = 0$ and $\mathbf{v}_1 = \gamma_1(\mathbf{x})$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

     Choose:

- $\mathbf{v}_2 = \gamma_2(\mathbf{x})$,
- $K_2' = w.K_2$, and
- $w' = w$.

To show:

- $w.K_2 \triangleright \gamma_2(\mathbf{x}) \longmapsto^* w.K_2 \triangleright \gamma_2(\mathbf{x})$,
- $(j, w) \sqsupseteq (j, w)$,
- $w.K_1 = w.K_1$,
- $w.K_2 = w.K_2$, and
- $(j, w, \gamma_1(\mathbf{x}), \gamma_2(\mathbf{x})) \in \mathcal{V}_M[\![\tau]\!]\delta$.

The first four of these conjuncts are immediate, and the last follows from the premise that $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

- Case (MAbs): $\dfrac{\Delta; \Gamma, \mathbf{x} : \tau_1 \vdash_M \mathbf{e} : \tau_2}{\Delta; \Gamma \vdash_M \lambda \mathbf{x} : \tau_1.\ \mathbf{e} : \tau_1 \to \tau_2}$

To show: $\Delta; \Gamma \vdash_M \lambda \mathbf{x} : \tau_1.\ \mathbf{e} \lesssim_M \lambda \mathbf{x} : \tau_1.\ \mathbf{e} : \tau_1 \to \tau_2$.

Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

To show: $(j, w, \delta_1(\gamma_1(\lambda \mathbf{x} : \tau_1.\ \mathbf{e})), \delta_2(\gamma_2(\lambda \mathbf{x} : \tau_1.\ \mathbf{e}))) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$, or, equivalently, that $(j, w, \lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e})), \lambda \mathbf{x} : \delta_2(\tau_1).\ \delta_2(\gamma_2(\mathbf{e}))) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$.

The proof is in two parts:

1. (Error conjunct): Consider arbitrary i, str such that:
   - $i < j$, and
   - $w.K_1 \triangleright \lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e})) \longmapsto^i$ **Error**: str.

   But $\lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e}))$ is a value (not **Error**: str), so this case cannot occur.
2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
   - $i < j$, and
   - $w.K_1 \triangleright \lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e})) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

   Since $\lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e}))$ is a value, it must be the case that $i = 0$ and $\mathbf{v}_1 = \lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e}))$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

   Choose:
   - $\mathbf{v}_2 = \lambda \mathbf{x} : \delta_2(\tau_1).\ \delta_2(\gamma_2(\mathbf{e}))$,
   - $K_2' = w.K_2$, and
   - $w' = w$.

   To show:
   - $w.K_2 \triangleright \lambda \mathbf{x} : \delta_2(\tau_1).\ \delta_2(\gamma_2(\mathbf{e})) \longmapsto^* w.K_2 \triangleright \lambda \mathbf{x} : \delta_2(\tau_1).\ \delta_2(\gamma_2(\mathbf{e}))$,
   - $(j, w) \sqsupseteq (j, w)$,

- $w.K_1 = w.K_1$,
- $w.K_2 = w.K_2$, and
- $(j, w, \lambda \mathbf{x} : \delta_1(\tau_1).\ \delta_1(\gamma_1(\mathbf{e})), \lambda \mathbf{x} : \delta_2(\tau_1).\ \delta_2(\gamma_2(\mathbf{e}))) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$.

The first four conjuncts are immediate, and the last we show as follows:
Consider arbitrary $(j', w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1]\!]\delta$ such that $(j', w') \sqsupseteq (j, w)$.
To show: $(j', w', \delta_1(\gamma_1(\mathbf{e}))[\mathbf{x} := \mathbf{v}_1], \delta_2(\gamma_2(\mathbf{e}))[\mathbf{x} := \mathbf{v}_2]) \in \mathcal{E}_M[\![\tau_2]\!]\delta$.
Applying the induction hypothesis to $\Delta; \Gamma, \mathbf{x} : \tau_1 \vdash_M \mathbf{e} : \tau_2$, we have that $\Delta; \Gamma, \mathbf{x} : \tau_1 \vdash_M \mathbf{e} \lesssim_M \mathbf{e} : \tau_2$,
which we instantiate with $n, j', \delta, \gamma_1[\mathbf{x} := \mathbf{v}_1], \gamma_2[\mathbf{x} := \mathbf{v}_2], w'$. Note that:

- $n \geq 0$,
- $j' < n$ (since $j' \leq j$ and $j < n$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j', w', \gamma_1[\mathbf{x} := \mathbf{v}_1], \gamma_2[\mathbf{x} := \mathbf{v}_2]) \in \mathcal{G}[\![\Gamma, \mathbf{x} : \tau_1]\!]\delta$, which follows from:
  - $(j', w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$, and
  - $(j', w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_1]\!]\delta$, by downward closure of $\mathcal{V}_M[\![.]\!]$.

We therefore have that $(j', w', \delta_1(\gamma_1[\mathbf{x} := \mathbf{v}_1](\mathbf{e})), \delta_2(\gamma_2[\mathbf{x} := \mathbf{v}_2](\mathbf{e}))) \in \mathcal{E}_M[\![\tau_2]\!]\delta$, which is equivalent to
$(j', w', \delta_1(\gamma_1(\mathbf{e}))[\mathbf{x} := \mathbf{v}_1], \delta_2(\gamma_2(\mathbf{e}))[\mathbf{x} := \mathbf{v}_2]) \in \mathcal{E}_M[\![\tau_2]\!]\delta$, as we were required to show.

- Case (MApp): $\dfrac{\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau_1 \to \tau_2 \qquad \Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau_1}{\Delta; \Gamma \vdash_M (\mathbf{e}_1\ \mathbf{e}_2) : \tau_2}$

To show: $\Delta; \Gamma \vdash_M (\mathbf{e}_1\ \mathbf{e}_2) \lesssim_M (\mathbf{e}_1\ \mathbf{e}_2) : \tau_2$.

Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

To show: $(j, w, \delta_1(\gamma_1(\mathbf{e}_1\ \mathbf{e}_2)), \delta_2(\gamma_2(\mathbf{e}_1\ \mathbf{e}_2))) \in \mathcal{E}_M[\![\tau_2]\!]\delta$, or, equivalently,
$(j, w, (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))), (\delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2)))) \in \mathcal{E}_M[\![\tau_2]\!]\delta$.
The proof is in two parts:

1. (Error conjunct): Consider arbitrary i, str such that:
   - $i < j$, and
   - $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) \longmapsto^i$ **Error**: str.

   To show: $w.K_2 \triangleright (\delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^*$ **Error**: str.
   By the operational semantics, there are three cases to consider, corresponding to the three times during the evaluation of $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2)))$ during which an error could be raised: during the evaluation of the function, during the evaluation of the argument, or during the evaluation of the body of the function.

   (a) $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))) \longmapsto^i$ **Error**: str.
   Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau_1 \to \tau_2$, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau_1 \to \tau_2$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$. Instantiate the first conjunct of this with i, str. Note that:

- $i < j$, and
- $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))) \longmapsto^i$ **Error**: str.

Therefore $w.K_2 \triangleright (\delta_2(\gamma_2(\mathbf{e}_1))) \longmapsto^*$ **Error**: str.
It follows that $w.K_2 \triangleright (\delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^*$ **Error**: str, as we were required to show.

(b) $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$ and $K_{11} \triangleright (\delta_1(\gamma_1(\mathbf{e}_2))) \longmapsto^{i_2}$ **Error**: str and $i = i_1 + i_2$.
Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau_1 \to \tau_2$ as in the previous case, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau_1 \to \tau_2$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$.
Instantiate the second conjunct of this with $i_1, K_{11}, \mathbf{v}_{11}$. Note that:

- $i_1 < j$ (which follows from $i = i_1 + i_2$ and $i < j$), and
- $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.

Therefore there exist $K_{21}, \mathbf{v}_{21}, w'$ such that:

- $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1)) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}$,
- $(j - i_1, w') \sqsupseteq (j, w)$,
- $w'.K_1 = K_{11}$,
- $w'.K_2 = K_{21}$, and
- $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$.

Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau_1$, we have that
$\Delta; \Gamma \vdash_M \mathbf{e}_2 \lesssim_M \mathbf{e}_2 : \tau_1$,
which we instantiate with $n, j - i_1, \delta, \gamma_1, \gamma_2, w'$. Note that:

- $n \geq 0$,
- $j - i_1 < n$ (since $j < n$ and $i_1 < j$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j - i_1, w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$.

Therefore $(j - i_1, w', \delta_1(\gamma_1(\mathbf{e}_2)), \delta_2(\gamma_2(\mathbf{e}_2))) \in \mathcal{E}_M[\![\tau_1]\!]\delta$. Instantiate the first conjunct of this with i_2, str. Note that:

- $i_2 < j$ (which follows from $i = i_1 + i_2$ and $i < j$), and
- $w'.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2}$ **Error**: str (from premises, since $w'.K_1 = K_{11}$).

Therefore $w'.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^*$ **Error**: str.
Since $w'.K_2 = K_{21}$, by the operational semantics we have that

$$w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}\ \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^* \textbf{Error}: \text{str},$$

as we were required to show.

(c) $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$, and
$K_{11} \triangleright (\delta_1(\gamma_1(\mathbf{e}_2))) \longmapsto^{i_2} K_{12} \triangleright \mathbf{v}_{12}$, and
$K_{12} \triangleright (\mathbf{v}_{11}\ \mathbf{v}_{12}) \longmapsto^{i_3} \textbf{Error}$: str, and
$i = i_1 + i_2 + i_3$.
Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau_1 \to \tau_2$ as in the previous case, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau_1 \to \tau_2$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$.
Instantiate the second conjunct of this with $i_1, K_{11}, \mathbf{v}_{11}$. Note that:

- $i_1 < j$ (which follows from $i = i_1 + i_2 + i_3$ and $i < j$), and
- $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.

Therefore there exist $K_{21}, \mathbf{v}_{21}, w'$ such that:

- $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1)) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}$,
- $(j - i_1, w') \sqsupseteq (j, w)$,
- $w'.K_1 = K_{11}$,
- $w'.K_2 = K_{21}$, and
- $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$.

Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau_1$, we have that
$\Delta; \Gamma \vdash_M \mathbf{e}_2 \lesssim_M \mathbf{e}_2 : \tau_1$,
which we instantiate with $n, j - i_1, \delta, \gamma_1, \gamma_2, w'$. Note that:

- $n \geq 0$,
- $j - i_1 < n$ (since $j < n$ and $i_1 < j$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j - i_1, w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$.

Therefore $(j - i_1, w', \delta_1(\gamma_1(\mathbf{e}_2)), \delta_2(\gamma_2(\mathbf{e}_2))) \in \mathcal{E}_M[\![\tau_1]\!]\delta$.
Instantiate the second conjunct of this with $i_2, K_{12}, \mathbf{v}_{12}$. Note that:

- $i_2 < j$ (which follows from $i = i_1 + i_2 + i_3$ and $i < j$), and
- $w'.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathbf{v}_{12}$ (from above, since $w'.K_1 = K_{11}$).

Therefore there exist $K_{22}, \mathbf{v}_{22}, w''$ such that:

- $w'.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^* K_{22} \triangleright \mathbf{v}_{22}$,
- $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$,
- $w''.K_1 = K_{12}$,
- $w''.K_2 = K_{22}$, and
- $(j - i_1 - i_2, w'', \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau_1]\!]\delta$.

By the operational semantics, we have that

$$\begin{array}{lll} w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathbf{v}_{11}\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \longmapsto^{i_2} & K_{12} \triangleright (\mathbf{v}_{11}\ \mathbf{v}_{12}) \\ & \longmapsto^{i_3} & \textbf{Error}\text{: str} \end{array}$$

where $i = i_1 + i_2 + i_3$.
Recall that $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$ and instantiate with $j - i_1 - i_2, w'', \mathbf{v}_{12}, \mathbf{v}_{22}$.
Note that:

- $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$ (from above, by downward closure of world extension under decreasing step index), and
- $(j - i_1 - i_2, w'', \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau_1]\!]\delta$ (from above, by downward closure of $\mathcal{V}_M[\![.]\!]$).

Further note that it must be the case that $\mathbf{v}_{11}$ and $\mathbf{v}_{21}$ are functions:
$\mathbf{v}_{11} \equiv \lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_{11}$, and
$\mathbf{v}_{21} \equiv \lambda \mathbf{x} : \delta_2(\tau_1).\ \mathbf{e}_{21}$
for some $\mathbf{e}_{11}$ and $\mathbf{e}_{21}$.
Therefore $(j - i_1 - i_2, w'', \mathbf{e}_{11}[\mathbf{x} := \mathbf{v}_{12}], \mathbf{e}_{21}[\mathbf{x} := \mathbf{v}_{22}]) \in \mathcal{E}_M[\![\tau_2]\!]\delta$. Instantiate the first conjunct with $i_3 - 1$ and str. Note that:

- $i_3 - 1 < j - i_1 - i_2$ (since $i = i_1 + i_2 + i_3$, so $i_3 = i - i_1 - i_2$, and $i < j$, so we have $i_3 < j - i_1 - i_2$ and therefore clearly $i_3 - 1 < j - i_1 - i_2$), and
- $w''.K_1 \triangleright \mathbf{e}_{11}[\mathbf{x} := \mathbf{v}_{12}] \longmapsto^{i_3 - 1}$ **Error**: str, which follows from above, since:
  - $w''.K_1 = K_{12}$, and
  - $K_{12} \triangleright (\mathbf{v}_{11}\ \mathbf{v}_{12}) \longmapsto^1 K_{12} \triangleright \mathbf{e}_{11}[\mathbf{x} := \mathbf{v}_{12}]$, by operational semantics, and
  - $K_{12} \triangleright (\mathbf{v}_{11}\ \mathbf{v}_{12}) \longmapsto^{i_3}$ **Error**: str, from above.

We therefore have that $w''.K_2 \triangleright \mathbf{e}_{21}[\mathbf{x} := \mathbf{v}_{22}] \longmapsto^*$ **Error**: str.
Since $w''.K_2 = K_{22}$, by the operational semantics we have that

$$\begin{array}{lll} w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2)) & \longmapsto^* & K_{21} \triangleright \mathbf{v}_{21}\ \delta_2(\gamma_2(\mathbf{e}_2)) \\ & \longmapsto^* & K_{22} \triangleright \mathbf{v}_{21}\ \mathbf{v}_{22} \\ & \longmapsto^1 & K_{22} \triangleright \mathbf{e}_{21}[\mathbf{x} := \mathbf{v}_{22}] \\ & \longmapsto^* & \textbf{Error}\text{: str}, \end{array}$$

as we were required to show.

2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
   - $i < j$, and
   - $w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

   Note that by the operational semantics there exist $i_1 \leq i, K_{11}, \mathbf{v}_{11}$ such that $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.
   To show: There exist $K_2', \mathbf{v}_2, w'$ such that:
   - $w.K_2 \triangleright (\delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
   - $(j - i, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_1'$,
   - $w'.K_2 = K_2'$, and
   - $(j - i, w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_2]\!]\delta$.

   Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau_1 \to \tau_2$, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau_1 \to \tau_2$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau_1 \to \tau_2]\!]\delta$. Instantiate the second conjunct with $i_1, K_{11}, \mathbf{v}_{11}$. Note that:

- $i_1 < j$ (which follows from $i_1 \leq i$ and $i < j$), and
- $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.

Therefore there exist $K_{21}, \mathbf{v}_{21}, w'$ such that:

- $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1)) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}$,
- $(j - i_1, w') \sqsupseteq (j, w)$,
- $w'.K_1 = K_{11}$,
- $w'.K_2 = K_{21}$, and
- $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$.

Therefore it must be the case that
$\mathbf{v}_{11} \equiv \lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_{11}$, and
$\mathbf{v}_{21} \equiv \lambda \mathbf{x} : \delta_2(\tau_1).\ \mathbf{e}_{21}$.
Note that

$$\begin{array}{lll} w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathbf{v}_{11}\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \equiv & K_{11} \triangleright ((\lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_{11})\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \longmapsto^{i - i_1} & K_1' \triangleright \mathbf{v}_1. \end{array}$$

Therefore, by the operational semantics there exist i_2, K_{12}, and $\mathbf{v}_{12}$ such that:

- $K_{11} \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathbf{v}_{12}$, and
- $i_2 \leq i - i_1$.

Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau_1$, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_2 \lesssim_M \mathbf{e}_2 : \tau_1$, which we instantiate with $n, j - i_1, \delta, \gamma_1, \gamma_2, w'$. Note that:

- $n \geq 0$,
- $j - i_1 < n$ (since $j < n$ and $i_1 < j$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j - i_1, w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$.

Therefore $(j - i_1, w', \delta_1(\gamma_1(\mathbf{e}_2)), \delta_2(\gamma_2(\mathbf{e}_2))) \in \mathcal{E}_M[\![\tau_1]\!]\delta$. Instantiate the second conjunct with $i_2, K_{12}, \mathbf{v}_{12}$. Note that:

- $i_2 < j - i_1$ (since $i_2 \leq i - i_2$ and $i < j$), and
- $w'.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathbf{v}_{12}$, which follows from above since $w'.K_1 = K_{11}$.

Therefore there exist $K_{22}, \mathbf{v}_{22}, w''$ such that:

- $w'.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^* K_{22} \triangleright \mathbf{v}_{22}$,
- $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$,
- $w''.K_1 = K_{12}$,
- $w''.K_2 = K_{22}$, and
- $(j - i_1 - i_2, w'', \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau_1]\!]\delta$.

By the operational semantics, we have that

$$\begin{array}{lll} w.K_1 \triangleright (\delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathbf{v}_{11}\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \equiv & K_{11} \triangleright ((\lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_{11})\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \longmapsto^{i_2} & K_{12} \triangleright ((\lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_{11})\ \mathbf{v}_{12}) \\ & \longmapsto^{1} & K_{12} \triangleright \mathbf{e}_{11}[\mathbf{x} := \mathbf{v}_{12}] \\ & \longmapsto^{i_3} & K_1' \triangleright \mathbf{v}_1 \end{array}$$

where $i = i_1 + i_2 + 1 + i_3$. (Note that the key store cannot change from the third to the fourth configuration, since no new keys are generated during β-substitution.)

Recall that $(j - i_1, w', \lambda \mathbf{x} : \delta_1(\tau_1).\ \mathbf{e}_{11}, \lambda \mathbf{x} : \delta_2(\tau_1).\ \mathbf{e}_{21}) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$ and instantiate with $j - i_1 - i_2 - 1, w'', \mathbf{v}_{12}, \mathbf{v}_{22}$. Note that:

- $(j - i_1 - i_2 - 1, w'') \sqsupseteq (j - i_1, w')$ (from above, since $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$, by downward closure of world extension under decreasing step index), and
- $(j - i_1 - i_2 - 1, w'', \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau_1]\!]\delta$ (from above, since $(j - i_1 - i_2, w'', \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau_1]\!]\delta$, by downward closure of $\mathcal{V}_M[\![.]\!]$).

Therefore $(j - i_1 - i_2 - 1, w'', \mathbf{e}_{11}[\mathbf{x} := \mathbf{v}_{12}], \mathbf{e}_{21}[\mathbf{x} := \mathbf{v}_{22}]) \in \mathcal{E}_M[\![\tau_2]\!]\delta$. Instantiate the second conjunct with i_3, K_1', and $\mathbf{v}_1$.

Note that:

- $i_3 < j - i_1 - i_2 - 1$ (since $i = i_1 + i_2 + i_3$, so $i_3 = i - i_1 - i_2 - 1$, and $i < j$), and
- $w''.K_1 \triangleright \mathbf{e}_{11}[\mathbf{x} := \mathbf{v}_{12}] \longmapsto^{i_3} K_1' \triangleright \mathbf{v}_1$, which follows from above, since $w''.K_1 = K_{12}$.

Therefore, there exist $K_2', \mathbf{v}_2, w'''$ such that:

- $w''.K_2 \triangleright \mathbf{e}_{21}[\mathbf{x} := \mathbf{v}_{22}] \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
- $(j - i_1 - i_2 - 1 - i_3, w''') \sqsupseteq (j - i_1 - i_2 - 1, w'')$,
- $w'''.K_1 = K_1'$,
- $w'''.K_2 = K_2'$, and
- $(j - i_1 - i_2 - 1 - i_3, w''', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_2]\!]\delta$,

fulfilling our proof obligation.

- Case (MNil): $\dfrac{}{\Delta; \Gamma \vdash_M \mathbf{nil} : \tau^*}$

To show: $\Delta; \Gamma \vdash_M \mathbf{nil} \lesssim_M \mathbf{nil} : \tau^*$.

Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

To show: $(j, w, \delta_1(\gamma_1(\mathbf{nil})), \delta_2(\gamma_2(\mathbf{nil}))) \in \mathcal{E}_M[\![\tau^*]\!]\delta$, or, equivalently (since there are no variables in $\mathbf{nil}$), that $(j, w, \mathbf{nil}, \mathbf{nil}) \in \mathcal{E}_M[\![\tau]\!]\delta$.

The proof is in two parts:

1. (Error conjunct): Consider arbitrary i, str such that:
   - $i < j$, and

- $w.K_1 \triangleright \mathbf{nil} \longmapsto^i \mathbf{Error}$: str.

But $\mathbf{nil}$ is a value (not **Error**: str), so this case cannot occur.

2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
   - $i < j$, and
   - $w.K_1 \triangleright \mathbf{nil} \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

   Since $\mathbf{nil}$ is a value, it must be the case that $i = 0$ and $\mathbf{v}_1 = \mathbf{nil}$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

   Choose:
   - $\mathbf{v}_2 = \mathbf{nil}$,
   - $K_2' = w.K_2$, and
   - $w' = w$.

   To show:
   - $w.K_2 \triangleright \mathbf{nil} \longmapsto^* w.K_2 \triangleright \mathbf{nil}$,
   - $(j, w) \sqsupseteq (j, w)$,
   - $w.K_1 = w.K_1$,
   - $w.K_2 = w.K_2$, and
   - $(j, w, \mathbf{nil}, \mathbf{nil}) \in \mathcal{V}_M[\![\tau^*]\!]\delta$.

   The first four of these conjuncts are immediate, and the last follows immediately from the definition of $\mathcal{V}_M[\![\tau^*]\!]$.

- Case (MPair):

$$\frac{\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau \qquad \Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau^*}{\Delta; \Gamma \vdash_M (\mathbf{cons}\ \mathbf{e}_1\ \mathbf{e}_2) : \tau^*}$$

To show: $\Delta; \Gamma \vdash_M (\mathbf{cons}\ \mathbf{e}_1\ \mathbf{e}_2) \lesssim_M (\mathbf{cons}\ \mathbf{e}_1\ \mathbf{e}_2) : \tau^*$.

Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

To show: $(j, w, \delta_1(\gamma_1((\mathbf{cons}\ \mathbf{e}_1\ \mathbf{e}_2)), \delta_2(\gamma_2((\mathbf{cons}\ \mathbf{e}_1\ \mathbf{e}_2))))) \in \mathcal{E}_M[\![\tau^*]\!]\delta$, or, equivalently, that $(j, w, (\mathbf{cons}\ \delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))), (\mathbf{cons}\ \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))),) \in \mathcal{E}_M[\![\tau^*]\!]\delta$.

The proof is in two parts:

1. (Error conjunct): Consider arbitrary i, str such that:
   - $i < j$, and
   - $w.K_1 \triangleright (\mathbf{cons}\ \delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) \longmapsto^i \mathbf{Error}$: str.

   To show: $w.K_2 \triangleright (\mathbf{cons}\ \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^* \mathbf{Error}$: str.

   By the operational semantics, there are two cases to consider, corresponding to the two times during the evaluation of $w.K_1 \triangleright (\mathbf{cons}\ \delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2)))$ during which an error could be raised: during the evaluation of $\delta_1(\gamma_1(\mathbf{e}_1))$, or during the evaluation of $\delta_1(\gamma_1(\mathbf{e}_2))$.

(a) $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^i$ **Error**: str.
Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau$, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau]\!]\delta$. Instantiate the first conjunct of this with i, str. Note that:

- $i < j$, and
- $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^i$ **Error**: str.

Therefore $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1)) \longmapsto^*$ **Error**: str.
It follows that $w.K_2 \triangleright (\mathbf{cons}\ \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^*$ **Error**: str, as we were required to show.

(b) $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$ and $K_{11} \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2}$ **Error**: str and $i = i_1 + i_2$.
Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau$ as in the previous case, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:

- $n \geq 0$,
- $j < n$,
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau]\!]\delta$.
Instantiate the second conjunct of this with $i_1, K_{11}, \mathbf{v}_{11}$. Note that:

- $i_1 < j$ (which follows from $i = i_1 + i_2$ and $i < j$), and
- $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.

Therefore there exist $K_{21}, \mathbf{v}_{21}, w'$ such that:

- $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1)) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}$,
- $(j - i_1, w') \sqsupseteq (j, w)$,
- $w'.K_1 = K_{11}$,
- $w'.K_2 = K_{21}$, and
- $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau_1 \to \tau_2]\!]\delta$.

Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau^*$, we have that
$\Delta; \Gamma \vdash_M \mathbf{e}_2 \lesssim_M \mathbf{e}_2 : \tau^*$,
which we instantiate with $n, j - i_1, \delta, \gamma_1, \gamma_2, w'$. Note that:

- $n \geq 0$,
- $j - i_1 < n$ (since $j < n$ and $i_1 < j$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j - i_1, w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$.

Therefore $(j - i_1, w', \delta_1(\gamma_1(\mathbf{e}_2)), \delta_2(\gamma_2(\mathbf{e}_2))) \in \mathcal{E}_M[\![\tau^*]\!]\delta$. Instantiate the first conjunct of this with i_2, str. Note that:

- $i_2 < j$ (which follows from $i = i_1 + i_2$ and $i < j$), and
- $w'.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2}$ **Error**: str (from premises, since $w'.K_1 = K_{11}$).

Therefore $w'.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^*$ **Error**: str.
Since $w'.K_2 = K_{21}$, by the operational semantics we have that

$$\begin{array}{lll} w.K_2 \triangleright (\mathbf{cons}\ \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) & \longmapsto^* & K_{21} \triangleright (\mathbf{cons}\ \mathbf{v}_{21}\ \delta_2(\gamma_2(\mathbf{e}_2))) \\ & \longmapsto^* & \textbf{Error}: \text{str}, \end{array}$$

as we were required to show.

2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
   - $i < j$, and
   - $w.K_1 \triangleright (\mathbf{cons}\ \delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

   Note that by the operational semantics there exist $i_1 \leq i, K_{11}, \mathbf{v}_{11}$ such that $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.
   To show: There exist $K_2', \mathbf{v}_2, w'$ such that:
   - $w.K_2 \triangleright (\delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
   - $(j - i, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_1'$,
   - $w'.K_2 = K_2'$, and
   - $(j - i, w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau_2]\!]\delta$.

   Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_1 : \tau$, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_1 \lesssim_M \mathbf{e}_1 : \tau$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:
   - $n \geq 0$,
   - $j < n$,
   - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
   - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

   Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e}_1)), \delta_2(\gamma_2(\mathbf{e}_1))) \in \mathcal{E}_M[\![\tau]\!]\delta$. Instantiate the second conjunct with $i_1, K_{11}, \mathbf{v}_{11}$. Note that:
   - $i_1 < j$ (which follows from $i_1 \leq i$ and $i < j$), and
   - $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$ (from above).

   Therefore there exist $K_{21}, \mathbf{v}_{21}, w'$ such that:
   - $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_1)) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}$,
   - $(j - i_1, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_{11}$,
   - $w'.K_2 = K_{21}$, and
   - $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau]\!]\delta$.

   Note that

$$\begin{array}{lll} w.K_1 \triangleright (\mathbf{cons}\ \delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathbf{cons}\ \mathbf{v}_{11}\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \longmapsto^{i - i_1} & K_1' \triangleright \mathbf{v}_1. \end{array}$$

   Therefore, by the operational semantics there exist i_2, K_{12}, and $\mathbf{v}_{12}$ such that:
   - $K_{11} \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathbf{v}_{12}$, and
   - $i_2 \leq i - i_1$.

   Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e}_2 : \tau^*$, we have that $\Delta; \Gamma \vdash_M \mathbf{e}_2 \lesssim_M \mathbf{e}_2 : \tau^*$, which we instantiate with $n, j - i_1, \delta, \gamma_1, \gamma_2, w'$. Note that:

- $n \geq 0$,
- $j - i_1 < n$ (since $j < n$ and $i_1 < j$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j - i_1, w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$.

Therefore $(j - i_1, w', \delta_1(\gamma_1(\mathbf{e}_2)), \delta_2(\gamma_2(\mathbf{e}_2))) \in \mathcal{E}_M[\![\tau^*]\!]\delta$. Instantiate the second conjunct with $i_2, K_{12}, \mathbf{v}_{12}$. Note that:

- $i_2 < j - i_1$ (since $i_2 \leq i - i_1$, and $i_1 \leq i$, and $i < j$), and
- $w'.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathbf{v}_{12}$, which follows from above since $w'.K_1 = K_{11}$.

Therefore there exist $K_{22}, \mathbf{v}_{22}, w''$ such that:

- $w'.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}_2)) \longmapsto^* K_{22} \triangleright \mathbf{v}_{22}$,
- $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$,
- $w''.K_1 = K_{12}$,
- $w''.K_2 = K_{22}$, and
- $(j - i_1 - i_2, w'', \mathbf{v}_{12}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau^*]\!]\delta$.

By the operational semantics, we have that

$$\begin{array}{lll} w.K_1 \triangleright (\mathbf{cons}\ \delta_1(\gamma_1(\mathbf{e}_1))\ \delta_1(\gamma_1(\mathbf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathbf{cons}\ \mathbf{v}_{11}\ \delta_1(\gamma_1(\mathbf{e}_2))) \\ & \longmapsto^{i_2} & K_{12} \triangleright (\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12}) \\ & \equiv & K_1' \triangleright \mathbf{v}_1, \end{array}$$

since $(\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12})$ is a value. Therefore $i = i_1 + i_2$.

From above, and from the operational semantics, we have that

$$\begin{array}{lll} w.K_2 \triangleright (\mathbf{cons}\ \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) & \longmapsto^* & K_{21} \triangleright (\mathbf{cons}\ \mathbf{v}_{21}\ \delta_2(\gamma_2(\mathbf{e}_2))) \\ & \longmapsto^* & K_{22} \triangleright (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22}). \end{array}$$

since $w'.K_2 = K_{21}$.

Choose:

- $\mathbf{v}_2 = (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})$,
- $K_2' = w''.K_2$, and
- $w' = w''$.

To show:

- $w.K_2 \triangleright (\mathbf{cons}\ \delta_2(\gamma_2(\mathbf{e}_1))\ \delta_2(\gamma_2(\mathbf{e}_2))) \longmapsto^* w''.K_2 \triangleright (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})$, which follows from above since $w''.K_2 = K_{22}$,
- $(j - i, w'') \sqsupseteq (j, w)$, which follows from above and from Lemma 1.1 since $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$ and $(j - i_1, w') \sqsupseteq (j, w)$,
- $w''.K_1 = K_1'$, which follows from above, since $w''.K_1 = K_{12} = K_1'$,
- $w''.K_2 = w''.K_2$ (immediate), and
- $(j - i, w'', (\mathbf{cons}\ \mathbf{v}_{11}\ \mathbf{v}_{12}), (\mathbf{cons}\ \mathbf{v}_{21}\ \mathbf{v}_{22})) \in \mathcal{V}_M[\![\tau^*]\!]\delta$, which follows from:
  - $(j - i, w'', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau]\!]\delta$ (since $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\tau]\!]\delta$, and $\mathcal{V}_M[\![.]\!]$ is closed under world extension and decreasing step-index), and
  - $(j - i, w'', \mathbf{v}_{21}, \mathbf{v}_{22}) \in \mathcal{V}_M[\![\tau^*]\!]\delta$ (immediate from above, since $i = i_1 + i_2$).

- Case (MFst): $\dfrac{\Delta; \Gamma \vdash_M \mathbf{e} : \tau^*}{\Delta; \Gamma \vdash_M (\mathbf{fst}\ \mathbf{e}) : \tau}$

  Straightforward.

- Case (MSnd): $\dfrac{\Delta;\Gamma \vdash_M \mathbf{e} : \tau^*}{\Delta;\Gamma \vdash_M (\mathbf{rst}\ \mathbf{e}) : \tau^*}$

  Straightforward.

- Case (MNat): $\dfrac{}{\Delta;\Gamma \vdash_M \overline{n} : \mathbf{Nat}}$

  To show: $\Delta;\Gamma \vdash_M \overline{n} \lesssim_M \overline{n} : \mathbf{Nat}$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(\overline{n})), \delta_2(\gamma_2(\overline{n}))) \in \mathcal{E}_M[\![\mathbf{Nat}]\!]\delta$, or, equivalently (since there are no variables in $\overline{n}$), that $(j, w, \overline{n}, \overline{n}) \in \mathcal{E}_M[\![\mathbf{Nat}]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright \overline{n} \longmapsto^i$ **Error**: str.

     But $\overline{n}$ is a value (not **Error**: str), so this case cannot occur.
  2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
     - $i < j$, and
     - $w.K_1 \triangleright \overline{n} \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

     Since $\overline{n}$ is a value, it must be the case that $i = 0$ and $\mathbf{v}_1 = \overline{n}$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

     Choose:
     - $\mathbf{v}_2 = \overline{n}$,
     - $K_2' = w.K_2$, and
     - $w' = w$.

     To show:
     - $w.K_2 \triangleright \overline{n} \longmapsto^* w.K_2 \triangleright \overline{n}$,
     - $(j, w) \sqsupseteq (j, w)$,
     - $w.K_1 = w.K_1$,
     - $w.K_2 = w.K_2$, and
     - $(j, w, \overline{n}, \overline{n}) \in \mathcal{V}_M[\![\mathbf{Nat}]\!]\delta$.

     The first four of these conjuncts are immediate, and the last follows immediately from the definition of $\mathcal{V}_M[\![\mathbf{Nat}]\!]$.

- Case (MOp): $\dfrac{\Delta;\Gamma \vdash_M \mathbf{e}_1 : \mathbf{Nat} \qquad \Delta;\Gamma \vdash_M \mathbf{e}_2 : \mathbf{Nat}}{\Delta;\Gamma \vdash_M (op\ \mathbf{e}_1\ \mathbf{e}_2) : \mathbf{Nat}}$

  Straightforward.

- Case (MIfZero): $\dfrac{\Delta;\Gamma \vdash_M \mathbf{e}_1 : \mathbf{Nat} \qquad \Delta;\Gamma \vdash_M \mathbf{e}_2 : \tau \qquad \Delta;\Gamma \vdash_M \mathbf{e}_3 : \tau}{\Delta;\Gamma \vdash_M (\mathbf{if0}\ \mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3) : \tau}$

  Straightforward.

- Case (MTAbs): $\dfrac{\Delta, \alpha;\Gamma \vdash_M \mathbf{e} : \tau}{\Delta;\Gamma \vdash_M (\Lambda\alpha.\ \mathbf{e}) : \forall\alpha.\ \tau}$

  To show: $\Delta;\Gamma \vdash_M (\Lambda\alpha.\ \mathbf{e}) \lesssim_M (\Lambda\alpha.\ \mathbf{e}) : \forall\alpha.\ \tau$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(\Lambda\alpha.\ \mathbf{e})), \delta_2(\gamma_2(\Lambda\alpha.\ \mathbf{e}))) \in \mathcal{E}_M[\![\forall\alpha.\ \tau]\!]\delta$, or, equivalently, that $(j, w, \Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e})), \Lambda\alpha.\ \delta_2(\gamma_2(\mathbf{e}))) \in \mathcal{E}_M[\![\forall\alpha.\ \tau]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright \Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e})) \longmapsto^i \mathbf{Error}$: str.

     But $\Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e}))$ is a value (not **Error**: str), so this case cannot occur.
  2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
     - $i < j$, and
     - $w.K_1 \triangleright \Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e})) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

     Since $\Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e}))$ is a value, it must be the case that $i = 0$ and $\mathbf{v}_1 = \Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e}))$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

     Choose:
     - $\mathbf{v}_2 = \Lambda\alpha.\ \delta_2(\gamma_2(\mathbf{e}))$,
     - $K_2' = w.K_2$, and
     - $w' = w$.

     To show:
     - $w.K_2 \triangleright \Lambda\alpha.\ \delta_2(\gamma_2(\mathbf{e})) \longmapsto^* w.K_2 \triangleright \Lambda\alpha.\ \delta_2(\gamma_2(\mathbf{e}))$,
     - $(j, w) \sqsupseteq (j, w)$,
     - $w.K_1 = w.K_1$,
     - $w.K_2 = w.K_2$, and
     - $(j, w, \Lambda\alpha.\ \delta_1(\gamma_1(\mathbf{e})), \Lambda\alpha.\ \delta_2(\gamma_2(\mathbf{e}))) \in \mathcal{V}_M[\![\forall\alpha.\ \tau]\!]\delta$.

     The first four conjuncts are immediate, and the last we show as follows:

     Consider arbitrary $j', w', \tau_1, \tau_2, \mathbf{R}, k_1, k_2$ such that:
     - $(j', w') \sqsupseteq (j, w)$,
     - $(\tau_1, \tau_2, \mathbf{R}) \in \text{SomeRel}_{j'}$, and
     - $k_1 \notin w'.K_1$, and

– $k_2 \notin w'.K_2$.

To show:
$(j', w' \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \mathbf{R}), \delta_1(\gamma_1(\mathbf{e}))[\alpha := \langle k_1; \tau_1\rangle], \delta_2(\gamma_2(\mathbf{e}))[\alpha := \langle k_2; \tau_2\rangle]) \in \blacktriangleright \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$, or, equivalently, that
$(j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1}), \delta_1(\gamma_1(\mathbf{e}))[\alpha := \langle k_1; \tau_1\rangle], \delta_2(\gamma_2(\mathbf{e}))[\alpha := \langle k_2; \tau_2\rangle]) \in \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$.
Applying the induction hypothesis to $\Delta, \alpha; \Gamma \vdash_M \mathbf{e} : \tau$, we have that $\Delta, \alpha; \Gamma \vdash_M \mathbf{e} \lesssim_M \mathbf{e} : \tau$, which we instantiate with $j', j' - 1, \delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})], \gamma_1, \gamma_2, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1})$. Note that:

– $j' \geq 0$,
– $j' - 1 < j'$,
– $(j', \delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]) \in \mathcal{D}[\![\Delta, \alpha]\!]$, which we show as follows:
From the definition of $\mathcal{D}[\![.]\!]$, we're required to show that:
  * $\mathrm{dom}(\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]) = \Delta, \alpha$, which follows from $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, since $\mathrm{dom}(\delta) = \Delta$ and extending δ with a binding for α adds α to its domain; and
  * $\forall \alpha' \in \Delta, \alpha.\ \delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})](\alpha') \in \mathrm{SomeRel}_{j'}$, which we show as follows: Since $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, we have that $\forall \alpha' \in \Delta.\ \delta(\alpha') \in \mathrm{SomeRel}_n$. Therefore, since $j' < n$ and SomeRel is downward closed, $\delta(\alpha') \in \mathrm{SomeRel}_{j'}$ as well. Finally, $\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})](\alpha) = (\tau_1, \tau_2, \mathbf{R})$, and our premises state that $(\tau_1, \tau_2, \mathbf{R})$ is also a member of $\mathrm{SomeRel}_{j'}$.
– $(j'-1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1}), \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$, which we show as follows:
From the definition of $\mathcal{G}[\![.]\!]$, we're required to show that:
  * $\mathrm{dom}(\Gamma) = \mathrm{dom}(\gamma_1) = \mathrm{dom}(\gamma_2)$, which follows from the premise that $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$; and
  * $\forall x \in \mathrm{dom}(\Gamma).\ (j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1}), \gamma_1(x), \gamma_2(x)) \in \mathcal{V}_*[\![\Gamma(x)]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$, which we show as follows:
  From the premise $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, we have that
  $\forall x \in \mathrm{dom}(\Gamma).\ (j, w, \gamma_1(x), \gamma_2(x)) \in \mathcal{V}_*[\![\Gamma(x)]\!]\delta$.
  Hence, by Lemma 1.4, we have that $\forall x \in \mathrm{dom}(\Gamma).\ (j, w, \gamma_1(x), \gamma_2(x)) \in \mathcal{V}_*[\![\Gamma(x)]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$.
  Next, note that $(j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1})) \sqsupseteq (j, w)$, which follows from Lemma 1.1, applied to:
    · $(j', w') \sqsupseteq (j, w)$ (from premises);
    · $(j' - 1, \lfloor w' \rfloor_{j'-1}) \sqsupseteq (j', w')$ (from the definition of $\sqsupseteq$, since $\lfloor w' \rfloor_{j'-1}.\Psi = \lfloor w'.\Psi \rfloor_{j'-1}$);
    · $(j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1})) \sqsupseteq (j' - 1, \lfloor w' \rfloor_{j'-1})$ (from Lemma 1.9).

  So, since $(j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1})) \sqsupseteq (j, w)$, we have from Lemma 1.3 that
  $\forall x \in \mathrm{dom}(\Gamma).\ (j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1}), \gamma_1(x), \gamma_2(x)) \in \mathcal{V}_*[\![\Gamma(x)]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$,
  as we were required to show.

Therefore we have that
$(j'-1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1}), \delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]_1(\gamma_1(\mathbf{e})), \delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]_2(\gamma_2(\mathbf{e}))) \in \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$.

which is equivalent to:
$(j' - 1, \lfloor w' \rfloor_{j'-1} \boxplus (\alpha \mapsto k_1, k_2, \tau_1, \tau_2, \lfloor \mathbf{R} \rfloor_{j'-1}), \delta_1(\gamma_1(\mathbf{e}))[\alpha := \langle k_1; \tau_1 \rangle], \delta_2(\gamma_2(\mathbf{e}))[\alpha := \langle k_2; \tau_2 \rangle]) \in \mathcal{E}_M[\![\tau]\!]\delta[\alpha \mapsto (\tau_1, \tau_2, \mathbf{R})]$,
as we were required to show.

- Case (MTApp): $\dfrac{\Delta; \Gamma \vdash_M \mathbf{e} : \forall\alpha.\ \tau' \qquad \Delta \vdash \tau}{\Delta; \Gamma \vdash_M \mathbf{e}\ \tau : \tau'[\alpha := \tau]}$

  To show: $\Delta; \Gamma \vdash_M \mathbf{e}\ \tau \lesssim_M \mathbf{e}\ \tau : \tau'[\alpha := \tau]$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(\mathbf{e}\ \tau)), \delta_2(\gamma_2(\mathbf{e}\ \tau))) \in \mathcal{E}_M[\![\tau'[\alpha := \tau]]\!]\delta$, or, equivalently,
  $(j, w, \delta_1(\gamma_1(\mathbf{e}))\ \delta_1(\tau), \delta_2(\gamma_2(\mathbf{e}))\ \delta_2(\tau)) \in \mathcal{E}_M[\![\tau'[\alpha := \tau]]\!]\delta$.

  The proof is in two parts:

  1. (Error conjunct): Straightforward from the operational semantics and above assumptions.
  2. (Value conjunct): Consider arbitrary $i, K_1', \mathbf{v}_1$ such that:
     - $i < j$, and
     - $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}))\ \delta_1(\tau) \longmapsto^i K_1' \triangleright \mathbf{v}_1$.

     Note that by the operational semantics there exist $i_1 \leq i, K_{11}, \mathbf{v}_{11}$ such that $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e})) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.
     To show: There exist $K_2', \mathbf{v}_2, w'$ such that:
     - $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}))\ \delta_2(\tau) \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
     - $(j - i, w') \sqsupseteq (j, w)$,
     - $w'.K_1 = K_1'$,
     - $w'.K_2 = K_2'$, and
     - $(j - i, w', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau'[\alpha := \tau]]\!]\delta$.

     Applying the induction hypothesis to
     $\Delta; \Gamma \vdash_M \mathbf{e} : \forall\alpha.\ \tau'$, we have that $\Delta; \Gamma \vdash_M \mathbf{e} \lesssim_M \mathbf{e} : \forall\alpha.\ \tau'$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:
     - $n \geq 0$,
     - $j < n$,
     - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
     - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

     Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e})), \delta_2(\gamma_2(\mathbf{e}))) \in \mathcal{E}_M[\![\forall\alpha.\ \tau']\!]\delta$. Instantiate the second conjunct with $i_1, K_{11}, \mathbf{v}_{11}$. Note that:
     - $i_1 < j$ (which follows from $i_1 \leq i$ and $i < j$), and
     - $w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e})) \longmapsto^{i_1} K_{11} \triangleright \mathbf{v}_{11}$.

     Therefore there exist $K_{21}, \mathbf{v}_{21}, w'$ such that:

- $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e})) \longmapsto^* K_{21} \triangleright \mathbf{v}_{21}$,
- $(j - i_1, w') \sqsupseteq (j, w)$,
- $w'.K_1 = K_{11}$,
- $w'.K_2 = K_{21}$, and
- $(j - i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\forall\alpha.\ \tau']\!]\delta$.

By the definition of $\mathcal{V}_M[\![\forall\alpha.\ \tau]\!]$, it must be the case that
$\mathbf{v}_{11} \equiv \Lambda\alpha.\ \mathbf{e}_{11}$, and
$\mathbf{v}_{21} \equiv \Lambda\alpha.\ \mathbf{e}_{21}$
for some $\mathbf{e}_{11}$ and $\mathbf{e}_{21}$.
Note that by the operational semantics,

$$\begin{array}{lll} w.K_1 \triangleright \delta_1(\gamma_1(\mathbf{e}))\ \delta_1(\tau) & \longmapsto^{i_1} & K_{11} \triangleright \mathbf{v}_{11}\ \delta_1(\tau) \\ & \equiv & K_{11} \triangleright (\Lambda\alpha.\ \mathbf{e}_{11})\ \delta_1(\tau) \\ & \longmapsto^{1} & K_{11}, k_1 \triangleright \mathbf{e}_{11}[\alpha := \langle k_1;\ \delta_1(\tau)\rangle] \\ & \longmapsto^{i_2} & K_1' \triangleright \mathbf{v}_1 \end{array}$$

where $i_2 = i - i_1 - 1$ and k_1 is a freshly generated key, i.e., $k_1 \notin K_{11}$.
We also have from the operational semantics that

$$\begin{array}{lll} w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}))\ \delta_2(\tau) & \longmapsto^{*} & K_{21} \triangleright \mathbf{v}_{21}\ \delta_2(\tau) \\ & \equiv & K_{21} \triangleright (\Lambda\alpha.\ \mathbf{e}_{21})\ \delta_2(\tau) \\ & \longmapsto^{1} & K_{21}, k_2 \triangleright \mathbf{e}_{21}[\alpha := \langle k_2;\ \delta_2(\tau)\rangle] \end{array}$$

where k_2 is a freshly generated key, i.e., $k_2 \notin K_{21}$.
Let $\mathbf{R} = \mathcal{V}_M[\![\tau]\!]\delta$, and instantiate $(j{-}i_1, w', \mathbf{v}_{11}, \mathbf{v}_{21}) \in \mathcal{V}_M[\![\forall\alpha.\ \tau']\!]\delta$ with $j{-}i_1, w', \delta_1(\tau), \delta_2(\tau), \mathbf{R}, k_1, k_2$.
Note that:

- $(j - i_1, w') \sqsupseteq (j - i_1, w')$,
- $(\delta_1(\tau), \delta_2(\tau), \mathbf{R}) \in \text{SomeRel}_{j-i_1}$, which follows from:
  - $FV(\delta_1(\tau)) = FV(\delta_2(\tau)) = \emptyset$, since δ_1 and δ_2 bind all free variables in τ, and
  - $\mathcal{V}_M[\![\tau]\!]\delta \in \text{Rel}_{j-i_1}[\delta_1\tau, \delta_2\tau]$, which follows from Lemma 1.3;
- $k_1 \notin w'.K_1$, which follows from $k_1 \notin K_{11}$ and $w'.K_1 = K_{11}$;
- $k_2 \notin w'.K_2$, which follows from $k_2 \notin K_{21}$ and $w'.K_2 = K_{21}$.

Therefore, we have that:
$(j - i_1, w' \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \mathbf{R}), \mathbf{e}_{11}[\alpha := \langle k_1;\ \delta_1(\tau)\rangle], \mathbf{e}_{21}[\alpha := \langle k_2;\ \delta_2(\tau)\rangle]) \in \blacktriangleright \mathcal{E}_M[\![\tau']\!]\delta[\alpha \mapsto (\delta_1(\tau), \delta_2(\tau), \mathbf{R})]$, or, equivalently,
$(j{-}i_1{-}1, \lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1}), \mathbf{e}_{11}[\alpha := \langle k_1;\ \delta_1(\tau)\rangle], \mathbf{e}_{21}[\alpha := \langle k_2;\ \delta_2(\tau)\rangle]) \in \mathcal{E}_M[\![\tau']\!]\delta[\alpha \mapsto (\delta_1(\tau), \delta_2(\tau), \mathbf{R})]$.
Instantiate the second conjunct of this with $i_2, K_1', \mathbf{v}_1$.
(Note that the K_1 and K_2 components of $\lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1})$ are $w'.K_1$ extended with k_1 and $w'.K_2$ extended with k_2, respectively. Therefore, in the below we write $w'.K_1, k_1$ and $w'.K_2, k_2$ in place of $\lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1}).K_1$ and $\lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1}).K_2$, respectively.)
Note that:

- $i_2 < j - i_1 - 1$ (since $i_2 = i - i_1 - 1$, and $i < j$), and

– $w'.K_1, k_1 \triangleright \mathbf{e}_{11}[\alpha := \langle k_1;\ \delta_1(\tau)\rangle] \longmapsto^{i_2} K_1' \triangleright \mathbf{v}_1$, which follows from above, since $w'.K_1 = K_{11}$.

Therefore, there exist $K_2', \mathbf{v}_2, w''$ such that:

– $w'.K_2, k_2 \triangleright \mathbf{e}_{21}[\alpha := \langle k_2;\ \delta_2(\tau)\rangle] \longmapsto^* K_2' \triangleright \mathbf{v}_2$,
– $(j - i_1 - 1 - i_2, w'') \sqsupseteq (j - i_1 - 1, \lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1}))$,
– $w''.K_1 = K_1'$,
– $w''.K_2 = K_2'$, and
– $(j - i_1 - 1 - i_2, w'', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau']\!]\delta[\alpha \mapsto (\delta_1(\tau), \delta_2(\tau), \mathbf{R})]$.

Choose:

– $\mathbf{v}_2 = \mathbf{v}_2$,
– $K_2' = K_2'$, and
– $w' = w''$.

To show:

– $w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}))\ \delta_2(\tau) \longmapsto^* K_2' \triangleright \mathbf{v}_2$, which follows from:

$$\begin{array}{lll} w.K_2 \triangleright \delta_2(\gamma_2(\mathbf{e}))\ \delta_1(\tau) & \longmapsto^* & K_{21} \triangleright \mathbf{v}_{21}\ \delta_2(\tau) \\ & \equiv & K_{21} \triangleright (\Lambda\alpha.\ \mathbf{e}_{21})\ \delta_2(\tau) \\ & \longmapsto^1 & K_{21}, k_2 \triangleright \mathbf{e}_{21}[\alpha := \langle k_2;\ \delta_2(\tau)\rangle] \\ & \longmapsto^* & K_2' \triangleright \mathbf{v}_2 \end{array}$$

(which follows from above, since $w'.K_2 = K_{21}$, so $K_{21}, k_2 = w'.K_2, k_2$),

– $(j - i, w'') \sqsupseteq (j, w)$, which follows from Lemma 1.1, since:
  * $(j - i_1, w') \sqsupseteq (j, w)$ (from above);
  * $(j - i_1 - 1, \lfloor w' \rfloor_{j-i_1-1}) \sqsupseteq (j - i_1, w')$ (from Lemma 1.8);
  * $(j - i_1 - 1, \lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1})) \sqsupseteq (j - i_1 - 1, \lfloor w' \rfloor_{j-i_1-1})$ (from Lemma 1.9); and
  * $(j - i, w'') \sqsupseteq (j - i_1 - 1, \lfloor w' \rfloor_{j-i_1-1} \boxplus (\alpha \mapsto k_1, k_2, \delta_1(\tau), \delta_2(\tau), \lfloor \mathbf{R} \rfloor_{j-i_1-1}))$ (from above, since $i = i_1 + i_2 + 1$);
– $w''.K_1 = K_1'$, which follows from above,
– $w''.K_2 = K_2'$, which follows from above, and
– $(j - i, w'', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau'[\alpha := \tau]]\!]\delta$, which follows from $(j - i_1 - 1 - i_2, w'', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_M[\![\tau']\!]\delta[\alpha \mapsto (\delta_1(\tau), \delta_2(\tau), \mathbf{R})]$, by compositionality (Lemma 2.1).

- Case (SVar): $\dfrac{}{\Delta; \Gamma, \mathsf{x} : \mathbf{TST} \vdash_S \mathsf{x} : \mathbf{TST}}$

To show: $\Delta; \Gamma, \mathsf{x} : \mathbf{TST} \vdash_S \mathsf{x} \lesssim_S \mathsf{x} : \mathbf{TST}$.

Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

– $n \geq 0$,
– $j < n$,
– $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
– $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

To show: $(j, w, \delta_1(\gamma_1(\mathsf{x})), \delta_2(\gamma_2(\mathsf{x}))) \in \mathcal{E}_S$, or, equivalently, that
$(j, w, \gamma_1(\mathsf{x}), \gamma_2(\mathsf{x})) \in \mathcal{E}_S$.

The proof is in two parts:

1. (Error conjunct): Consider arbitrary i, str such that:
   - $i < j$, and
   - $w.K_1 \triangleright \gamma_1(\mathsf{x}) \longmapsto^i \textbf{Error}$: str.

   But $\gamma_1(\mathsf{x})$ is a value (not **Error**: str), so this case cannot occur.
2. (Value conjunct): Consider arbitrary i, K_1', v_1 such that:
   - $i < j$, and
   - $w.K_1 \triangleright \gamma_1(\mathsf{x}) \longmapsto^i K_1' \triangleright \mathsf{v}_1$.

   Since $\gamma_1(\mathsf{x})$ is a value, it must be the case that $i = 0$ and $\mathsf{v}_1 = \gamma_1(\mathsf{x})$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

   Choose:
   - $\mathsf{v}_2 = \gamma_2(\mathsf{x})$,
   - $K_2' = w.K_2$, and
   - $w' = w$.

   To show:
   - $w.K_2 \triangleright \gamma_2(\mathsf{x}) \longmapsto^* w.K_2 \triangleright \gamma_2(\mathsf{x})$,
   - $(j, w) \sqsupseteq (j, w)$,
   - $w.K_1 = w.K_1$,
   - $w.K_2 = w.K_2$, and
   - $(j, w, \gamma_1(\mathsf{x}), \gamma_2(\mathsf{x})) \in \mathcal{V}_S$.

   The first four of these conjuncts are immediate, and the last follows from the premise that $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

- Case (SAbs): $\dfrac{\Delta; \Gamma, \mathsf{x} : \textbf{TST} \vdash_S \mathsf{e} : \textbf{TST}}{\Delta; \Gamma \vdash_S \lambda\mathsf{x}.\ \mathsf{e} : \textbf{TST}}$

  To show: $\Delta; \Gamma \vdash_S \lambda\mathsf{x}.\ \mathsf{e} \lesssim_S \lambda\mathsf{x}.\ \mathsf{e} : \textbf{TST}$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:
  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(\lambda\mathsf{x}.\ \mathsf{e})), \delta_2(\gamma_2(\lambda\mathsf{x}.\ \mathsf{e}))) \in \mathcal{E}_S$, or, equivalently, that
  $(j, w, \lambda\mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e})), \lambda\mathsf{x}.\ \delta_2(\gamma_2(\mathsf{e}))) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright \lambda\mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e})) \longmapsto^i \textbf{Error}$: str.

     But $\lambda\mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e}))$ is a value (not **Error**: str), so this case cannot occur.
  2. (Value conjunct): Consider arbitrary i, K_1', v_1 such that:
     - $i < j$, and

– $w.K_1 \triangleright \lambda \mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e})) \longmapsto^i K_1' \triangleright \mathsf{v}_1$.

Since $\lambda \mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e}))$ is a value, it must be the case that $i = 0$ and $\mathsf{v}_1 = \lambda \mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e}))$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

Choose:

– $\mathsf{v}_2 = \lambda \mathsf{x}.\ \delta_2(\gamma_2(\mathsf{e}))$,
– $K_2' = w.K_2$, and
– $w' = w$.

To show:

– $w.K_2 \triangleright \lambda \mathsf{x}.\ \delta_2(\gamma_2(\mathsf{e})) \longmapsto^* w.K_2 \triangleright \lambda \mathsf{x}.\ \delta_2(\gamma_2(\mathsf{e}))$,
– $(j, w) \sqsupseteq (j, w)$,
– $w.K_1 = w.K_1$,
– $w.K_2 = w.K_2$, and
– $(j, w, \lambda \mathsf{x}.\ \delta_1(\gamma_1(\mathsf{e})), \lambda \mathsf{x}.\ \delta_2(\gamma_2(\mathsf{e}))) \in \mathcal{V}_S$.

The first four conjuncts are immediate, and the last we show as follows:

Consider arbitrary $(j', w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$ such that $(j', w') \sqsupset (j, w)$.

To show: $(j', w', \delta_1(\gamma_1(\mathsf{e}))[\mathsf{x} := \mathsf{v}_1], \delta_2(\gamma_2(\mathsf{e}))[\mathsf{x} := \mathsf{v}_2]) \in \mathcal{E}_S$.

Applying the induction hypothesis to $\Delta; \Gamma, \mathsf{x} : \mathbf{TST} \vdash_S \mathsf{e} : \mathbf{TST}$, we have that $\Delta; \Gamma, \mathsf{x} : \mathbf{TST} \vdash_S \mathsf{e} \lesssim_S \mathsf{e} : \mathbf{TST}$,

which we instantiate with $n, j', \delta, \gamma_1[\mathsf{x} := \mathsf{v}_1], \gamma_2[\mathsf{x} := \mathsf{v}_2], w'$. Note that:

– $n \geq 0$,
– $j' < n$ (since $j' < j$ and $j < n$),
– $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
– $(j', w', \gamma_1[\mathsf{x} := \mathsf{v}_1], \gamma_2[\mathsf{x} := \mathsf{v}_2]) \in \mathcal{G}[\![\Gamma, \mathsf{x} : \mathbf{TST}]\!]\delta$, which follows from:
  * $(j', w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$, and
  * $(j', w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$, by downward closure of $\mathcal{V}_S$.

We therefore have that $(j', w', \delta_1(\gamma_1[\mathsf{x} := \mathsf{v}_1](\mathsf{e})), \delta_2(\gamma_2[\mathsf{x} := \mathsf{v}_2](\mathsf{e}))) \in \mathcal{E}_S$, which is equivalent to

$(j', w', \delta_1(\gamma_1(\mathsf{e}))[\mathsf{x} := \mathsf{v}_1], \delta_2(\gamma_2(\mathsf{e}))[\mathsf{x} := \mathsf{v}_2]) \in \mathcal{E}_S$, as we were required to show.

- Case (SApp): $\dfrac{\Delta; \Gamma \vdash_S \mathsf{e}_1 : \mathbf{TST} \qquad \Delta; \Gamma \vdash_S \mathsf{e}_2 : \mathbf{TST}}{\Delta; \Gamma \vdash_S (\mathsf{e}_1\ \mathsf{e}_2) : \mathbf{TST}}$

To show: $\Delta; \Gamma \vdash_S (\mathsf{e}_1\ \mathsf{e}_2) \lesssim_S (\mathsf{e}_1\ \mathsf{e}_2) : \mathbf{TST}$.

Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

– $n \geq 0$,
– $j < n$,
– $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
– $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

To show: $(j, w, \delta_1(\gamma_1(\mathsf{e}_1\ \mathsf{e}_2)), \delta_2(\gamma_2(\mathsf{e}_1\ \mathsf{e}_2))) \in \mathcal{E}_S$, or, equivalently,

$(j, w, (\delta_1(\gamma_1(\mathsf{e}_1))\ \delta_1(\gamma_1(\mathsf{e}_2))), (\delta_2(\gamma_2(\mathsf{e}_1))\ \delta_2(\gamma_2(\mathsf{e}_2)))) \in \mathcal{E}_S$.

The proof is in two parts:

1. (Error conjunct): Straightforward from the operational semantics and above assumptions.
2. (Value conjunct): Consider arbitrary i, K_1', v_1 such that:
   - $i < j$, and
   - $w.K_1 \triangleright (\delta_1(\gamma_1(\mathsf{e}_1))\ \delta_1(\gamma_1(\mathsf{e}_2))) \longmapsto^{i} K_1' \triangleright \mathsf{v}_1$.

   Note that by the operational semantics there exist $i_1 \leq i, K_{11}, \mathsf{v}_{11}$ such that $w.K_1 \triangleright \delta_1(\gamma_1(\mathsf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathsf{v}_{11}$.

   To show: There exist K_2', v_2, w' such that:
   - $w.K_2 \triangleright (\delta_2(\gamma_2(\mathsf{e}_1))\ \delta_2(\gamma_2(\mathsf{e}_2))) \longmapsto^{*} K_2' \triangleright \mathsf{v}_2$,
   - $(j - i, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_1'$,
   - $w'.K_2 = K_2'$, and
   - $(j - i, w', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$.

   Applying the induction hypothesis to $\Delta; \Gamma \vdash_S \mathsf{e}_1 : \mathbf{TST}$, we have that $\Delta; \Gamma \vdash_S \mathsf{e}_1 \lesssim_S \mathsf{e}_1 : \mathbf{TST}$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$. Note that:
   - $n \geq 0$,
   - $j < n$,
   - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
   - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

   Therefore $(j, w, \delta_1(\gamma_1(\mathsf{e}_1)), \delta_2(\gamma_2(\mathsf{e}_1))) \in \mathcal{E}_S$. Instantiate the second conjunct with $i_1, K_{11}, \mathsf{v}_{11}$. Note that:
   - $i_1 < j$ (which follows from $i_1 \leq i$ and $i < j$), and
   - $w.K_1 \triangleright \delta_1(\gamma_1(\mathsf{e}_1)) \longmapsto^{i_1} K_{11} \triangleright \mathsf{v}_{11}$.

   Therefore there exist $K_{21}, \mathsf{v}_{21}, w'$ such that:
   - $w.K_2 \triangleright \delta_2(\gamma_2(\mathsf{e}_1)) \longmapsto^{*} K_{21} \triangleright \mathsf{v}_{21}$,
   - $(j - i_1, w') \sqsupseteq (j, w)$,
   - $w'.K_1 = K_{11}$,
   - $w'.K_2 = K_{21}$, and
   - $(j - i_1, w', \mathsf{v}_{11}, \mathsf{v}_{21}) \in \mathcal{V}_S$.

   Therefore it must be the case that

   $\mathsf{v}_{11} \equiv \lambda \mathsf{x}.\ \mathsf{e}_{11}$, and

   $\mathsf{v}_{21} \equiv \lambda \mathsf{x}.\ \mathsf{e}_{21}$.

   Note that

$$\begin{array}{lll} w.K_1 \triangleright (\delta_1(\gamma_1(\mathsf{e}_1))\ \delta_1(\gamma_1(\mathsf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathsf{v}_{11}\ \delta_1(\gamma_1(\mathsf{e}_2))) \\ & \equiv & K_{11} \triangleright ((\lambda \mathsf{x}.\ \mathsf{e}_{11})\ \delta_1(\gamma_1(\mathsf{e}_2))) \\ & \longmapsto^{i - i_1} & K_1' \triangleright \mathsf{v}_1. \end{array}$$

   Therefore, by the operational semantics there exist i_2, K_{12}, and v_{12} such that:
   - $K_{11} \triangleright \delta_1(\gamma_1(\mathsf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathsf{v}_{12}$, and
   - $i_2 \leq i - i_1$.

   Applying the induction hypothesis to $\Delta; \Gamma \vdash_S \mathsf{e}_2 : \mathbf{TST}$, we have that $\Delta; \Gamma \vdash_S \mathsf{e}_2 \lesssim_S \mathsf{e}_2 : \mathbf{TST}$, which we instantiate with $n, j - i_1, \delta, \gamma_1, \gamma_2, w'$. Note that:

- $n \geq 0$,
- $j - i_1 < n$ (since $j < n$ and $i_1 < j$),
- $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
- $(j - i_1, w', \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$, by downward closure of $\mathcal{G}[\![.]\!]$.

Therefore $(j - i_1, w', \delta_1(\gamma_1(\mathsf{e}_2)), \delta_2(\gamma_2(\mathsf{e}_2))) \in \mathcal{E}_S$. Instantiate the second conjunct with $i_2, K_{12}, \mathsf{v}_{12}$. Note that:

- $i_2 < j - i_1$ (since $i_2 \leq i - i_2$ and $i < j$), and
- $w'.K_1 \triangleright \delta_1(\gamma_1(\mathsf{e}_2)) \longmapsto^{i_2} K_{12} \triangleright \mathsf{v}_{12}$, which follows from above since $w'.K_1 = K_{11}$.

Therefore there exist $K_{22}, \mathsf{v}_{22}, w''$ such that:

- $w'.K_2 \triangleright \delta_2(\gamma_2(\mathsf{e}_2)) \longmapsto^* K_{22} \triangleright \mathsf{v}_{22}$,
- $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$,
- $w''.K_1 = K_{12}$,
- $w''.K_2 = K_{22}$, and
- $(j - i_1 - i_2, w'', \mathsf{v}_{12}, \mathsf{v}_{22}) \in \mathcal{V}_S$.

By the operational semantics, we have that

$$\begin{array}{lll} w.K_1 \triangleright (\delta_1(\gamma_1(\mathsf{e}_1))\ \delta_1(\gamma_1(\mathsf{e}_2))) & \longmapsto^{i_1} & K_{11} \triangleright (\mathsf{v}_{11}\ \delta_1(\gamma_1(\mathsf{e}_2))) \\ & \equiv & K_{11} \triangleright ((\lambda \mathsf{x}.\ \mathsf{e}_{11})\ \delta_1(\gamma_1(\mathsf{e}_2))) \\ & \longmapsto^{i_2} & K_{12} \triangleright ((\lambda \mathsf{x}.\ \mathsf{e}_{11})\ \mathsf{v}_{12}) \\ & \longmapsto^{1} & K_{12} \triangleright \mathsf{e}_{11}[\mathsf{x} := \mathsf{v}_{12}] \\ & \longmapsto^{i_3} & K_1' \triangleright \mathsf{v}_1 \end{array}$$

where $i = i_1 + i_2 + 1 + i_3$. (Note that the key store cannot change from the third to the fourth configuration, since no new keys are generated during β-substitution.)

Recall that $(j - i_1, w', \lambda \mathsf{x}.\ \mathsf{e}_{11}, \lambda \mathsf{x}.\ \mathsf{e}_{21}) \in \mathcal{V}_S$ and instantiate with $j - i_1 - i_2 - 1, w'', \mathsf{v}_{12}, \mathsf{v}_{22}$. Note that:

- $(j - i_1 - i_2 - 1, w'') \sqsupseteq (j - i_1, w')$ (from above, since $(j - i_1 - i_2, w'') \sqsupseteq (j - i_1, w')$, by downward closure of world extension under decreasing step index), and
- $(j - i_1 - i_2 - 1, w'', \mathsf{v}_{12}, \mathsf{v}_{22}) \in \mathcal{V}_S$ (from above, since $(j - i_1 - i_2, w'', \mathsf{v}_{12}, \mathsf{v}_{22}) \in \mathcal{V}_S$, by downward closure of $\mathcal{V}_S$).

Therefore $(j - i_1 - i_2 - 1, w'', \mathsf{e}_{11}[\mathsf{x} := \mathsf{v}_{12}], \mathsf{e}_{21}[\mathsf{x} := \mathsf{v}_{22}]) \in \mathcal{E}_S$. Instantiate the second conjunct with i_3, K_1', and v_1.

Note that:

- $i_3 < j - i_1 - i_2 - 1$ (since $i = i_1 + i_2 + i_3$, so $i_3 = i - i_1 - i_2 - 1$, and $i < j$), and
- $w''.K_1 \triangleright \mathsf{e}_{11}[\mathsf{x} := \mathsf{v}_{12}] \longmapsto^{i_3} K_1' \triangleright \mathsf{v}_1$, which follows from above, since $w''.K_1 = K_{12}$.

Therefore, there exist K_2', v_2, w''' such that:

- $w''.K_2 \triangleright \mathsf{e}_{21}[\mathsf{x} := \mathsf{v}_{22}] \longmapsto^* K_2' \triangleright \mathsf{v}_2$,
- $(j - i_1 - i_2 - 1 - i_3, w''') \sqsupseteq (j - i_1 - i_2 - 1, w'')$,
- $w'''.K_1 = K_1'$,
- $w'''.K_2 = K_2'$, and
- $(j - i_1 - i_2 - 1 - i_3, w''', \mathsf{v}_1, \mathsf{v}_2) \in \mathcal{V}_S$,

fulfilling our proof obligation.

- Case (SNil): $\frac{}{\Delta;\Gamma \vdash_S \mathsf{nil} : \mathbf{TST}}$

  To show: $\Delta;\Gamma \vdash_S \mathsf{nil} \lesssim_S \mathsf{nil} : \mathbf{TST}$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(\mathsf{nil})), \delta_2(\gamma_2(\mathsf{nil}))) \in \mathcal{E}_S$, or, equivalently (since there are no variables in nil), that $(j, w, \mathsf{nil}, \mathsf{nil}) \in \mathcal{E}_S$.

  The proof is in two parts:

  1. (Error conjunct): Consider arbitrary i, str such that:
     - $i < j$, and
     - $w.K_1 \triangleright \mathsf{nil} \longmapsto^i \mathbf{Error}$: str.

     But nil is a value (not **Error**: str), so this case cannot occur.
  2. (Value conjunct): Consider arbitrary i, K_1', v_1 such that:
     - $i < j$, and
     - $w.K_1 \triangleright \mathsf{nil} \longmapsto^i K_1' \triangleright \mathsf{v}_1$.

     Since nil is a value, it must be the case that $i = 0$ and $\mathsf{v}_1 = \mathsf{nil}$. Therefore $K_1' = w.K_1$, since no new keys can be generated in 0 steps.

     Choose:
     - $\mathsf{v}_2 = \mathsf{nil}$,
     - $K_2' = w.K_2$, and
     - $w' = w$.

     To show:
     - $w.K_2 \triangleright \mathsf{nil} \longmapsto^* w.K_2 \triangleright \mathsf{nil}$,
     - $(j, w) \sqsupseteq (j, w)$,
     - $w.K_1 = w.K_1$,
     - $w.K_2 = w.K_2$, and
     - $(j, w, \mathsf{nil}, \mathsf{nil}) \in \mathcal{V}_S$.

     The first four of these conjuncts are immediate, and the last follows immediately from the definition of $\mathcal{V}_S$.

- Case (SPair): $\frac{\Delta;\Gamma \vdash_S \mathsf{e}_1 : \mathbf{TST} \qquad \Delta;\Gamma \vdash_S \mathsf{e}_2 : \mathbf{TST}}{\Delta;\Gamma \vdash_S (\mathsf{cons}\ \mathsf{e}_1\ \mathsf{e}_2) : \mathbf{TST}}$

  Straightforward.

- Case (SFst): $\frac{\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}}{\Delta;\Gamma \vdash_S (\mathsf{fst}\ \mathsf{e}) : \mathbf{TST}}$

  Straightforward.

- Case (SSnd): $\dfrac{\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}}{\Delta;\Gamma \vdash_S (\mathsf{rst}\ \mathsf{e}) : \mathbf{TST}}$

  Straightforward.

- Case (SNat): $\dfrac{}{\Delta;\Gamma \vdash_S \overline{n} : \mathbf{TST}}$

  Straightforward.

- Case (SOp): $\dfrac{\Delta;\Gamma \vdash_S \mathsf{e}_1 : \mathbf{TST} \qquad \Delta;\Gamma \vdash_S \mathsf{e}_2 : \mathbf{TST}}{\Delta;\Gamma \vdash_S (op\ \mathsf{e}_1\ \mathsf{e}_2) : \mathbf{TST}}$

  Straightforward.

- Case (SIfZero): $\dfrac{\Delta;\Gamma \vdash_S \mathsf{e}_1 : \mathbf{TST} \qquad \Delta;\Gamma \vdash_S \mathsf{e}_2 : \mathbf{TST} \qquad \Delta;\Gamma \vdash_S \mathsf{e}_3 : \mathbf{TST}}{\Delta;\Gamma \vdash_S (\mathsf{if0}\ \mathsf{e}_1\ \mathsf{e}_2\ \mathsf{e}_3) : \mathbf{TST}}$

  Straightforward.

- Case (SProc?):

  Straightforward.

- Case (SNat?):

  Straightforward.

- Case (SNil?):

  Straightforward.

- Case (SPair?):

  Straightforward.

- Case (MS): $\dfrac{\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST} \qquad \Delta \vdash |\kappa|}{\Delta;\Gamma \vdash_M ({}^{\kappa}MS\ \mathsf{e}) : |\kappa|}$

  To show: $\Delta;\Gamma \vdash_M ({}^{\kappa}MS\ \mathsf{e}) \lesssim_M ({}^{\kappa}MS\ \mathsf{e}) : |\kappa|$.

  Recall that from the statement of the parametricity theorem, $({}^{\kappa}MS\ \mathsf{e})$ is a key-free term. Since $({}^{\kappa}MS\ \mathsf{e})$ is key-free, κ cannot contain keys, so it must be some type τ, and its key erasure $|\kappa|$ is also simply τ. It therefore suffices to show that

  $\Delta;\Gamma \vdash_M ({}^{\tau}MS\ \mathsf{e}) \lesssim_M ({}^{\tau}MS\ \mathsf{e}) : \tau$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1({}^{\tau}MS\ \mathsf{e})), \delta_2(\gamma_2({}^{\tau}MS\ \mathsf{e}))) \in \mathcal{E}_M[\![\tau]\!]\delta$,

  or, equivalently, that $(j, w, ({}^{\delta_1(\tau)}MS\ \delta_1(\gamma_1(\mathsf{e}))), ({}^{\delta_2(\tau)}MS\ \delta_2(\gamma_2(\mathsf{e})))) \in \mathcal{E}_M[\![\tau]\!]\delta$.

  Applying the induction hypothesis to $\Delta;\Gamma \vdash_S \mathsf{e} : \mathbf{TST}$, we have that $\Delta;\Gamma \vdash_S \mathsf{e} \lesssim_S \mathsf{e} : \mathbf{TST}$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$.

  Therefore $(j, w, \delta_1(\gamma_1(\mathsf{e})), \delta_2(\gamma_2(\mathsf{e}))) \in \mathcal{E}_S$.

  Hence, by the bridge lemma, $(j, w, ({}^{\delta_1(\tau)}MS\ \delta_1(\gamma_1(\mathsf{e}))), ({}^{\delta_2(\tau)}MS\ \delta_2(\gamma_2(\mathsf{e})))) \in \mathcal{E}_M[\![\tau]\!]\delta$, as we were required to show.

- Case (SM): $\dfrac{\Delta; \Gamma \vdash_M \mathbf{e} : |\kappa| \qquad \Delta \vdash |\kappa|}{\Delta; \Gamma \vdash_S (SM^{\kappa}\ \mathbf{e}) : \mathbf{TST}}$

  To show: $\Delta; \Gamma \vdash_S (SM^{\kappa}\ \mathbf{e}) \lesssim_S (SM^{\kappa}\ \mathbf{e}) : \mathbf{TST}$.

  Analogously to the previous case, $(SM^{\kappa}\ \mathbf{e})$ is a key-free term, so κ and $|\kappa|$ must be some type τ, and it suffices to show that

  $\Delta; \Gamma \vdash_S (SM^{\tau}\ \mathbf{e}) \lesssim_S (SM^{\tau}\ \mathbf{e}) : \mathbf{TST}$.

  Consider arbitrary $n, j, \delta, \gamma_1, \gamma_2, w$ such that:

  - $n \geq 0$,
  - $j < n$,
  - $(n, \delta) \in \mathcal{D}[\![\Delta]\!]$, and
  - $(j, w, \gamma_1, \gamma_2) \in \mathcal{G}[\![\Gamma]\!]\delta$.

  To show: $(j, w, \delta_1(\gamma_1(SM^{\tau}\ \mathbf{e})), \delta_2(\gamma_2(SM^{\tau}\ \mathbf{e}))) \in \mathcal{E}_S$,

  or, equivalently, that $(j, w, (SM^{\delta_1(\tau)}\ \delta_1(\gamma_1(\mathbf{e}))), (SM^{\delta_2(\tau)}\ \delta_2(\gamma_2(\mathbf{e})))) \in \mathcal{E}_S$.

  Applying the induction hypothesis to $\Delta; \Gamma \vdash_M \mathbf{e} : \tau$, we have that $\Delta; \Gamma \vdash_M \mathbf{e} \lesssim_M \mathbf{e} : \tau$, which we instantiate with $n, j, \delta, \gamma_1, \gamma_2, w$.

  Therefore $(j, w, \delta_1(\gamma_1(\mathbf{e})), \delta_2(\gamma_2(\mathbf{e}))) \in \mathcal{E}_M[\![\tau]\!]\delta$.

  Hence, by the bridge lemma, $(j, w, (SM^{\delta_1(\tau)}\ \delta_1(\gamma_1(\mathbf{e}))), (SM^{\delta_2(\tau)}\ \delta_2(\gamma_2(\mathbf{e})))) \in \mathcal{E}_S$,

  as we were required to show.

□