Coherence of Type Class Resolution

GERT-JAN BOTTU, KU Leuven, Belgium
NINGNING XIE, The University of Hong Kong, China
KOAR MARNTIROSIAN, KU Leuven, Belgium
TOM SCHRIJVERS, KU Leuven, Belgium

Elaboration-based type class resolution, as found in languages like Haskell, Mercury and PureScript, is generally nondeterministic: there can be multiple ways to satisfy a wanted constraint in terms of global instances and locally given constraints. Coherence is the key property that keeps this sane; it guarantees that, despite the nondeterminism, programs still behave predictably. Even though elaboration-based resolution is generally assumed coherent, as far as we know, there is no formal proof of this property in the presence of sources of nondeterminism, like superclasses and flexible contexts.

This paper provides a formal proof to remedy the situation. The proof is non-trivial because the semantics elaborates resolution into a target language where different elaborations can be distinguished by contexts that do not have a source language counterpart. Inspired by the notion of full abstraction, we present a two-step strategy that first elaborates nondeterministically into an intermediate language that preserves contextual equivalence, and then deterministically elaborates from there into the target language. We use an approach based on logical relations to establish contextual equivalence and thus coherence for the first step of elaboration, while the second step's determinism straightforwardly preserves this coherence property.

91

CCS Concepts: • **Theory of computation → Type theory**; • **Software and its engineering → Correctness**; **Functional languages**.

Additional Key Words and Phrases: type class resolution, coherence, logical relations

ACM Reference Format:
Gert-Jan Bottu, Ningning Xie, Koar Marntirosian, and Tom Schrijvers. 2019. Coherence of Type Class Resolution. *Proc. ACM Program. Lang.* 3, ICFP, Article 91 (August 2019), 143 pages. https://doi.org/10.1145/3341695

As a guide to the reader, we present the source language $\boldsymbol{\lambda_{TC}}$ in blue, the intermediate language $\boldsymbol{F_D}$ in green and the target language $\boldsymbol{F_{\{\}}}$ in red. We thus encourage the reader to view / print this paper in color.

1 INTRODUCTION

Type classes were initially introduced in Haskell [Peyton Jones 2003] by Wadler and Blott [Wadler and Blott 1989] to make ad-hoc overloading less ad hoc, and they have since become one of Haskell's core abstraction features. Moreover, their resounding success has spread far beyond Haskell: several languages have adopted them (e.g., Mercury [Henderson et al. 1996], Coq [Sozeau and Oury 2008], PureScript [Freeman 2017], Lean [de Moura et al. 2015]), and they have inspired various alternative language features (e.g., Scala's implicits [Martin Odersky and Venners 2008; Odersky et al. 2017], Rust's traits [Mozilla Research 2017], C++'s concepts [Gregor et al. 2006], Agda's instance arguments [Devriese and Piessens 2011]).

Type classes have also received a lot of attention from researchers with many proposals for extensions and improvements, including functional dependencies [Jones 2000], associated types [Chakravarty et al. 2005], quantified constraints [Bottu et al. 2017] among other extensions.

Given the extensive attention that type classes have received, it may be surprising that the metatheory of their elaboration-based semantics [Hall et al. 1996] has not yet been exhaustively studied. In particular, as far as we know, while there have been many informal arguments, the formal notion of *coherence* has never been proven. Reynolds [1991] has defined coherence as follows:

> *"When a programming language has a sufficiently rich type structure, there can be more than one proof of the same typing judgment; potentially this can lead to semantic ambiguity since the semantics of a typed language is a function of such proofs. When no such ambiguity arises, we say that the language is coherent."*

Type classes give rise to two main (potential) sources of incoherence. The first source are *ambiguous type schemes*, such as that of the function foo:

```
> foo :: (Show a, Read a) => String -> String
> foo s = show (read s)
```

Authors' addresses: Gert-Jan Bottu, Department of Computer Science, KU Leuven, Belgium, gertjan.bottu@kuleuven.be; Ningning Xie, Department of Computer Science, The University of Hong Kong, China, nnxie@cs.hku.hk; Koar Marntirosian, Department of Computer Science, KU Leuven, Belgium, koar.marntirosian@kuleuven.be; Tom Schrijvers, Department of Computer Science, KU Leuven, Belgium, tom.schrijvers@kuleuven.be.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2019 Copyright held by the owner/author(s).
2475-1421/2019/8-ART91
https://doi.org/10.1145/3341695

The type scheme of `foo` requires that the type with which `a` will be instantiated must have `Show` and `Read` instances. This restriction alone is too permissive, because the type part (`String -> String`) of `foo`'s type scheme is not sufficient for a deterministic instantiation of `a` from the calling context. `a` can thus be instantiated arbitrarily to any type with `Show` and `Read` instances. Yet, the choice of type may lead to a different behavior of `show` and `read`, and thus of `foo` as a whole. For instance, `foo "1"` yields `"1"` when `a` is instantiated to `Int`, and `"1.0"` when it is instantiated to `Float`. To rule out this source of incoherence, Jones [1993] requires type schemes to be unambiguous and has formally proven that, for his system, this guarantees coherence.

The second source of ambiguity arises from the type class resolution mechanism itself. Such mechanisms check whether a particular type class constraint holds. Usually, they are styled after resolution-based proof search in logic, where type class instances act as Horn clauses and type scheme constraints as additional facts. Generally, this process is nondeterministic, but languages like Haskell, Mercury or PureScript contain it by requiring that type class instances do not overlap with each other or with locally given constraints. Nevertheless, superclasses remain as a source of nondeterminism; indeed, a superclass constraint can be resolved through any of its subclass constraints. Hence, in the presence of superclasses, type class resolution should properly be considered as a potential source for incoherence. Moreover, overlap between locally wanted constraints and global instances is often allowed (e.g., through GHC's `FlexibleContexts` pragma), but a formal argument for its harmlessness is also lacking. Jones [1993] considered neither of these aspects and simply assumed the coherence of resolution as a given. Morris [2014] side-steps these issues with a denotational semantics that is disconnected from the original elaboration-based semantics and its implementations (e.g., Hugs and GHC).

This paper aims to fill this gap in the metatheory of programming languages featuring type classes, including industrial grade languages such as Haskell, by formally establishing that elaboration-based type class resolution is coherent in the presence of superclasses and flexible contexts. The proof of this property is considerably complicated by the indirect, elaboration-based approach that is used to give meaning to programs with type classes. Indeed, the meaning of such programs is commonly given in terms of their translation to a core language [Hall et al. 1996], like System F, the meaning of which is defined in the form of an operational semantics. In this translation process, type classes are elaborated into explicitly passed function dictionaries. These dictionaries can, however, often be constructed in more than one way, resulting in multiple possible translations for a single program. The problem is that different translations of the same source program actually may have different meanings in the core language. The reason for this discrepancy is that the core language is more expressive than the source language and admits programs — that cannot be expressed in the source language — in which the different dictionaries can be distinguished.

We solve this problem with a new two-step approach that splits the problem into two subproblems. The midway point is an intermediate language that makes type class dictionaries explicit, but—inspired by fully abstract compilation—cannot distinguish between different elaborations from the same source language term [Abadi 1999]. We use a logical-relations approach to show that the nondeterministic elaboration from the source language to this intermediate language is coherent. Showing coherence for the elaboration from the intermediate language to the target language is much simpler, because we can formulate it in a deterministic fashion.

In summary, the contributions of this work are:

- We present a simple calculus λ_{TC} with full-blown type class resolution (incl. superclasses), which isolates nondeterministic resolution. Furthermore, we present an elaboration from λ_{TC} to the target language $F_{\{\}}$, System F with records, which are used to encode dictionaries.
- We present an intermediate language F_{D} with explicit dictionary-passing. This language enforces the uniqueness of dictionaries, which captures the intention of type class instances. We study its metatheory, and define a logical relation to prove contextual equivalence.
- We present elaborations from λ_{TC} to F_{D} and from F_{D} to $F_{\{\}}$, and prove that a direct translation from λ_{TC} to $F_{\{\}}$ can always be decomposed into an equivalent translation through F_{D}.
- We prove coherence of the elaboration between λ_{TC} and F_{D}, using logical relations.
- We prove that coherence is also preserved through the elaboration from F_{D} to $F_{\{\}}$. As a consequence, by combining this with the previous result, we prove that the elaboration between λ_{TC} and $F_{\{\}}$ is coherent. The latter coherence result implies coherence of elaboration-based type class resolution in the presence of superclasses and flexible contexts.

The full formalization and coherence proof can be found in the accompanying 122-page appendix.

The purpose of our work is twofold: 1) To develop a proof technique to establish coherence of type class resolution. Because this result is achieved on a minimal calculus, this work becomes a basis for researchers investigating type class extensions and larger languages, as well as their impact on coherence. 2) To present a formal proof of coherence for language designers considering to adopt type classes. In doing so, we show that the informally trivial argument for the coherence of type class resolution is surprisingly hard to formalize.

2 OVERVIEW

This section provides some background on dictionary-passing elaboration of type class resolution and discusses the potential nondeterminism introduced by superclasses and local constraints. We then briefly introduce our calculi and discuss the key

```
> class Eq a where
>   (==) :: a -> a -> Bool
>
> instance Eq Int where
>   (==) = primEqInt
>
> instance (Eq a, Eq b) => Eq (a, b) where
>   (x1,y1) == (x2,y2)  =  x1 == x2 && y1 == y2
>
> refl :: Eq a => a -> Bool
> refl x = x == x
>
> main :: Bool
> main = refl (5,42)
```

Example 1. Program with type classes.

ideas of the coherence proof. Throughout the section we use Haskell-like syntax as the source language for examples, and to simplify our informal discussion we use the same syntax without type classes as the target language.

2.1 Dictionary-Passing Elaboration

A program is coherent if it has exactly one meaning — i.e., its semantics is unambiguously determined. For type classes this is not as straightforward as it seems, because their dynamic semantics are not expressed directly but rather by type-directed elaboration into a simpler language without type classes such as System F. Thus the dynamic semantics of type classes are given indirectly as the dynamic semantics of their elaborated forms.

Basic Elaboration. Consider the small program with type classes in Example 1. We declare a type class `Eq` and instances for the `Int` and pair types. The function `refl` trivially tests whether an expression is equivalent to itself, which is called in `main`.

The dictionary-passing elaboration translates this program into a System F-like core language that does not feature type classes. The main idea of the elaboration is to map a type class declaration onto a datatype that contains the method implementations, a so-called *(function) dictionary.*

```
> data EqD a = EqD { (==) :: a -> a -> Bool }
```

Then simple instances give rise to dictionary values:

```
> eqInt :: EqD Int
> eqInt = EqD { (==) = primEqInt }
```

Instances with a non-empty context are translated to functions that take context dictionaries to the instance dictionary.

```
> eqPair :: (EqD a, EqD b) -> Eq (a,b)
> eqPair (da, db) =
>   EqD { (==) = \(x1,y1) (x2,y2) -> (==) da x1 x2 && (==) db y1 y2 }
```

Functions with qualified types, like `refl`, are translated to functions that take explicit dictionaries as arguments.

```
> refl :: EqD a -> a -> Bool
> refl d x = (==) d x x
```

Finally, calls to functions with a qualified type are mapped to calls that explicitly pass the appropriate dictionary.

```
> main :: Bool
> main = refl (eqPair eqInt eqInt) (5,42)
```

Elaboration of Superclasses. Superclasses require a small extension to the above elaboration scheme. Consider the small program in Example 2 where `Sub1` is a subclass of `Base`. The function `test1` has `Sub1 a` in the context and calls `sub1` and `base` in its definition.

The standard approach to encode superclass is to embed the superclass dictionary in that of the subclass. For this case, `Sub1D a` contains a field `super1` that points to the superclass:

```
> data BaseD a = BaseD { base :: a -> Bool }
> data Sub1D a = Sub1D { super1 :: BaseD a
>                      , sub1 :: a -> Bool }
```

This way we can extract the superclass from the subclass when needed. The function `test1` is then encoded as:

```
> class Base a where
>   base :: a -> Bool
>
> class Base a => Sub1 a where
>   sub1 :: a -> Bool
>
> test1 :: Sub1 a => a -> Bool
> test1 x = sub1 x && base x
```

Example 2. Program with superclasses.

```
> test1 :: Sub1 a -> a -> Bool
> test1 d x = sub1 d x && base (super1 d) x
```

Resolution. Calls to functions with a qualified type generate type class constraints. The process for checking whether these constraints can be satisfied, is known as *resolution*. For the sake of dictionary-passing elaboration, this resolution process is augmented with the construction of the appropriate dictionary that witnesses the satisfiability of the constraint.

2.2 Nondeterminism and Coherence

For Haskell'98 programs there is usually only one way to construct a dictionary for a type class constraint. Yet, in the presence of superclasses, there may be multiple ways. Suppose we extend Example 2 with an additional subclass and the following function:

```
> class Base a => Sub2 a where
>   sub2 :: a -> Bool
>
> test2 :: (Sub1 a, Sub2 a) => a -> Bool
> test2 x = base x
```

There are two possible ways to resolve the `Base a` constraint that arises from the call to `base` in function `test2`, resulting in the following two translations: we can either establish the desired constraint as the superclass of the given `Sub1 a` constraint or as the superclass of the given `Sub2 a` constraint.

```
> test2a, test2b :: (Sub1D a, Sub2D a) -> a -> Bool
> test2a (d1,d2) x = base (super1 d1) x
> test2b (d1,d2) x = base (super2 d2) x
```

Fortunately, this nondeterminism is harmless because the difference between the two elaborations cannot be observed. Indeed, for any given type `A`, Haskell'98 only allows a single instance `Base A`, and it does not matter whether we access its dictionary directly or through one of its subclass instances. More generally, this suggests that type class resolution in Haskell'98 is coherent.

If we relax the Haskell'98 non-overlap condition for locally given constraints and adopt flexible contexts (allowing for arbitrary types in class constraints, rather than simple type variables), another source of nondeterminism arises. Consider:

```
> isZero :: Eq Int => Int -> Bool
> isZero n = n == 0
```

There are two ways to resolve the wanted `Eq Int` constraint that arises from the use of `(==)`. Either we use the global `Eq Int` constraint (in `isZero1`), or we use the locally given `Eq Int` constraint, passed as argument `d` (in `isZero2`):

```
> isZero1, isZero2 :: EqD Int -> Int -> Bool
> isZero1 d n = (==) eqInt n 0
> isZero2 d n = (==) d     n 0
```

Haskell'98 does not allow the `Eq Int` constraint in `isZero`'s signature, which overlaps with the global `Eq Int` instance; it only allows constraints on type variables in function signatures. This prevents the above nondeterminism in the elaboration. Yet, the nondeterminism is, once more, harmless; there is no way that the supplied dictionary `d` can be anything other than the global instance's dictionary `eqInt`. Informally, resolution remains coherent in the presence of flexible contexts.

2.3 Contextual Difference

While it is easy to provide an informal argument for the coherence of type class resolution, formally establishing the property is much harder. The indirect, elaboration-based attribution of a dynamic semantics in particular is a complicating factor, since it requires us to reason about two languages simultaneously. Unfortunately, there is another factor that further complicates the proof: different elaborations of the same source program can actually be distinguished in the target language. Consider, for instance, the target program below:

```
> discern :: ((Sub1D (), Sub2D ()) -> () -> Bool) -> Bool
> discern f =
>   let b1 = BaseD { base = \() -> True }
>       b2 = BaseD { base = \() -> False }
>       d1 = Sub1D { super1 = b1 }
>       d2 = Sub2D { super2 = b2 }
>   in f (d1,d2) ()
```

We find that `discern test2a` evaluates to `True` and `discern test2b` evaluates to `False`. Hence, since `discern` can differentiate between them, `test2a` and `test2b` clearly do not have the same meaning in the target language.

The dictionaries for `Sub1 ()` and `Sub2 ()` have different implementations for their `Base ()` superclass. The source language would never allow this, but the target language has no notion of type classes and happily admits `discern`'s violation of source language rules.

The problem is that the target language is more expressive than the source language. While `test2a` and `test2b` cannot be distinguished in any program context that arises from the source language, we can write target programs like `discern` that are not the image of any source program and thus do not have to play by the source language rules.

2.4 Our Approach to Proving Coherence

To avoid the problem with contextual difference in the target language, we employ a novel two-step approach. We prove that any elaboration from a source language program into a dictionary-passing encoding in the target language, can be decomposed in two separate elaborations through an intermediate language. We thus obtain two simpler problems for proving coherence of type class resolution.

The source language, λ_{TC} (presented in blue), features full-fledged type class resolution, and simplifies term typing with a bidirectional type system (a technique popularized by Pierce and Turner [2000]) to not distract from the main objective of coherent resolution.

The intermediate language, F_D (presented in green), is an extension of System F that explicitly passes type class dictionaries, and preserves the source language invariant that there is at most one such dictionary value for any combination of class and type. We show F_D is type-safe and strongly normalizing, and define a logical relation that captures the contextual equivalence of two F_D terms.

The target language, $F_{\{\}}$ (presented in red), is a different variant of System F without direct support for type class dictionaries; instead it features records, which can be used to encode dictionaries, but does not enforce uniqueness of instances.

The different calculi are presented in Figure 1, where the edges denote possible elaborations.

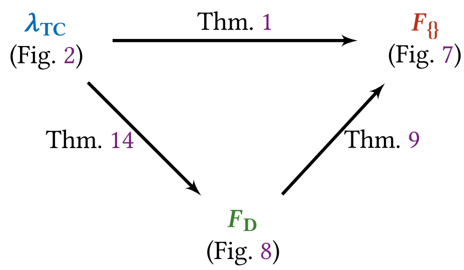

Fig. 1. The different calculi with elaborations

The coherence proof consists of two main parts:

Coherent Elaboration from λ_{TC} to F_D. Our elaboration from λ_{TC} into F_D is nondeterministic, but type preserving. Furthermore, we show that any two F_D elaborations of the same λ_{TC} term are logically related, and prove that this logical relation implies contextual equivalence. This establishes that the elaboration from λ_{TC} to F_D is coherent.

Deterministic Elaboration from F_D to $F_{\{\}}$. Because of the syntactic similarity between F_D and $F_{\{\}}$, the elaboration from the former into the latter is a more straightforward affair. In addition to being type preserving, it is also deterministic, and preserves contextual equivalence.

These results are easily combined to show the coherence of the elaboration from λ_{TC} to $F_{\{\}}$, which implies coherence of elaboration-based type class resolution. The full proofs can be found in the appendix. Note that the proofs depend on a number of standard boilerplate conjectures (e.g., substitution lemmas), which can be found in Sections J.1 and K.1 of the appendix.

$$
\begin{array}{lcll}
\tau & ::= & Bool \mid a \mid \tau_1 \rightarrow \tau_2 & \textit{monotype} \\
\rho & ::= & \tau \mid Q \Rightarrow \rho & \textit{qualified type} \\
\sigma & ::= & \rho \mid \forall a.\sigma & \textit{type scheme} \\
\\
Q & ::= & TC\,\tau & \textit{class constraint} \\
C & ::= & \forall \overline{a}.\overline{Q} \Rightarrow Q & \textit{constraint scheme} \\
\\
e & ::= & True \mid False \mid x \mid m \mid \lambda x.e \mid e_1\, e_2 \mid \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2 \mid e :: \tau & \textit{term} \\
\\
pgm & ::= & e \mid cls; pgm \mid inst; pgm & \lambda_{\text{TC}}\ \textit{program} \\
cls & ::= & \textbf{class}\ \overline{TC_i\ a} \Rightarrow TC\, a\ \textbf{where}\ \{m : \sigma\} & \textit{class decl.} \\
inst & ::= & \textbf{instance}\ \overline{Q} \Rightarrow TC\,\tau\ \textbf{where}\ \{m = e\} & \textit{instance decl.} \\
\\
\Gamma & ::= & \bullet \mid \Gamma, x : \sigma \mid \Gamma, a \mid \Gamma, \delta : Q & \textit{typing environment} \\
\Gamma_C & ::= & \bullet \mid \Gamma_C, m : \overline{TC_i\ a} \Rightarrow TC\, a : \sigma & \textit{class environment} \\
P & ::= & \bullet \mid P, (D : C).m \mapsto \Gamma : e & \textit{program context} \\
M & ::= & [\bullet] \mid \lambda x.M \mid M\, e \mid e\, M \mid M :: \tau & \textit{evaluation context} \\
 & \mid & \textbf{let}\ x : \sigma = M\ \textbf{in}\ e \mid \textbf{let}\ x : \sigma = e\ \textbf{in}\ M &
\end{array}
$$

Fig. 2. λ_{TC} syntax

3 SOURCE LANGUAGE λ_{TC}

This section presents our source language λ_{TC}, a basic calculus which isolates nondeterministic resolution. The calculus only supports features that are essential for type class resolution and its coherence.

Consequently, the language is strongly normalizing, and thus does not support recursive let expressions, mutual recursion or recursive methods. This is a sensible choice, as recursion does not affect the fundamentals of the coherence proof. This work could include recursion through step indexing [Ahmed 2006], a well-known technique, but this would significantly clutter the proof. Recursion is discussed in more detail in Section 8.

Furthermore, two notable design decisions were made in the support of superclasses in λ_{TC}. Firstly, similar to GHC, λ_{TC} derives all possible superclass constraints from their subclass constraints in advance, instead of deriving them "just-in-time" during resolution. The advantage of this approach is that it streamlines the actual resolution process.

Secondly, similar to Coq [Sozeau and Oury 2008] and unlike Wadler and Blott [1989], we pass superclass dictionaries alongside their subclass dictionaries, i.e., in a *flattened* form, instead of nesting them inside their subclass dictionaries. As it is not too difficult to see that both approaches are isomorphic, flattening the superclasses does not impact the coherence of resolution. It does, however, considerably simplify the proof, since this way neither our type class resolution mechanism, nor the intermediate language F_{D} (Section 6) need to have any support for superclasses and can treat them as regular local constraints. A more structured representation would give rise to additional complexity, but would not alter the essence of the proof.

Syntax. Figure 2 presents the, mostly standard, syntax. Programs consist of a number of class (with superclasses) and instance declarations, and an expression. For the sake of simplicity and well-foundedness, the declarations are ordered and can only refer to previous declarations.

Following Jones [1994]'s qualified types framework, we distinguish between three sorts of types: monotypes τ, qualified types ρ which include constraints, and type schemes σ which include type abstractions. Constraints differentiate between full constraint schemes C and simple class constraints Q. Observe that we allow flexible contexts in the qualified types; they are not restricted to constraints on type variables.

The definition of expressions e is standard, but with a few notable exceptions. Firstly, the language differentiates syntactically between regular variables x and method names m, which are introduced in class declarations. Secondly, type annotations $e :: \tau$ allow the programmer to manually assign a monotype to an expression. This is useful for resolving ambiguity—see the *Typing* paragraph below. Finally, let bindings include type annotations with a type scheme σ, allowing the programmer to introduce local constraints—also discussed in the *Typing* paragraph. Note that we use Haskell syntax for class and instance declarations.

There are three λ_{TC} environments: two global ones and one local environment. Firstly, the global class environment Γ_C stores all class declarations. Each entry in Γ_C contains the method name m, any superclasses $\overline{TC_i\ a}$, the class $TC\, a$ itself and the corresponding method type σ.

Secondly, the global program context P contains all instance declarations. Each entry in P consists of a unique dictionary constructor D, its corresponding constraint C, the method name m and its implementation e, together with the context Γ under which e should be interpreted. This context contains the local axioms available in this instance declaration, as well as any axioms which explicitly annotate the method type signature.

Thirdly, the local typing environment Γ, besides containing the default term and type variables x and a, also stores any local axioms Q. As opposed to the program context P, Γ does not contain any type class instances. Instead, the (local) axioms are associated with a dictionary variable δ. Sections 6 and 6.1 explain the use of these dictionaries.

Typing. Our type system features two design choices to eliminate the possibility of ambiguous type schemes. This allows us to focus on the coherence of type class resolution, by making our proof orthogonal to ambiguous type schemes, the source of ambiguity which has already been studied by Jones [1993]. We thus side-step an already solved problem and focus on tackling the full problem of resolution coherence.

Firstly, we require type signatures to be unambiguous (Figure 4, right-hand side) to make sure that all newly introduced type variables are bound in the head of the type (the remaining monotype after dropping all type and constraint abstractions). This prevents ambiguous expressions such as:

```
> let f : forall a . Eq a => Int -> Int -- ambiguous
>       = \ x . x + 1 in f 42
```

Secondly, we use a bidirectional type system rather than a fully declarative one. A bidirectional type system distinguishes between two typing modes: *inference* and *check* mode. The former synthesizes a type from the given expression, while the latter checks whether a given expression is of a given type. Special in our setting is that variables can only be typed in check mode, to ensure that only a single instantiation exists. This avoids the ambiguity that can arise when instantiating type variables in inference mode. Consider the following example:

```
> let y : forall a . Eq a => a -> a = ...
> in const 1 y
```

where `const x` is the constant function, which evaluates to `x` for any input. The instantiation of `y`'s type scheme is not uniquely determined by the context in which it is used. In a declarative type system or in inference mode, this ambiguity would result in multiple distinct typings and corresponding elaborations. While this ambiguity is harmless, it is not the focus of this work. Hence, to focus exclusively on the resolution, we use a bidirectional type system with check mode for variables to eliminate this irrelevant source of ambiguity.

Figure 3 [1] shows selected typing rules. The full set of rules can be found in Section B.2 of the appendix. We ignore the red (elaboration-related) parts for now and explain them in detail in Section 4.1. The judgments $P;\Gamma_C;\Gamma \vdash_{tm} e \Rightarrow \tau$ and $P;\Gamma_C;\Gamma \vdash_{tm} e \Leftarrow \tau$ denote *infer*ring a monotype τ for expression e and *check*ing e to have a monotype τ respectively, in environments P, Γ_C and Γ. Note that the constraint and type well-formedness relations $\vdash_Q$ and $\vdash_{ty}$ are omitted, as they are standard well-scopedness checks. They can be found in Section B.1 of the appendix.

$$\boxed{P;\Gamma_C;\Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e} \qquad (\boldsymbol{\lambda_{TC}}\ \textit{Term Inference})$$

$$\frac{\begin{array}{c} x \notin \mathbf{dom}(\Gamma) \qquad \mathbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1) \\ \mathbf{closure}(\Gamma_C;\overline{Q}_i) = \overline{Q}_k \qquad (\Gamma_C;\Gamma \vdash_Q Q_k \rightsquigarrow \sigma_k\ ,\ \forall k) \qquad \Gamma_C;\Gamma \vdash_{ty} \forall \overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{\sigma}_k \to \sigma \\ \overline{\delta}_k\ \mathbf{fresh} \qquad P;\Gamma_C;\Gamma,\overline{a}_j,\overline{\delta}_k : \overline{Q}_k \vdash_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \qquad P;\Gamma_C;\Gamma, x : \forall \overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \vdash_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \end{array}}{P;\Gamma_C;\Gamma \vdash_{tm} \mathbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\ \mathbf{in}\ e_2 \Rightarrow \tau_2 \qquad \rightsquigarrow \mathbf{let}\ x : \forall \overline{a}_j.\overline{\sigma}_k \to \sigma_1 = \Lambda \overline{a}_j.\lambda\, \overline{\delta_k : \sigma_k}^{k}.e_1\ \mathbf{in}\ e_2}\ \textsc{sTm-infT-let}$$

$$\boxed{P;\Gamma_C;\Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e} \qquad (\boldsymbol{\lambda_{TC}}\ \textit{Term Checking})$$

$$\frac{\begin{array}{c} (m : \overline{Q}'_k \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau') \in \Gamma_C \qquad \mathbf{unambig}(\forall \overline{a}_j, a.\overline{Q}_i \Rightarrow \tau') \qquad P;\Gamma_C;\Gamma \vDash TC\,\tau \rightsquigarrow e \\ \Gamma_C;\Gamma \vdash_{ty} \tau \rightsquigarrow \sigma \qquad (P;\Gamma_C;\Gamma \vDash [\overline{\tau}_j/\overline{a}_j][\tau/a]Q_i \rightsquigarrow e_i\ ,\ \forall i) \qquad (\Gamma_C;\Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j\ ,\ \forall j) \qquad \vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma \end{array}}{P;\Gamma_C;\Gamma \vdash_{tm} m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow e.m\,\overline{\sigma}_j\,\overline{e}_i}\ \textsc{sTm-checkT-meth}$$

Fig. 3. $\boldsymbol{\lambda_{TC}}$ typing, selected rules

[1] Note that lists, such as $\overline{\tau}_i$, are denoted by overlines, whereas collections of predicates are annotated by their range. For instance, $(\Gamma_C;\Gamma \vdash_{ty} \tau_i \rightsquigarrow \sigma_i\ ,\ \forall i)$ iterates over i.

$\boxed{\mathbf{closure}(\Gamma_C; \overline{Q}_i) = \overline{Q}_j}$ *(Superclass Closure)*

$$\frac{}{\mathbf{closure}(\Gamma_C; \bullet) = \bullet}\ \text{sClosure-empty}$$

$$\frac{(m : \overline{Q}_m \Rightarrow TC\ a : \sigma) \in \Gamma_C \qquad \mathbf{closure}(\Gamma_C; \overline{Q}_i, \overline{Q}_m) = \overline{Q}_j}{\mathbf{closure}(\Gamma_C; \overline{Q}_i, TC\ a) = \overline{Q}_j, TC\ a}\ \text{sClosure-TC}$$

$\boxed{\mathbf{unambig}(\sigma)}$ *(Unambiguity for Type Schemes)*

$$\frac{\overline{a}_j \in \mathbf{fv}(\tau)}{\mathbf{unambig}(\forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau)}\ \text{sUnambig-scheme}$$

$\boxed{\mathbf{unambig}(C)}$ *(Unambiguity for Constraints)*

$$\frac{\overline{a}_j \in \mathbf{fv}(\tau)}{\mathbf{unambig}(\forall \overline{a}_j . \overline{Q}_i \Rightarrow TC\ \tau)}\ \text{sUnambig-constraint}$$

Fig. 4. Closure and unambiguity relations

$\boxed{P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow e}$ *(Constraint Entailment)*

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow \delta}\ \text{sEntailT-local}$$

$$\frac{\begin{array}{c} P = P_1, (D : \forall \overline{a}_j . \overline{Q}'_i \Rightarrow Q').m \mapsto \Gamma' : e, P_2 \qquad \Gamma' = \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h \\ Q = [\overline{\tau}_j / \overline{a}_j] Q' \qquad P_1; \Gamma_C; \Gamma' \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \qquad (\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j\ ,\ \forall j) \\ (\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q'_i \rightsquigarrow \sigma'_i\ ,\ \forall i) \qquad (\Gamma_C; \bullet, \overline{a}_j, \overline{b}_k \vdash_Q Q_h \rightsquigarrow \sigma''_h\ ,\ \forall h) \qquad (P; \Gamma_C; \Gamma \vDash [\overline{\tau}_j / \overline{a}_j] Q'_i \rightsquigarrow e_i\ ,\ \forall i) \end{array}}{P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow (\Lambda \overline{a}_j . \lambda \overline{\delta'_i : \sigma'_i}^{i} . \{m = \Lambda \overline{b}_k . \lambda \overline{\delta_h : \sigma''_h}^{h} . e\})\, \overline{\sigma}_j\, \overline{e}_i}\ \text{sEntailT-inst}$$

Fig. 5. $\lambda_{\mathbf{TC}}$ constraint entailment

$\boxed{P; \Gamma_C \vdash_{inst} inst : P'}$ *(Instance Decl Typing)*

$$\frac{\begin{array}{c} (m : \overline{Q}'_i \Rightarrow TC\ a : \forall \overline{a}_j . \overline{Q}'_h \Rightarrow \tau_1) \in \Gamma_C \qquad \overline{b}_k = \mathbf{fv}(\tau) \qquad \Gamma_C; \bullet, \overline{b}_k \vdash_{ty} \tau \rightsquigarrow \sigma \\ \mathbf{closure}(\Gamma_C; \overline{Q}_p) = \overline{Q}_q \qquad \mathbf{unambig}(\forall \overline{b}_k . \overline{Q}_q \Rightarrow TC\ \tau) \qquad (\Gamma_C; \bullet, \overline{b}_k \vdash_Q Q_q \rightsquigarrow \sigma_q\ ,\ \forall q) \qquad D\ \mathbf{fresh} \\ \overline{\delta}_q\ \mathbf{fresh} \qquad \overline{\delta}'_h\ \mathbf{fresh} \qquad (P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_q : \overline{Q}_q \vDash [\tau / a] Q'_i \rightsquigarrow e_i\ ,\ \forall i) \qquad \Gamma' = \bullet, \overline{b}_k, \overline{\delta}_q : \overline{Q}_q, \overline{a}_j, \overline{\delta}'_h : [\tau / a] \overline{Q}'_h \\ P; \Gamma_C; \Gamma' \vdash_{tm} e \Leftarrow [\tau / a] \tau_1 \rightsquigarrow e \qquad (D' : \forall \overline{b}'_m . \overline{Q}'_n \Rightarrow TC\ \tau_2).m' \mapsto \Gamma'' : e' \notin P \qquad \textit{where}\ [\overline{\tau}'_m / \overline{b}'_m] \tau_2 = [\overline{\tau}'_k / \overline{b}_k] \tau \end{array}}{P; \Gamma_C \vdash_{inst} \mathbf{instance}\ \overline{Q}_p \Rightarrow TC\ \tau\ \mathbf{where}\ \{m = e\} : (D : \forall \overline{b}_k . \overline{Q}_q \Rightarrow TC\ \tau).m \mapsto \Gamma' : e}\ \text{sInstT-inst}$$

Fig. 6. $\lambda_{\mathbf{TC}}$ instance declaration typing

Through a let binding (rule sTm-infT-let), the programmer provides a type scheme for a variable, thus potentially introducing local constraints. As explained above, the unambiguity check from Figure 4 (right-hand side) requires the provided type scheme to be unambiguous. In order to flatten the superclasses, the rule takes the closure over the superclass relation (left-hand side of Figure 4) of the user provided constraints $\overline{Q}_i$. It then adds the resulting set of constraints $\overline{Q}_k$ to the typing environment, under which to typecheck e_1. Finally, the type of e_2 is inferred under the extended environment.

Rule sTm-checkT-meth types a method call m in check mode, like regular variables, to avoid any ambiguity in the instantiation of the type variables in the method's type scheme. This includes both the type variable a from the class and any additional free variables $\overline{a}_j$ in the method type. Furthermore, the rules uses the unambig-relation to avoid ambiguity in the method type scheme itself, by requiring that both sets of type variables have to occur in the head of the method type. The rule also checks that all required constraints $\overline{Q}_i$ from the method type can be entailed.

The instance typing rule can be found in Figure 6. The relation $P; \Gamma_C \vdash_{inst} inst : P'$ denotes that an instance declaration $inst$ results in a $\lambda_{\mathbf{TC}}$ program context P', while being typed under environments P and Γ_C. The unambig-relation for constraints (Figure 4, bottom right), similarly to the unambig-relation for types, checks that all free type variables $\overline{b}_k$ in the instance context occur in the instance type τ as well, in order to avoid ambiguity. Like in the sTm-infT-let rule explained above, the superclasses of the instance context $\overline{Q}_p$ are flattened into additional local constraints $\overline{Q}_q$ and added to the environment. The superclasses $\overline{Q}'_i$ of the instantiated type class are then checked to be entailed under this extended environment. The rule checks that no overlapping instance declarations D' have been defined. Finally, the program context is extended with the new instance axiom D, consisting of a constraint scheme that requires the full set of local constraints $\overline{Q}_q$.

$$\begin{array}{rcll} \sigma & ::= & Bool \mid a \mid \forall a.\sigma \mid \sigma_1 \to \sigma_2 \mid \{\, \overline{m_i : \sigma_i}^{\,i<n} \,\} & F_{\{\}}\ type \\ e & ::= & True \mid False \mid x \mid \lambda x : \sigma.e \mid e_1\, e_2 \mid \Lambda a.e \mid e\, \sigma & F_{\{\}}\ term \\ & & \mid \{\, \overline{m_i = e_i}^{\,i<n} \,\} \mid e.m \mid \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2 & \\ \Gamma & ::= & \bullet \mid \Gamma, a \mid \Gamma, x : \sigma & F_{\{\}}\ context \end{array}$$

Fig. 7. Target language syntax

Type Class Resolution. The type class resolution rules can be found in Figure 5, where $P; \Gamma_C; \Gamma \models Q$ denotes that a class constraint Q is entailed under the environments P, Γ_C and Γ. A wanted constraint Q can either be resolved using a locally available constraint δ (sEntailT-local) or through a global instance declaration D (sEntailT-inst). The former is entirely straightforward. The latter is more involved as an instance D may have an instance context $\overline{Q}'_i$, which has to be recursively resolved. Before resolving the context, the type variables $\overline{a}_j$ are instantiated with the corresponding concrete types $\overline{\tau}_j$, originating from the wanted constraint Q.

Note that the type class resolution mechanism does not require any specific support for superclasses, as these have all been flattened into regular local constraints.

4 TARGET LANGUAGE $F_{\{\}}$

This section covers our target language $F_{\{\}}$, and the elaboration from λ_{TC} to $F_{\{\}}$.

The target language is System F with records, which we consider a reasonable subcalculus of those used by Haskell compilers. Its syntax is shown in Figure 7. We omit its standard typing rules and call-by-name operational semantics and refer the reader to Pierce [2002, Chapter 23], or Section E of the appendix.

4.1 Elaboration from λ_{TC} to $F_{\{\}}$

The red aspects in Figure 3 denote the elaboration of λ_{TC} terms to $F_{\{\}}$. We have adopted the convention that any red $F_{\{\}}$ types are the elaborated forms of their identically named blue λ_{TC} counterparts. This elaboration maps most λ_{TC} forms on identical $F_{\{\}}$ terms, with the exception of a few notable cases: (a) The interesting aspect of elaborating let expressions (sTm-infT-let) is that, as mentioned previously, superclasses are flattened into additional local constraints. The elaborated expression thus explicitly requires both the type variables and the full closure of the local constraints. (b) As opposed to λ_{TC}, dictionary and type application are made explicit in $F_{\{\}}$. When elaborating variables x and method references m (sTm-checkT-meth), all previously substituted types $\overline{\tau}_j$ are now explicitly applied, together with the dictionary expressions $\overline{e}_i$. Furthermore, method names m are elaborated to $F_{\{\}}$ record labels m and therefore cannot appear by themselves, but must be applied to a record expression e, which originates from resolving the class constraint.

Type class resolution (Figure 5) of a λ_{TC} constraint Q results in a $F_{\{\}}$ expression e. When resolving the wanted constraint using a locally available constraint δ (sEntailT-local), this results in a regular term variable δ (which keeps the name of its λ_{TC} counterpart for readability). On the other hand, when resolving with the use of a global instance declaration D (sEntailT-inst), a record expression is constructed, containing the method name m and its corresponding implementation e. This method implementation now explicitly abstracts over the type variables $\overline{b}_k$ and term variables $\overline{\delta}_h$ originating from the method types's class constraints $\overline{Q}_h$, which annotate the class declaration. Furthermore, the record expression is nested in abstractions over the type variables $\overline{a}_j$ and term variables $\overline{\delta}'_i$ arising from the corresponding instance constraints $\overline{Q}'_i$. These abstractions are immediately instantiated by applying (a) the types $\overline{\sigma}_j$ needed for matching the wanted constraint Q to the instance declaration Q' and (b) the expressions $\overline{e}_i$ constructed by resolving the instance context constraints $\overline{Q}'_i$.

Example 1 λ_{TC} to $F_{\{\}}$. Typing the Example 1 program results in the following environments:

$$\Gamma_C = (==) : Eq\,a : a \to a \to Bool$$

$$\begin{aligned} P = {} & (D_1 : Eq\,Int).(==) \mapsto \bullet : primEqInt \\ & , (D_2 : \forall a, b. Eq\,a \Rightarrow Eq\,b \Rightarrow Eq\,(a, b)).(==) \mapsto a, b, \delta_1 : Eq\,a, \delta_2 : Eq\,b : \\ & \quad \lambda(x_1, y_1).\lambda(x_2, y_2).\ (\&\&)\,((==)\,x_1\,x_2)\,((==)\,y_1\,y_2) \end{aligned}$$

The Eq class straightforwardly gets stored in the class environment Γ_C. Instances are stored in the λ_{TC} program context P (containing the dictionary constructor, the corresponding constraint, the method implementation and the environment under which to interpret this expression). Storing the instance declaration for $Eq\,Int$ is clear-cut. The instance for tuples on the other hand is somewhat more complex, since it requires an instance context, containing the local constraints $Eq\,a$ and $Eq\,b$. These constraints are made explicit, that is, the corresponding dictionaries are required by the elaborated implementation.

Elaborating the $\boldsymbol{\lambda_{TC}}$ expression results in the following $\boldsymbol{F_{\{\}}}$ expression:

$$\begin{aligned}
&\textbf{let } \mathit{refl}\colon \forall a.\{(==) : a \to a \to \mathit{Bool}\} \to a \to \mathit{Bool} \\
&\qquad = \Lambda a.\lambda\delta_3 : \{(==) : a \to a \to \mathit{Bool}\}.\lambda x : a.\delta_3.(==)\, x\, x \\
&\textbf{in let } \mathit{main}\colon \mathit{Bool} \\
&\qquad\qquad = \mathit{refl}\,(\mathit{Int}, \mathit{Int}) \\
&\qquad\qquad\quad (\Lambda a.\Lambda b.\lambda\delta_4 : \{(==) : a \to a \to \mathit{Bool}\}.\lambda\delta_5 : \{(==) : b \to b \to \mathit{Bool}\}. \\
&\qquad\qquad\qquad \{(==) = \lambda(x_1, y_1) : (a, b).\lambda(x_2, y_2) : (a, b). \\
&\qquad\qquad\qquad\quad (\&\&)\,(\delta_4.(==)\, x_1\, x_2)\,(\delta_5.(==)\, y_1\, y_2)\}) \\
&\qquad\qquad\quad \mathit{Int}\ \mathit{Int}\ \{(==) = \mathit{primEqInt}\}\ \{(==) = \mathit{primEqInt}\}\,(5, 42) \\
&\textbf{in } \mathit{main}
\end{aligned}$$

Note that the $\mathit{Eq}\ a$ constraint is made explicit in the implementation of refl, by abstracting over the constraint (elaborated to $\boldsymbol{F_{\{\}}}$ as the record type $\{(==) : a \to a \to \mathit{Bool}\}$, which stores the method name and its corresponding type) with the use of the record variable δ_3. When this function is called in main, both the type and the dictionary variable are instantiated. The latter is performed by (recursively) constructing a dictionary expression, using the type class resolution mechanism, as explained above in Section 4.1.

Example 2 $\boldsymbol{\lambda_{TC}}$ to $\boldsymbol{F_{\{\}}}$. Below is the environment generated by typing the Example 2 program (including the Section 2.2 extension), which features superclasses.

$$\begin{aligned}
\Gamma_C &= \mathit{base} : \mathit{Base}\ a : a \to \mathit{Bool} \\
&\ , \mathit{sub}_1 : \mathit{Base}\ a \Rightarrow \mathit{Sub}_1\ a : a \to \mathit{Bool} \\
&\ , \mathit{sub}_2 : \mathit{Base}\ a \Rightarrow \mathit{Sub}_2\ a : a \to \mathit{Bool}
\end{aligned}$$

The class environment Γ_C contains three classes, two of which have superclasses. However, since the example does not contain any instance declarations, the resulting program context P is empty.

For space reasons, we focus solely on elaborating `test2`, which results in the following $\boldsymbol{F_{\{\}}}$ expression:

$$\begin{aligned}
&\textbf{let } \mathit{test}_2\colon \forall a.\{\mathit{base} : a \to \mathit{Bool}\} \to \{\mathit{sub}_1 : a \to \mathit{Bool}\} \\
&\qquad\qquad \to \{\mathit{base} : a \to \mathit{Bool}\} \to \{\mathit{sub}_2 : a \to \mathit{Bool}\} \to a \to \mathit{Bool} \\
&\qquad = \Lambda a.\lambda\delta_1 : \{\mathit{base} : a \to \mathit{Bool}\}.\lambda\delta_2 : \{\mathit{sub}_1 : a \to \mathit{Bool}\}. \\
&\qquad\quad \lambda\delta_3 : \{\mathit{base} : a \to \mathit{Bool}\}.\lambda\delta_4 : \{\mathit{sub}_2 : a \to \mathit{Bool}\}.\lambda x : a.\ \delta_1.\mathit{base}\ x \\
&\textbf{in } \mathit{True}
\end{aligned}$$

Note that the $\boldsymbol{\lambda_{TC}}$ expression requires two local constraints: $\mathit{Sub}_1\ a$ and $\mathit{Sub}_2\ a$. However, after flattening the superclasses and adding them to the local constraints, the elaborated $\boldsymbol{F_{\{\}}}$ expression requires (the elaborated form of) the $\mathit{Base}\ a$, $\mathit{Sub}_1\ a$, $\mathit{Base}\ a$ and $\mathit{Sub}_2\ a$ constraints. Notice the duplicate $\mathit{Base}\ a$ entry. Either of these two entries can be used for calling the method base. We have arbitrarily selected the first here. The next section proves that both options are equivalent and can be used interchangeably.

5 COHERENCE

This section provides an outline for our coherence proof, and defines the required notions. We first provide a definition of *contextual equivalence* [Morris Jr 1969], which captures that two expressions have the same meaning.

5.1 Contextual Equivalence

In order to formally discuss the concept of contextual equivalence, we first define the notion of an *expression context.*

Expression Contexts. An expression context M is an expression with a single hole, for which another expression e can be filled in, denoted as $M[e]$. The syntax can be found in Figure 2.

The typing judgment for an expression context M is of the form $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M$. This means that for any expression e such that $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e$, we have $P; \Gamma_C; \Gamma' \vdash_{tm} M[e] \Rightarrow \tau' \rightsquigarrow e'$. Following regular $\boldsymbol{\lambda_{TC}}$ term typing, context typing spans all combinations of type inference and checking mode: $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M$, $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M$ and $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M$.

For example, the simplest expression context is the empty context $[\bullet] : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma \Rightarrow \tau) \rightsquigarrow [\bullet]$.

Now we can formally define contextual equivalence. Note that the small step operational semantics can be found in Section E.4 of the appendix. The environment and type well-formedness judgments can be found in Sections B.4 and B.1 of the appendix respectively.

Definition 1 (Kleene Equivalence).
Two $F_{\{\}}$ *expressions* e_1 *and* e_2 *are Kleene equivalent, written* $e_1 \simeq e_2$*,*
if there exists a value v *such that* $e_1 \longrightarrow^* v$*, and* $e_2 \longrightarrow^* v$*.*

Definition 2 (Contextual Equivalence).
Two expressions $\Gamma \vdash_{tm} e_1 : \sigma$ *and* $\Gamma \vdash_{tm} e_2 : \sigma$*,*
where $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C; \Gamma \vdash_{ty} \tau \rightsquigarrow \sigma$*,*
are contextually equivalent, written $P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$*,*
if forall $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1$
and $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2$
implies $M_1[e_1] \simeq M_2[e_2]$*.*

The definition is adapted from Harper [2016, Chapter 46]. Intuitively, contextual equivalence means that two open expressions are observationally indistinguishable, when used in any program that instantiates the expressions' free variables.

5.2 Coherence

We can now make a first attempt to prove that different translations of the same source program are contextually equivalent. The program typing judgment can be found in Section B.2 of the appendix.

Theorem 1 (Coherence).
If $\bullet; \bullet \vdash_{pgm} pgm : \tau; P_1; \Gamma_{C1} \rightsquigarrow e_1$ *and* $\bullet; \bullet \vdash_{pgm} pgm : \tau; P_2; \Gamma_{C2} \rightsquigarrow e_2$
then $\Gamma_{C1} = \Gamma_{C2}, P_1 = P_2$ *and* $P_1; \Gamma_{C1}; \bullet \vdash e_1 \simeq_{ctx} e_2 : \tau$*.*

We first set out to prove the simpler variant, which only considers expressions [2].

Theorem 2 (Expression Coherence).
If $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e_1$ *and* $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e_2$
then $P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$*.*

The main requirement which makes type class resolution coherent is that type class instances do not overlap. However, since $F_{\{\}}$ uses records to encode dictionaries, the $F_{\{\}}$ language does not enforce this crucial uniqueness property. In order to prove Theorem 1, we introduce an additional intermediate language F_D, which captures the invariant that type class instances do not overlap, and makes it explicit.

6 INTERMEDIATE LANGUAGE F_D

This section presents our intermediate language F_D. The language is modeled with three main goals in mind: (a) F_D should explicitly pass type class dictionaries, which are implicit in λ_{TC}; (b) the F_D type system should capture the uniqueness of dictionaries, thus enforcing the elaboration from λ_{TC} to preserve full abstraction; and (c) F_D expressions should elaborate straightforwardly and deterministically to the target language $F_{\{\}}$ (System F with records, see Section 4).

To this end, F_D is an extension of System F, with built-in support for dictionaries. These dictionaries differ from those commonly used in Haskell compilers in that they are special constants rather than a record of method implementations. A separate global map Σ from dictionaries to method implementations gives access to the latter. Note that this setup does not allow programs to introduce new (and possibly overlapping) dictionaries dynamically. All dictionaries have to be provided upfront, where uniqueness is easily enforced.

Syntax. Figure 8 shows selected syntax of F_D; the basic System F constructs are omitted and can be found in Section A.2 of the appendix.

F_D introduces a new syntactic sort of dictionaries d that can either be a dictionary variable δ or a dictionary constructor D. A dictionary constructor has a number (possibly zero) of type and dictionary parameters and always appears in fully-applied form. Each constructor corresponds to a unique instance declaration, and is mapped to its method implementation by the global environment Σ.

Expressions have explicit application and abstraction forms for dictionaries. Furthermore, similarly to $F_{\{\}}$, method names can no longer be used on their own. Instead, they have to be applied explicitly to a dictionary, in the form $d.m$.

F_D types σ or τ are identical to the well-known System F types, with the addition of a special function type $Q \Rightarrow \sigma$ for dictionary abstractions.

Similarly to λ_{TC}, F_D features two global and a single local environment Γ. The latter is similar to the λ_{TC} typing environment Γ. However, there are notable differences between the global environments. The F_D class environment Γ_C does not contain any superclass information. The reason for this is that, as previously mentioned in Section 3, superclass constraints in the source language λ_{TC} are flattened into local constraints, and stored in the typing environment Γ. The analog to the λ_{TC} program context P is the F_D method environment Σ, storing information about all dictionary constructors D. Each constructor corresponds to a unique instance declaration, and stores the accompanying method implementations.

[2] Theorem 2 also has a type checking mode counterpart, which has been omitted here for space reasons.

$$\begin{array}{llll}
\sigma, \tau & ::= & \ldots \mid Q \Rightarrow \sigma & \textit{type} \\
Q & ::= & TC\,\sigma & \textit{class constraint} \\
C & ::= & \forall \overline{a}.\overline{Q} \Rightarrow Q & \textit{constraint} \\
d & ::= & \delta \mid D\,\overline{\sigma}\,\overline{d} & \textit{dictionary} \\
dv & ::= & D\,\overline{\sigma}\,\overline{dv} & \textit{dictionary value} \\
e & ::= & \ldots \mid \lambda\delta : Q.e \mid e\,d \mid d.m & \textit{expression} \\
\Gamma & ::= & \bullet \mid \Gamma, x : \sigma \mid \Gamma, a \mid \Gamma, \delta : Q & \textit{typing environment} \\
\Gamma_C & ::= & \bullet \mid \Gamma_C, m : TC\,a : \sigma & \textit{class environment} \\
\Sigma & ::= & \bullet \mid \Sigma, (D : C).m \mapsto e & \textit{method environment}
\end{array}$$

Fig. 8. $\boldsymbol{F}_{\mathbf{D}}$, selected syntax

$\boxed{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e}$ *(F_D Term Typing)*

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \rightsquigarrow e \qquad (m : TC\,a : \sigma') \in \Gamma_C}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma' \rightsquigarrow e.m}\ \textsc{iTm-method}$$

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \rightsquigarrow e_1 \qquad \Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow e_2}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\,d : \sigma \rightsquigarrow e_1\,e_2}\ \textsc{iTm-constrE}$$

$$\frac{\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e \qquad \Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma \qquad e' = \lambda\delta : \sigma.e}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma \rightsquigarrow e'}\ \textsc{iTm-constrI}$$

$\boxed{\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma}$ *(Constr. Well-Formedness)*

$$\frac{\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \qquad \Gamma_C = \Gamma_{C1}, m : TC\,a : \sigma', \Gamma_{C2} \qquad \Gamma_{C1}; \bullet, a \vdash_{ty} \sigma' \rightsquigarrow \sigma'}{\Gamma_C; \Gamma \vdash_Q TC\,\sigma \rightsquigarrow [\sigma/a]\{m : \sigma'\}}\ \textsc{iQ-TC}$$

$\boxed{\Sigma \vdash e \longrightarrow e'}$ *(F_D Evaluation)*

$$\frac{\Sigma \vdash e \longrightarrow e'}{\Sigma \vdash e\,d \longrightarrow e'\,d}\ \textsc{iEval-DApp}$$

$$\frac{}{\Sigma \vdash (\lambda\delta : Q.e)\,d \longrightarrow [d/\delta]e}\ \textsc{iEval-DAppAbs}$$

$$\frac{(D : C).m \mapsto e \in \Sigma}{\Sigma \vdash (D\,\overline{\sigma}\,\overline{d}).m \longrightarrow e\,\overline{\sigma}\,\overline{d}}\ \textsc{iEval-method}$$

Fig. 9. $\boldsymbol{F}_{\mathbf{D}}$ typing and operational semantics, selected rules

$\boxed{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}$ *(F_D Environment Well-Formedness)*

$$\frac{\begin{array}{c}\mathbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma) \\ \Gamma_C; \bullet \vdash_C \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma \qquad (m : TC\,a : \sigma') \in \Gamma_C \qquad \Sigma; \Gamma_C; \bullet \vdash_{tm} e : \forall \overline{a}_j.\overline{Q}_i \Rightarrow [\sigma/a]\sigma' \rightsquigarrow e \\ D \notin \mathbf{dom}(\Sigma) \qquad (D' : \forall \overline{a}'_m.\overline{Q}'_n \Rightarrow TC\,\sigma').m' \mapsto e' \notin \Sigma \qquad \mathbf{where}[\overline{\sigma}_j/\overline{a}_j]\sigma = [\overline{\sigma}'_m/\overline{a}'_m]\sigma' \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma\end{array}}{\vdash_{ctx} \Sigma, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma).m \mapsto e; \Gamma_C; \Gamma}\ \textsc{iCtx-MEnv}$$

Fig. 10. $\boldsymbol{F}_{\mathbf{D}}$ environment well-formedness, selected rules

Typing. Figure 9 (left-hand side) shows selected typing rules for $\boldsymbol{F}_{\mathbf{D}}$ expressions. The red parts can be safely ignored for now, as they will be explained in detail in Section 6.2. The judgment $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$ expresses that the $\boldsymbol{F}_{\mathbf{D}}$ term e of type σ is well-typed under environments Σ, Γ_C and Γ. As shown by rule iTm-method, the type of a method variable applied to a dictionary is simply the corresponding method type (as stored in the static class environment), where the type variable has been substituted for the corresponding dictionary type.

Figure 11 shows the typing rules for dictionaries. The relation $\Sigma; \Gamma_C; \Gamma \vdash_d d : Q$ denotes that dictionary d of dictionary type Q is well-formed under environments Σ, Γ_C and Γ. Similarly to regular term variables x (iTm-var), the type of a dictionary variable δ (D-var) is obtained from the typing environment Γ. The type of a dictionary constructor D (D-con), on the other hand, is obtained by finding the corresponding entry in the method environment Σ and substituting any types $\overline{\sigma}_j$ applied to it in the corresponding class constraint $TC\,\sigma_q$. All applied dictionaries $\overline{d}_i$ have to be well-typed with the corresponding constraint. Finally, the corresponding method implementation has to be well-typed in the reduced method environment Σ_1, which only contains the instances declared before D. As mentioned in Section 3, this reduced environment disallows recursive method implementations, as this would significantly clutter the coherence proof while, as a feature, recursion is completely orthogonal to the desired property.

$\boxed{\Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow e}$ *(Dictionary Typing)*

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_d \delta : Q \rightsquigarrow \delta} \text{ D-VAR}$$

$$\frac{\begin{array}{c}(D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow TC\, \sigma_q).m \mapsto \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e \in \Sigma \\ (\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma'_i\ , \forall i) \qquad (\Gamma_C; \Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j\ , \forall j) \qquad \Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e : [\sigma_q/a]\sigma_m \rightsquigarrow e \\ (\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i \rightsquigarrow e_i\ , \forall i) \qquad \Sigma = \Sigma_1, (D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow TC\, \sigma_q).m \mapsto \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e, \Sigma_2\end{array}}{\Sigma; \Gamma_C; \Gamma \vdash_d D\, \overline{\sigma}_j\, \overline{d}_i : TC\, [\overline{\sigma}_j/\overline{a}_j]\sigma_q \rightsquigarrow (\Lambda \overline{a}_j . \lambda\, \overline{\delta_i : \sigma'_i}^{\,i} . \{m = e\})\, \overline{\sigma}_j\, \overline{e}_i} \text{ D-CON}$$

Fig. 11. F_D dictionary typing

Non-Overlapping Instances. The main requirement for achieving coherence of type class resolution, is that type class instances do not overlap. This requirement is common in Haskell and is for example enforced in GHC (though strongly discouraged, the `OverlappingInstances` pragma disables it). By storing all method implementations (with their corresponding instances) in a single environment Σ, this invariant can easily be made explicit.

Figure 10 shows the environment well-formedness condition $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$ for the method environment. Besides stating well-scopedness, it denotes that the method environment Σ cannot contain a second instance, for which the head of the constraint overlaps with $TC\, \sigma$, up to renaming. This key property will be exploited in our coherence proof.

Operational Semantics. As F_D is an extension of System F, its call-by-name operational semantics are mostly standard. The non-standard rules can be found in Figure 9 (bottom right), where $\Sigma \vdash e \longrightarrow e'$ denotes expression e evaluating to e' in a single step, under method environment Σ.

The evaluation rules for dictionary application (iEval-DApp and iEval-DAppAbs) are identical to those for term and type application. More interesting, however, is the evaluation for methods (iEval-method). A method name applied to a dictionary evaluates in one step to the method implementation, as stored in the environment Σ.

Metatheory. F_D is type safe. That is, the common progress and preservation properties hold:

Theorem 3 (Progress).
If $\Sigma; \bullet; \bullet \vdash_{tm} e : \sigma$*, then either* e *is a value, or there exists* e' *such that* $\Sigma \vdash e \longrightarrow e'$.

Theorem 4 (Preservation).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$*, and* $\Sigma \vdash e \longrightarrow e'$*, then* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e' : \sigma$.

Analogously to λ_{TC}, F_D rejects recursive expressions (including mutual recursion and recursive methods). This allows for a normalizing language, that is, any well-typed expression evaluates to a value, after a finite number of steps. Note that since the small step operational semantics are deterministic, normalization implies strong normalization.

Theorem 5 (Strong Normalization).
If $\Sigma; \Gamma_C; \bullet \vdash_{tm} e : \sigma$ *then all possible evaluation derivations for* e *terminate* : $\exists v : \Sigma \vdash e \longrightarrow^* v$.

The proof follows the familiar structure for proving normalization using logical relations, as presented by Ahmed [Ahmed 2015], and can be found in Section K.4 of the appendix.

6.1 Elaboration from λ_{TC} to F_D

The green aspects in Figure 12 denote the elaboration of λ_{TC} terms to F_D. Similarly to the elaboration from λ_{TC} to $F_{\{\}}$, we have adopted the convention that any green F_D types or constraints are the elaborated forms of their identically named blue λ_{TC} counterparts. This elaboration works analogously to the elaboration from λ_{TC} to $F_{\{\}}$, as shown in Figure 3. The full set of rules can be found in Section C.2 of the appendix.

The only notable case is sTm-check-meth, where the entailment relation for solving the type class constraint $TC\, \tau$ now results in a dictionary d. As explained at the start of Section 6, unlike $F_{\{\}}$, F_D differentiates syntactically between dictionaries and normal expressions.

Type class resolution (Figure 13) of a λ_{TC} constraint Q results in a F_D dictionary d. When using a locally available constraint to resolve the wanted constraint (sEntail-local), the corresponding dictionary variable δ is returned. On the other hand, when resolving using a global instance declaration (sEntail-inst), a dictionary is constructed by taking the corresponding constructor D and applying (a) the types $\overline{\sigma}_j$ needed for matching the wanted constraint to the instance declaration and (b) the dictionaries $\overline{d}_i$, constructed by resolving the instance context constraints.

$\boxed{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e}$ *(Source Term Checking)*

$$\frac{\begin{array}{c}(m : \overline{Q}'_k \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau') \in \Gamma_C \\ \textbf{unambig}(\forall \overline{a}_j, a.\overline{Q}_i \Rightarrow \tau') \qquad P;\Gamma_C;\Gamma \vDash^M TC\,\tau \rightsquigarrow d \qquad \Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \\ (P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j][\tau/a]Q_i \rightsquigarrow d_i\ ,\ \forall i) \qquad (\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j\ ,\ \forall j) \qquad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma\end{array}}{P;\Gamma_C;\Gamma \vdash^M_{tm} m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow d.m\,\overline{\tau}_j\,\overline{d}_i}\ \text{sTm-check-meth}$$

Fig. 12. λ_{TC} typing with elaboration to F_D, selected rules

$\boxed{P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow d}$ *(Constraint Entailment)*

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma}{P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow \delta}\ \text{sEntail-local}$$

$$\frac{\begin{array}{c}P = P_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto \Gamma' : e, P_2 \qquad \Gamma' = \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h \\ Q = [\overline{\tau}_j/\overline{a}_j]Q' \qquad (\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j\ ,\ \forall j) \qquad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma \qquad (P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i\ ,\ \forall i)\end{array}}{P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow D\,\overline{\sigma}_j\,\overline{d}_i}\ \text{sEntail-inst}$$

Fig. 13. λ_{TC} constraint entailment with elaboration to F_D

Metatheory. We discuss the coherence of the elaboration from λ_{TC} to F_D in detail in Section 7, and mention here that it is type preserving:

Theorem 6 (Typing Preservation - Expressions).
If $P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e$, *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$, *and* $\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$,
then $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma$.

The same theorem holds for check mode, but is omitted for space reasons. The full proofs can be found in Section J.3 of the appendix.

Example 1 λ_{TC} ***to*** F_D. Elaborating the λ_{TC} environments that originate from Example 1, results in the following F_D environments:

$$\Gamma_C = (==) : Eq\ a : a \rightarrow a \rightarrow Bool$$

$$\begin{aligned}\Sigma = {} & (D_1 : Eq\,Int).(==) \mapsto primEqInt \\ & , (D_2 : \forall a, b.Eq\ a \Rightarrow Eq\ b \Rightarrow Eq\,(a, b)).(==) \mapsto \\ & \quad \Lambda a.\Lambda b.\lambda\delta_1 : Eq\ a.\lambda\delta_2 : Eq\ b. \\ & \quad \lambda(x_1, y_1) : (a, b).\lambda(x_2, y_2) : (a, b).(\&\&)\,(\delta_1.(==)\,x_1\,x_2)\,(\delta_2.(==)\,y_1\,y_2)\end{aligned}$$

Note that both the class environment Γ_C and the program context Σ are largely direct translations of their λ_{TC} counterparts. One notable difference is the fact that the environment Γ under which to interpret the λ_{TC} method implementation is now explicitly abstracted over in the F_D method implementation. Consider for instance the case of D_2, where the variables a, b, δ_1 and δ_2, which in λ_{TC} are implicitly provided by the typing environment, are now explicit in the term level.

Elaborating the λ_{TC} expression results in the following F_D expression:

$$\begin{aligned}& \textbf{let}\ \mathit{refl} : \forall a.Eq\ a \Rightarrow a \rightarrow Bool \\ & \qquad = \Lambda a.\lambda\delta_3 : Eq\ a.\lambda x : a.\delta_3.(==)\,x\,x \\ & \textbf{in let}\ main : Bool \\ & \qquad = \mathit{refl}\,(Int, Int)\,(D_2\,Int\,Int\,D_1\,D_1)\,(5, 42) \\ & \textbf{in}\ main\end{aligned}$$

Unlike the corresponding $F_{\{\}}$ expression, shown in Section 4.1, records storing the method types and implementations do not need to be passed around explicitly. In F_D, they are replaced by class constraints and dictionaries, respectively. The construction of these dictionaries through type class resolution is shown in Figure 13.

Example 2 λ_{TC} ***to*** F_D. Elaborating Example 2, including the extension from Section 2.2, results in the following F_D class environment (since no instance declarations exist, the method environment Σ remains empty):

$$\begin{aligned}\Gamma_C = \ & base : Base\ a : a \to Bool\\ , \ & sub_1 : Sub_1\ a : a \to Bool\\ , \ & sub_2 : Sub_2\ a : a \to Bool\end{aligned}$$

The F_D class environment no longer needs to store superclasses, as these are all flattened into additional local constraints during elaboration.

Similarly to Section 4.1, we focus solely on elaborating `test2`, which results in the following F_D expression:

$$\begin{aligned}&\textbf{let}\ test_2 \colon \forall a.Base\ a \Rightarrow Sub_1\ a \Rightarrow Base\ a \Rightarrow Sub_2\ a \Rightarrow a \to Bool\\ &\qquad = \Lambda a.\lambda\delta_1 : Base\ a.\lambda\delta_2 : Sub_1\ a.\lambda\delta_3 : Base\ a.\lambda\delta_4 : Sub_2\ a.\\ &\qquad\quad \lambda x : a.\delta_1.base\ x\\ &\textbf{in}\ True\end{aligned}$$

The only difference with the $F_{\{\}}$ elaboration is that we now use class constraints instead of passing around a record type (storing the method types).

6.2 Elaboration from F_D to $F_{\{\}}$

As both F_D and $F_{\{\}}$ are extensions of System F, the elaboration from former to latter is mostly trivial, leaving common features unchanged. The mapping of F_D dictionaries into $F_{\{\}}$ records, however, is non-trivial. Briefly, dictionary types are elaborated into record types, as shown in Figure 9 (top right), and dictionaries into record expressions, possibly nested within type and term abstractions and applications, as shown in Figure 11.

In particular, a dictionary type, TC, which corresponds to a unique entry $(m : TC\ a : \sigma')$ in the class environment Γ_C, elaborates to a record type whose field has the same name as the dictionary type's method name, m, and the type of that field is determined by the elaboration of σ'. A $TC\ \sigma$ dictionary elaborates to a record expression which is surrounded, firstly, by abstractions over type and term variables that arise from the method type's class constraints and, secondly, by type and term applications that properly instantiate those abstractions.

Metatheory. The following theorems confirm that the F_D-to-$F_{\{\}}$ elaboration is indeed appropriate.

The first theorem states that a well-typed F_D expression always elaborates to a $F_{\{\}}$ expression that is also well-typed in the translated context.

Theorem 7 (Type Preservation).
If $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$,
then there are unique Γ *and* σ *such that* $\Gamma \vdash_{tm} e : \sigma$,
where $\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$.

Secondly, and more importantly, the dynamic semantics is also preserved by the elaboration.

Theorem 8 (Semantic Preservation).
If $\Sigma;\Gamma_C;\bullet \vdash_{tm} e : \sigma \rightsquigarrow e$ *and* $\Sigma \vdash e \longrightarrow^* v$
then there exists a v *such that* $\Sigma;\Gamma_C;\bullet \vdash_{tm} v : \sigma \rightsquigarrow v$ *and* $e \simeq v$.

Thirdly, the elaboration is entirely deterministic.

Theorem 9 (Determinism).
If $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma \rightsquigarrow e_1$ *and* $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma \rightsquigarrow e_2$, *then* $e_1 = e_2$.

6.3 Elaboration Decomposition

An elaboration from λ_{TC} to $F_{\{\}}$ can always be decomposed into two elaborations through F_D. This intuition is formalized in Theorems 10 and 11 respectively.

Theorem 10 (Elaboration Equivalence - Expressions).
If $P;\Gamma_C;\Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e$ *and* $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$
then $P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e$ *and* $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$
where $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$.

Theorem 11 (Elaboration Equivalence - Dictionaries).
If $P;\Gamma_C;\Gamma \vDash Q \rightsquigarrow e$ *and* $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$
then $P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow d$ *and* $\Sigma;\Gamma_C;\Gamma \vdash_d d : Q \rightsquigarrow e$
where $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$.

Theorem 10 also has a type checking mode counterpart, which has been omitted for space reasons. The full proofs can be found in Section L of the appendix.

7 COHERENCE REVISITED

As mentioned previously in Section 5, the invariant that type class instances do not overlap is crucial in proving Theorem 1. This uniqueness property is made explicit in F_D. Our proof thus proceeds by elaborating the λ_{TC} expression to two possibly different F_D expressions and subsequently elaborating these F_D expressions to $F_{\{\}}$ expressions. Consequently, the proof is split in two main steps. The first part is the most involved, where we use a technique based on logical relations to prove that any two F_D expressions originating from the same λ_{TC} expression are contextually equivalent. The second part proves that the elaboration from F_D to $F_{\{\}}$ is contextual equivalence preserving. This step follows straightforwardly from the fact that the F_D-to-$F_{\{\}}$ elaboration is deterministic. Together, these prove that the elaboration from λ_{TC} to $F_{\{\}}$ through F_D is coherent. Theorem 2 follows from this result, together with Theorem 10.

The remainder of this section explains the techniques we used to prove Theorem 1 in detail.

7.1 Coherent Elaboration from λ_{TC} to F_D

7.1.1 ***Logical Relations.*** Logical relations [Plotkin 1973; Statman 1985; Tait 1967] are key to proving contextual equivalence. In our type system, the logical relation for expressions is mostly standard, though the relation for dictionaries is novel.

Dictionaries. The logical relation over two open dictionaries is defined by means of an auxiliary relation on closed dictionaries. We define this value relation for closed dictionaries as follows. Note that from now on, we will omit elaborations when they are entirely irrelevant. The appendix uses the same convention.

Definition 3 (Value Relation for Dictionaries).
The dictionary values $D\,\overline{\sigma}_j\,\overline{dv}_{1\,i}$ *and* $D\,\overline{\sigma}_j\,\overline{dv}_{2\,i}$ *are in the value relation, defined as:*

$$(\Sigma_1 : D\,\overline{\sigma}_j\,\overline{dv}_{1\,i}, \Sigma_2 : D\,\overline{\sigma}_j\,\overline{dv}_{2\,i}) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \triangleq ((\Sigma_1 : dv_{1\,i}, \Sigma_2 : dv_{2\,i}) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j]Q_i]\!]_R^{\Gamma_C}\ ,\ \forall i)$$
$$\wedge\ \Sigma_1;\Gamma_C;\bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv}_{1\,i} : R(Q)\ \wedge\ \Sigma_2;\Gamma_C;\bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv}_{2\,i} : R(Q),$$
$$\textit{where } (D : \forall\overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto e_1 \in \Sigma_1 \wedge Q = [\overline{\sigma}_j/\overline{a}_j]Q'$$

The value relation is indexed by the dictionary type Q. We require both dictionaries to be well-typed, and their dictionary arguments to be in the value relation as well. The relation has four additional parameters: the contexts Σ_1 and Σ_2, which annotate the dictionaries, the class environment Γ_C, used in the well-typing condition, and the type substitution R.

In order to define logical equivalence between open dictionaries, we substitute all free variables with closed terms, thus reducing them to closed dictionary values. Three kinds of variables exist (term variables x, type variables a and dictionary variables δ). This results in three separate semantic interpretations of the typing context Γ. The type substitution $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$ maps all type variables $a \in \Gamma$ onto closed types. $\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ maps each term variable $x \in \Gamma$ to two expressions e_1 and e_2 that are in the expression value relation (see Definition 5), and $\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ maps each dictionary variable $\delta \in \Gamma$ to two logically related dictionary values. We use ϕ_1 and ϕ_2 to denote the substitution for the first and second expression, respectively.

Definition 4 (Logical Equivalence for Dictionaries).
Two dictionaries d_1 *and* d_2 *are logically equivalent, defined as:*

$$\Gamma_C;\Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : Q \triangleq\ \forall R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C},\ \phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C},\ \gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} :$$
$$(\Sigma_1 : \gamma_1(\phi_1(R(d_1))), \Sigma_2 : \gamma_2(\phi_2(R(d_2)))) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C}$$

Two dictionaries d_1 and d_2 are logically equivalent if any substitution of their free variables (with related expressions / dictionaries) results in related dictionary values.

Expressions. The value relation for expressions is mostly standard, with two notable deviations. Firstly, the relation is defined over two different method environments Σ_1 and Σ_2. Hence, both expressions are annotated with their respective environment. Secondly, the dictionary abstraction case is novel.

Definition 5 (Value Relation for Expressions).
Two values v_1 *and* v_2 *are in the value relation, defined as:*

$(\Sigma_1 : True, \Sigma_2 : True) \in \mathcal{V}[\![Bool]\!]^{\Gamma_C}_R$

$(\Sigma_1 : False, \Sigma_2 : False) \in \mathcal{V}[\![Bool]\!]^{\Gamma_C}_R$

$$(\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![a]\!]^{\Gamma_C}_R \triangleq$$
$$(a \mapsto (\sigma, r)) \in R \;\wedge\; (v_1, v_2) \in r \;\wedge\; \Sigma_1; \Gamma_C; \bullet \vdash_{tm} v_1 : \sigma \;\wedge\; \Sigma_2; \Gamma_C; \bullet \vdash_{tm} v_2 : \sigma$$

$$(\Sigma_1 : \lambda x : \sigma_1.e_1, \Sigma_2 : \lambda x : \sigma_1.e_2) \in \mathcal{V}[\![\sigma_1 \to \sigma_2]\!]^{\Gamma_C}_R \triangleq$$
$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda x : \sigma.e_1 : R(\sigma_1 \to \sigma_2) \;\wedge\; \Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda x : \sigma.e_2 : R(\sigma_1 \to \sigma_2)$$
$$\wedge\, \forall (\Sigma_1 : e_3, \Sigma_2 : e_4) \in \mathcal{E}[\![\sigma_1]\!]^{\Gamma_C}_R : (\Sigma_1 : (\lambda x : \sigma.e_1)\, e_3, \Sigma_2 : (\lambda x : \sigma.e_2)\, e_4) \in \mathcal{E}[\![\sigma_2]\!]^{\Gamma_C}_R$$

$$(\Sigma_1 : \lambda \delta : Q.e_1, \Sigma_2 : \lambda \delta : Q.e_2) \in \mathcal{V}[\![Q \Rightarrow \sigma]\!]^{\Gamma_C}_R \triangleq$$
$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda \delta : Q.e_1 : R(Q \Rightarrow \sigma) \;\wedge\; \Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda \delta : Q.e_2 : R(Q \Rightarrow \sigma)$$
$$\wedge\, \forall (\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_R : (\Sigma_1 : (\lambda \delta : Q.e_1)\, dv_1, \Sigma_2 : (\lambda \delta : Q.e_2)\, dv_2) \in \mathcal{E}[\![\sigma]\!]^{\Gamma_C}_R$$

$$(\Sigma_1 : \Lambda a.e_1, \Sigma_2 : \Lambda a.e_2) \in \mathcal{V}[\![\forall a.\sigma]\!]^{\Gamma_C}_R \triangleq$$
$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda a.e_1 : R(\forall a.\sigma) \;\wedge\; \Sigma_2; \Gamma_C; \bullet \vdash_{tm} \Lambda a.e_2 : R(\forall a.\sigma)$$
$$\wedge\, \forall \sigma', \forall r \in Rel[\sigma'] :\; \Gamma_C; \bullet \vdash_{ty} \sigma' \Rightarrow (\Sigma_1 : (\Lambda a.e_1)\, \sigma', \Sigma_2 : (\Lambda a.e_2)\, \sigma') \in \mathcal{E}[\![\sigma]\!]^{\Gamma_C}_{R, a \mapsto (\sigma', r)}$$

Consider the interesting case of dictionary abstraction. The relation requires the terms to be well-typed, and the applications for all related input dictionaries to be in the expression relation $\mathcal{E}$. The definition of this $\mathcal{E}$ relation is as follows:

Definition 6 (Expression Relation).
Two expressions e_1 *and* e_2 *are in the expression relation, defined as:*

$$(\Sigma_1 : e_1, \Sigma_2 : e_2) \in \mathcal{E}[\![\sigma]\!]^{\Gamma_C}_R \triangleq \Sigma_1; \Gamma_C; \bullet \vdash_{tm} e_1 : R(\sigma) \;\wedge \Sigma_2; \Gamma_C; \bullet \vdash_{tm} e_2 : R(\sigma)$$
$$\wedge\, \exists v_1, v_2, \Sigma_1 \vdash e_1 \longrightarrow^* v_1, \Sigma_2 \vdash e_2 \longrightarrow^* v_2, (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]^{\Gamma_C}_R$$

In this definition, expressions are reduced to values and those values must be in the value relation. This is well-defined because F_D is strongly normalizing (Theorem 5).

Finally, we can give the definition of logical equivalence for open expressions:

Definition 7 (Logical Equivalence for Expressions).
Two expressions e_1 *and* e_2 *are logically equivalent, defined as:*

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma \triangleq \forall R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C},\; \phi \in \mathcal{G}[\![\Gamma]\!]^{\Sigma_1, \Sigma_2, \Gamma_C}_R,\; \gamma \in \mathcal{H}[\![\Gamma]\!]^{\Sigma_1, \Sigma_2, \Gamma_C}_R :$$
$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1))), \Sigma_2 : \gamma_2(\phi_2(R(e_2)))) \in \mathcal{E}[\![\sigma]\!]^{\Gamma_C}_R$$

We also provide a definition of logical equivalence for contexts:

Definition 8 (Logical Equivalence for Contexts).
Two contexts M_1 *and* M_2 *are logically equivalent, defined as:*

$$\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma' \Rightarrow \sigma') \triangleq$$
$$\forall e_1, e_2 : \Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma \;\Rightarrow\; \Gamma_C; \Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma'$$

7.1.2 ***Proof of*** λ_{TC}***-to-***F_D ***Coherence.*** With the above definitions we are ready to formally state the metatheory and establish the coherence theorems from λ_{TC} to F_D.

Design Principle of F_D***.*** We emphasize that F_D captures the intention of type class instances. Theorem 12 states that any two dictionary values for the same constraint are logically related:

Theorem 12 (Value Relation for Dictionary Values).
If $\Sigma_1; \Gamma_C; \bullet \vdash_d dv_1 : Q$ *and* $\Sigma_2; \Gamma_C; \bullet \vdash_d dv_2 : Q$ *and* $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$
then $(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_\bullet$.

Note that two environments Σ_1 and Σ_2 are logically equivalent under Γ_C, written $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$, when they contain the same dictionary constructors and the corresponding method implementations are logically equivalent. The full definition can be found in Section G.3 of the appendix.

Coherent Resolution. We now prove that constraint resolution is semantically coherent, that is, if multiple resolutions of the same constraint exist, they are logically equivalent.

Theorem 13 (Logical Coherence of Dictionary Resolution).
If $P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow d_1$ *and* $P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow d_2$
and $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_1;\Gamma_C;\Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_2;\Gamma_C;\Gamma$
then $\Gamma_C;\Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : Q$ *where* $\Gamma_C;\Gamma \vdash^M_Q Q \rightsquigarrow Q$.

Coherent Elaboration. Furthermore, in order to prove that the elaboration from λ_{TC} to F_{D} is coherent, we show that all elaborations of the same expression are logically equivalent [3].

Theorem 14 (Logical Coherence of Expression Elaboration).
If $P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e_1$ *and* $P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e_2$
and $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_1;\Gamma_C;\Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_2;\Gamma_C;\Gamma$
then $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$ *where* $\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$.

Contextual Equivalence. We prove that all logically equivalent expressions are contextually equivalent. Together with Theorem 14, this shows coherence of the λ_{TC}-to-F_{D} elaboration.

We first provide a formal definition of contextual equivalence for F_{D} expressions. Kleene equivalence for F_{D} is defined similarly to Definition 1 and can be found in Section I.1 of the appendix.

Definition 9 (Contextual Equivalence for F_{D} Expressions).
Two expressions $\Sigma_1;\Gamma_C;\Gamma \vdash_{tm} e_1 : \sigma$ *and* $\Sigma_2;\Gamma_C;\Gamma \vdash_{tm} e_2 : \sigma$,
are contextually equivalent, written $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$,
if forall $M_1 : (\Sigma_1;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma_1;\Gamma_C;\bullet \Rightarrow Bool)$
and forall $M_2 : (\Sigma_2;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma_2;\Gamma_C;\bullet \Rightarrow Bool)$
where $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\bullet \Rightarrow Bool)$,
we have that $\Sigma_1 : M_1[e_1] \simeq \Sigma_2 : M_2[e_2]$.

Theorem 15 (Logical Equivalence implies Contextual Equivalence).
If $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$ *then* $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$.

7.2 Deterministic Elaboration from F_{D} to $F_{\{\}}$

7.2.1 ***Contextual Equivalence.*** Similarly to expressions, the elaboration from a λ_{TC} context M to a $F_{\{\}}$ context M can always be decomposed into two elaborations, through a F_{D} context M. The syntax and typing judgments can be found in Sections A and F of the appendix, respectively.

We now formally define contextual equivalence for $F_{\{\}}$ expressions through F_{D} contexts.

Definition 10 (Contextual Equivalence in F_{D} Context).
Two expressions $\Gamma \vdash_{tm} e_1 : \sigma$ *and* $\Gamma \vdash_{tm} e_2 : \sigma$,
where $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C;\Gamma \vdash_{ty} \tau \rightsquigarrow \sigma$,
are contextually equivalent, written $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$,
if forall $M_1 : (\Sigma_1;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma_1;\Gamma_C;\bullet \Rightarrow Bool) \rightsquigarrow M_1$
and forall $M_2 : (\Sigma_2;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma_2;\Gamma_C;\bullet \Rightarrow Bool) \rightsquigarrow M_2$
where $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\bullet \Rightarrow Bool)$,
we have that $M_1[e_1] \simeq M_2[e_2]$,
where $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *and* $\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$.

7.2.2 ***Proof of F_{D}-to-$F_{\{\}}$ Coherence.*** We continue by proving that contextual equivalence is preserved by the elaboration from F_{D} to $F_{\{\}}$:

Theorem 16 (Elaboration preserves Contextual Equivalence).
If $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$
and $\Sigma_1;\Gamma_C;\Gamma \vdash_{tm} e_1 : \sigma \rightsquigarrow e_1$ *and* $\Sigma_2;\Gamma_C;\Gamma \vdash_{tm} e_2 : \sigma \rightsquigarrow e_2$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$
then $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$.

7.2.3 ***Proof of λ_{TC}-to-$F_{\{\}}$ Coherence.*** Finally, in order to link back to Theorem 2 (which has no notion of F_{D}), we prove that contextual equivalence with F_{D} contexts implies contextual equivalence with λ_{TC} contexts:

Theorem 17 (Contextual Equivalence in F_{D} implies Contextual Equivalence in λ_{TC}).
If $\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$
and $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_1;\Gamma_C;\Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_2;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$
then $P;\Gamma_C;\Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$.

[3] Theorem 14 also has a type checking mode counterpart, which has been omitted here for space reasons.

For clarity, we restate coherence Theorems 2 and 1:

Theorem 2 (Expression Coherence - Restated).
If $P;\Gamma_C;\Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e_1$ *and* $P;\Gamma_C;\Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e_2$
then $P;\Gamma_C;\Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$.

Theorem 2 follows by combining Theorems 10, 14, 15, 16 and 17.

Theorem 1 (Coherence - Restated).
If $\bullet;\bullet \vdash_{pgm} pgm : \tau; P_1; \Gamma_{C1} \rightsquigarrow e_1$ *and* $\bullet;\bullet \vdash_{pgm} pgm : \tau; P_2; \Gamma_{C2} \rightsquigarrow e_2$
then $\Gamma_{C1} = \Gamma_{C2}, P_1 = P_2$ *and* $P_1; \Gamma_{C1}; \bullet \vdash e_1 \simeq_{ctx} e_2 : \tau$.

Finally, we show that Theorem 1 follows from Theorem 2. The full proofs can be found in Section M of the appendix.

8 DISCUSSION OF POSSIBLE EXTENSIONS

As the goal of this work was to find a proof technique to formally establish coherence for type class resolution, a stripped down source calculus was employed in order not to clutter the proof. This section provides a brief discussion of extending our coherence proof to support several mainstream language features.

Ambiguous Type Schemes. As mentioned previously, our work is orthogonal to ambiguous type schemes, which have already been extensively studied by Jones [1993]. We believe our work and the proof by Jones can be combined, which would then relax the restriction of bidirectional type checking, and prove coherence for both ambiguous type schemes and type class resolution.

General Recursion. Recursion is an important feature, present in any real world programming language. It is important to note that, while $\lambda_{\mathbf{TC}}$ does not feature recursion on the expression level (as it does not affect the essence of the coherence proof), type class resolution itself is recursive. Dictionary values are constructed dynamically from a statically given set of dictionary constructors (one constructor per type class instance). The system can thus recursively generate an arbitrary number of dictionaries from a finite set of instances.

Our logical relations can be adapted to support general recursion, through well-known techniques, such as step indexing [Ahmed 2006]. While this results in a significantly longer and more cluttered proof, we do not anticipate any major complications.

Multi-Parameter Type Classes. Just like regular type class instances, multi-parameter instances (as supported by GHC) are subject to the no-overlap rule. Hence, they respect our main assumption. They may indeed give rise to more ambiguity, but this is the kind of ambiguity that is studied by Jones [1993], not the kind that shows up during resolution. Note that functional dependencies were originally introduced by Jones [2000] as a way to resolve the ambiguity caused by multi-parameter instances.

Dependent Types. Dependently typed languages, e.g., Agda [Devriese and Piessens 2011] and Idris [Brady 2013], include language features that are inspired by type classes. Proving resolution coherence in a dependently typed setting requires significant extension of our calculi, as dependent types collapse the term and type levels into a single level and thus enable more powerful type signatures for classes and instances. Furthermore, our logical relation needs to be extended to support dependent types [Bernardy et al. 2012] as well. Fortunately, the essence of our proof strategy still applies. That is, the intermediate language incorporates separate binding structures for dictionaries, and enforces the uniqueness of dictionaries. We thus believe a non-trivial extension of our proof methodology can be used to prove coherence for type class resolution in the setting of dependently typed languages.

Non-overlapping Instances. Our work is built on top of the assumption that type class instances do not overlap. This is enforced during the type checking of instance declarations, and made explicit in the intermediate language. Whether a constraint is entailed directly from an instance, through user provided constraints in a type annotation, or through local evidence, is not actually relevant, as all evidence ultimately has to originate from a non-overlapping instance declaration.

Therefore, our work can be extended to include features where the assumption holds true. This includes, among others, GADT's [Peyton Jones et al. 2006], implication constraints [Bottu et al. 2017], type constructors, higher kinded types and constraint kinds [Orchard and Schrijvers 2010], e.g., Bottu et al. [2017] informally discuss the coherence of implication constraints based on the same assumption. These features are all included in GHC.

Modules. Modules, as supported by GHC, pose an interesting challenge, as they are known to cause a form of ambiguity.[4] GHC does not statically check the uniqueness of instances across modules, thus indirectly allowing users to write overlapping instances, as long as no ambiguity arises during resolution. Adapting our global uniqueness assumption to accommodate this additional freedom remains an interesting challenge.

[4] http://blog.ezyang.com/2014/07/type-classes-confluence-coherence-global-uniqueness/

Laziness. The operational semantics of the F_D and $F_{\{\}}$ calculi in this work are given through standard call-by-name semantics, in order to approximate Haskell's laziness. The system can easily be adapted to either call-by-value or call-by-need, with little impact on the proofs.

It is important to note though, that while expressions are evaluated lazily, type class resolution itself is eager, and constructs the full dictionaries at compile time. This complicates supporting certain GHC features that rely on laziness, like cyclic and infinite dictionaries. They could be supported through loop detection and deferring the construction of dictionaries to runtime, but these would nonetheless pose a significant challenge.

9 RELATED WORK

Type Classes. Jones [1993, 1994] formally proves coherence for the framework of qualified types, which generalizes from type classes to arbitrary evidence-backed type constraints. He focuses on nondeterminism in the typing derivation, and assumes that resolution is coherent.

Morris [2014] presents an alternative, denotational semantics for type classes (without superclasses) that avoids elaboration and instead interprets qualified type schemes as the set of denotations of all its monomorphic instantiations that satisfy the qualifiers. The nondeterminism of resolution does not affect these semantics.

Kahl and Scheffczyk [2001] present named type class instances that are not used during resolution, but can be explicitly passed to functions. Nevertheless, they violate the uniqueness of instances, and give rise to incoherence of the form illustrated by our `discern` function in Section 2.3.

Unlike most other languages with type classes (such as Haskell, Mercury or PureScript) Coq [Sozeau and Oury 2008] does not enforce the non-overlapping instances condition. Consequently, coherence does not hold for type class resolution in Coq. The reason for this alternative design choice is twofold: (a) Since Coq's type system is more complex than that found in regular programming languages, it is not always possible to decide whether two instances overlap [Lampropoulos and Pierce 2018, Chapter 2: Typeclasses]. (b) Type class members in Coq are often proofs and, unlike for expressions, users are often indifferent to coherence in the presence of proofs (even though from a semantic point of view, Coq differentiates between them). This concept is known as "proof irrelevance" [Gilbert et al. 2019], that is, as long as at least one proof exists, the concrete choice between these proofs is irrelevant. Users can deal with this lack of coherence by either assigning priorities to overlapping instances, or by manually curating the instance database and locally removing specific instances.

Winant and Devriese [2018] introduce explicit dictionary application to the Haskell language, and prove coherence for this extended system. Their proof is parametric in the constraint entailment judgment and thus assumes that the constraint solver produces "canonical" evidence. They proceed by introducing a disjointness condition to explicitly applied dictionaries, in order to ensure that coherence is preserved by their extension. Our paper proves their aforementioned assumption, by establishing coherence for type class resolution.

Dreyer et al. [2007] blend ML modules with Haskell type class resolution. Unlike Haskell, they feature multiple global (or outer) scopes; instances within one such global scope must not overlap. Moreover, global instances are shadowed by those given through type signatures. While their language has been formalized, no formal proof of coherence is given.

Implicits. Cochis [Schrijvers et al. 2019] is a calculus with highly expressive implicit resolution, including local instances. It achieves coherence by imposing restrictions on the implicit context and enforcing a deterministic resolution process. This allows for a much simpler coherence proof.

OCaml's modular implicits [White et al. 2014] do not enforce uniqueness of "instances" but dynamically ensure coherence by rejecting programs where there are multiple possible resolution derivations. This approach has not been formalized yet.

Other. Reynolds [Reynolds 1991] introduced the notion of coherence in the context of the Forsythe language's intersection types; he proved coherence directly in terms of the denotational semantics of the language.

In contrast, Bi et al. [2018, 2019] consider a setting where subtyping for intersection types is elaborated to coercions. Inspired by Biernacki and Polesiuk [2015], they use an approach based on contextual equivalence and logical relations, which has inspired us in turn. However, they do not create an intermediate language to avoid the problem of a more expressive target language. This leads to a notion of contextual equivalence that straddles two languages and complicates their proofs.

10 CONCLUSION

We have formally proven that type class resolution is coherent by means of logical relations and an intermediate language with explicit dictionaries. In future work we would like to mechanize the proof and adapt it to extensions such as quantified class constraints and GADT's.

ACKNOWLEDGMENTS

This work would not have been possible without the enlightening discussions with Dominique Devriese and George Karachalias. Furthermore, we would like to thank Alexander Vandenbroucke, Ruben Pieters and Steven Keuchel, as well as the anonymous ICFP 2019 and Haskell Symposium 2018 reviewers, for their constructive feedback. This research was partially supported by the Flemish Fund for Scientific Research (FWO) and the Hong Kong Research Grant Council projects number 17210617 and 17258816.

REFERENCES

Martín Abadi. 1999. *Protection in Programming-Language Translations.* Springer, 19–34.

Amal Ahmed. 2006. Step-indexed syntactic logical relations for recursive and quantified types. In *European Symposium on Programming (ESOP).*

Amal Ahmed. 2015. Logical Relations. https://www.cs.uoregon.edu/research/summerschool/summer15/curriculum.html.

Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. 2012. Proofs for free: Parametricity for dependent types. *Journal of Functional Programming* 22, 2 (2012), 107–152.

Xuan Bi, Bruno C. d. S. Oliveira, and Tom Schrijvers. 2018. The essence of nested composition. In *ECOOP.*

Xuan Bi, Ningning Xie, Bruno C. d. S. Oliveira, and Tom Schrijvers. 2019. Distributive Disjoint Polymorphism for Compositional Programming. (2019).

Dariusz Biernacki and Piotr Polesiuk. 2015. Logical relations for coherence of effect subtyping. In *LIPIcs.*

Gert-Jan Bottu, Georgios Karachalias, Tom Schrijvers, Bruno C. d. S. Oliveira, and Philip Wadler. 2017. Quantified Class Constraints. In *Haskell 2017.* ACM, 148–161.

Edwin Brady. 2013. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming* 23, 5 (2013), 552–593.

Manuel M. T. Chakravarty, Gabriele Keller, and Simon Peyton Jones. 2005. Associated Type Synonyms. *SIGPLAN Not.* 40, 9 (2005), 241–253.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 2015. The Lean theorem prover. (2015).

Dominique Devriese and Frank Piessens. 2011. On the Bright Side of Type Classes: Instance Arguments in Agda. In *ICFP '11.* ACM, 143–155.

Derek Dreyer, Robert Harper, Manuel M. T. Chakravarty, and Gabriele Keller. 2007. Modular Type Classes. In *POPL '07.* ACM, 63–70.

Phil Freeman. 2017. *PureScript by Example.* Leanpub. https://leanpub.com/purescript.

Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. 2019. Definitional Proof-Irrelevance without K. *Proceedings of the ACM on Programming Languages* (Jan. 2019), 1–28. https://doi.org/10.1145/329031610.1145/3290316

Douglas Gregor, Jaakko Järvi, Jeremy Siek, Bjarne Stroustrup, Gabriel Dos Reis, and Andrew Lumsdaine. 2006. Concepts: Linguistic Support for Generic Programming in C++. *SIGPLAN Not.* 41, 10 (2006), 291–310.

Cordelia V. Hall, Kevin Hammond, Simon L. Peyton Jones, and Philip L. Wadler. 1996. Type Classes in Haskell. *ACM Trans. Program. Lang. Syst.* 18, 2 (1996), 109–138.

Robert Harper. 2016. *Practical Foundations for Programming Languages* (2nd ed.). Cambridge University Press.

Fergus Henderson, Thomas Conway, Zoltan Somogyi, David Jeffery, Peter Schachte, Simon Taylor, and Chris Speirs. 1996. *The Mercury Language Reference Manual.* Technical Report.

M.P. Jones. 1993. *Coherence for qualified types.* Research Report YALEU/DCS/RR-989. Yale University, Dept. of Computer Science.

Mark P. Jones. 1994. *Qualified Types: Theory and Practice.* Cambridge University Press.

Mark P. Jones. 2000. Type Classes with Functional Dependencies. In *Programming Languages and Systems.* LNCS, Vol. 1782. Springer, 230–244.

Wolfram Kahl and Jan Scheffczyk. 2001. Named Instances for Haskell Type Classes. In *Proc. Haskell Workshop 2001*, Ralf Hinze (Ed.), Vol. 59.

Leonidas Lampropoulos and Benjamin C. Pierce. 2018. *QuickChick: Property-Based Testing in Coq* (1st ed.). Software Foundations, Vol. 4.

Lex Spoon Martin Odersky and Bill Venners. 2008. Implicit Conversions and Parameters. In *Programming in Scala.* Chapter 21.

J. Garrett Morris. 2014. A simple semantics for Haskell overloading. In *Haskell 2014*, Wouter Swierstra (Ed.). ACM, 107–118.

James Hiram Morris Jr. 1969. *Lambda-calculus models of programming languages.* Ph.D. Dissertation. Massachusetts Institute of Technology.

Team Mozilla Research. 2017. *The Rust Programming Language.* https://www.rust-lang.org/en-US/.

Martin Odersky, Olivier Blanvillain, Fengyun Liu, Aggelos Biboudis, Heather Miller, and Sandro Stucki. 2017. Simplicitly: Foundations and Applications of Implicit Function Types. In *POPL '18.*

Dominic Orchard and Tom Schrijvers. 2010. Haskell Type Constraints Unleashed. In *Functional and Logic Programming*, Matthias Blume, Naoki Kobayashi, and Germán Vidal (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 56–71.

Simon Peyton Jones. 2003. *Haskell 98 Language and Libraries: The Revised Report.* Cambridge University Press.

Simon Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Geoffrey Washburn. 2006. Simple Unification-based Type Inference for GADTs. In *Proceedings of the Eleventh ACM SIGPLAN International Conference on Functional Programming (ICFP '06).* ACM, New York, NY, USA, 50–61. https://doi.org/10.1145/1159803.1159811

Benjamin C. Pierce. 2002. *Types and Programming Languages* (1st ed.). The MIT Press.

Benjamin C. Pierce and David N. Turner. 2000. Local Type Inference. *ACM Trans. Program. Lang. Syst.* 22, 1 (2000), 1–44.

Gordon Plotkin. 1973. *Lambda-definability and logical relations.* Edinburgh University.

John C. Reynolds. 1991. The Coherence of Languages with Intersection Types. In *TACS '91.* Springer-Verlag, 675–700.

Tom Schrijvers, Bruno C.D.S. Oliveira, Philip Wadler, and Koar Marntirosian. 2019. COCHIS: Stable and coherent implicits. *Journal of Functional Programming* 29 (2019), e3. https://doi.org/10.1017/S0956796818000242

Matthieu Sozeau and Nicolas Oury. 2008. First-Class Type Classes. In *TPHOLs '08.* Springer-Verlag, 278–293.

Richard Statman. 1985. Logical relations and the typed λ-calculus. *Information and Control* 65, 2-3 (1985), 85–97.

William W Tait. 1967. Intensional interpretations of functionals of finite type I. *The journal of symbolic logic* 32, 2 (1967), 198–212.

P. Wadler and S. Blott. 1989. How to Make Ad-hoc Polymorphism Less Ad Hoc. In *POPL '89.* ACM.

Leo White, Frédéric Bour, and Jeremy Yallop. 2014. Modular implicits. In *ML/OCaml 2014.*

Thomas Winant and Dominique Devriese. 2018. Coherent Explicit Dictionary Application for Haskell. In *Proceedings of the 11th ACM SIGPLAN International Symposium on Haskell (Haskell 2018).* ACM, New York, NY, USA, 81–93.

Appendix

CONTENTS

. List of Definitions 24
. List of Lemmas and Theorems 24
A. Syntax 27
A.1 λ_{TC} : Source Language 27
A.2 F_D : Intermediate Language 27
A.3 $F_{\{\}}$: Target language 27
B. λ_{TC} Judgments and Elaboration 28
B.1 λ_{TC} Type & Constraint Well-Formedness 28
B.2 λ_{TC} Term Typing 28
B.3 Constraint Proving 29
B.4 λ_{TC} Environment Well-Formedness 30
C. λ_{TC} Judgments and Elaboration through F_D 31
C.1 λ_{TC} Type & Constraint Well-Formedness 31
C.2 λ_{TC} Term Typing 31
C.3 Constraint Proving 32
C.4 λ_{TC} Environment Well-Formedness 32
D. F_D Judgments and Elaboration 34
D.1 F_D Type & Constraint Well-Formedness 34
D.2 Dictionary Typing 34
D.3 F_D Term Typing 34
D.4 F_D Environment Well-Formedness 35
D.5 F_D Environment Elaboration 35
D.6 F_D Operational Semantics 35
E. $F_{\{\}}$ Judgments 36
E.1 $F_{\{\}}$ Type Well-Formedness 36
E.2 $F_{\{\}}$ Term Typing 36
E.3 $F_{\{\}}$ Environment Well-Formedness 36
E.4 $F_{\{\}}$ Operational Semantics 36
F. Context Typing 37
F.1 λ_{TC} Context Typing and Elaboration 37
F.2 λ_{TC} Context Typing and Elaboration through F_D 38
F.3 F_D Context Typing and Elaboration 39
F.4 $F_{\{\}}$ Context Typing 40
G. Logical Relations 41
G.1 Dictionary Relation 41
G.2 Expression Relation 41
G.3 Environment Relation 42
H. Strong Normalization Relations 43
I. Equivalence Relations 44
I.1 Kleene Equivalence Relations 44
I.2 Contextual Equivalence Relations 44
J. λ_{TC} Theorems 45
J.1 Conjectures 45
J.2 Lemmas 45
J.3 Typing Preservation 47
K. F_D Theorems 59
K.1 Conjectures 59
K.2 Lemmas 60
K.3 Type Safety 61
K.4 Strong Normalization 65
L. Elaboration Equivalence Theorems 75
M. Coherence Theorems 88
M.1 Compatibility Lemmas 88

M.2 Helper Theorems . . . 99
M.3 Partial Coherence Theorems . . . 114
M.4 Main Coherence Theorems . . . 122
N. F_{D}-to-$F_{\{\}}$ Theorems 126
N.1 Lemmas . . . 126
N.2 Soundness . . . 127
N.3 Determinism . . . 133
N.4 Semantic Preservation . . . 138

List of Definitions

1 Definition (Interpretation of type variables in type contexts) . . . 42
2 Definition (Interpretation of term variables in type contexts) . . . 42
3 Definition (Interpretation of dictionary variables dictionary contexts) . . . 42
4 Definition (Interpretation of type variables in type contexts for strong normalization) . . . 43
5 Definition (Interpretation of term variables in type contexts for strong normalization) . . . 43
6 Definition (Interpretation of dictionary variables dictionary contexts for strong normalization) . . . 43

List of Lemmas and Theorems

1 Lemma (Type Variable Substitution in λ_{TC} Constraint Typing) . . . 45
2 Lemma (Type Well-Formedness Environment Weakening) . . . 45
3 Lemma (Class Constraint Well-Formedness Environment Weakening) . . . 45
4 Lemma (Constraint Well-Formedness Environment Weakening) . . . 45
5 Lemma (Context Well-Formedness Class Environment Weakening) . . . 45
6 Lemma (Context Well-Formedness Typing Environment Weakening) . . . 45
7 Lemma (Determinism of Context Typing) . . . 45
8 Lemma (Class Constraint Elaboration to F_D Uniqueness) . . . 45
9 Lemma (Type Elaboration to F_D Uniqueness) . . . 45
10 Lemma (Constraint Elaboration to F_D Uniqueness) . . . 45
11 Lemma (Environment Elaboration to F_D Uniqueness) . . . 46
12 Lemma (Environment Well-Formedness of λ_{TC} Typing) . . . 46
13 Lemma (Environment Well-Formedness of λ_{TC} Typing through F_D) . . . 46
14 Lemma (Well-Formedness of λ_{TC} Typing Result) . . . 46
15 Lemma (Preservation of Environment Term Variables from λ_{TC} to F_D) . . . 46
16 Lemma (Preservation of Environment Type Variables from λ_{TC} to F_D) . . . 46
17 Lemma (Preservation of Environment Dictionary Variables from λ_{TC} to F_D) . . . 46
18 Lemma (Environment Well-Formedness Strengthening) . . . 46
19 Lemma (Environment Well-Formedness with F_D Elaboration Strengthening) . . . 47
1 Theorem (Typing Preservation - Expressions) . . . 47
2 Theorem (Typing Preservation - Instance) . . . 51
3 Theorem (Typing Preservation - Classes) . . . 52
4 Theorem (Typing Preservation - Programs) . . . 52
5 Theorem (Typing Preservation - Types and Class Constraints) . . . 53
6 Theorem (Typing Preservation - Constraints Proving) . . . 54
7 Theorem (Typing Preservation - Environment Well-Formedness) . . . 56
20 Lemma (Type Variable Substitution in Types) . . . 59
21 Lemma (Type Variable Substitution in Dictionaries) . . . 59
22 Lemma (Variable Substitution) . . . 59
23 Lemma (Reverse Variable Substitution) . . . 59
24 Lemma (Dictionary Variable Substitution) . . . 59
25 Lemma (Reverse Dictionary Variable Substitution) . . . 59
26 Lemma (Type Variable Substitution) . . . 59
27 Lemma (Reverse Type Variable Substitution) . . . 59
28 Lemma (Dictionary Variable Substitution in Dictionaries) . . . 59
29 Lemma (Type Variable Substitution in Dictionaries) . . . 59
30 Lemma (Expression Well-Typed Method Environment Weakening) . . . 59
31 Lemma (Type Well-Formedness Dictionary Environment Weakening) . . . 59
32 Lemma (Class Constraint Well-Formedness Environment Weakening) . . . 59
33 Lemma (Logical Equivalence Environment Weakening) . . . 59
34 Lemma (Strong Normalization Relation Method Environment Weakening) . . . 60
35 Lemma (Dictionary Value Relation Preserved under Substitution) . . . 60
36 Lemma (Environment Well-Formedness Strengthening) . . . 60
37 Lemma (Variable Strengthening in Dictionaries) . . . 60
38 Lemma (Variable Strengthening in Types) . . . 60
39 Lemma (Method Type Well-Formedness) . . . 60
40 Lemma (Method Environment Well-Formedness) . . . 60
41 Lemma (Determinism of Evaluation) . . . 60

42 Lemma (Preservation of Environment Type Variables from F_D to $F_{\{\}}$) . . . 60
43 Lemma (Well-Formedness of F_D Typing Result) . . . 61
44 Lemma (Context Well-Formedness of F_D Typing) . . . 61
45 Lemma (Context Well-Formedness of Dictionary Typing) . . . 61
8 Theorem (Preservation) . . . 61
9 Theorem (Progress) . . . 63
10 Theorem (Strong Normalization) . . . 65
46 Lemma (Well Typedness from Strong Normalization) . . . 65
47 Lemma (Strong Normalization preserved by forward/backward reduction) . . . 65
48 Lemma (Substitution for Context Interpretation) . . . 66
49 Lemma (Compositionality for Strong Normalization) . . . 66
1 Corollary (Compositionality for Strong Normalization (Context Interpretation)) . . . 68
11 Theorem (Strong Normalization - Part A) . . . 69
12 Theorem (Strong Normalization - Part B) . . . 74
13 Theorem (Equivalence - Environments) . . . 75
14 Theorem (Equivalence - Types and Constraints) . . . 77
15 Theorem (Equivalence - Dictionaries) . . . 80
16 Theorem (Equivalence - Expressions) . . . 82
17 Theorem (Equivalence - Contexts) . . . 86
50 Lemma (Compatibility - Term Abstraction) . . . 88
51 Lemma (Compatibility - Term Application) . . . 89
52 Lemma (Compatibility - Dictionary Abstraction) . . . 90
53 Lemma (Compatibility - Dictionary Application) . . . 92
54 Lemma (Compatibility - Type Abstraction) . . . 93
55 Lemma (Compatibility - Type Application) . . . 94
56 Lemma (Compatibility - Let Binding) . . . 95
57 Lemma (Compatibility - Method) . . . 97
18 Theorem (Congruence - Expressions) . . . 99
19 Theorem ($F_{\{\}}$ Context Preserved by Elaboration) . . . 99
20 Theorem (Logical Equivalence Preserved by Forward/Backward Reduction) . . . 102
21 Theorem (Dictionary Reflexivity) . . . 102
22 Theorem (Expression Reflexivity) . . . 104
23 Theorem (Context Reflexivity) . . . 107
24 Theorem (Value Relation for Dictionary Values) . . . 112
25 Theorem (Environment Equivalence Preservation) . . . 113
26 Theorem (Contextual Equivalence in F_D Implies Contextual Equivalence in λ_{TC}) . . . 114
27 Theorem (Coherence - Dictionaries - Part A) . . . 114
28 Theorem (Coherence - Expressions - Part A) . . . 117
29 Theorem (Coherence - Expressions - Part B) . . . 121
30 Theorem (Coherence - Expressions - Part C) . . . 122
31 Theorem (Coherence) . . . 122
32 Theorem (Coherence - Expressions) . . . 123
33 Theorem (Coherence - Programs) . . . 123
58 Lemma (Dictionary Elaboration Uniqueness) . . . 126
59 Lemma (Type Elaboration Uniqueness) . . . 126
60 Lemma (Context Elaboration Uniqueness) . . . 126
61 Lemma (Determinism of Evaluation) . . . 126
62 Lemma (Dictionary Variable Elaboration Soundness) . . . 126
63 Lemma (Type Elaboration Soundness) . . . 126
64 Lemma (Term Variable Elaboration Soundness) . . . 126
65 Lemma (Dictionary Variable in Environment Elaboration Soundness) . . . 126
66 Lemma (Environment Elaboration Soundness) . . . 127
67 Lemma (Canonical Forms for Functions) . . . 127
68 Lemma (Canonical Forms for Type Abstractions) . . . 127
69 Lemma (Distribution of tEval-app) . . . 127
70 Lemma (Distribution of tEval-Tapp) . . . 127
34 Theorem (Term Elaboration Soundness) . . . 127
35 Theorem (Dictionary Elaboration Soundness) . . . 131

36 Theorem (Deterministic Dictionary Elaboration) 133
37 Theorem (Deterministic Term Elaboration) 134
38 Theorem (Deterministic Context Elaboration) 138
39 Theorem (Semantic Preservation) 138
71 Lemma ($F_{\{\}}$ Preservation of Values) 142
40 Theorem (Value Semantic Preservation) 143

THEOREM NUMBER MAPPING

Theorem	Paper	Appendix
Coherence	1	31
Expression Coherence	2	32
Progress	3	9
Preservation	4	8
Strong Normalization	5	10
Typing Preservation - Expressions	6	1
Type Preservation	7	34
Semantic Preservation	8	40
Determinism	9	37
Elaboration Equivalence - Expressions	10	16
Elaboration Equivalence - Dictionaries	11	15
Value Relation for Dictionary Values	12	24
Logical Coherence of Dictionary Resolution	13	27
Logical Coherence of Expression Elaboration	14	28
Logical implies Contextual Equivalence	15	29
Elaboration preserves Contextual Equivalence	16	30
Contextual Equivalence in F_D implies Contextual Equivalence in λ_{TC}	17	26

A SYNTAX

A.1 λ_{TC} : Source Language

$pgm ::= e \mid cls; pgm \mid inst; pgm$	*spgm*
$cls ::= \textbf{class}\ \overline{TC_i\ a} \Rightarrow TC\, a\ \textbf{where}\ \{m : \sigma\}$	*class decl.*
$inst ::= \textbf{instance}\ \overline{Q} \Rightarrow TC\, \tau\ \textbf{where}\ \{m = e\}$	*instance decl.*
$e ::= True \mid False \mid x \mid m \mid \lambda x.e \mid e_1\, e_2 \mid \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2 \mid e :: \tau$	*term*
$\tau ::= Bool \mid a \mid \tau_1 \to \tau_2$	*monotype*
$\rho ::= \tau \mid Q \Rightarrow \rho$	*qualified type*
$\sigma ::= \rho \mid \forall a.\sigma$	*type scheme*
$Q ::= TC\, \tau$	*class constraint*
$C ::= \forall \overline{a}.\overline{Q} \Rightarrow Q$	*constraint*
$\Gamma ::= \bullet \mid \Gamma, x : \sigma \mid \Gamma, a \mid \Gamma, \delta : Q$	*typing environment*
$\Gamma_C ::= \bullet \mid \Gamma_C, m : \overline{TC_i\ a} \Rightarrow TC\, a : \sigma$	*class environment*
$P ::= \bullet \mid P, (D : C).m \mapsto \Gamma : e$	*program context*
$M ::= [\bullet] \mid \lambda x.M \mid M\, e \mid e\, M \mid M :: \tau$ $\mid \textbf{let}\ x : \sigma = M\ \textbf{in}\ e \mid \textbf{let}\ x : \sigma = e\ \textbf{in}\ M$	*evaluation context*

A.2 F_{D} : Intermediate Language

$e ::= True \mid False \mid x \mid \lambda x : \sigma.e \mid e_1\, e_2 \mid \lambda\delta : Q.e \mid e\, d$ $\Lambda a.e \mid e\, \sigma \mid d.m \mid \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2$	*expression*
$v ::= True \mid False \mid \lambda x : \sigma.e \mid \lambda\delta : Q.e \mid \Lambda a.e$	*value*
$\sigma ::= Bool \mid a \mid \sigma_1 \to \sigma_2 \mid Q \Rightarrow \sigma \mid \forall a.\sigma$	*type*
$Q ::= TC\, \sigma$	*class constraint*
$C ::= \forall \overline{a}.\overline{Q} \Rightarrow Q$	*constraint*
$\Gamma ::= \bullet \mid \Gamma, x : \sigma \mid \Gamma, a \mid \Gamma, \delta : Q$	*typing environment*
$\Gamma_C ::= \bullet \mid \Gamma_C, m : TC\, a : \sigma$	*class environment*
$\Sigma ::= \bullet \mid \Sigma, (D : C).m \mapsto e$	*method environment*
$M ::= [\bullet] \mid \lambda x : \sigma.M \mid \lambda\delta : Q.M \mid e\, M \mid M\, e \mid M\, d$ $\mid \Lambda a.M \mid M\, \sigma \mid \textbf{let}\ x : \sigma = M\ \textbf{in}\ e$ $\mid \textbf{let}\ x : \sigma = e\ \textbf{in}\ M$	*evaluation context*

A.2.1 *Dictionaries.*

$d ::= \delta \mid D\, \overline{\sigma}\, \overline{d}$	*dictionary*
$dv ::= D\, \overline{\sigma}\, \overline{dv}$	*dictionary value*

A.3 $F_{\{\}}$: Target language

$e ::= True \mid False \mid x \mid \lambda x : \sigma.e \mid e_1\, e_2 \mid \Lambda a.e \mid e\, \sigma$ $\mid \{\overline{m_i = e_i}^{\,i<n}\} \mid e.m \mid \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2$	*target term*
$v ::= True \mid False \mid \lambda x : \sigma.e \mid \Lambda a.e \mid \{\overline{m_i = e_i}^{\,i<n}\}$	*target value*
$\sigma ::= Bool \mid a \mid \forall a.\sigma \mid \sigma_1 \to \sigma_2 \mid \{\overline{m_i : \sigma_i}^{\,i<n}\}$	*target type*
$\Gamma ::= \bullet \mid \Gamma, a \mid \Gamma, x : \sigma$	*target context*
$M ::= [\bullet] \mid \lambda x : \sigma.M \mid e\, M \mid M\, e \mid \Lambda a.M \mid M\, \sigma \mid \{\overline{m_i = M_i}^{\,i\in 1..n}\}$ $\mid M.m \mid \textbf{let}\ x : \sigma = M\ \textbf{in}\ e \mid \textbf{let}\ x : \sigma = e\ \textbf{in}\ M$	*target evaluation context*

B λ_{TC} JUDGMENTS AND ELABORATION

B.1 λ_{TC} Type & Constraint Well-Formedness

$\boxed{\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma}$ *(λ_{TC} Class Constraint Well-Formedness)*

$$\frac{\Gamma_C; \Gamma \vdash_{ty} \tau \rightsquigarrow \sigma' \qquad \Gamma_C = \Gamma_{C1}, m : \overline{Q}_i \Rightarrow TC\, a : \sigma, \Gamma_{C2} \qquad \Gamma_{C1}; \bullet, a \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C; \Gamma \vdash_Q TC\, \tau \rightsquigarrow [\sigma'/a]\{m : \sigma\}} \text{ sQT-TC}$$

$\boxed{\Gamma_C; \Gamma \vdash_C C \rightsquigarrow \sigma}$ *(λ_{TC} Constraint Well-Formedness)*

$$\frac{\overline{\Gamma_C; \Gamma, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma_i}^{\,i} \qquad \Gamma_C; \Gamma, \overline{a}_j \vdash_Q Q \rightsquigarrow \sigma \qquad \overline{a}_j \notin \Gamma}{\Gamma_C; \Gamma \vdash_C \forall \overline{a}_j . \overline{Q}_i \Rightarrow Q \rightsquigarrow \forall \overline{a}_j . \overline{\sigma}_i \rightarrow \sigma} \text{ sCT-ABS}$$

$\boxed{\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma}$ *(λ_{TC} Type Well-Formedness)*

$$\frac{}{\Gamma_C; \Gamma \vdash_{ty} Bool \rightsquigarrow Bool} \text{ sTyT-BOOL} \qquad \frac{a \in \Gamma}{\Gamma_C; \Gamma \vdash_{ty} a \rightsquigarrow a} \text{ sTyT-VAR} \qquad \frac{\Gamma_C; \Gamma \vdash_{ty} \tau_1 \rightsquigarrow \sigma_1 \qquad \Gamma_C; \Gamma \vdash_{ty} \tau_2 \rightsquigarrow \sigma_2}{\Gamma_C; \Gamma \vdash_{ty} \tau_1 \rightarrow \tau_2 \rightsquigarrow \sigma_1 \rightarrow \sigma_2} \text{ sTyT-ARROW}$$

$$\frac{\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_1 \qquad \Gamma_C; \Gamma \vdash_{ty} \rho \rightsquigarrow \sigma_2}{\Gamma_C; \Gamma \vdash_{ty} Q \Rightarrow \rho \rightsquigarrow \sigma_1 \rightarrow \sigma_2} \text{ sTyT-QUAL} \qquad \frac{a \notin \Gamma \qquad \Gamma_C; \Gamma, a \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C; \Gamma \vdash_{ty} \forall a . \sigma \rightsquigarrow \forall a . \sigma} \text{ sTyT-SCHEME}$$

B.2 λ_{TC} Term Typing

$\boxed{P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e}$ *(λ_{TC} Term Inference)*

$$\frac{\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{P; \Gamma_C; \Gamma \vdash_{tm} True \Rightarrow Bool \rightsquigarrow True} \text{ sTm-infT-True} \qquad \frac{\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{P; \Gamma_C; \Gamma \vdash_{tm} False \Rightarrow Bool \rightsquigarrow False} \text{ sTm-infT-False}$$

$$\frac{\begin{array}{c} x \notin \mathbf{dom}(\Gamma) \qquad \mathbf{unambig}(\forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1) \qquad \mathbf{closure}(\Gamma_C; \overline{Q}_i) = \overline{Q}_k \qquad \overline{\Gamma_C; \Gamma \vdash_Q Q_k \rightsquigarrow \sigma_k}^{\,k} \\ \Gamma_C; \Gamma \vdash_{ty} \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j . \overline{\sigma}_k \rightarrow \sigma \qquad \overline{\delta}_k \ \mathbf{fresh} \qquad P; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k \vdash_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ P; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau_1 \vdash_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \qquad e = \mathbf{let}\ x : \forall \overline{a}_j . \overline{\sigma}_k \rightarrow \sigma = \Lambda \overline{a}_j . \lambda \overline{\delta_k : \sigma_k}^{\,k} . e_1\ \mathbf{in}\ e_2 \end{array}}{P; \Gamma_C; \Gamma \vdash_{tm} \mathbf{let}\ x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 = e_1\ \mathbf{in}\ e_2 \Rightarrow \tau_2 \rightsquigarrow e} \text{ sTm-infT-Let}$$

$$\frac{P; \Gamma_C; \Gamma \vdash_{tm} e_1 \Rightarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow e_1 \qquad P; \Gamma_C; \Gamma \vdash_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{P; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 \Rightarrow \tau_2 \rightsquigarrow e_1\, e_2} \text{ sTm-infT-ArrE} \qquad \frac{P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e}{P; \Gamma_C; \Gamma \vdash_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e} \text{ sTm-infT-Ann}$$

$\boxed{P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e}$ *(λ_{TC} Term Checking)*

$$\frac{\begin{array}{c} (x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau) \in \Gamma \\ \mathbf{unambig}(\forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau) \qquad \overline{P; \Gamma_C; \Gamma \vDash [\overline{\tau}_j / \overline{a}_j] Q_i \rightsquigarrow e_i}^{\,i} \qquad \overline{\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \end{array}}{P; \Gamma_C; \Gamma \vdash_{tm} x \Leftarrow [\overline{\tau}_j / \overline{a}_j] \tau \rightsquigarrow x\, \overline{\sigma}_j\, \overline{e}_i} \text{ sTm-checkT-Var}$$

$$\frac{\begin{array}{c} (m : \overline{Q}'_k \Rightarrow TC\, a : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau') \in \Gamma_C \qquad \mathbf{unambig}(\forall \overline{a}_j, a . \overline{Q}_i \Rightarrow \tau') \qquad P; \Gamma_C; \Gamma \vDash TC\, \tau \rightsquigarrow e \\ \Gamma_C; \Gamma \vdash_{ty} \tau \rightsquigarrow \sigma \qquad \overline{P; \Gamma_C; \Gamma \vDash [\overline{\tau}_j / \overline{a}_j][\tau / a] Q_i \rightsquigarrow e_i}^{\,i} \qquad \overline{\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \end{array}}{P; \Gamma_C; \Gamma \vdash_{tm} m \Leftarrow [\overline{\tau}_j / \overline{a}_j][\tau / a] \tau' \rightsquigarrow e.m\, \overline{\sigma}_j\, \overline{e}_i} \text{ sTm-checkT-Meth}$$

$$\frac{x \notin \mathbf{dom}(\Gamma) \qquad P; \Gamma_C; \Gamma, x : \tau_1 \vdash_{tm} e \Leftarrow \tau_2 \rightsquigarrow e \qquad \Gamma_C; \Gamma \vdash_{ty} \tau_1 \rightsquigarrow \sigma}{P; \Gamma_C; \Gamma \vdash_{tm} \lambda x . e \Leftarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x : \sigma . e} \text{ sTm-checkT-ArrI}$$

$$\frac{P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e}{P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e} \text{ sTm-checkT-Inf}$$

$\boxed{\Gamma_C \vdash_{cls} cls : \Gamma_C'}$ *(Class Decl Typing)*

$$\frac{\begin{array}{c} m \notin \mathbf{dom}(\Gamma_C) \qquad \mathbf{closure}(\Gamma_C; \overline{Q_k}) = \overline{Q_p} \qquad \Gamma_C; \bullet, a \vdash_{ty} \forall \overline{a_j}.\overline{Q_p} \Rightarrow \tau \rightsquigarrow \sigma \qquad \mathbf{unambig}(\forall \overline{a_j}, a.\overline{Q_p} \Rightarrow \tau) \\ \overline{\Gamma_C; \bullet, a \vdash_Q TC_i\, a \rightsquigarrow \sigma_i}^{\,i} \qquad \nexists TC' : (m : \overline{Q'_m} \Rightarrow TC'\, b : \sigma') \in \Gamma_C \qquad \nexists m' : (m' : \overline{Q'_m} \Rightarrow TC\, a : \sigma') \in \Gamma_C \end{array}}{\Gamma_C \vdash_{cls} \mathbf{class}\ \overline{TC_i\, a} \Rightarrow TC\, a\ \mathbf{where}\ \{m : \forall \overline{a_j}.\overline{Q_k} \Rightarrow \tau\} : \bullet, m : \overline{TC_i\, a} \Rightarrow TC\, a : \forall \overline{a_j}.\overline{Q_p} \Rightarrow \tau} \;\textsc{sClsT-cls}$$

$\boxed{P; \Gamma_C \vdash_{inst} inst : P'}$ *(Instance Decl Typing)*

$$\frac{\begin{array}{c} (m : \overline{Q'_i} \Rightarrow TC\, a : \forall \overline{a_j}.\overline{Q'_h} \Rightarrow \tau_1) \in \Gamma_C \qquad \overline{b_k} = \mathbf{fv}(\tau) \\ \Gamma_C; \bullet, \overline{b_k} \vdash_{ty} \tau \rightsquigarrow \sigma \qquad \mathbf{closure}(\Gamma_C; \overline{Q_p}) = \overline{Q_q} \qquad \mathbf{unambig}(\forall \overline{b_k}.\overline{Q_q} \Rightarrow TC\, \tau) \qquad \overline{\Gamma_C; \bullet, \overline{b_k} \vdash_Q Q_q \rightsquigarrow \sigma_q}^{\,q} \\ \overline{P; \Gamma_C; \bullet, \overline{b_k}, \overline{\delta_q : Q_q} \vDash [\tau/a]Q'_i \rightsquigarrow e_i}^{\,i} \qquad P; \Gamma_C; \bullet, \overline{b_k}, \overline{\delta_q : Q_q}, \overline{a_j}, \overline{\delta'_h : [\tau/a]Q'_h} \vdash_{tm} e \Leftarrow [\tau/a]\tau_1 \rightsquigarrow e \\ D\ \mathbf{fresh} \qquad \overline{\delta_q}\ \mathbf{fresh} \qquad \overline{\delta'_h}\ \mathbf{fresh} \qquad (D' : \forall \overline{b'_s}.\overline{Q'_n} \Rightarrow TC\, \tau_2).m' \mapsto \Gamma' : e' \notin P \text{ where } [\overline{\tau'_s}/\overline{b'_s}]\tau_2 = [\overline{\tau'_k}/\overline{b_k}]\tau \\ P' = (D : \forall \overline{b_k}.\overline{Q_q} \Rightarrow TC\, \tau).m \mapsto \bullet, \overline{b_k}, \overline{\delta_q : Q_q}, \overline{a_j}, \overline{\delta'_h : [\tau/a]Q'_h} : e \end{array}}{P; \Gamma_C \vdash_{inst} \mathbf{instance}\ \overline{Q_p} \Rightarrow TC\, \tau\ \mathbf{where}\ \{m = e\} : P'} \;\textsc{sInstT-inst}$$

$\boxed{P; \Gamma_C \vdash_{pgm} pgm : \tau; P'; \Gamma_C' \rightsquigarrow e}$ *(λ_{TC} Program Typing)*

$$\frac{\Gamma_C \vdash_{cls} cls : \Gamma_C' \qquad P; \Gamma_C, \Gamma_C' \vdash_{pgm} pgm : \tau; P'; \Gamma_C'' \rightsquigarrow e}{P; \Gamma_C \vdash_{pgm} cls; pgm : \tau; P'; \Gamma_C', \Gamma_C'' \rightsquigarrow e} \;\textsc{sPgmT-cls}$$

$$\frac{P; \Gamma_C \vdash_{inst} inst : P' \qquad P, P'; \Gamma_C \vdash_{pgm} pgm : \tau; P''; \Gamma_C' \rightsquigarrow e}{P; \Gamma_C \vdash_{pgm} inst; pgm : \tau; P', P''; \Gamma_C' \rightsquigarrow e} \;\textsc{sPgmT-inst} \qquad \frac{P; \Gamma_C; \bullet \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e}{P; \Gamma_C \vdash_{pgm} e : \tau; \bullet; \bullet \rightsquigarrow e} \;\textsc{sPgmT-expr}$$

$\boxed{\mathbf{closure}(\Gamma_C; \overline{Q_i}) = \overline{Q_j}}$ *(Closure over Superclass Relation)*

$$\frac{}{\mathbf{closure}(\Gamma_C; \bullet) = \bullet} \;\textsc{sClosure-empty} \qquad \frac{(m : \overline{Q_m} \Rightarrow TC\, a : \sigma) \in \Gamma_C \qquad \mathbf{closure}(\Gamma_C; \overline{Q_i}, \overline{Q_m}) = \overline{Q_j}}{\mathbf{closure}(\Gamma_C; \overline{Q_i}, TC\, a) = \overline{Q_j}, TC\, a} \;\textsc{sClosure-TC}$$

$\boxed{\mathbf{unambig}(\sigma)}$ *(Unambiguity for Type Schemes)*

$$\frac{\overline{a_j} \in \mathbf{fv}(\tau)}{\mathbf{unambig}(\forall \overline{a_j}.\overline{Q_i} \Rightarrow \tau)} \;\textsc{sUnambig-scheme}$$

$\boxed{\mathbf{unambig}(C)}$ *(Unambiguity for Constraints)*

$$\frac{\overline{a_j} \in \mathbf{fv}(\tau)}{\mathbf{unambig}(\forall \overline{a_j}.\overline{Q_i} \Rightarrow TC\, \tau)} \;\textsc{sUnambig-constraint}$$

B.3 Constraint Proving

$\boxed{P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow e}$ *(Constraint Entailment)*

$$\frac{\begin{array}{c} P = P_1, (D : \forall \overline{a_j}.\overline{Q'_i} \Rightarrow Q').m \mapsto \bullet, \overline{a_j}, \overline{\delta_i : Q'_i}, \overline{b_k}, \overline{\delta_h : Q_h} : e, P_2 \\ Q = [\overline{\tau_j}/\overline{a_j}]Q' \qquad P_1; \Gamma_C; \bullet, \overline{a_j}, \overline{\delta_i : Q'_i}, \overline{b_k}, \overline{\delta_h : Q_h} \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \\ \overline{\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \overline{\Gamma_C; \bullet, \overline{a_j} \vdash_Q Q'_i \rightsquigarrow \sigma'_i}^{\,i} \qquad \overline{\Gamma_C; \bullet, \overline{a_j}, \overline{b_k} \vdash_Q Q_h \rightsquigarrow \sigma''_h}^{\,h} \qquad \overline{P; \Gamma_C; \Gamma \vDash [\overline{\tau_j}/\overline{a_j}]Q'_i \rightsquigarrow e_i}^{\,i} \end{array}}{P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow (\Lambda \overline{a_j}.\lambda \overline{\delta'_i : \sigma'_i}^{\,i}.\{m = \Lambda \overline{b_k}.\lambda \overline{\delta_h : \sigma''_h}^{\,h}.e\})\, \overline{\sigma_j}\, \overline{e_i}} \;\textsc{sEntailT-inst}$$

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow \delta} \;\textsc{sEntailT-local}$$

B.4 λ_{TC} Environment Well-Formedness

$\boxed{\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma}$ *(λ_{TC} Environment Well-Formedness)*

$$\frac{}{\vdash_{ctx} \bullet; \bullet; \bullet \rightsquigarrow \bullet}\ \textsc{sCtxT-empty}$$

$$\frac{\overline{a}_j, a = \mathbf{fv}(\tau) \qquad \dfrac{\Gamma_C; \bullet, a \vdash_{ty} \forall \overline{a}_j . \overline{TC_i\, a}^{\,i} \Rightarrow \tau \rightsquigarrow \sigma}{\overline{\Gamma_C; \bullet, a \vdash_Q TC_i\, a \rightsquigarrow \sigma_i}^{\,i}} \qquad m \notin \mathbf{dom}(\Gamma_C) \qquad TC\, b \notin \mathbf{dom}(\Gamma_C) \qquad \vdash_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet}{\vdash_{ctx} \bullet; \Gamma_C, m : \overline{TC_i\; a} \Rightarrow TC\, a : \forall \overline{a}_j . \overline{TC_i\, a}^{\,i} \Rightarrow \tau; \bullet \rightsquigarrow \bullet}\ \textsc{sCtxT-clsEnv}$$

$$\frac{\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \qquad x \notin \mathbf{dom}(\Gamma) \qquad \vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{\vdash_{ctx} \bullet; \Gamma_C; \Gamma, x : \sigma \rightsquigarrow \Gamma, x : \sigma}\ \textsc{sCtxT-tyEnvTm}$$

$$\frac{a \notin \Gamma \qquad \vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{\vdash_{ctx} \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \Gamma, a}\ \textsc{sCtxT-tyEnvTy}$$

$$\frac{\Gamma_C; \Gamma \vdash_Q TC\, \tau \rightsquigarrow \sigma \qquad \delta \notin \mathbf{dom}(\Gamma) \qquad \vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma}{\vdash_{ctx} \bullet; \Gamma_C; \Gamma, \delta : TC\, \tau \rightsquigarrow \Gamma, \delta : \sigma}\ \textsc{sCtxT-tyEnvD}$$

$$\frac{\begin{array}{c} \mathbf{unambig}(\forall \overline{b}_j . \overline{Q}_i \Rightarrow TC\, \tau) \qquad \Gamma_C; \bullet \vdash_C \forall \overline{b}_j . \overline{Q}_i \Rightarrow TC\, \tau \rightsquigarrow \forall \overline{b}_j . \overline{\sigma}_i \rightarrow [\sigma / a]\{m : \forall \overline{a}_k . \overline{\sigma}'_h \rightarrow \sigma'\} \\ (m : \overline{Q}'_m \Rightarrow TC\; a : \forall \overline{a}_k . \overline{Q}'_h \Rightarrow \tau') \in \Gamma_C \qquad \Gamma_C; \bullet, a \vdash_{ty} \forall \overline{a}_k . \overline{Q}'_h \Rightarrow \tau' \rightsquigarrow \forall \overline{a}_k . \overline{\sigma}'_h \rightarrow \sigma' \\ \Gamma_C; \bullet, \overline{b}_j \vdash_{ty} \tau \rightsquigarrow \sigma \qquad P; \Gamma_C; \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau / a]\overline{Q}'_h \vdash_{tm} e \Leftarrow [\tau / a]\tau' \rightsquigarrow e \qquad D \notin \mathbf{dom}(P) \\ (D' : \forall \overline{b}'_k . \overline{Q}''_h \Rightarrow TC\, \tau'').m' \mapsto \Gamma' : e' \notin P \qquad \mathbf{where}[\overline{\tau}_j / \overline{b}_j]\tau = [\overline{\tau}'_k / \overline{b}'_k]\tau'' \qquad \vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \end{array}}{\vdash_{ctx} P, (D : \forall \overline{b}_j . \overline{Q}_i \Rightarrow TC\, \tau).m \mapsto \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau / a]\overline{Q}'_h : e; \Gamma_C; \Gamma \rightsquigarrow \Gamma}\ \textsc{sCtxT-pgmInst}$$

C λ_{TC} JUDGMENTS AND ELABORATION THROUGH F_D

C.1 λ_{TC} Type & Constraint Well-Formedness

$\boxed{\Gamma_C;\Gamma \vdash^M_Q Q \rightsquigarrow Q}$ *(λ_{TC} Class Constraint Well-Formedness)*

$$\frac{\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \qquad \Gamma_C = \Gamma_{C1}, m : \overline{Q}_i \Rightarrow TC\,a : \sigma, \Gamma_{C2} \qquad \Gamma_{C1}; \bullet, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma'}{\Gamma_C;\Gamma \vdash^M_Q TC\,\tau \rightsquigarrow TC\,\sigma}\;\textsc{sQ-TC}$$

$\boxed{\Gamma_C;\Gamma \vdash^M_C C \rightsquigarrow C}$ *(λ_{TC} Constraint Well-Formedness)*

$$\frac{\overline{\Gamma_C;\Gamma,\overline{a}_j \vdash^M_Q Q_i \rightsquigarrow Q_i}^{\,i} \qquad \Gamma_C;\Gamma,\overline{a}_j \vdash^M_Q Q \rightsquigarrow Q \qquad \overline{a}_j \notin \Gamma}{\Gamma_C;\Gamma \vdash^M_C \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q}\;\textsc{sC-Abs}$$

$\boxed{\Gamma_C;\Gamma \vdash^M_{ty} \sigma \rightsquigarrow \sigma}$ *(λ_{TC} Type Well-Formedness)*

$$\frac{}{\Gamma_C;\Gamma \vdash^M_{ty} Bool \rightsquigarrow Bool}\;\textsc{sTy-Bool} \qquad \frac{a \in \Gamma}{\Gamma_C;\Gamma \vdash^M_{ty} a \rightsquigarrow a}\;\textsc{sTy-Var} \qquad \frac{\Gamma_C;\Gamma \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma_1 \qquad \Gamma_C;\Gamma \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2}{\Gamma_C;\Gamma \vdash^M_{ty} \tau_1 \to \tau_2 \rightsquigarrow \sigma_1 \to \sigma_2}\;\textsc{sTy-Arrow}$$

$$\frac{\Gamma_C;\Gamma \vdash^M_Q Q \rightsquigarrow Q \qquad \Gamma_C;\Gamma \vdash^M_{ty} \rho \rightsquigarrow \sigma}{\Gamma_C;\Gamma \vdash^M_{ty} Q \Rightarrow \rho \rightsquigarrow Q \Rightarrow \sigma}\;\textsc{sTy-Qual} \qquad \frac{a \notin \Gamma \qquad \Gamma_C;\Gamma, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C;\Gamma \vdash^M_{ty} \forall a.\sigma \rightsquigarrow \forall a.\sigma}\;\textsc{sTy-Scheme}$$

C.2 λ_{TC} Term Typing

$\boxed{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e}$ *(λ_{TC} Term Inference)*

$$\frac{\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma}{P;\Gamma_C;\Gamma \vdash^M_{tm} True \Rightarrow Bool \rightsquigarrow True}\;\textsc{sTm-Inf-True} \qquad \frac{\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma}{P;\Gamma_C;\Gamma \vdash^M_{tm} False \Rightarrow Bool \rightsquigarrow False}\;\textsc{sTm-Inf-False}$$

$$\frac{\begin{array}{c} x \notin \textbf{dom}(\Gamma) \qquad \textbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1) \qquad \textbf{closure}(\Gamma_C;\overline{Q}_i) = \overline{Q}_k \\ \Gamma_C;\Gamma \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{Q}_k \Rightarrow \sigma \qquad \overline{\delta}_k\ \textbf{fresh} \qquad P;\Gamma_C;\Gamma,\overline{a}_j,\overline{\delta}_k : \overline{Q}_k \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ P;\Gamma_C;\Gamma, x : \forall \overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \vdash^M_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \qquad e = \textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_k \Rightarrow \sigma = \Lambda \overline{a}_j.\lambda \overline{\delta}_k : \overline{Q}_k.e_1\ \textbf{in}\ e_2 \end{array}}{P;\Gamma_C;\Gamma \vdash^M_{tm} \textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\ \textbf{in}\ e_2 \Rightarrow \tau_2 \rightsquigarrow e}\;\textsc{sTm-Inf-Let}$$

$$\frac{P;\Gamma_C;\Gamma \vdash^M_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1 \qquad P;\Gamma_C;\Gamma \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{P;\Gamma_C;\Gamma \vdash^M_{tm} e_1\,e_2 \Rightarrow \tau_2 \rightsquigarrow e_1\,e_2}\;\textsc{sTm-Inf-ArrE} \qquad \frac{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e}{P;\Gamma_C;\Gamma \vdash^M_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e}\;\textsc{sTm-Inf-Ann}$$

$\boxed{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e}$ *(λ_{TC} Term Checking)*

$$\frac{\begin{array}{c} (x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \in \Gamma \\ \textbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \qquad \overline{P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i}^{\,i} \qquad \overline{\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma \end{array}}{P;\Gamma_C;\Gamma \vdash^M_{tm} x \Leftarrow [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow x\,\overline{\sigma}_j\,\overline{d}_i}\;\textsc{sTm-Check-Var}$$

$$\frac{\begin{array}{c} (m : \overline{Q'}_k \Rightarrow TC\,a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau') \in \Gamma_C \qquad \textbf{unambig}(\forall \overline{a}_j, a.\overline{Q}_i \Rightarrow \tau') \qquad P;\Gamma_C;\Gamma \vDash^M TC\,\tau \rightsquigarrow d \\ \Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \qquad \overline{P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j][\tau/a]Q_i \rightsquigarrow d_i}^{\,i} \qquad \overline{\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma \end{array}}{P;\Gamma_C;\Gamma \vdash^M_{tm} m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow d.m\,\overline{\sigma}_j\,\overline{d}_i}\;\textsc{sTm-Check-Meth}$$

$$\frac{x \notin \textbf{dom}(\Gamma) \qquad P;\Gamma_C;\Gamma, x : \tau_1 \vdash^M_{tm} e \Leftarrow \tau_2 \rightsquigarrow e \qquad \Gamma_C;\Gamma \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma}{P;\Gamma_C;\Gamma \vdash^M_{tm} \lambda x.e \Leftarrow \tau_1 \to \tau_2 \rightsquigarrow \lambda x : \sigma.e}\;\textsc{sTm-Check-ArrI} \qquad \frac{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e}{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e}\;\textsc{sTm-Check-Inf}$$

$\boxed{\Gamma_C \vdash^M_{cls} cls : \Gamma_C'}$ *(Class Decl Typing)*

$$\frac{\begin{array}{c} m \notin \mathbf{dom}(\Gamma_C) \qquad \mathbf{closure}(\Gamma_C; \overline{Q}_k) = \overline{Q}_p \qquad \Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a}_j . \overline{Q}_p \Rightarrow \tau \rightsquigarrow \sigma \qquad \mathbf{unambig}(\forall \overline{a}_j, a. \overline{Q}_p \Rightarrow \tau) \\ \overline{\Gamma_C; \bullet, a \vdash^M_Q TC_i\, a \rightsquigarrow Q_i}^{\,i} \qquad \nexists TC' : (m : \overline{Q}'_m \Rightarrow TC'\, b : \sigma') \in \Gamma_C \qquad \nexists m' : (m' : \overline{Q}'_m \Rightarrow TC\, a : \sigma') \in \Gamma_C \end{array}}{\Gamma_C \vdash^M_{cls} \mathbf{class}\ \overline{TC_i\, a} \Rightarrow TC\, a\ \mathbf{where}\ \{m : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau\} : \bullet, m : \overline{TC_i\, a} \Rightarrow TC\, a : \forall \overline{a}_j . \overline{Q}_p \Rightarrow \tau} \text{ sCls-cls}$$

$\boxed{P; \Gamma_C \vdash^M_{inst} inst : P'}$ *(Instance Decl Typing)*

$$\frac{\begin{array}{c} (m : \overline{Q}'_i \Rightarrow TC\, a : \forall \overline{a}_j . \overline{Q}_h \Rightarrow \tau_1) \in \Gamma_C \qquad \overline{b}_k = \mathbf{fv}(\tau) \\ \Gamma_C; \bullet, \overline{b}_k \vdash^M_{ty} \tau \rightsquigarrow \sigma \qquad \mathbf{closure}(\Gamma_C; \overline{Q}_p) = \overline{Q}_q \qquad \mathbf{unambig}(\forall \overline{b}_k . \overline{Q}_q \Rightarrow TC\, \tau) \qquad \overline{\Gamma_C; \bullet, \overline{b}_k \vdash^M_Q Q_q \rightsquigarrow Q_q}^{\,q} \\ \overline{P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_q : \overline{Q}_q \vDash^M [\tau/a] Q'_i \rightsquigarrow d_i}^{\,i} \qquad P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_q : \overline{Q}_q, \overline{a}_j, \overline{\delta}_h : [\tau/a]\overline{Q}_h \vdash^M_{tm} e \Leftarrow [\tau/a]\tau_1 \rightsquigarrow e \\ D\ \mathbf{fresh} \qquad \overline{\delta}_h\ \mathbf{fresh} \qquad \overline{\delta}_q\ \mathbf{fresh} \qquad (D' : \forall \overline{b}'_m . \overline{Q}'_n \Rightarrow TC\, \tau_2).m' \mapsto \Gamma' : e' \notin P \text{ where } [\overline{\tau}'_m / \overline{b}'_m]\tau_2 = [\overline{\tau}'_k / \overline{b}_k]\tau \\ P' = (D : \forall \overline{b}_k . \overline{Q}_q \Rightarrow TC\, \tau).m \mapsto \bullet, \overline{b}_k, \overline{\delta}_q : \overline{Q}_q, \overline{a}_j, \overline{\delta}_h : [\tau/a]\overline{Q}_h : e \end{array}}{P; \Gamma_C \vdash^M_{inst} \mathbf{instance}\ \overline{Q}_p \Rightarrow TC\, \tau\ \mathbf{where}\ \{m = e\} : P'} \text{ sInst-inst}$$

$\boxed{P; \Gamma_C \vdash^M_{pgm} pgm : \tau; P'; \Gamma_C' \rightsquigarrow e}$ *(λ_{TC} Program Typing)*

$$\frac{\Gamma_C \vdash^M_{cls} cls : \Gamma_C' \qquad P; \Gamma_C, \Gamma_C' \vdash^M_{pgm} pgm : \tau; P'; \Gamma_C'' \rightsquigarrow e}{P; \Gamma_C \vdash^M_{pgm} cls; pgm : \tau; P'; \Gamma_C', \Gamma_C'' \rightsquigarrow e} \text{ sPgm-cls}$$

$$\frac{P; \Gamma_C \vdash^M_{inst} inst : P' \qquad P, P'; \Gamma_C \vdash^M_{pgm} pgm : \tau; P''; \Gamma_C' \rightsquigarrow e}{P; \Gamma_C \vdash^M_{pgm} inst; pgm : \tau; P', P''; \Gamma_C' \rightsquigarrow e} \text{ sPgm-inst} \qquad \frac{P; \Gamma_C; \bullet \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e}{P; \Gamma_C \vdash^M_{pgm} e : \tau; \bullet; \bullet \rightsquigarrow e} \text{ sPgm-expr}$$

C.3 Constraint Proving

$\boxed{P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow d}$ *(Constraint Entailment)*

$$\frac{\begin{array}{c} P = P_1, (D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow Q').m \mapsto \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h : e, P_2 \\ Q = [\overline{\tau}_j / \overline{a}_j] Q' \qquad \vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \qquad \overline{\Gamma_C; \Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \overline{P; \Gamma_C; \Gamma \vDash^M [\overline{\tau}_j / \overline{a}_j] Q_i \rightsquigarrow d_i}^{\,i} \end{array}}{P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow D\, \overline{\sigma}_j\, \overline{d}_i} \text{ sEntail-inst}$$

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma}{P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow \delta} \text{ sEntail-local}$$

C.4 λ_{TC} Environment Well-Formedness

$\boxed{\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma}$ *(λ_{TC} Environment Well-Formedness)*

$$\frac{}{\vdash^M_{ctx} \bullet; \bullet; \bullet \rightsquigarrow \bullet; \bullet; \bullet} \text{ sCtx-empty}$$

$$\frac{\begin{array}{c} \Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a}_j . \overline{TC_i\, a}^{\,i} \Rightarrow \tau \rightsquigarrow \sigma \\ \overline{a}_j, a = \mathbf{fv}(\tau) \qquad \overline{\Gamma_C; \bullet, a \vdash^M_Q TC_i\, a \rightsquigarrow Q_i}^{\,i} \qquad m \notin \mathbf{dom}(\Gamma_C) \qquad TC\, b \notin \mathbf{dom}(\Gamma_C) \qquad \vdash^M_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet \end{array}}{\vdash^M_{ctx} \bullet; \Gamma_C, m : \overline{TC_i\, a} \Rightarrow TC\, a : \forall \overline{a}_j . \overline{TC_i\, a}^{\,i} \Rightarrow \tau; \bullet \rightsquigarrow \bullet; \Gamma_C, m : TC\, a : \sigma; \bullet} \text{ sCtx-clsEnv}$$

$$\frac{\Gamma_C; \Gamma \vdash^M_{ty} \sigma \rightsquigarrow \sigma \qquad x \notin \mathbf{dom}(\Gamma) \qquad \vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma}{\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma, x : \sigma \rightsquigarrow \bullet; \Gamma_C; \Gamma, x : \sigma} \text{ sCtx-tyEnvTm} \qquad \frac{a \notin \Gamma \qquad \vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma}{\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \bullet; \Gamma_C; \Gamma, a} \text{ sCtx-tyEnvTy}$$

$$\frac{\Gamma_C;\Gamma \vdash^M_Q TC\,\tau \rightsquigarrow Q \qquad \delta \notin \mathbf{dom}(\Gamma) \qquad \vdash^M_{ctx} \bullet;\Gamma_C;\Gamma \rightsquigarrow \bullet;\Gamma_C;\Gamma}{\vdash^M_{ctx} \bullet;\Gamma_C;\Gamma,\delta : TC\,\tau \rightsquigarrow \bullet;\Gamma_C;\Gamma,\delta : Q}\ \text{sCtx-tyEnvD}$$

$$\frac{\begin{array}{c}\mathbf{unambig}(\forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau) \\ \Gamma_C;\bullet \vdash^M_C \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau \rightsquigarrow \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\sigma \qquad (m : \overline{Q}'_m \Rightarrow TC\,a : \forall \overline{a}_k.\overline{Q}_h \Rightarrow \tau') \in \Gamma_C \\ P;\Gamma_C;\bullet,\overline{b}_j,\overline{\delta}_i : \overline{Q}_i,\overline{a}_k,\overline{\delta}_h : [\tau/a]\overline{Q}_h \vdash^M_{tm} e \Leftarrow [\tau/a]\tau' \rightsquigarrow e \qquad \Gamma_C;\bullet,a \vdash^M_{ty} \forall \overline{a}_k.\overline{Q}_h \Rightarrow \tau' \rightsquigarrow \forall \overline{a}_k.\overline{Q}_h \Rightarrow \sigma' \\ D \notin \mathbf{dom}(P) \qquad (D' : \forall \overline{b}'_k.\overline{Q}''_h \Rightarrow TC\,\tau'').m' \mapsto \Gamma' : e' \notin P \qquad \mathbf{where}[\overline{\tau}_j/\overline{b}_j]\tau = [\overline{\tau}'_k/\overline{b}'_k]\tau'' \\ \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma \qquad \Sigma' = \Sigma,(D : \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\sigma).m \mapsto \Lambda \overline{b}_j.\lambda \overline{\delta}_i : \overline{Q}_i.\Lambda \overline{a}_k.\lambda \overline{\delta}_h : [\sigma/a]\overline{Q}_h.e\end{array}}{\vdash^M_{ctx} P,(D : \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau).m \mapsto \bullet,\overline{b}_j,\overline{\delta}_i : \overline{Q}_i,\overline{a}_k,\overline{\delta}_h : [\tau/a]\overline{Q}_h : e;\Gamma_C;\Gamma \rightsquigarrow \Sigma';\Gamma_C;\Gamma}\ \text{sCtx-pgmInst}$$

D F_D JUDGMENTS AND ELABORATION

D.1 F_D Type & Constraint Well-Formedness

$\boxed{\Gamma_C;\Gamma \vdash_Q Q \rightsquigarrow \sigma}$ *(F_D Dictionary Type Well-Formedness)*

$$\frac{\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \qquad \Gamma_C = \Gamma_{C1}, m : TC\, a : \sigma', \Gamma_{C2} \qquad \Gamma_{C1}; \bullet, a \vdash_{ty} \sigma' \rightsquigarrow \sigma'}{\Gamma_C;\Gamma \vdash_Q TC\,\sigma \rightsquigarrow [\sigma/a]\{m : \sigma'\}}\ \textsc{iQ-TC}$$

$\boxed{\Gamma_C;\Gamma \vdash_C C}$ *(F_D Constraint Well-Formedness)*

$$\frac{\overline{\Gamma_C;\Gamma,\bar{a}_j \vdash_Q Q_i \rightsquigarrow \sigma_i}^{i\in 1..n} \qquad \Gamma_C;\Gamma,\bar{a}_j \vdash_Q Q \rightsquigarrow \sigma \qquad \bar{a}_j \notin \Gamma}{\Gamma_C;\Gamma \vdash_C \forall \bar{a}_j.\bar{Q}_i \Rightarrow Q}\ \textsc{iC-Abs}$$

$\boxed{\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma}$ *(F_D Type Well-Formedness)*

$$\frac{}{\Gamma_C;\Gamma \vdash_{ty} Bool \rightsquigarrow Bool}\ \textsc{iTy-Bool} \qquad \frac{a \in \Gamma}{\Gamma_C;\Gamma \vdash_{ty} a \rightsquigarrow a}\ \textsc{iTy-Var} \qquad \frac{\Gamma_C;\Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \qquad \Gamma_C;\Gamma \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2}{\Gamma_C;\Gamma \vdash_{ty} \sigma_1 \to \sigma_2 \rightsquigarrow \sigma_1 \to \sigma_2}\ \textsc{iTy-Arrow}$$

$$\frac{\Gamma_C;\Gamma \vdash_Q Q \rightsquigarrow \sigma' \qquad \Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C;\Gamma \vdash_{ty} Q \Rightarrow \sigma \rightsquigarrow \sigma' \to \sigma}\ \textsc{iTy-Qual} \qquad \frac{\Gamma_C;\Gamma, a \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C;\Gamma \vdash_{ty} \forall a.\sigma \rightsquigarrow \forall a.\sigma}\ \textsc{iTy-Scheme}$$

D.2 Dictionary Typing

$\boxed{\Sigma;\Gamma_C;\Gamma \vdash_d d : Q \rightsquigarrow e}$ *(Dictionary Typing)*

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_d \delta : Q \rightsquigarrow \delta}\ \textsc{D-Var}$$

$$\frac{\begin{array}{c}\Sigma = \Sigma_1, (D : \forall \bar{a}_j.\bar{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda \bar{a}_j.\lambda \bar{\delta}_i : \bar{Q}_i.e, \Sigma_2 \qquad \vdash_{ctx} \Sigma;\Gamma_C;\Gamma \qquad \overline{\Gamma_C;\bullet,\bar{a}_j \vdash_Q Q_i \rightsquigarrow \sigma'_i}^{i} \\ \overline{\Gamma_C;\Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^{j} \qquad \Sigma_1;\Gamma_C;\bullet,\bar{a}_j,\bar{\delta}_i : \bar{Q}_i \vdash_{tm} e : [\sigma_q/a]\sigma_m \rightsquigarrow e \qquad \overline{\Sigma;\Gamma_C;\Gamma \vdash_d d_i : [\bar{\sigma}_j/\bar{a}_j]Q_i \rightsquigarrow e_i}^{i}\end{array}}{\Sigma;\Gamma_C;\Gamma \vdash_d D\,\bar{\sigma}_j\,\bar{d}_i : TC\,[\bar{\sigma}_j/\bar{a}_j]\sigma_q \rightsquigarrow (\Lambda \bar{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{i}.\{m = e\})\,\bar{\sigma}_j\,\bar{e}_i}\ \textsc{D-Con}$$

D.3 F_D Term Typing

$\boxed{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma \rightsquigarrow e}$ *(F_D Term Typing)*

$$\frac{\vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} True : Bool \rightsquigarrow True}\ \textsc{iTm-True} \qquad \frac{\vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} False : Bool \rightsquigarrow False}\ \textsc{iTm-False} \qquad \frac{(x : \sigma) \in \Gamma \qquad \vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} x : \sigma \rightsquigarrow x}\ \textsc{iTm-Var}$$

$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1 \qquad \Sigma;\Gamma_C;\Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2 \qquad \Gamma_C;\Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2 : \sigma_2 \rightsquigarrow \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2}\ \textsc{iTm-Let}$$

$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_d d : TC\,\sigma \rightsquigarrow e \qquad (m : TC\,a : \sigma') \in \Gamma_C}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} d.m : [\sigma/a]\sigma' \rightsquigarrow e.m}\ \textsc{iTm-Method} \qquad \frac{\Sigma;\Gamma_C;\Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \rightsquigarrow e \qquad \Gamma_C;\Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \to \sigma_2 \rightsquigarrow \lambda x : \sigma_1.e}\ \textsc{iTm-ArrI}$$

$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \rightsquigarrow e_1 \qquad \Sigma;\Gamma_C;\Gamma \vdash_{tm} e_2 : \sigma_1 \rightsquigarrow e_2}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_1\,e_2 : \sigma_2 \rightsquigarrow e_1\,e_2}\ \textsc{iTm-ArrE}$$

$$\frac{\Sigma;\Gamma_C;\Gamma, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e \qquad \Gamma_C;\Gamma \vdash_Q Q \rightsquigarrow \sigma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma \rightsquigarrow \lambda\delta : \sigma.e}\ \textsc{iTm-ConstrI}$$

$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : Q \Rightarrow \sigma \rightsquigarrow e_1 \qquad \Sigma;\Gamma_C;\Gamma \vdash_d d : Q \rightsquigarrow e_2}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e\,d : \sigma \rightsquigarrow e_1\,e_2}\ \textsc{iTm-ConstrE} \qquad \frac{\Sigma;\Gamma_C;\Gamma, a \vdash_{tm} e : \sigma \rightsquigarrow e}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma \rightsquigarrow \Lambda a.e}\ \textsc{iTm-ForallI}$$

$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \forall a.\sigma' \rightsquigarrow e \qquad \Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e\,\sigma : [\sigma/a]\sigma' \rightsquigarrow e\,\sigma}\ \textsc{iTm-ForallE}$$

D.4 F_D Environment Well-Formedness

$\boxed{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}$ *(F_D Environment Well-Formedness)*

$$\frac{}{\vdash_{ctx} \bullet; \bullet; \bullet}\ \text{ICTX-EMPTY} \qquad \frac{\Gamma_C; \bullet, a \vdash_{ty} \sigma \rightsquigarrow \sigma \quad m \notin \mathbf{dom}(\Gamma_C) \quad TC\, b \notin \mathbf{dom}(\Gamma_C) \quad \vdash_{ctx} \bullet; \Gamma_C; \bullet}{\vdash_{ctx} \bullet; \Gamma_C, m : TC\, a : \sigma; \bullet}\ \text{ICTX-CLSENV}$$

$$\frac{\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \quad x \notin \mathbf{dom}(\Gamma) \quad \vdash_{ctx} \bullet; \Gamma_C; \Gamma}{\vdash_{ctx} \bullet; \Gamma_C; \Gamma, x : \sigma}\ \text{ICTX-TYENVTM} \qquad \frac{a \notin \Gamma \quad \vdash_{ctx} \bullet; \Gamma_C; \Gamma}{\vdash_{ctx} \bullet; \Gamma_C; \Gamma, a}\ \text{ICTX-TYENVTY}$$

$$\frac{\Gamma_C; \Gamma \vdash_Q TC\, \sigma \rightsquigarrow \sigma \quad \delta \notin \mathbf{dom}(\Gamma) \quad \vdash_{ctx} \bullet; \Gamma_C; \Gamma}{\vdash_{ctx} \bullet; \Gamma_C; \Gamma, \delta : TC\, \sigma}\ \text{ICTX-TYENVD}$$

$$\frac{\begin{array}{c}\mathbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\, \sigma) \\ \Gamma_C; \bullet \vdash_C \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\, \sigma \quad (m : TC\, a : \sigma') \in \Gamma_C \quad \Sigma; \Gamma_C; \bullet \vdash_{tm} e : \forall \overline{a}_j.\overline{Q}_i \Rightarrow [\sigma/a]\sigma' \rightsquigarrow e \\ D \notin \mathbf{dom}(\Sigma) \quad (D' : \forall \overline{a}'_m.\overline{Q}''_n \Rightarrow TC\, \sigma'').m' \mapsto e' \notin \Sigma \quad \mathbf{where}[\overline{\sigma}_j/\overline{a}_j]\sigma = [\overline{\sigma}'_m/\overline{a}'_m]\sigma'' \quad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma\end{array}}{\vdash_{ctx} \Sigma, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\, \sigma).m \mapsto e; \Gamma_C; \Gamma}\ \text{ICTX-MENV}$$

$\boxed{\mathbf{unambig}(C)}$ *(Unambiguity for Constraints)*

$$\frac{\overline{a}_j \in \mathbf{fv}(\sigma)}{\mathbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\, \sigma)}\ \text{IUNAMBIG-CONSTRAINT}$$

D.5 F_D Environment Elaboration

$\boxed{\Gamma_C; \Gamma \rightsquigarrow \Gamma}$ *(F_D-to-$F_{\{\}}$ environment translation)*

$$\frac{}{\Gamma_C; \bullet \rightsquigarrow \bullet}\ \text{CTX-EMPTY} \qquad \frac{\Gamma_C; \Gamma \rightsquigarrow \Gamma \quad \Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C; \Gamma, x : \sigma \rightsquigarrow \Gamma, x : \sigma}\ \text{CTX-VAR} \qquad \frac{\Gamma_C; \Gamma \rightsquigarrow \Gamma \quad \Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma}{\Gamma_C; \Gamma, \delta : Q \rightsquigarrow \Gamma, \delta : \sigma}\ \text{CTX-DVAR}$$

$$\frac{\Gamma_C; \Gamma \rightsquigarrow \Gamma}{\Gamma_C; \Gamma, a \rightsquigarrow \Gamma, a}\ \text{CTX-TVAR}$$

In the translation mechanism, we have assumed namespace translation functions which take a F_D type, term or dictionary-variable name and return the same identifier representing a $F_{\{\}}$ type or term variable. There are four such functions, each with a different namespace as domain:

Type variables: It translates a type variable of the F_D language, a, to the $F_{\{\}}$ type variable with the same name, a.
Term variables: Similar to type variables, but for the term sort.
Dictionary variables: It translates a dictionary variable, δ, to a $F_{\{\}}$ term variable with the same name.
Dictionary labels: It translates a dictionary method, m, to a record-field label, m, with the same name.

This identifier translation is assumed in all judgments that involve elaboration, such as the F_D term typing. When we regard identifiers, the font-color change implies such a translation. However, this convention is not used in other language sorts (types, non-variable terms, etc.). For example, two types with the same identifier but of different color mean only two types, a F_D and a $F_{\{\}}$ type, that are not related to each other. Any specification of the relation between the two types is given by the judgments they appear in.

D.6 F_D Operational Semantics

$\boxed{\Sigma \vdash e \longrightarrow e'}$ *(F_D Evaluation)*

$$\frac{\Sigma \vdash e_1 \longrightarrow e'_1}{\Sigma \vdash e_1\, e_2 \longrightarrow e'_1\, e_2}\ \text{IEVAL-APP} \qquad \frac{}{\Sigma \vdash (\lambda x : \sigma.e_1)\, e_2 \longrightarrow [e_2/x]e_1}\ \text{IEVAL-APPABS} \qquad \frac{\Sigma \vdash e \longrightarrow e'}{\Sigma \vdash e\, \sigma \longrightarrow e'\, \sigma}\ \text{IEVAL-TYAPP}$$

$$\frac{}{\Sigma \vdash (\Lambda a.e)\, \sigma \longrightarrow [\sigma/a]e}\ \text{IEVAL-TYAPPABS} \qquad \frac{\Sigma \vdash e \longrightarrow e'}{\Sigma \vdash e\, d \longrightarrow e'\, d}\ \text{IEVAL-DAPP} \qquad \frac{}{\Sigma \vdash (\lambda \delta : Q.e)\, d \longrightarrow [d/\delta]e}\ \text{IEVAL-DAPPABS}$$

$$\frac{(D : C).m \mapsto e \in \Sigma}{\Sigma \vdash (D\, \overline{\sigma}\, \overline{d}).m \longrightarrow e\, \overline{\sigma}\, \overline{d}}\ \text{IEVAL-METHOD} \qquad \frac{}{\Sigma \vdash \mathbf{let}\ x : \sigma = e_1\ \mathbf{in}\ e_2 \longrightarrow [e_1/x]e_2}\ \text{IEVAL-LET}$$

E $F_{\{\}}$ JUDGMENTS

E.1 $F_{\{\}}$ Type Well-Formedness

$\boxed{\Gamma \vdash_{ty} \sigma}$ *(Well-formed $F_{\{\}}$ types)*

$$\frac{}{\Gamma \vdash_{ty} Bool}\ \text{TTY-BOOL} \qquad \frac{}{\Gamma \vdash_{ty} a}\ \text{TTY-VAR} \qquad \frac{\Gamma, a \vdash_{ty} \sigma}{\Gamma \vdash_{ty} \forall a.\sigma}\ \text{TTY-ABS} \qquad \frac{\Gamma \vdash_{ty} \sigma_1 \quad \Gamma \vdash_{ty} \sigma_2}{\Gamma \vdash_{ty} \sigma_1 \to \sigma_2}\ \text{TTY-ARR} \qquad \frac{\overline{\Gamma \vdash_{ty} \sigma_i}^{\,i<n}}{\Gamma \vdash_{ty} \{\, \overline{m_i : \sigma_i}^{\,i<n} \,\}}\ \text{TTY-REC}$$

E.2 $F_{\{\}}$ Term Typing

$\boxed{\Gamma \vdash_{tm} e : \sigma}$ *(Well typed $F_{\{\}}$ terms)*

$$\frac{\vdash_{ctx} \Gamma}{\Gamma \vdash_{tm} True : Bool}\ \text{TTM-TRUE} \qquad \frac{\vdash_{ctx} \Gamma}{\Gamma \vdash_{tm} False : Bool}\ \text{TTM-FALSE} \qquad \frac{(x : \sigma) \in \Gamma \quad \vdash_{ctx} \Gamma}{\Gamma \vdash_{tm} x : \sigma}\ \text{TTM-VAR}$$

$$\frac{\Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2}{\Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \to \sigma_2}\ \text{TTM-ABS} \qquad \frac{\Gamma \vdash_{tm} e_1 : \sigma \to \sigma' \quad \Gamma \vdash_{tm} e_2 : \sigma}{\Gamma \vdash_{tm} e_1\, e_2 : \sigma'}\ \text{TTM-APP} \qquad \frac{\Gamma, a \vdash_{tm} e : \sigma}{\Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma}\ \text{TTM-TABS}$$

$$\frac{\Gamma \vdash_{tm} e : \forall a.\sigma_1 \quad \Gamma \vdash_{ty} \sigma_2}{\Gamma \vdash_{tm} e\, \sigma : [\sigma_2/a]\sigma_1}\ \text{TTM-TAPP} \qquad \frac{\overline{\Gamma \vdash_{tm} e_i : \sigma_i}^{\,i<n}}{\Gamma \vdash_{tm} \{\, \overline{m_i = e_i}^{\,i<n} \,\} : \{\, \overline{m_i : \sigma_i}^{\,i<n} \,\}}\ \text{TTM-REC} \qquad \frac{\Gamma \vdash_{tm} e : \{\, \overline{m_i : \sigma_i}^{\,i<n} \,\}}{\Gamma \vdash_{tm} e.m_j : \sigma_j}\ \text{TTM-PROJ}$$

$$\frac{\Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \quad \Gamma \vdash_{tm} e_1 : \sigma_1}{\Gamma \vdash_{tm} \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2 : \sigma_2}\ \text{TTM-LET}$$

E.3 $F_{\{\}}$ Environment Well-Formedness

$\boxed{\vdash_{ctx} \Gamma}$ *(Well-formed $F_{\{\}}$ environment)*

$$\frac{}{\vdash_{ctx} \bullet}\ \text{TCX-EMPTY} \qquad \frac{\vdash_{ctx} \Gamma \quad a \notin \Gamma}{\vdash_{ctx} \Gamma, a}\ \text{TCX-TVAR} \qquad \frac{\vdash_{ctx} \Gamma \quad \Gamma \vdash_{ty} \sigma \quad x \notin \Gamma}{\vdash_{ctx} \Gamma, x : \sigma}\ \text{TCX-VAR}$$

E.4 $F_{\{\}}$ Operational Semantics

$\boxed{e \longrightarrow e'}$ *($F_{\{\}}$ evaluation)*

$$\frac{}{(\lambda x : \sigma.e_1)\, e_2 \longrightarrow [e_1/x]e_2}\ \text{TEVAL-APPABS} \qquad \frac{}{(\Lambda a.e)\, \sigma \longrightarrow [\sigma/a]e}\ \text{TEVAL-TAPPTABS} \qquad \frac{}{\{\, \overline{m_i = e_i}^{\,i\in 1..n} \,\}.m_j \longrightarrow e_j}\ \text{TEVAL-PROJ}$$

$$\frac{}{\mathbf{let}\ x : \sigma = e_1\ \mathbf{in}\ e_2 \longrightarrow [e_1/x]e_2}\ \text{TEVAL-LET} \qquad \frac{e_1 \longrightarrow e_1'}{e_1\, e_2 \longrightarrow e_1'\, e_2}\ \text{TEVAL-APP} \qquad \frac{e_1 \longrightarrow e_1'}{e_1\, \sigma \longrightarrow e_1'\, \sigma}\ \text{TEVAL-TAPP}$$

$$\frac{e \longrightarrow e'}{e.m_j \longrightarrow e'.m_j}\ \text{TEVAL-REC}$$

F CONTEXT TYPING

F.1 λ_{TC} Context Typing and Elaboration

$\boxed{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}$ *(λ_{TC} Context Inference - Inference)*

$$\frac{}{[\bullet] : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma \Rightarrow \tau) \rightsquigarrow [\bullet]}\ \text{sM-inf-infT-Empty}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_1 \to \tau_2) \rightsquigarrow M \qquad P;\Gamma_C;\Gamma' \vdash_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{M\,e_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M\,e_2}\ \text{sM-inf-infT-AppL}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau_1) \rightsquigarrow M \qquad P;\Gamma_C;\Gamma' \vdash_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1}{e_1\,M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow e_1\,M}\ \text{sM-inf-infT-AppR}$$

$$\frac{\begin{array}{c} M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma',\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \Leftarrow \tau_1) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P;\Gamma_C;\Gamma',x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \vdash_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \\ \Gamma_C;\Gamma' \vdash_{ty} \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall\overline{a}_j.\overline{\sigma}_i \to \sigma_1 \qquad M' = \textbf{let}\ x : \forall\overline{a}_j.\overline{\sigma}_i \to \sigma_1 = \Lambda\overline{a}_j.\lambda\overline{\delta_i : \sigma_i}^i.M\,\textbf{in}\ e_2 \end{array}}{\textbf{let}\ x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M\,\textbf{in}\ e_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \text{sM-inf-infT-LetL}$$

$$\frac{\begin{array}{c} M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma',x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \Rightarrow \tau_2) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P;\Gamma_C;\Gamma',\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \vdash_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ \Gamma_C;\Gamma' \vdash_{ty} \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall\overline{a}_j.\overline{\sigma}_i \to \sigma_1 \qquad M' = \textbf{let}\ x : \forall\overline{a}_j.\overline{\sigma}_i \to \sigma_1 = \Lambda\overline{a}_j.\lambda\overline{\delta_i : \sigma_i}^i.e_1\,\textbf{in}\ M \end{array}}{\textbf{let}\ x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\,\textbf{in}\ M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \text{sM-inf-infT-LetR}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}{M :: \tau' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}\ \text{sM-inf-infT-Ann}$$

$\boxed{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}$ *(λ_{TC} Context Inference - Checking)*

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau_1) \mapsto (P;\Gamma_C;\Gamma',x : \tau \Leftarrow \tau_2) \rightsquigarrow M \qquad \Gamma_C;\Gamma' \vdash_{ty} \tau \rightsquigarrow \sigma}{\lambda x.M : (P;\Gamma_C;\Gamma \Rightarrow \tau_1) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau \to \tau_2) \rightsquigarrow \lambda x : \sigma.M}\ \text{sM-inf-checkT-Abs}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}\ \text{sM-inf-checkT-Inf}$$

$\boxed{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}$ *(λ_{TC} Context Checking - Inference)*

$$\frac{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_1 \to \tau_2) \rightsquigarrow M \qquad P;\Gamma_C;\Gamma' \vdash_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{M\,e_2 : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M\,e_2}\ \text{sM-check-infT-AppL}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau_1) \rightsquigarrow M \qquad P;\Gamma_C;\Gamma' \vdash_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1}{e_1\,M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow e_1\,M}\ \text{sM-check-infT-AppR}$$

$$\frac{\begin{array}{c} M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma',\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \Leftarrow \tau_1) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P;\Gamma_C;\Gamma',x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \vdash_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \\ \Gamma_C;\Gamma' \vdash_{ty} \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall\overline{a}_j.\overline{\sigma}_i \to \sigma_1 \qquad M' = \textbf{let}\ x : \forall\overline{a}_j.\overline{\sigma}_i \to \sigma_1 = \Lambda\overline{a}_j.\lambda\overline{\delta_i : \sigma_i}^i.M\,\textbf{in}\ e_2 \end{array}}{\textbf{let}\ x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M\,\textbf{in}\ e_2 : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \text{sM-check-infT-LetL}$$

$$\frac{\begin{array}{c} M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma', x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \Rightarrow \tau_2) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P;\Gamma_C;\Gamma',\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \vdash_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ \Gamma_C;\Gamma' \vdash_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{\sigma}_i \to \sigma_1 \qquad M' = \textbf{let}\ x : \forall \overline{a}_j.\overline{\sigma}_i \to \sigma_1 = \Lambda \overline{a}_j.\lambda\, \overline{\delta_i : \sigma_i}^{\,i}.e_1\ \textbf{in}\ M \end{array}}{\textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\ \textbf{in}\ M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \textsc{sM-check-infT-letR}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}{M :: \tau' : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}\ \textsc{sM-check-infT-ann}$$

$\boxed{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}$ *(λ_{TC} Context Checking - Checking)*

$$\frac{}{[\bullet] : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma \Leftarrow \tau) \rightsquigarrow [\bullet]}\ \textsc{sM-check-checkT-empty}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Leftarrow \tau_1) \mapsto (P;\Gamma_C;\Gamma', x : \tau \Leftarrow \tau_2) \rightsquigarrow M \qquad \Gamma_C;\Gamma' \vdash_{ty} \tau \rightsquigarrow \sigma}{\lambda x.M : (P;\Gamma_C;\Gamma \Leftarrow \tau_1) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau \to \tau_2) \rightsquigarrow \lambda x : \sigma.M}\ \textsc{sM-check-checkT-abs}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}{M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}\ \textsc{sM-check-checkT-inf}$$

F.2 λ_{TC} Context Typing and Elaboration through F_D

$\boxed{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}$ *(λ_{TC} Context Inference - Inference)*

$$\frac{}{[\bullet] : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma \Rightarrow \tau) \rightsquigarrow [\bullet]}\ \textsc{sM-inf-inf-empty}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_1 \to \tau_2) \rightsquigarrow M \qquad P;\Gamma_C;\Gamma' \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{M\, e_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M\, e_2}\ \textsc{sM-inf-inf-appL}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau_1) \rightsquigarrow M \qquad P;\Gamma_C;\Gamma' \vdash^M_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1}{e_1\, M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow e_1\, M}\ \textsc{sM-inf-inf-appR}$$

$$\frac{\begin{array}{c} M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma',\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \Leftarrow \tau_1) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P;\Gamma_C;\Gamma', x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \vdash^M_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \\ \Gamma_C;\Gamma' \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 \qquad M' = \textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.M\ \textbf{in}\ e_2 \end{array}}{\textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M\ \textbf{in}\ e_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \textsc{sM-inf-inf-letL}$$

$$\frac{\begin{array}{c} M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma', x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \Rightarrow \tau_2) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P;\Gamma_C;\Gamma',\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ \Gamma_C;\Gamma' \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 \qquad M' = \textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.e_1\ \textbf{in}\ M \end{array}}{\textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\ \textbf{in}\ M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \textsc{sM-inf-inf-letR}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}{M :: \tau' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}\ \textsc{sM-inf-inf-ann}$$

$\boxed{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}$ *(λ_{TC} Context Inference - Checking)*

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau_1) \mapsto (P;\Gamma_C;\Gamma', x : \tau \Leftarrow \tau_2) \rightsquigarrow M \qquad \Gamma_C;\Gamma' \vdash^M_{ty} \tau \rightsquigarrow \sigma}{\lambda x.M : (P;\Gamma_C;\Gamma \Rightarrow \tau_1) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau \to \tau_2) \rightsquigarrow \lambda x : \sigma.M}\ \textsc{sM-inf-check-abs}$$

$$\frac{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M}{M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M}\ \textsc{sM-inf-check-inf}$$

$\boxed{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M}$ (λ_{TC} *Context Checking - Inference*)

$$\frac{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_1 \to \tau_2) \rightsquigarrow M \qquad P; \Gamma_C; \Gamma' \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{M\, e_2 : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M\, e_2}\ \textsc{sM-check-inf-appL}$$

$$\frac{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_1) \rightsquigarrow M \qquad P; \Gamma_C; \Gamma' \vdash^M_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1}{e_1\, M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow e_1\, M}\ \textsc{sM-check-inf-appR}$$

$$\frac{\begin{array}{c} M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \Leftarrow \tau_1) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P; \Gamma_C; \Gamma', x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \vdash^M_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \\ \Gamma_C; \Gamma' \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 \qquad M' = \textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.M\ \textbf{in}\ e_2 \end{array}}{\textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M\ \textbf{in}\ e_2 : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \textsc{sM-check-inf-letL}$$

$$\frac{\begin{array}{c} M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma', x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \Rightarrow \tau_2) \rightsquigarrow M \\ \overline{\delta}_i\ \textbf{fresh} \qquad x \notin \textbf{dom}(\Gamma') \qquad P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ \Gamma_C; \Gamma' \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 \qquad M' = \textbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.e_1\ \textbf{in}\ M \end{array}}{\textbf{let}\ x : \sigma_1 = e_1\ \textbf{in}\ M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M'}\ \textsc{sM-check-inf-letR}$$

$$\frac{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M}{M :: \tau' : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M}\ \textsc{sM-check-inf-ann}$$

$\boxed{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M}$ (λ_{TC} *Context Checking - Checking*)

$$\frac{}{[\bullet] : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma \Leftarrow \tau) \rightsquigarrow [\bullet]}\ \textsc{sM-check-check-empty}$$

$$\frac{M : (P; \Gamma_C; \Gamma \Leftarrow \tau_1) \mapsto (P; \Gamma_C; \Gamma', x : \tau \Leftarrow \tau_2) \rightsquigarrow M \qquad \Gamma_C; \Gamma' \vdash^M_{ty} \tau \rightsquigarrow \sigma}{\lambda x.M : (P; \Gamma_C; \Gamma \Leftarrow \tau_1) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau \to \tau_2) \rightsquigarrow \lambda x : \sigma.M}\ \textsc{sM-check-check-abs}$$

$$\frac{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M}{M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M}\ \textsc{sM-check-check-inf}$$

F.3 F_D Context Typing and Elaboration

$\boxed{M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M}$ (F_D *Context Typing*)

$$\frac{}{[\bullet] : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \rightsquigarrow [\bullet]}\ \textsc{iM-empty}$$

$$\frac{M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma_1) \mapsto (\Sigma; \Gamma_C; \Gamma', x : \sigma \Rightarrow \sigma_2) \rightsquigarrow M \qquad \Gamma_C; \Gamma' \vdash_{ty} \sigma \rightsquigarrow \sigma}{\lambda x : \sigma.M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma_1) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma \to \sigma_2) \rightsquigarrow \lambda x : \sigma.M}\ \textsc{iM-abs}$$

$$\frac{M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_1 \to \sigma_2) \rightsquigarrow M \qquad \Sigma; \Gamma_C; \Gamma' \vdash_{tm} e_2 : \sigma_1 \rightsquigarrow e_2}{M\, e_2 : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_2) \rightsquigarrow M\, e_2}\ \textsc{iM-appL}$$

$$\frac{M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_1) \rightsquigarrow M \qquad \Sigma; \Gamma_C; \Gamma' \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \rightsquigarrow e_1}{e_1\, M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_2) \rightsquigarrow e_1\, M}\ \textsc{iM-appR}$$

$$\frac{M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma', \delta : Q \Rightarrow \sigma_1) \rightsquigarrow M \qquad \Gamma_C; \Gamma' \vdash_Q Q \rightsquigarrow \sigma}{\lambda \delta : Q.M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow Q \Rightarrow \sigma_1) \rightsquigarrow \lambda \delta : \sigma.M}\ \textsc{iM-dictAbs}$$

$$\frac{M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow Q \Rightarrow \sigma_1) \rightsquigarrow M \qquad \Sigma; \Gamma_C; \Gamma' \vdash_d d : Q \rightsquigarrow e}{M\, d : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_1) \rightsquigarrow M\, e}\ \textsc{iM-dictApp}$$

$$\frac{M : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma_1) \mapsto (\Sigma;\Gamma_C;\Gamma', a \Rightarrow \sigma_2) \rightsquigarrow M}{\Lambda a.M : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma_1) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \forall a.\sigma_2) \rightsquigarrow \Lambda a.M}\ \textsc{iM-tyAbs}$$

$$\frac{M : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma_1) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \forall a.\sigma_2) \rightsquigarrow M \qquad \Gamma_C;\Gamma' \vdash_{ty} \sigma \rightsquigarrow \sigma}{M\,\sigma : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma_1) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow [\sigma/a]\sigma_2) \rightsquigarrow M\,\sigma}\ \textsc{iM-tyApp}$$

$$\frac{M : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \sigma_1) \rightsquigarrow M \qquad \Sigma;\Gamma_C;\Gamma', x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2 \qquad \Gamma_C;\Gamma' \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1}{\mathbf{let}\ x : \sigma_1 = M\,\mathbf{in}\ e_2 : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \sigma_2) \rightsquigarrow \mathbf{let}\ x : \sigma_1 = M\,\mathbf{in}\ e_2}\ \textsc{iM-letL}$$

$$\frac{M : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma', x : \sigma_1 \Rightarrow \sigma_2) \rightsquigarrow M \qquad \Sigma;\Gamma_C;\Gamma' \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1 \qquad \Gamma_C;\Gamma' \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1}{\mathbf{let}\ x : \sigma_1 = e_1\,\mathbf{in}\ M : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \sigma_2) \rightsquigarrow \mathbf{let}\ x : \sigma_1 = e_1\,\mathbf{in}\ M}\ \textsc{iM-letR}$$

F.4 $F_{\{\}}$ Context Typing

$\boxed{M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma')}$ *($F_{\{\}}$ Context Typing)*

$$\frac{}{[\bullet] : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma \Rightarrow \sigma)}\ \textsc{tM-empty}$$

$$\frac{M : (\Gamma \Rightarrow \sigma_1) \mapsto (\Gamma', x : \sigma \Rightarrow \sigma_2) \qquad \Gamma' \vdash_{ty} \sigma}{\lambda x : \sigma.M : (\Gamma \Rightarrow \sigma_1) \mapsto (\Gamma' \Rightarrow \sigma \rightarrow \sigma_2)}\ \textsc{tM-abs}$$

$$\frac{M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_1 \rightarrow \sigma_2) \qquad \Gamma' \vdash_{tm} e_2 : \sigma_1}{M\,e_2 : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_2)}\ \textsc{tM-appL}$$

$$\frac{M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_1) \qquad \Gamma' \vdash_{tm} e_1 : \sigma_1 \rightarrow \sigma_2}{e_1\,M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_2)}\ \textsc{tM-appR}$$

$$\frac{M : (\Gamma \Rightarrow \sigma_1) \mapsto (\Gamma', a \Rightarrow \sigma_2)}{\Lambda a.M : (\Gamma \Rightarrow \sigma_1) \mapsto (\Gamma' \Rightarrow \forall a.\sigma_2)}\ \textsc{tM-tyAbs}$$

$$\frac{M : (\Gamma \Rightarrow \sigma_1) \mapsto (\Gamma' \Rightarrow \forall a.\sigma_2) \qquad \Gamma' \vdash_{ty} \sigma}{M\,\sigma : (\Gamma \Rightarrow \sigma_1) \mapsto (\Gamma' \Rightarrow [\sigma/a]\sigma_2)}\ \textsc{tM-tyApp}$$

$$\frac{M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_1) \qquad \Gamma', x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \qquad \Gamma' \vdash_{ty} \sigma_1}{\mathbf{let}\ x : \sigma_1 = M\,\mathbf{in}\ e_2 : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_2)}\ \textsc{tM-letL}$$

$$\frac{M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma', x : \sigma_1 \Rightarrow \sigma_2) \qquad \Gamma' \vdash_{tm} e_1 : \sigma_1 \qquad \Gamma' \vdash_{ty} \sigma_1}{\mathbf{let}\ x : \sigma_1 = e_1\,\mathbf{in}\ M : (\Gamma \Rightarrow \sigma) \mapsto (\Gamma' \Rightarrow \sigma_2)}\ \textsc{tM-letR}$$

G LOGICAL RELATIONS

In the definitions for the logical relations below, $\gamma = \gamma', \delta \mapsto (dv_1, dv_2)$ and $\phi = \phi', x \mapsto (e_1, e_2)$ are substitutions which map all dictionary variables $\delta \in \Gamma$ and term variables $x \in \Gamma$ onto two (possibly different) dictionary values and term values respectively. Notation-wise, we adopted the convention that γ_1 maps the dictionary variable δ to the leftmost value dv_1 and γ_2 substitutes δ for the rightmost value dv_2. Similarly for ϕ_1 and ϕ_2.

The third kind of substitution $R = R', a \mapsto (\sigma, r)$ maps all type variables $a \in \Gamma$ onto closed types σ, while also storing a relation r. This relation r is an arbitrary member of the set of all relations $Rel[\sigma]$ which offer the following property:

$$Rel[\sigma] = \{r \in \mathcal{P}(v \times v) \mid \forall (v_1, v_2) \in r : \Sigma; \Gamma_C; \bullet \vdash_{tm} v_1 : \sigma \wedge \Sigma; \Gamma_C; \bullet \vdash_{tm} v_2 : \sigma\}$$

G.1 Dictionary Relation

$\boxed{(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C}}$ (Closed Dictionary Value Relation)

$$\begin{aligned}(\Sigma_1 : D\,\overline{\sigma}_j\,\overline{dv_1}_i, \Sigma_2 : D\,\overline{\sigma}_j\,\overline{dv_2}_i) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \triangleq\ & \overline{(\Sigma_1 : dv_{1\,i}, \Sigma_2 : dv_{2\,i}) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j]Q_i]\!]_R^{\Gamma_C}}^{\,i} \\ & \wedge \Sigma_1; \Gamma_C; \bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv_1}_i : R(Q) \\ & \wedge \Sigma_2; \Gamma_C; \bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv_2}_i : R(Q) \\ & \textit{where } (D : \forall \overline{a}_j. \overline{Q}_i \Rightarrow Q').m \mapsto e_1 \in \Sigma_1 \wedge Q = [\overline{\sigma}_j/\overline{a}_j]Q'\end{aligned}$$

$\boxed{\Gamma_C; \Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : Q}$ (Logical Equivalence for Open Dictionaries)

$$\begin{aligned}\Gamma_C; \Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : Q \triangleq\ & \forall R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}, \\ & \phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1, \Sigma_2, \Gamma_C}, \\ & \gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1, \Sigma_2, \Gamma_C}, \\ & (\Sigma_1 : \gamma_1(\phi_1(R(d_1))), \Sigma_2 : \gamma_2(\phi_2(R(d_2)))) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C}\end{aligned}$$

G.2 Expression Relation

$\boxed{(\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C}}$ (Closed Expression Value Relation)

$$\begin{aligned}(\Sigma_1 : \mathit{True}, \Sigma_2 : \mathit{True}) &\in \mathcal{V}[\![\mathit{Bool}]\!]_R^{\Gamma_C} \\ (\Sigma_1 : \mathit{False}, \Sigma_2 : \mathit{False}) &\in \mathcal{V}[\![\mathit{Bool}]\!]_R^{\Gamma_C} \\ (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![a]\!]_R^{\Gamma_C} &\triangleq (a \mapsto (\sigma, r)) \in R \\ & \wedge \Sigma_1; \Gamma_C; \bullet \vdash_{tm} v_1 : \sigma \\ & \wedge \Sigma_2; \Gamma_C; \bullet \vdash_{tm} v_2 : \sigma \\ & \wedge (v_1, v_2) \in r \\ (\Sigma_1 : \lambda x : \sigma_1.e_1, \Sigma_2 : \lambda x : \sigma_1.e_2) \in \mathcal{V}[\![\sigma_1 \to \sigma_2]\!]_R^{\Gamma_C} &\triangleq \Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda x : \sigma.e_1 : R(\sigma_1 \to \sigma_2) \\ & \wedge \Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda x : \sigma.e_2 : R(\sigma_1 \to \sigma_2) \\ & \wedge \forall (\Sigma_1 : e_3, \Sigma_2 : e_4) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C} : \\ & (\Sigma_1 : (\lambda x : \sigma.e_1)\,e_3, \Sigma_2 : (\lambda x : \sigma.e_2)\,e_4) \in \mathcal{E}[\![\sigma_2]\!]_R^{\Gamma_C} \\ (\Sigma_1 : \lambda\delta : Q.e_1, \Sigma_2 : \lambda\delta : Q.e_2) \in \mathcal{V}[\![Q \Rightarrow \sigma]\!]_R^{\Gamma_C} &\triangleq \Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : Q.e_1 : R(Q \Rightarrow \sigma) \\ & \wedge \Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : Q.e_2 : R(Q \Rightarrow \sigma) \\ & \wedge \forall (\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} : \\ & (\Sigma_1 : (\lambda\delta : Q.e_1)\,dv_1, \Sigma_2 : (\lambda\delta : Q.e_2)\,dv_2) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \\ (\Sigma_1 : \Lambda a.e_1, \Sigma_2 : \Lambda a.e_2) \in \mathcal{V}[\![\forall a.\sigma]\!]_R^{\Gamma_C} &\triangleq \Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda a.e_1 : R(\forall a.\sigma) \\ & \wedge \Sigma_2; \Gamma_C; \bullet \vdash_{tm} \Lambda a.e_2 : R(\forall a.\sigma) \\ & \wedge \forall \sigma', \forall r \in Rel[\sigma'] : \\ & \Gamma_C; \bullet \vdash_{ty} \sigma' \Rightarrow \\ & (\Sigma_1 : (\Lambda a.e_1)\,\sigma', \Sigma_2 : (\Lambda a.e_2)\,\sigma') \in \mathcal{E}[\![\sigma]\!]_{R, a \mapsto (\sigma', r)}^{\Gamma_C}\end{aligned}$$

$\boxed{(\Sigma_1 : e_1, \Sigma_2 : e_2) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C}}$ (Closed Expression Relation)

$$(\Sigma_1 : e_1, \Sigma_2 : e_2) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \triangleq \Sigma_1; \Gamma_C; \bullet \vdash_{tm} e_1 : R(\sigma) \\ \land \Sigma_2; \Gamma_C; \bullet \vdash_{tm} e_2 : R(\sigma) \\ \land \exists v_1, v_2, \Sigma_1 \vdash e_1 \longrightarrow^* v_1, \Sigma_2 \vdash e_2 \longrightarrow^* v_2, (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C}$$

$\boxed{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma}$ (Logical Equivalence for Open Expressions)

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma \triangleq \forall R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}, \\ \phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}, \\ \gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}, \\ (\Sigma_1 : \gamma_1(\phi_1(R(e_1))), \Sigma_2 : \gamma_2(\phi_2(R(e_2)))) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C}$$

$\boxed{\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma' \Rightarrow \sigma')}$ (Logical Equivalence for Contexts)

$$\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma' \Rightarrow \sigma') \triangleq \forall e_1, e_2 : \Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma \\ \Rightarrow \Gamma_C; \Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma'$$

Definition 1 (Interpretation of type variables in type contexts).

$$\frac{}{\bullet \in \mathcal{F}[\![\bullet]\!]^{\Gamma_C}} \qquad \frac{R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}}{R \in \mathcal{F}[\![\Gamma, x : \sigma]\!]^{\Gamma_C}} \qquad \frac{R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C} \quad r \in Rel[\sigma] \quad \Gamma_C; \bullet \vdash_{ty} \sigma}{R, a \mapsto (\sigma, r) \in \mathcal{F}[\![\Gamma, a]\!]^{\Gamma_C}} \qquad \frac{R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}}{R \in \mathcal{F}[\![\Gamma, \delta : Q]\!]^{\Gamma_C}}$$

Definition 2 (Interpretation of term variables in type contexts).

$$\frac{}{\bullet \in \mathcal{G}[\![\bullet]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}} \qquad \frac{\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}{\phi \in \mathcal{G}[\![\Gamma, a]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}} \qquad \frac{\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \quad (\Sigma_1 : e_1, \Sigma_2 : e_2) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C}}{\phi, x \mapsto (e_1, e_2) \in \mathcal{G}[\![\Gamma, x : \sigma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}$$

$$\frac{\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}{\phi \in \mathcal{G}[\![\Gamma, \delta : Q]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}$$

Definition 3 (Interpretation of dictionary variables dictionary contexts).

$$\frac{}{\bullet \in \mathcal{H}[\![\bullet]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}} \qquad \frac{\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}{\gamma \in \mathcal{H}[\![\Gamma, a]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}} \qquad \frac{\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}{\gamma \in \mathcal{H}[\![\Gamma, x : \sigma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}$$

$$\frac{\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \quad (\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C}}{\gamma, \delta \mapsto (dv_1, dv_2) \in \mathcal{H}[\![\Gamma, \delta : Q]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}}$$

G.3 Environment Relation

$\boxed{\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2}$ *(Logical Equivalence for Environments)*

$$\frac{}{\Gamma_C \vdash \bullet \simeq_{log} \bullet}\ \text{ctxLog-empty} \qquad \frac{\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2 \quad \Gamma_C; \bullet \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma'}{\Gamma_C \vdash \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma).m \mapsto e_1 \simeq_{log} \Sigma_2, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma).m \mapsto e_2}\ \text{ctxLog-cons}$$

H STRONG NORMALIZATION RELATIONS

As opposed to Section G, the relations and substitutions in the strong normalization relations described below, are unary. The substitutions $\gamma^{SN} = \gamma^{SN}{}', \delta \mapsto d$ and $\phi^{SN} = \phi^{SN}{}', x \mapsto e$ map all dictionary variables $\delta \in \Gamma$ and term variables $x \in \Gamma$ onto well-typed dictionaries d and expressions $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. The final kind of substitution $R^{SN} = R^{SN}{}', a \mapsto (\sigma, r)$ maps all type variables $a \in \Gamma$ onto closed types σ, while also storing a relation r. This relation r is an arbitrary member of the set of all relations $Rel[\sigma]$ which offer the following property:

$$Rel[\sigma] = \{r \in \mathcal{P}(e) \mid \forall e \in r : \Sigma; \Gamma_C; \bullet \vdash_{tm} e : \sigma\}$$

We adopted the convention that $R^{SN}{}_1(a)$ maps the type variable a onto the closed type σ and $R^{SN}{}_2(a)$ denotes the contained set of expressions r.

$\boxed{e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}$ (Strong Normalization Relation)

$$\begin{aligned}
e \in \mathcal{SN}[\![Bool]\!]^{\Sigma,\Gamma_C}_{R^{SN}} &\triangleq \Sigma; \Gamma_C; \bullet \vdash_{tm} e : Bool \\
&\wedge \exists v : \Sigma \vdash e \longrightarrow^* v \\
e \in \mathcal{SN}[\![a]\!]^{\Sigma,\Gamma_C}_{R^{SN}} &\triangleq \Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}{}_1(a) \\
&\wedge \exists v : \Sigma \vdash e \longrightarrow^* v \\
&\wedge v \in R^{SN}{}_2(a) \\
e \in \mathcal{SN}[\![\sigma_1 \to \sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} &\triangleq \Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}{}_1(\sigma_1 \to \sigma_2) \\
&\wedge \exists v : \Sigma \vdash e \longrightarrow^* v \\
&\wedge \forall e' : e' \in \mathcal{SN}[\![\sigma_1]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \Rightarrow e\, e' \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \\
e \in \mathcal{SN}[\![Q \Rightarrow \sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} &\triangleq \Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}{}_1(Q \Rightarrow \sigma) \\
&\wedge \exists v : \Sigma \vdash e \longrightarrow^* v \\
&\wedge \forall d : \Sigma; \Gamma_C; \bullet \vdash_d d : R^{SN}{}_1(Q) \Rightarrow e\, d \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \\
e \in \mathcal{SN}[\![\forall a.\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} &\triangleq \Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}{}_1(\forall a.\sigma) \\
&\wedge \exists v : \Sigma \vdash e \longrightarrow^* v \\
&\wedge \forall \sigma', r \in Rel[\sigma'] : e\, \sigma' \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (\sigma', r)}
\end{aligned}$$

Definition 4 (Interpretation of type variables in type contexts for strong normalization).

$$\frac{}{\bullet \in \mathcal{F}^{SN}[\![\bullet]\!]^{\Sigma,\Gamma_C}} \qquad \frac{R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}}{R^{SN} \in \mathcal{F}^{SN}[\![\Gamma, x : \sigma]\!]^{\Sigma,\Gamma_C}}$$

$$\frac{R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C} \qquad r = \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \qquad \Gamma_C; \bullet \vdash_{ty} R^{SN}{}_1(\sigma)}{R^{SN}, a \mapsto (R^{SN}{}_1(\sigma), r) \in \mathcal{F}^{SN}[\![\Gamma, a]\!]^{\Sigma,\Gamma_C}} \qquad \frac{R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}}{R^{SN} \in \mathcal{F}^{SN}[\![\Gamma, \delta : Q]\!]^{\Sigma,\Gamma_C}}$$

Definition 5 (Interpretation of term variables in type contexts for strong normalization).

$$\frac{}{\bullet \in \mathcal{G}^{SN}[\![\bullet]\!]^{\Sigma,\Gamma_C}_{R^{SN}}} \qquad \frac{\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}{\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma, a]\!]^{\Sigma,\Gamma_C}_{R^{SN}}} \qquad \frac{\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \qquad e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}{\phi^{SN}, x \mapsto e \in \mathcal{G}^{SN}[\![\Gamma, x : \sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}$$

$$\frac{\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}{\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma, \delta : Q]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}$$

Definition 6 (Interpretation of dictionary variables dictionary contexts for strong normalization).

$$\frac{}{\bullet \in \mathcal{H}^{SN}[\![\bullet]\!]^{\Sigma,\Gamma_C}_{R^{SN}}} \qquad \frac{\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}{\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma, a]\!]^{\Sigma,\Gamma_C}_{R^{SN}}} \qquad \frac{\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}{\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma, x : \sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}$$

$$\frac{\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \qquad \Sigma; \Gamma_C; \bullet \vdash_d d : R^{SN}{}_1(Q)}{\gamma^{SN}, \delta \mapsto d \in \mathcal{H}^{SN}[\![\Gamma, \delta : Q]\!]^{\Sigma,\Gamma_C}_{R^{SN}}}$$

I EQUIVALENCE RELATIONS

I.1 Kleene Equivalence Relations

$\boxed{\Sigma_1 : e_1 \simeq \Sigma_2 : e_2}$ (Kleene Equivalence for F_{D} Expressions)

$$\Sigma_1 : e_1 \simeq \Sigma_2 : e_2 \triangleq \exists v : \Sigma_1 \vdash e_1 \longrightarrow^* v \land \Sigma_2 \vdash e_2 \longrightarrow^* v$$

$\boxed{e_1 \simeq e_2}$ (Kleene Equivalence for $F_{\{\}}$ Expressions)

$$e_1 \simeq e_2 \triangleq \exists v : e_1 \longrightarrow^* v \land e_2 \longrightarrow^* v$$

I.2 Contextual Equivalence Relations

$\boxed{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma}$ (Contextual Equivalence for F_{D} Expressions)

$$\begin{aligned}\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma \triangleq\ & \forall M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \bullet \Rightarrow Bool) \\ & \land \forall M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \\ & \land \Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \\ & \Rightarrow \Sigma_1 : M_1[e_1] \simeq \Sigma_2 : M_2[e_2]\end{aligned}$$

$\boxed{P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau}$ (Contextual Equivalence for $F_{\{\}}$ Expressions)

$$\begin{aligned}P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau \triangleq\ & \forall M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1 \\ & \land M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2 \\ & \Rightarrow M_1[e_1] \simeq M_2[e_2]\end{aligned}$$

$\boxed{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma}$ (Contextual Equivalence for $F_{\{\}}$ Expressions in F_{D} context)

$$\begin{aligned}\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma \triangleq\ & \forall M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1 \\ & \land \forall M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2 \\ & \land \Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \\ & \Rightarrow M_1[e_1] \simeq M_2[e_2]\end{aligned}$$

J λ_{TC} THEOREMS

J.1 Conjectures

We are confident that the following lemmas can be proven using well-known proof techniques.

Lemma 1 (Type Variable Substitution in λ_{TC} Constraint Typing).
If $\Gamma_C; \Gamma_1, a, \Gamma_2 \vdash^M_Q Q \rightsquigarrow Q$ *and* $\Gamma_C; \Gamma_1 \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *then* $\Gamma_C; \Gamma_1, [\tau/a]\Gamma_2 \vdash^M_Q [\tau/a]Q \rightsquigarrow [\sigma/a]Q$.

Lemma 2 (Type Well-Formedness Environment Weakening).
If $\Gamma_{C1}; \Gamma_1 \vdash^M_{ty} \sigma \rightsquigarrow \sigma$ *and* $\vdash^M_{ctx} \bullet; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \rightsquigarrow \bullet; \Gamma_C; \Gamma$ *then* $\Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \vdash^M_{ty} \sigma \rightsquigarrow \sigma$.

Lemma 3 (Class Constraint Well-Formedness Environment Weakening).
If $\Gamma_{C1}; \Gamma_1 \vdash^M_Q Q \rightsquigarrow Q'$ *and* $\vdash^M_{ctx} \bullet; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \rightsquigarrow \bullet; \Gamma_C; \Gamma$ *then* $\Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \vdash^M_Q Q \rightsquigarrow Q'$.

Lemma 4 (Constraint Well-Formedness Environment Weakening).
If $\Gamma_{C1}; \Gamma_1 \vdash^M_C C \rightsquigarrow C'$ *and* $\vdash^M_{ctx} \bullet; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \rightsquigarrow \bullet; \Gamma_C; \Gamma$ *then* $\Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \vdash^M_C C \rightsquigarrow C'$.

Lemma 5 (Context Well-Formedness Class Environment Weakening).
If $\vdash^M_{ctx} P; \Gamma_{C1}; \Gamma \rightsquigarrow \Sigma; \Gamma_{C1}; \Gamma$ *and* $\vdash^M_{ctx} \bullet; \Gamma_{C1}, \Gamma_{C2}; \Gamma \rightsquigarrow \bullet; \Gamma_{C1}, \Gamma_{C2}; \Gamma$ *then* $\vdash^M_{ctx} P; \Gamma_{C1}, \Gamma_{C2}; \Gamma \rightsquigarrow \Sigma; \Gamma_{C1}, \Gamma_{C2}; \Gamma$.

Lemma 6 (Context Well-Formedness Typing Environment Weakening).
If $\vdash^M_{ctx} P; \Gamma_C; \Gamma_1 \rightsquigarrow \Sigma; \Gamma_C; \Gamma_1$ *and* $\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma_1, \Gamma_2 \rightsquigarrow \bullet; \Gamma_C; \Gamma_1, \Gamma_2$ *then* $\vdash^M_{ctx} P; \Gamma_C; \Gamma_1, \Gamma_2 \rightsquigarrow \Sigma; \Gamma_C; \Gamma_1, \Gamma_2$.

J.2 Lemmas

Lemma 7 (Determinism of Context Typing).

- *If* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_1) \rightsquigarrow M_1$ *and* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M_2$ *then* $\tau_1 = \tau_2$.
- *If* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_1) \rightsquigarrow M_1$ *and* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_2) \rightsquigarrow M_2$ *then* $\tau_1 = \tau_2$.
- *If* $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_1) \rightsquigarrow M_1$ *and* $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M_2$ *then* $\tau_1 = \tau_2$.
- *If* $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_1) \rightsquigarrow M_1$ *and* $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_2) \rightsquigarrow M_2$ *then* $\tau_1 = \tau_2$.

Proof. By straightforward induction on the first typing derivation, in combination with case analysis on the second derivation. □

Lemma 8 (Class Constraint Elaboration to F_D Uniqueness).
If $\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q_1$ *and* $\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q_2$, *then* $Q_1 = Q_2$.

Proof. By mutual induction on both well-formedness derivations, together with Lemma 9. □

Lemma 9 (Type Elaboration to F_D Uniqueness).
If $\Gamma_C; \Gamma \vdash^M_{ty} \sigma \rightsquigarrow \sigma_1$ *and* $\Gamma_C; \Gamma \vdash^M_{ty} \sigma \rightsquigarrow \sigma_2$, *then* $\sigma_1 = \sigma_2$.

Proof. By mutual induction on both well-formedness derivations, together with Lemma 8. □

Lemma 10 (Constraint Elaboration to F_D Uniqueness).
If $\Gamma_C; \Gamma \vdash^M_C C \rightsquigarrow C_1$ *and* $\Gamma_C; \Gamma \vdash^M_C C \rightsquigarrow C_2$, *then* $C_1 = C_2$.

Proof. By straightforward induction on both well-formedness derivations, in combination with Lemma 8. □

Lemma 11 (Environment Elaboration to F_D Uniqueness). *If* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_{C1}; \Gamma_2$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_{C2}; \Gamma_2$*, then* $\Gamma_{C1} = \Gamma_{C2}$ *and* $\Gamma_1 = \Gamma_2$.

Proof. By straightforward induction on both well-formedness derivations, in combination with Lemmas 8, 9 and 10. □

Lemma 12 (Environment Well-Formedness of λ_{TC} Typing).

- *If* $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e$ *then* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$.
- *If* $P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e$ *then* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$.

Proof. By straightforward induction on the typing derivation. □

Lemma 13 (Environment Well-Formedness of λ_{TC} Typing through F_D).

- *If* $P; \Gamma_C; \Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e$ *then* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$.
- *If* $P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e$ *then* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$.

Proof. By straightforward induction on the typing derivation. □

Lemma 14 (Well-Formedness of λ_{TC} Typing Result).

- *If* $P; \Gamma_C; \Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e$ *then* $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$.
- *If* $P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e$ *then* $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$.

Proof. By straightforward induction on the typing derivation. □

Lemma 15 (Preservation of Environment Term Variables from λ_{TC} to F_D).

- *If* $(x : \sigma) \in \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $(x : \sigma) \in \Gamma$ *where* $\Gamma_C; \Gamma \vdash^M_{ty} \sigma \rightsquigarrow \sigma$.
- *If* $x \notin \textbf{dom}(\Gamma)$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $x \notin \textbf{dom}(\Gamma)$.

Proof. By straightforward induction on the environment elaboration derivation. □

Lemma 16 (Preservation of Environment Type Variables from λ_{TC} to F_D).

- *If* $a \in \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $a \in \Gamma$.
- *If* $a \notin \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $a \notin \Gamma$.

Proof. By straightforward induction on the environment elaboration derivation. □

Lemma 17 (Preservation of Environment Dictionary Variables from λ_{TC} to F_D).

- *If* $(\delta : Q) \in \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $(\delta : Q) \in \Gamma$ *where* $\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q$.
- *If* $\delta \notin \textbf{dom}(\Gamma)$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $\delta \notin \textbf{dom}(\Gamma)$.

Proof. By straightforward induction on the environment elaboration derivation. □

Lemma 18 (Environment Well-Formedness Strengthening). *If* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $\vdash_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \bullet$.

Proof. By case analysis on the hypothesis, the last rules used to construct it must be (possibly zero) consecutive applications of sCtxT-pgmInst. Revert those rules, to obtain $\vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma$. By further case analysis (sCtxT-tyEnvTm, sCtxT-tyEnvTy and sCtxT-tyEnvD), we get $\vdash_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet$. The goal follows by consecutively re-applying rule sCtxT-pgmInst with the appropriate premises. ◻

Lemma 19 (Environment Well-Formedness with F_D Elaboration Strengthening). *If* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *then* $\vdash^M_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \Sigma; \Gamma_C; \bullet$.

Proof. By case analysis on the hypothesis, the last rules used to construct it must be (possibly zero) consecutive applications of sCtx-pgmInst. Revert those rules, to obtain $\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma$. By further case analysis (sCtx-tyEnvTm, sCtx-tyEnvTy and sCtx-tyEnvD), we get $\vdash^M_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet$. The goal follows by consecutively re-applying rule sCtx-pgmInst with the appropriate premises. ◻

J.3 Typing Preservation

Theorem 1 (Typing Preservation - Expressions).

- *If* $P; \Gamma_C; \Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e$, *and* $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$, *then* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$, *and* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$.
- *If* $P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e$, *and* $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$, *then* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$, *and* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$.

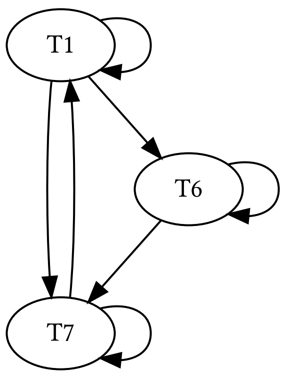

Fig. 14. Dependency graph for Theorems 1, 6 and 7

Proof. This theorem is mutually proven with Theorems 6 and 7. This mutual dependency is illustrated in Figure 14, where an arrow from A to B denotes A being dependent on B. Note that at the dependency from Theorem 7 to 1, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Consequently, the size of P is strictly decreasing in every possible cycle. The induction thus remains well-founded.

By applying Lemma 13 to the first hypothesis, we get:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \quad (1)$$

We continue by induction on the lexicographic order of the tuple (size of the expression, typing mode). Regarding typing mode, we define type checking to be larger than type inference. In each mutual dependency, we know that the tuple size decreases, meaning that the induction is well-founded.

Part 1

sTm-inf-true

$$\frac{\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma}{P; \Gamma_C; \Gamma \vdash^M_{tm} True \Rightarrow Bool \rightsquigarrow True} \text{ sTm-inf-true}$$

By sTy-bool, we know that:

$$\Gamma_C; \Gamma \vdash^M_{ty} Bool \rightsquigarrow Bool$$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} True : Bool$$

From Theorem 7, we know that:

$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$$

The goal follows from iTm-true.

sTm-inf-false
$$\frac{\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma}{P; \Gamma_C; \Gamma \vdash^M_{tm} \mathit{False} \Rightarrow \mathit{Bool} \rightsquigarrow \mathit{False}} \text{ sTm-inf-false}$$

Similar to the sTm-inf-true case.

sTm-inf-let

$$\frac{\begin{array}{c} x \notin \mathbf{dom}(\Gamma) \qquad \mathbf{unambig}(\forall \overline{a_j}.\overline{Q_i} \Rightarrow \tau_1) \qquad \mathbf{closure}(\Gamma_C; \overline{Q_i}) = \overline{Q_k} \\ \Gamma_C; \Gamma \vdash^M_{ty} \forall \overline{a_j}.\overline{Q_k} \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma \qquad \overline{\delta_k}\ \mathbf{fresh} \qquad P; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \\ P; \Gamma_C; \Gamma, x : \forall \overline{a_j}.\overline{Q_k} \Rightarrow \tau_1 \vdash^M_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \qquad e = \mathbf{let}\ x : \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma = \Lambda \overline{a_j}.\lambda \overline{\delta_k : Q_k}.e_1\ \mathbf{in}\ e_2 \end{array}}{P; \Gamma_C; \Gamma \vdash^M_{tm} \mathbf{let}\ x : \forall \overline{a_j}.\overline{Q_i} \Rightarrow \tau_1 = e_1\ \mathbf{in}\ e_2 \Rightarrow \tau_2 \rightsquigarrow e} \text{ sTm-inf-let}$$

Given

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2$$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \mathbf{let}\ x : \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma = \Lambda \overline{a_j}.\lambda \overline{\delta_k : Q_k}.e_1\ \mathbf{in}\ e_2 : \sigma_2$$

By case analysis on Equation 1 (sCtx-pgmInst), we know:

$$\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma \tag{2}$$

From the rule premise we know that:

$$\Gamma_C; \Gamma \vdash^M_{ty} \forall \overline{a_j}.\overline{Q_k} \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma \tag{3}$$

Applying Theorem 5 to Equations 2 and 3, we get that:

$$\Gamma_C; \Gamma \vdash_{ty} \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma \tag{4}$$

By repeated case analysis on Equation 3 (sTy-scheme and sTy-qual), we get that:

$$\overline{a_j} \notin \Gamma$$

$$\overline{\frac{}{\Gamma_C; \Gamma, \overline{a_j} \vdash^M_Q Q_k \rightsquigarrow Q_k}}^{k}$$

Applying these results, together with Equation 2, to sCtx-tyEnvTy and sCtx-tyEnvD, we get:

$$\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \rightsquigarrow \bullet; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \tag{5}$$

By weakening (Lemma 6) on Equations 1 and 5, we get:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \rightsquigarrow \Sigma; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \tag{6}$$

The rule premise also gives us that:

$$P; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \tag{7}$$

By applying induction hypothesis with Equations 3 and 7, we get that:

$$\Sigma; \Gamma_C; \Gamma, \overline{a_j}, \overline{\delta_k : Q_k} \vdash_{tm} e_1 : \sigma$$

Because of iTm-constrI and iTm-forallI, it is equivalent to say that:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \Lambda \overline{a_j}.\lambda \overline{\delta_k : Q_k}.e_1 : \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma \tag{8}$$

Through a similar analysis, we get that:

$$\Sigma; \Gamma_C; \Gamma, x : \forall \overline{a_j}.\overline{Q_k} \Rightarrow \sigma \vdash_{tm} e_2 : \sigma_2 \tag{9}$$

By iTm-let, in combination with Equations 4, 8 and 9, the goal has been proven.

sTm-inf-ArrE

$$\frac{P;\Gamma_C;\Gamma \vdash^M_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1 \qquad P;\Gamma_C;\Gamma \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2}{P;\Gamma_C;\Gamma \vdash^M_{tm} e_1\, e_2 \Rightarrow \tau_2 \rightsquigarrow e_1\, e_2}\ \textsc{sTm-inf-ArrE}$$

From the rule premise:

$$P;\Gamma_C;\Gamma \vdash^M_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1 \tag{10}$$

$$P;\Gamma_C;\Gamma \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2 \tag{11}$$

The goal to be proven is the following:

$$\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_1\, e_2 : \sigma_2$$

where $\Gamma_C;\Gamma \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2$.
Because the typing result is well-formed (Lemma 14), we know:

$$\Gamma_C;\Gamma \vdash^M_{ty} \tau_1 \to \tau_2 \rightsquigarrow \sigma_1 \to \sigma_2$$

By applying the induction hypothesis on Equations 10 and 11, we know respectively:

$$\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2$$

$$\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_2 : \sigma_1$$

The goal follows from ɪTm-ArrE.

sTm-inf-Ann

$$\frac{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e}{P;\Gamma_C;\Gamma \vdash^M_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e}\ \textsc{sTm-inf-Ann}$$

Follows directly from the induction hypothesis.

Part 2

sTm-check-Var

$$\frac{\textbf{unambig}(\forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \qquad \overline{P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i}^{\,i} \quad \overset{(x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \in \Gamma}{\overline{\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j}} \qquad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma}{P;\Gamma_C;\Gamma \vdash^M_{tm} x \Leftarrow [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow x\, \overline{\sigma}_j\, \overline{d}_i}\ \textsc{sTm-check-Var}$$

From the rule premise:

$$(x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \in \Gamma$$

By repeated case analysis on Equation 1 (sCtx-pgmInst), we get that:

$$\vdash^M_{ctx} \bullet;\Gamma_C;\Gamma \rightsquigarrow \bullet;\Gamma_C;\Gamma \tag{12}$$

By case analysis on Equation 12 (sCtx-tyEnvTm), we know:

$$(x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \in \Gamma \tag{13}$$

$$\Gamma_C;\Gamma_1 \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma \tag{14}$$

where $\Gamma = \Gamma_1, x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau, \Gamma_2$.
By applying Lemma 15 to Equation 13, we get:

$$(x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma) \in \Gamma \tag{15}$$

Furthermore, from the rule premise, we know that:

$$\overline{\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \tag{16}$$

By Typing Preservation - Types (Theorem 5), together with Equation 12, we have:

$$\overline{\Gamma_C;\Gamma \vdash_{ty} \sigma_j}^{\,j} \tag{17}$$

Similarly, the rule premise tells us that:

$$\overline{P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i}^{\,i} \tag{18}$$

By applying weakening (Lemma 2) to Equation 14, we get:

$$\Gamma_C;\Gamma \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma \tag{19}$$

By repeated case analysis on Equation 19 (sTy-qual), we get that:

$$\overline{\Gamma_C; \Gamma, \overline{a}_j \vdash^M_Q Q_i \rightsquigarrow Q_i}^{\,i} \tag{20}$$

By applying Lemma 1 on Equations 20 and 16, we get:

$$\overline{\Gamma_C; \Gamma \vdash^M_Q [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{21}$$

By Typing Preservation - Constraints Proving (Theorem 6), applied to Equations 18, 1 and 21, we have:

$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{22}$$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} x\, \overline{\sigma}_j\, \overline{d}_i : [\overline{\sigma}_j/\overline{a}_j]\sigma$$

where $\Gamma_C; \Gamma \vdash^M_{ty} [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow [\overline{\sigma}_j/\overline{a}_j]\sigma$.
From Equation 15, by applying iTm-var, we get

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma \tag{23}$$

By Equations 17, 23 and iTm-forallE, we get:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} x\, \overline{\sigma}_j : [\overline{\sigma}_j/\overline{a}_j]\overline{Q}_i \Rightarrow [\overline{\sigma}_j/\overline{a}_j]\sigma \tag{24}$$

By Equations 22 and 24, in combination with rule iTm-constrE, we get

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} x\, \overline{\sigma}_j\, \overline{d}_i : [\overline{\sigma}_j/\overline{a}_j]\sigma \tag{25}$$

which is exactly the goal.

sTm-check-meth

$$\frac{\begin{array}{c}(m : \overline{Q}'_k \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau') \in \Gamma_C \qquad \mathbf{unambig}(\forall \overline{a}_j, a.\overline{Q}_i \Rightarrow \tau') \qquad P; \Gamma_C; \Gamma \vDash^M TC\,\tau \rightsquigarrow d \\ \Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \qquad \overline{P; \Gamma_C; \Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j][\tau/a]Q_i \rightsquigarrow d_i}^{\,i} \qquad \overline{\Gamma_C; \Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \qquad \vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma\end{array}}{P; \Gamma_C; \Gamma \vdash^M_{tm} m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow d.m\, \overline{\sigma}_j\, \overline{d}_i}\ \textsc{sTm-check-meth}$$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m\, \overline{\sigma}_j\, \overline{d}_i : [\overline{\sigma}_j/\overline{a}_j][\sigma/a]\sigma' \tag{26}$$

where $\Gamma_C; \bullet, \overline{a}_j, a \vdash^M_{ty} \tau' \rightsquigarrow \sigma'$.
From the rule premise, we get that:

$$P; \Gamma_C; \Gamma \vDash^M TC\,\tau \rightsquigarrow d \tag{27}$$

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \tag{28}$$

By repeated case analysis on Equation 1 (sCtx-clsEnv), together with the first rule premise, we get:

$$\Gamma_{C1}; \bullet, a \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau' \rightsquigarrow \sigma''$$

where $\Gamma_C = \Gamma_{C1}, m : \overline{Q}'_k \Rightarrow TC\,a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau', \Gamma_{C2}$.
Following sQ-TC, in combination with this result, Equation 28 and the first rule premise, we have:

$$\Gamma_C; \Gamma \vdash^M_Q TC\,\tau \rightsquigarrow TC\,\sigma \tag{29}$$

Applying Typing Preservation - Constraints Proving (Theorem 6) on Equations 27 and 29, we get:

$$\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \tag{30}$$

Furthermore, we know from the rule premise that:

$$(m : \overline{Q}'_k \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau') \in \Gamma_C$$

Consequently, by repeated case analysis on Equation 1 (sCtx-clsEnv), we know that:

$$(m : TC\,a : \sigma'') \in \Gamma_C \tag{31}$$

By Equations 30 and 31, in combination with rule iTm-method, we get:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma''$$

The rest of the proof is similar to case sTm-check-var.

sTm-check-ArrI

$$\frac{x \notin \mathbf{dom}(\Gamma) \qquad P;\Gamma_C;\Gamma, x:\tau_1 \vdash^M_{tm} e \Leftarrow \tau_2 \rightsquigarrow e \qquad \Gamma_C;\Gamma \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma}{P;\Gamma_C;\Gamma \vdash^M_{tm} \lambda x.e \Leftarrow \tau_1 \to \tau_2 \rightsquigarrow \lambda x:\sigma.e}\ \textsc{sTm-check-ArrI}$$

The second hypothesis is:

$$\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$$

It is easy to verify that

$$\vdash^M_{ctx} P;\Gamma_C;\Gamma, x:\tau_1 \rightsquigarrow \Sigma;\Gamma_C;\Gamma, x:\sigma$$

The goal follows directly by applying the induction hypothesis, in combination with rule iTm-arrI.

sTm-check-Inf

$$\frac{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Rightarrow \tau \rightsquigarrow e}{P;\Gamma_C;\Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e}\ \textsc{sTm-check-Inf}$$

Follows directly from the induction hypothesis.

□

Theorem 2 (Typing Preservation - Instance). *If* $P;\Gamma_C \vdash^M_{inst} inst : P'$, *and* $\vdash^M_{ctx} P;\Gamma_C;\bullet \rightsquigarrow \Sigma;\Gamma_C;\bullet$ *then we have* $\vdash^M_{ctx} P,P';\Gamma_C;\bullet \rightsquigarrow \Sigma,\Sigma';\Gamma_C;\bullet$.

Proof. We restate the rule for typing instance declarations for reference:

$$\frac{\begin{array}{c}(m:\overline{Q'_i} \Rightarrow TC\ a : \forall \overline{a_j}.\overline{Q_h} \Rightarrow \tau_1) \in \Gamma_C \qquad \overline{b_k} = \mathbf{fv}(\tau) \qquad \overline{\dfrac{}{\Gamma_C;\bullet,\overline{b_k} \vdash^M_Q Q_q \rightsquigarrow Q_q}}^{\,q} \\ \Gamma_C;\bullet,\overline{b_k} \vdash^M_{ty} \tau \rightsquigarrow \sigma \qquad \mathbf{closure}(\Gamma_C;\overline{Q_p}) = \overline{Q_q} \qquad \mathbf{unambig}(\forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\tau) \\ \overline{P;\Gamma_C;\bullet,\overline{b_k},\overline{\delta_q}:\overline{Q_q} \vDash^M [\tau/a]Q'_i \rightsquigarrow d_i}^{\,i} \qquad P;\Gamma_C;\bullet,\overline{b_k},\overline{\delta_q}:\overline{Q_q},\overline{a_j},\overline{\delta_h}:[\tau/a]\overline{Q_h} \vdash^M_{tm} e \Leftarrow [\tau/a]\tau_1 \rightsquigarrow e \\ D\ \mathbf{fresh} \qquad \overline{\delta_h}\ \mathbf{fresh} \qquad \overline{\delta_q}\ \mathbf{fresh} \qquad (D' : \forall\overline{b'_m}.\overline{Q'_n} \Rightarrow TC\,\tau_2).m' \mapsto \Gamma' : e' \notin P \textit{ where } [\overline{\tau'_m}/\overline{b'_m}]\tau_2 = [\overline{\tau'_k}/\overline{b_k}]\tau \\ P' = (D:\forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\tau).m \mapsto \bullet,\overline{b_k},\overline{\delta_q}:\overline{Q_q},\overline{a_j},\overline{\delta_h}:[\tau/a]\overline{Q_h} : e\end{array}}{P;\Gamma_C \vdash^M_{inst} \mathbf{instance}\ \overline{Q_p} \Rightarrow TC\,\tau\ \mathbf{where}\ \{m = e\} : P'}\ \textsc{sInst-inst}$$

By inversion of rule sInst-inst, we know that:

$$P' = (D:\forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\tau).m \mapsto \bullet,\overline{b_k},\overline{\delta_q}:\overline{Q_q},\overline{a_j},\overline{\delta_h}:[\tau/a]\overline{Q_h} : e$$

Therefore our goal is

$$\vdash^M_{ctx} P,(D:\forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\tau).m \mapsto \bullet,\overline{b_k},\overline{\delta_q}:\overline{Q_q},\overline{a_j},\overline{\delta_h}:[\tau/a]\overline{Q_h} : e;\Gamma_C;\bullet \rightsquigarrow \Sigma,\Sigma';\Gamma_C;\bullet \tag{32}$$

From the hypothesis, we know that:

$$\vdash^M_{ctx} P;\Gamma_C;\bullet \rightsquigarrow \Sigma;\Gamma_C;\bullet \tag{33}$$

Goal 32 follows directly from sCtx-pgmInst with $\Sigma' = (D:\forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\sigma).m \mapsto \Lambda\overline{b_k}.\lambda\overline{\delta_q}:\overline{Q_q}.\Lambda\overline{a_j}.\lambda\overline{\delta_h}:[\sigma/a]\overline{Q_h}.e$, if we can show the following:

$$\mathbf{unambig}(\forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\tau) \tag{34}$$

$$\Gamma_C;\bullet \vdash^M_C \forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\tau \rightsquigarrow \forall\overline{b_k}.\overline{Q_q} \Rightarrow TC\,\sigma \tag{35}$$

$$(m:\overline{Q'_i} \Rightarrow TC\ a : \forall\overline{a_j}.\overline{Q_h} \Rightarrow \tau_1) \in \Gamma_C \tag{36}$$

$$P;\Gamma_C;\bullet,\overline{b_k},\overline{\delta_q}:\overline{Q_q},\overline{a_j},\overline{\delta_h}:[\tau/a]\overline{Q_h} \vdash^M_{tm} e \Leftarrow [\tau/a]\tau_1 \rightsquigarrow e \tag{37}$$

$$\Gamma_C;\bullet,a \vdash^M_{ty} \forall\overline{a_j}.\overline{Q_h} \Rightarrow \tau_1 \rightsquigarrow \sigma' \tag{38}$$

$$D \notin \mathbf{dom}(P) \tag{39}$$

$$(D' : \forall\overline{b'_k}.\overline{Q''_h} \Rightarrow TC\,\tau'').m' \mapsto \Gamma' : e' \notin P \tag{40}$$

$$\mathbf{where}[\overline{\tau_j}/\overline{b_j}]\tau = [\overline{\tau'_k}/\overline{b'_k}]\tau'' \tag{41}$$

$$\vdash^M_{ctx} P;\Gamma_C;\bullet \rightsquigarrow \Sigma;\Gamma_C;\bullet \tag{42}$$

Goal 42 is exactly Equation 33, which we already have. Goals 34 and 36 follow directly from the premise of sInst-inst. The premise also tells us that D is freshly generated, which satisfies Goal 39. Similarly Goals 37, 40 and 41 can be proven directly from the rule premise.

From the premise, we know:

$$\Gamma_C; \bullet, \overline{b}_k \vdash^M_{ty} \tau \rightsquigarrow \sigma \quad (43)$$

$$\overline{\Gamma_C; \bullet, \overline{b}_k \vdash^M_Q Q_q \rightsquigarrow Q_q}^{\,q} \quad (44)$$

Goal 35 follows directly from the definition of well-formedness of constraints and types. Goals 37 and 38 remain to be proven.

From the rule premise, we know that

$$(m : \overline{Q}'_i \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_h \Rightarrow \tau_1) \in \Gamma_C \quad (45)$$

From the definition of well-formedness of the source context, we know that:

$$\Gamma_{C1}; \bullet, a \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_h \Rightarrow \tau_1 \rightsquigarrow \sigma'$$

$$\Gamma_C = \Gamma_{C1}, \Gamma_{C2}$$

By weakening of class environment (Lemma 2), we can prove Goal 38.

□

Theorem 3 (Typing Preservation - Classes). *If $\Gamma_C \vdash^M_{cls} cls : \Gamma_C'$, and $\vdash^M_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \Sigma; \Gamma_C; \bullet$, then we have $\vdash^M_{ctx} P; \Gamma_C, \Gamma_C'; \bullet \rightsquigarrow \Sigma; \Gamma_C, \Gamma_C'; \bullet$.*

Proof. We restate the rule for class declaration typing for reference:

$$\frac{\begin{array}{c} m \notin \mathbf{dom}(\Gamma_C) \qquad \mathbf{closure}(\Gamma_C; \overline{Q}_k) = \overline{Q}_p \qquad \Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a}_j.\overline{Q}_p \Rightarrow \tau \rightsquigarrow \sigma \qquad \mathbf{unambig}(\forall \overline{a}_j, a.\overline{Q}_p \Rightarrow \tau) \\ \overline{\Gamma_C; \bullet, a \vdash^M_Q TC_i\ a \rightsquigarrow Q_i}^{\,i} \qquad \nexists TC' : (m : \overline{Q}'_m \Rightarrow TC'\ b : \sigma') \in \Gamma_C \qquad \nexists m' : (m' : \overline{Q}'_m \Rightarrow TC\ a : \sigma') \in \Gamma_C \end{array}}{\Gamma_C \vdash^M_{cls} \mathbf{class}\ \overline{TC_i\ a} \Rightarrow TC\ a\ \mathbf{where}\ \{m : \forall \overline{a}_j.\overline{Q}_k \Rightarrow \tau\} : \bullet, m : \overline{TC_i\ a} \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_p \Rightarrow \tau} \text{ sCls-cls}$$

By case analysis, we know that Γ_C' is of the form

$$\bullet, m : \overline{TC_i\ a} \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_p \Rightarrow \tau$$

The goal to be proven is the following:

$$\vdash^M_{ctx} P; \Gamma_C, m : \overline{TC_i\ a} \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_p \Rightarrow \tau; \bullet \rightsquigarrow \Sigma; \Gamma_C, m : TC\ a : \sigma; \bullet \quad (46)$$

We can derive from sCtx-clsEnv that

$$\vdash^M_{ctx} \bullet; \Gamma_C, m : \overline{TC_i\ a} \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_p \Rightarrow \tau; \bullet \rightsquigarrow \bullet; \Gamma_C, m : TC\ a : \sigma; \bullet \quad (47)$$

assuming we can show that:

$$\Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a}_k.\overline{Q}_p \Rightarrow \tau \rightsquigarrow \sigma \quad (48)$$

$$\overline{a}_j, a = \mathbf{fv}(\tau) \quad (49)$$

$$\overline{\Gamma_C; \bullet, a \vdash^M_Q TC_i\ a \rightsquigarrow Q_i}^{\,i} \quad (50)$$

$$m \notin \mathbf{dom}(\Gamma_C) \quad (51)$$

$$TC\ b \notin \mathbf{dom}(\Gamma_C) \quad (52)$$

$$\vdash^M_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet \quad (53)$$

Goals 48 till 52 follow directly from the premises and from the hypothesis. Goal 53 follows by repeated inversion on the second hypothesis. Finally, Goal 46 follows from Equation 47 by the definition of environment well-formedness and the second hypothesis.

□

Theorem 4 (Typing Preservation - Programs). *If $P; \Gamma_C \vdash^M_{pgm} pgm : \tau; P'; \Gamma_C' \rightsquigarrow e$, and $\vdash^M_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \Sigma; \Gamma_C; \bullet$, and $\Gamma_C, \Gamma_C'; \bullet \vdash^M_{ty} \tau \rightsquigarrow \sigma$ then we have $\vdash^M_{ctx} P, P'; \Gamma_C, \Gamma_C'; \bullet \rightsquigarrow \Sigma, \Sigma'; \Gamma_C, \Gamma_C'; \bullet$, and we have $\Sigma, \Sigma'; \Gamma_C, \Gamma_C'; \bullet \vdash_{tm} e : \sigma$.*

Proof. By structural induction on the typing derivation.

sPgmCls

$$\frac{\Gamma_C \vdash^M_{cls} cls : \Gamma_C' \qquad P; \Gamma_C, \Gamma_C' \vdash^M_{pgm} pgm : \tau; P'; \Gamma_C'' \rightsquigarrow e}{P; \Gamma_C \vdash^M_{pgm} cls; pgm : \tau; P'; \Gamma_C', \Gamma_C'' \rightsquigarrow e} \text{ sPgm-cls}$$

We know that

$$\vdash^{M}_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \Sigma; \Gamma_C; \bullet$$

By inversion it follows that:

$$\vdash^{M}_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet$$

By Typing Preservation - Classes (Theorem 3), we know

$$\vdash^{M}_{ctx} \bullet; \Gamma_C, \Gamma_C{}'; \bullet \rightsquigarrow \bullet; \Gamma_C, \Gamma_C{}'; \bullet$$

Through weakening (Lemma 5), we know that

$$\vdash^{M}_{ctx} P; \Gamma_C, \Gamma_C{}'; \bullet \rightsquigarrow \Sigma; \Gamma_C, \Gamma_C{}'; \bullet$$

The goal follows directly from the induction hypothesis.

sPgm-Inst

$$\frac{P; \Gamma_C \vdash^{M}_{inst} inst : P' \qquad P, P'; \Gamma_C \vdash^{M}_{pgm} pgm : \tau; P''; \Gamma_C{}' \rightsquigarrow e}{P; \Gamma_C \vdash^{M}_{pgm} inst; pgm : \tau; P', P''; \Gamma_C{}' \rightsquigarrow e}\ \text{sPgm-inst}$$

We know that

$$\vdash^{M}_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \Sigma; \Gamma_C; \bullet$$

By Typing Preservation - Instance (Theorem 2), we know that

$$\vdash^{M}_{ctx} P, P'; \Gamma_C; \bullet \rightsquigarrow \Sigma, \Sigma'; \Gamma_C; \bullet \qquad (54)$$

The goal follows directly from the induction hypothesis.

sPgm-expr

$$\frac{P; \Gamma_C; \bullet \vdash^{M}_{tm} e \Rightarrow \tau \rightsquigarrow e}{P; \Gamma_C \vdash^{M}_{pgm} e : \tau; \bullet; \bullet \rightsquigarrow e}\ \text{sPgm-expr}$$

Follows directly from Typing Preservation - Expressions (Theorem 1).

□

Theorem 5 (Typing Preservation - Types and Class Constraints).

- *If* $\Gamma_C; \Gamma \vdash^{M}_{ty} \sigma \rightsquigarrow \sigma$, *and* $\vdash^{M}_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma$, *then* $\Gamma_C; \Gamma \vdash_{ty} \sigma$.
- *If* $\Gamma_C; \Gamma \vdash^{M}_{Q} Q \rightsquigarrow Q$, *and* $\vdash^{M}_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma$, *then* $\Gamma_C; \Gamma \vdash_{Q} Q$.

Proof. By induction on the lexicographic order of the tuple (size of Γ_C, the derivation height of type well-formedness and the constraint well-formedness). In each mutual dependency, the size of the tuple is decreasing, so we know that the induction is well-founded.

Part 1

sTy-arrow

$$\frac{}{\Gamma_C; \Gamma \vdash^{M}_{ty} Bool \rightsquigarrow Bool}\ \text{sTy-bool}$$

Follows directly by iTy-bool.

sTy-var

$$\frac{a \in \Gamma}{\Gamma_C; \Gamma \vdash^{M}_{ty} a \rightsquigarrow a}\ \text{sTy-var}$$

It is easy to verify that for any environment for which $\vdash^{M}_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ holds, $a \in \Gamma$ implies $a \in \Gamma$. Therefore, the goal follows from iTy-var.

sTy-arrow

$$\frac{\Gamma_C; \Gamma \vdash^{M}_{ty} \tau_1 \rightsquigarrow \sigma_1 \qquad \Gamma_C; \Gamma \vdash^{M}_{ty} \tau_2 \rightsquigarrow \sigma_2}{\Gamma_C; \Gamma \vdash^{M}_{ty} \tau_1 \rightarrow \tau_2 \rightsquigarrow \sigma_1 \rightarrow \sigma_2}\ \text{sTy-arrow}$$

By induction hypothesis, we get

$$\Gamma_C; \Gamma \vdash_{ty} \sigma_1$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma_2$$

The goal follows directly from iTy-Arrow.

sTy-qual
$$\frac{\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q \qquad \Gamma_C; \Gamma \vdash^M_{ty} \rho \rightsquigarrow \sigma}{\Gamma_C; \Gamma \vdash^M_{ty} Q \Rightarrow \rho \rightsquigarrow Q \Rightarrow \sigma}\ \text{sTy-qual}$$

By induction hypothesis, we get

$$\Gamma_C; \Gamma \vdash_{ty} \sigma$$

By Part 2 of this lemma, we get

$$\Gamma_C; \Gamma \vdash_Q Q$$

The goal follows directly from iTy-Qual.

sTy-scheme
$$\frac{a \notin \Gamma \qquad \Gamma_C; \Gamma, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma}{\Gamma_C; \Gamma \vdash^M_{ty} \forall a.\sigma \rightsquigarrow \forall a.\sigma}\ \text{sTy-scheme}$$

Given $\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma$, by sCtx-EnvTy, we know that $\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \bullet; \Gamma_C; \Gamma, a$. By induction hypothesis, we get

$$\Gamma_C; \Gamma, a \vdash_{ty} \sigma$$

The goal follows directly by iTy-scheme.

Part 2

sQ-TC
$$\frac{\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \qquad \Gamma_C = \Gamma_{C1}, m : \overline{Q}_i \Rightarrow TC\, a : \sigma, \Gamma_{C2} \qquad \Gamma_{C1}; \bullet, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma'}{\Gamma_C; \Gamma \vdash^M_Q TC\, \tau \rightsquigarrow TC\, \sigma}\ \text{sQ-TC}$$

By Part 1 of this lemma, we know that

$$\Gamma_C; \Gamma \vdash_{ty} \sigma \tag{55}$$

It is easy to verify that given any environment for which

$$\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma$$
$$\Gamma_C = \Gamma_{C1}, m : \overline{Q}_i \Rightarrow TC\, a : \sigma, \Gamma_{C2} \tag{56}$$

then

$$\Gamma_C = \Gamma_{C1}, m : TC\, a : \sigma', \Gamma_{C2}$$
$$\vdash^M_{ctx} \bullet; \Gamma_{C1}; \bullet \rightsquigarrow \bullet; \Gamma_{C1}; \bullet$$
$$\Gamma_{C1}; \bullet, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma'$$

By sCtx-TyEnvTy, we get

$$\vdash^M_{ctx} \bullet; \Gamma_{C1}; \bullet, a \rightsquigarrow \bullet; \Gamma_{C1}; \bullet, a$$

The size of Γ_{C1} is trivially smaller than Γ_C. So by induction hypothesis, we have

$$\Gamma_C; \bullet, a \vdash_{ty} \sigma' \tag{57}$$

The goal follows directly from sQ-TC, and Equations 55, 56, 57.

□

Theorem 6 (Typing Preservation - Constraints Proving). *If* $P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow d$, *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$, *and* $\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q$, *then* $\Sigma; \Gamma_C; \Gamma \vdash_d d : Q$.

Proof. By induction on the constraint resolution derivation tree. This theorem is mutually proven with Theorems 1 and 7 (Figure 14). Note that at the dependency from Theorem 7 to 1, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Consequently, the size of P is strictly decreasing in every possible cycle. The induction thus remains well-founded.

sEntail-local
$$\frac{(\delta : Q) \in \Gamma \qquad \vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma}{P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow \delta}\ \text{sEntail-local}$$

It is easy to verify that given

$$\vdash^{M}_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$$
$$(\delta : Q) \in \Gamma$$

we can derive

$$(\delta : Q') \in \Gamma$$
$$\Gamma_C; \Gamma_1 \vdash^{M}_{Q} Q \rightsquigarrow Q'$$
$$\Gamma = \Gamma_1, \delta : Q, \Gamma_2$$

By weakening lemma (Lemma 3)

$$\Gamma_C; \Gamma \vdash^{M}_{Q} Q \rightsquigarrow Q'$$

We already know from the hypothesis:

$$\Gamma_C; \Gamma \vdash^{M}_{Q} Q \rightsquigarrow Q$$

Since the elaboration of $\vdash_Q$ is deterministic (Lemma 8), we know that $Q = Q'$. The goal follows directly by D-VAR.

sENTAIL-INST

$$\frac{\begin{array}{c} P = P_1, (D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow Q').m \mapsto \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h : e, P_2 \\ Q = [\overline{\tau}_j / \overline{a}_j] Q' \quad \vdash^{M}_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \quad \overline{\Gamma_C; \Gamma \vdash^{M}_{ty} \tau_j \rightsquigarrow \sigma_j}^{j} \quad \overline{P; \Gamma_C; \Gamma \vDash^{M} [\overline{\tau}_j / \overline{a}_j] Q_i \rightsquigarrow d_i}^{i} \end{array}}{P; \Gamma_C; \Gamma \vDash^{M} Q \rightsquigarrow D\, \overline{\sigma}_j\, \overline{d}_i} \text{sENTAIL-INST}$$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma \vdash_d D\, \overline{\sigma}_j\, \overline{d}_i : Q \textit{ where } Q = TC\, \sigma_q$$

Using D-CON, proving this is equivalent to proving:

$$(D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow TC\, \sigma'_q).m \mapsto \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e \in \Sigma \tag{58}$$
$$\textit{where } \sigma_q = [\overline{\sigma}_j / \overline{a}_j] \sigma'_q$$
$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \tag{59}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i}^{i} \tag{60}$$
$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{j} \tag{61}$$
$$\Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e : [\sigma'_q / a] \sigma_m \tag{62}$$
$$\textit{where } \Sigma = \Sigma_1, (D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow Q).m \mapsto e, \Sigma_2$$
$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j / \overline{a}_j] Q_i}^{i} \tag{63}$$

The rule premise tells us, among other things, that:

$$P = P_1, (D : \forall \overline{a}_j . \overline{Q}_i \Rightarrow Q').m \mapsto \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h : e, P_2 \tag{64}$$
$$\vdash^{M}_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \tag{65}$$
$$\overline{\Gamma_C; \Gamma \vdash^{M}_{ty} \tau_j \rightsquigarrow \sigma_j}^{j} \tag{66}$$
$$\overline{P; \Gamma_C; \Gamma \vDash^{M} [\overline{\tau}_j / \overline{a}_j] Q_i \rightsquigarrow d_i}^{i} \tag{67}$$

Goal 58 follows directly from Equations 64 and 65. Goal 59 follows by applying Equation 65 to preservation Theorem 7. Consequently, from Equation 59, in combination with rule ICTX-MENV, we know that:

$$\Gamma_C; \bullet \vdash_C \forall \overline{a}_j . \overline{Q}_i \Rightarrow TC\, \sigma'_q \tag{68}$$

Goal 60 follows from rule IC-ABS, in combination with Equation 68.
By inversion on Equation 65, we know that:

$$\vdash^{M}_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma \tag{69}$$

Goal 61 follows by applying Equations 66 and 69 to Theorem 5.
From rule ICTX-MENV, in combination with Equations 58 and 59, we know that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e : \forall \overline{a}_j . \overline{Q}_i \Rightarrow [\sigma'_q / a] \sigma_m \tag{70}$$

Goal 62 follows from inversion of rules iTm-forallI and iTm-constrI, on Equation 70.
Finally, Goal 63 follows by applying the induction hypothesis on Equation 67, assuming we can show that:

$$\overline{\Gamma_C; \Gamma \vdash_Q^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{71}$$

From rule sCtx-pgmInst, in combination with Equation 64, we know that:

$$\Gamma_C; \bullet \vdash_C^M \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q' \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma'_q \tag{72}$$

From weakening Lemma 4, we know that:

$$\Gamma_C; \Gamma \vdash_C^M \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q' \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma'_q \tag{73}$$

From rule sQ-TC, in combination with Equation 73, we know that:

$$\overline{\Gamma_C; \Gamma, \overline{a}_j \vdash_Q^M Q_i \rightsquigarrow Q_i}^{\,i} \tag{74}$$

Goal 71 follows by applying type substitution Lemma 1 on Equations 74 and 66 (in combination with weakening Lemma 2).

□

Theorem 7 (Typing Preservation - Environment Well-Formedness). *If* $\vdash_{ctx}^M P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$, *then* $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$.

Proof. By induction on the well-formedness derivation. This theorem is mutually proven with Theorems 1 and 6 (Figure 14). Note that at the dependency from Theorem 7 to 1, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Consequently, the size of P is strictly decreasing in every possible cycle. The induction thus remains well-founded.

sCtx-empty

$$\frac{}{\vdash_{ctx}^M \bullet; \bullet; \bullet \rightsquigarrow \bullet; \bullet; \bullet}\;\text{sCtx-empty}$$

Follows directly by iCtx-empty.

sCtx-tyEnvTm

$$\frac{\Gamma_C; \Gamma \vdash_{ty}^M \sigma \rightsquigarrow \sigma \qquad x \notin \mathbf{dom}(\Gamma) \qquad \vdash_{ctx}^M \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma}{\vdash_{ctx}^M \bullet; \Gamma_C; \Gamma, x : \sigma \rightsquigarrow \bullet; \Gamma_C; \Gamma, x : \sigma}\;\text{sCtx-tyEnvTm}$$

By induction hypothesis, we know

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma$$

By Typing Preservation - Types and Class Constraints (Theorem 5), we know

$$\Gamma_C; \Gamma \vdash_{ty} \sigma$$

Since $x \notin \mathbf{dom}(\Gamma)$, it is easy to verify that $x \notin \mathbf{dom}(\Gamma)$. Therefore the goal follows directly by iCtx-tyEnvTm.

sCtx-tyEnvTy

$$\frac{a \notin \Gamma \qquad \vdash_{ctx}^M \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma}{\vdash_{ctx}^M \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \bullet; \Gamma_C; \Gamma, a}\;\text{sCtx-tyEnvTy}$$

By induction hypothesis, we know

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma$$

Since we know $a \notin \Gamma$, it is easy to verify that $a \notin \Gamma$. Therefore the goal follows directly by iCtx-tyEnvTy.

sCtx-tyEnvD

$$\frac{\Gamma_C; \Gamma \vdash_Q^M TC\,\tau \rightsquigarrow Q \qquad \delta \notin \mathbf{dom}(\Gamma) \qquad \vdash_{ctx}^M \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma}{\vdash_{ctx}^M \bullet; \Gamma_C; \Gamma, \delta : TC\,\tau \rightsquigarrow \bullet; \Gamma_C; \Gamma, \delta : Q}\;\text{sCtx-tyEnvD}$$

By induction hypothesis, we know

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma$$

By Typing Preservation - Types and Class Constraints (Theorem 5), we know

$$\Gamma_C; \Gamma \vdash_Q Q$$

Since $\delta \notin \mathbf{dom}(\Gamma)$, it is easy to verify that $\delta \notin \mathbf{dom}(\Gamma)$. Therefore the goal follows directly by iCtx-tyEnvD.

sCtx-clsEnv

$$\frac{\overline{a_j}, a = \mathbf{fv}(\tau) \quad \dfrac{\Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a_j}.\overline{TC_i\, a}^i \Rightarrow \tau \rightsquigarrow \sigma}{\overline{\Gamma_C; \bullet, a \vdash^M_Q TC_i\, a \rightsquigarrow Q_i}^i} \quad m \notin \mathbf{dom}(\Gamma_C) \quad TC\, b \notin \mathbf{dom}(\Gamma_C) \quad \vdash^M_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet}{\vdash^M_{ctx} \bullet; \Gamma_C, m : \overline{TC_i\, a} \Rightarrow TC\, a : \forall \overline{a_j}.\overline{TC_i\, a}^i \Rightarrow \tau; \bullet \rightsquigarrow \bullet; \Gamma_C, m : TC\, a : \sigma; \bullet} \text{sCtx-clsEnv}$$

By induction hypothesis, we know

$$\vdash_{ctx} \bullet; \Gamma_C; \bullet$$

By sCtx-tyEnvTy, we know that

$$\vdash^M_{ctx} \bullet; \Gamma_C; \bullet, a \rightsquigarrow \bullet; \Gamma_C; \bullet, a$$

Then by Typing Preservation - Types and Class Constraints (Theorem 5), we know

$$\Gamma_C; \bullet, a \vdash_{ty} \sigma$$

It is easy to verify that given $m \notin \mathbf{dom}(\Gamma_C)$, $TC\, b \notin \mathbf{dom}(\Gamma_C)$, we can derive $m \notin \mathbf{dom}(\Gamma_C)$, $TC\, b \notin \mathbf{dom}(\Gamma_C)$. The goal follows directly by iCtx-ClsEnv.

sCtx-pgmInst

$$\frac{\begin{array}{c}\mathbf{unambig}(\forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \tau) \\ \Gamma_C; \bullet \vdash^M_C \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \tau \rightsquigarrow \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \sigma \quad (m : \overline{Q'_m} \Rightarrow TC\, a : \forall \overline{a_k}.\overline{Q_h} \Rightarrow \tau') \in \Gamma_C \\ P; \Gamma_C; \bullet, \overline{b_j}, \overline{\delta_i : Q_i}, \overline{a_k}, \overline{\delta_h : [\tau/a] Q_h} \vdash^M_{tm} e \Leftarrow [\tau/a]\tau' \rightsquigarrow e \quad \Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a_k}.\overline{Q_h} \Rightarrow \tau' \rightsquigarrow \forall \overline{a_k}.\overline{Q_h} \Rightarrow \sigma' \\ D \notin \mathbf{dom}(P) \quad (D' : \forall \overline{b'_k}.\overline{Q''_h} \Rightarrow TC\, \tau'').m' \mapsto \Gamma' : e' \notin P \quad \mathbf{where}[\overline{\tau_j}/\overline{b_j}]\tau = [\overline{\tau'_k}/\overline{b'_k}]\tau'' \\ \vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \quad \Sigma' = \Sigma, (D : \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \sigma).m \mapsto \Lambda \overline{b_j}.\lambda \overline{\delta_i : Q_i}.\Lambda \overline{a_k}.\lambda \overline{\delta_h : [\sigma/a] Q_h}.e\end{array}}{\vdash^M_{ctx} P, (D : \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \tau).m \mapsto \bullet, \overline{b_j}, \overline{\delta_i : Q_i}, \overline{a_k}, \overline{\delta_h : [\tau/a] Q_h} : e; \Gamma_C; \Gamma \rightsquigarrow \Sigma'; \Gamma_C; \Gamma} \text{sCtx-pgmInst}$$

By induction hypothesis, we know that

$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \tag{75}$$

Also, since we know that

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$$

by applying inversion, we get:

$$\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma \tag{76}$$

From the premise, we already know

$$\Gamma_C; \bullet \vdash^M_C \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \tau \rightsquigarrow \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \sigma \tag{77}$$

Then by Typing Preservation - Types and Class Constraints (Theorem 5) on Equations 76 and 77, we know

$$\Gamma_C; \bullet \vdash_C \forall \overline{b_j}.\overline{Q_i} \Rightarrow TC\, \sigma \tag{78}$$

From the 3rd rule premise, we know that

$$(m : \overline{Q'_m} \Rightarrow TC\, a : \forall \overline{a_k}.\overline{Q_h} \Rightarrow \tau') \in \Gamma_C \tag{79}$$

By inversion on Equation 76 (sCtx-clsEnv), together with Equation 79, we get

$$(m : TC\, a : \sigma_1) \in \Gamma_C \tag{80}$$

$$\Gamma_{C1}; \bullet, a \vdash^M_{ty} \forall \overline{a_k}.\overline{Q_h} \Rightarrow \tau' \rightsquigarrow \sigma_1 \tag{81}$$

$$\Gamma_C = \Gamma_{C1}, \Gamma_{C2} \tag{82}$$

By applying weakening (Lemma 2) on Equation 81, we have

$$\Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a_k}.\overline{Q_h} \Rightarrow \tau' \rightsquigarrow \sigma_1 \tag{83}$$

From the 5th rule premise, we know that

$$\Gamma_C; \bullet, a \vdash^M_{ty} \forall \overline{a_k}.\overline{Q_h} \Rightarrow \tau' \rightsquigarrow \forall \overline{a_k}.\overline{Q_h} \Rightarrow \sigma' \tag{84}$$

Because the elaboration of types is deterministic (Lemma 9), combining Equations 83 and 84, we know that $\sigma_1 = \forall \overline{a_k}.\overline{Q_h} \Rightarrow \sigma'$. By rewriting Equation 80, we get

$$(m : TC\, a : \forall \overline{a_k}.\overline{Q_h} \Rightarrow \sigma') \in \Gamma_C \tag{85}$$

By applying Theorem 1 to the 4th rule premise, we get:

$$\Sigma; \Gamma_C; \bullet, \overline{b_j}, \overline{\delta_i : Q_i}, \overline{a_k}, \overline{\delta_h : [\sigma/a] Q_h} \vdash_{tm} e : [\sigma/a]\sigma' \tag{86}$$

Lemma 44, applied to this result, gives us:

$$\vdash_{ctx} \Sigma; \Gamma_C; \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\sigma/a]\overline{Q}_h \tag{87}$$

Furthermore, by applying iTm-constrI and iTm-forallI to Equation 86, in combination with inversion on Equation 87, we get

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} \Lambda\overline{b}_j.\lambda\overline{\delta}_i : \overline{Q}_i.\Lambda\overline{a}_k.\lambda\overline{\delta}_h : [\sigma/a]\overline{Q}_h.e : \forall\overline{b}_j.\overline{Q}_i \Rightarrow \forall\overline{a}_k.[\sigma/a]\overline{Q}_h \Rightarrow [\sigma/a]\sigma'$$

which is equivalent to

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} \Lambda\overline{b}_j.\lambda\overline{\delta}_i : \overline{Q}_i.\Lambda\overline{a}_k.\lambda\overline{\delta}_h : [\sigma/a]\overline{Q}_h.e : \forall\overline{b}_j.\overline{Q}_i \Rightarrow [\sigma/a](\forall\overline{a}_k.\overline{Q}_h \Rightarrow \sigma') \tag{88}$$

Given

$$\mathbf{unambig}(\forall\overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau)$$
$$D \notin \mathbf{dom}(P)$$
$$(D' : \forall\overline{b}'_k.\overline{Q}''_h \Rightarrow TC\,\tau'').m' \mapsto \Gamma' : e' \notin P$$
$$\mathbf{where}[\overline{\tau}_j/\overline{b}_j]\tau = [\overline{\tau}'_k/\overline{b}'_k]\tau''$$

It is easy to verify that

$$\mathbf{unambig}(\forall\overline{b}_j.\overline{Q}_i \Rightarrow TC\,\sigma) \tag{89}$$
$$D \notin \mathbf{dom}(\Sigma) \tag{90}$$
$$(D' : \forall\overline{a}'_m.\overline{Q}''_n \Rightarrow TC\,\sigma'').m' \mapsto e' \notin \Sigma \tag{91}$$
$$\mathbf{where}[\overline{\sigma}_j/\overline{a}_j]\sigma = [\overline{\sigma}'_m/\overline{a}'_m]\sigma'' \tag{92}$$

The goal follows by combining Equations 75, 78, 85, 88, 89, 90, 91, 92, and the rule iCtx-MEnv.

□

K F_D THEOREMS

K.1 Conjectures

We are confident that the following lemmas can be proven using well-known proof techniques.

Lemma 20 (Type Variable Substitution in Types).
If $\Gamma_C; \Gamma_1, a, \Gamma_2 \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1$ *and* $\Gamma_C; \Gamma_1 \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2$ *then* $\Gamma_C; \Gamma_1, [\sigma_2/a]\Gamma_2 \vdash_{ty} [\sigma_2/a]\sigma_1 \rightsquigarrow [\sigma_2/a]\sigma_1$.

Lemma 21 (Type Variable Substitution in Dictionaries).
If $\Gamma_C; \Gamma_1, a, \Gamma_2 \vdash_Q Q \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma_1 \vdash_{ty} \sigma' \rightsquigarrow \sigma'$ *then* $\Gamma_C; \Gamma_1, [\sigma'/a]\Gamma_2 \vdash_Q [\sigma'/a]Q \rightsquigarrow [\sigma'/a]\sigma$.

Lemma 22 (Variable Substitution).
If $\Sigma; \Gamma_C; \Gamma_1, x : \sigma_2, \Gamma_2 \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1$ *and* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2$ *then* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_{tm} [e_2/x]e_1 : \sigma_1 \rightsquigarrow [e_2/x]e_1$.

Lemma 23 (Reverse Variable Substitution).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} [e_2/x]e_1 : \sigma_1$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_2$ *then* $\Sigma; \Gamma_C; \Gamma, x : \sigma_2 \vdash_{tm} e_1 : \sigma_1$.

Lemma 24 (Dictionary Variable Substitution).
If $\Sigma; \Gamma_C; \Gamma_1, \delta : Q, \Gamma_2 \vdash_{tm} e : \sigma \rightsquigarrow e$ *and* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_d d : Q \rightsquigarrow e'$ *then* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_{tm} [d/\delta]e : \sigma \rightsquigarrow [e'/\delta]e$.

Lemma 25 (Reverse Dictionary Variable Substitution).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} [d/\delta]e : \sigma$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_d d : Q$ *then* $\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma$.

Lemma 26 (Type Variable Substitution).
If $\Sigma; \Gamma_C; \Gamma_1, a, \Gamma_2 \vdash_{tm} e : \sigma \rightsquigarrow e$ *and* $\Gamma_C; \Gamma_1 \vdash_{ty} \sigma' \rightsquigarrow \sigma'$ *then* $\Sigma; \Gamma_C; \Gamma_1, [\sigma'/a]\Gamma_2 \vdash_{tm} [\sigma'/a]e : [\sigma'/a]\sigma \rightsquigarrow [\sigma'/a]e$.

Lemma 27 (Reverse Type Variable Substitution).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} [\sigma'/a]e : [\sigma'/a]\sigma$ *then* $\Sigma; \Gamma_C; \Gamma, a \vdash_{tm} e : \sigma$.

Lemma 28 (Dictionary Variable Substitution in Dictionaries).
If $\Sigma; \Gamma_C; \Gamma_1, \delta : Q', \Gamma_2 \vdash_d D\, \overline{\sigma}_j\, \overline{d}_i : Q$ *and* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_d d : Q'$ *then* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_d D\, \overline{\sigma}_j\, [d/\delta]\overline{d}_i : Q$.

Lemma 29 (Type Variable Substitution in Dictionaries).
If $\Sigma; \Gamma_C; \Gamma_1, a, \Gamma_2 \vdash_d D\, \overline{\sigma}_j\, \overline{d}_i : Q$ *and* $\Gamma_C; \Gamma_1 \vdash_{ty} \sigma$ *then* $\Sigma; \Gamma_C; \Gamma_1, [\sigma/a]\Gamma_2 \vdash_d D\, [\sigma/a]\overline{\sigma}_j\, [\sigma/a]\overline{d}_i : [\sigma/a]Q$.

Lemma 30 (Expression Well-Typed Method Environment Weakening).
If $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$ *and* $\vdash_{ctx} \Sigma_1, \Sigma_2; \Gamma_C; \Gamma$ *then* $\Sigma_1, \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$.

Lemma 31 (Type Well-Formedness Dictionary Environment Weakening).
If $\vdash_{ctx} \Sigma; \Gamma_{C1}; \Gamma_1$ *and* $\Gamma_{C1}; \Gamma_1 \vdash_{ty} \sigma \rightsquigarrow \sigma$ *and* $\vdash_{ctx} \Sigma; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2$, *then* $\Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \vdash_{ty} \sigma \rightsquigarrow \sigma$.

Lemma 32 (Class Constraint Well-Formedness Environment Weakening).
If $\Gamma_{C1}; \Gamma_1 \vdash_Q Q \rightsquigarrow \sigma$ *and* $\vdash_{ctx} \Sigma; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2$ *then* $\Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \vdash_Q Q \rightsquigarrow \sigma$.

Lemma 33 (Logical Equivalence Environment Weakening).
If $\Gamma_{C1}; \Gamma_1 \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$ *and* $\vdash_{ctx} \Sigma_1, \Sigma_1'; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2$ *and* $\vdash_{ctx} \Sigma_2, \Sigma_2'; \Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2$,
then $\Gamma_{C1}, \Gamma_{C2}; \Gamma_1, \Gamma_2 \vdash \Sigma_1, \Sigma_1' : e_1 \simeq_{log} \Sigma_2, \Sigma_2' : e_2 : \sigma$.

Lemma 34 (Strong Normalization Relation Method Environment Weakening).
If $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma_1,\Gamma_C}_{RSN}$ *and* $\vdash_{ctx} \Sigma_1, \Sigma_2; \Gamma_C; \Gamma$ *then* $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{RSN}$.

Lemma 35 (Dictionary Value Relation Preserved under Substitution).
$(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_{R_1,R_2}$ *if and only if* $(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![R_2(Q)]\!]^{\Gamma_C}_{R_1}$.

K.2 Lemmas

Lemma 36 (Environment Well-Formedness Strengthening).
If $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$ *then* $\vdash_{ctx} \Sigma; \Gamma_C; \bullet$.

Proof. By case analysis on the hypothesis, the last rules used to construct it must be (possibly zero) consecutive applications of iCtx-MEnv. Revert those rules, to obtain $\vdash_{ctx} \bullet; \Gamma_C; \Gamma$. By further case analysis (with iCtx-tyEnvTm, iCtx-tyEnvTy and iCtx-tyEnvD), we get $\vdash_{ctx} \bullet; \Gamma_C; \bullet$. The goal follows by consecutively re-applying rule iCtx-MEnv with the appropriate premises. □

Lemma 37 (Variable Strengthening in Dictionaries).
If $\Sigma; \Gamma_C; \Gamma_1, x : \sigma, \Gamma_2 \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : Q$ *then* $\Sigma; \Gamma_C; \Gamma_1, \Gamma_2 \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : Q$.

Proof. By straightforward induction on the well-formedness derivation. □

Lemma 38 (Variable Strengthening in Types).
If $\Gamma_C; \Gamma_1, x : \sigma, \Gamma_2 \vdash_{ty} \sigma'$ *then* $\Gamma_C; \Gamma_1, \Gamma_2 \vdash_{ty} \sigma'$.

Proof. By straightforward induction on the well-formedness derivation. □

Lemma 39 (Method Type Well-Formedness).
If $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$ *and* $\Gamma_C = \Gamma_{C1}, m : TC\,a : \sigma, \Gamma_{C2}$ *then there is a* σ *such, that* $\Gamma_{C1}; \bullet, a \vdash_{ty} \sigma \rightsquigarrow \sigma$.

Proof. By straightforward induction on the environment well-formedness derivation. □

Lemma 40 (Method Environment Well-Formedness).
If $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$ *and* $(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma).m \mapsto e \in \Sigma$, *where* $i = 1 \ldots n$, *then there are unique* a, σ_m, σ_m *and* $\overline{\sigma}_i$, *such that* $(m : TC\,a : \sigma_m) \in \Gamma_C$ *and* $\Gamma_C; \bullet, a \vdash_{ty} \sigma_m \rightsquigarrow \sigma_m$ *and* $\Sigma; \Gamma_C; \bullet \vdash_{tm} e : \forall \overline{a}_j.\overline{Q}_i \Rightarrow [\sigma/a]\sigma_m \rightsquigarrow e$ *and* $\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma_i}^{\,i}$. *and* $\Gamma_C; \bullet, \overline{a}_j \vdash_Q TC\,\sigma \rightsquigarrow [\sigma/a]\{m : \sigma_m\}$ *and* $\Gamma_C; \bullet, \overline{a}_j \vdash_{ty} \sigma \rightsquigarrow \sigma$.

Proof. By straightforward induction on the environment well-formedness derivation. □

Lemma 41 (Determinism of Evaluation).
If $\Sigma \vdash e \longrightarrow e_1$ *and* $\Sigma \vdash e \longrightarrow e_2$ *then* $e_1 = e_2$.

Proof. By straightforward induction on the evaluation derivation. □

Lemma 42 (Preservation of Environment Type Variables from F_D to $F_{\{\}}$).

- *If* $a \in \Gamma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $a \in \Gamma$.
- *If* $a \notin \Gamma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $a \notin \Gamma$.

Proof. By straightforward induction on the environment elaboration derivation. □

Lemma 43 (Well-Formedness of F_D Typing Result).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$ *then* $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$.

Proof. By straightforward induction on the typing derivation. □

Lemma 44 (Context Well-Formedness of F_D Typing).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$ *then* $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$.

Proof. By straightforward induction on the typing derivation. □

Lemma 45 (Context Well-Formedness of Dictionary Typing).
If $\Sigma; \Gamma_C; \Gamma \vdash_{d} d : Q$ *then* $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$.

Proof. By straightforward induction on the dictionary typing derivation. □

K.3 Type Safety

Theorem 8 (Preservation).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$, *and* $\Sigma \vdash e \longrightarrow e'$, *then* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e' : \sigma$.

Proof. By induction on the typing derivation.

iTm-true

$$\frac{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} True : Bool}\ \text{iTm-true}$$

True is already a value, so impossible case.

iTm-false

$$\frac{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} False : Bool}\ \text{iTm-false}$$

False is already a value, so impossible case.

iTm-var

$$\frac{(x : \sigma) \in \Gamma \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} x : \sigma}\ \text{iTm-var}$$

x cannot be reduced, so impossible case.

iTm-let

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \qquad \Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2 : \sigma_2}\ \text{iTm-let}$$

By inversion on the evaluation (iEval-let), we get that

$$e' = [e_1/x]e_2$$

Given

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1$$
$$\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2$$

The goal follows directly from Lemma 22.

iTm-method $$\frac{\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \qquad (m : TC\,a : \sigma') \in \Gamma_C}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma'}\ \text{iTm-method}$$

By inversion on the evaluation (iEval-method), we get that

$$e' = e\,\overline{\sigma}_j\,\overline{d}_i \tag{93}$$
$$d = D\,\overline{\sigma}_j\,\overline{d}_i \tag{94}$$

The goal to be proven thus becomes:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{d}_i : [\sigma/a]\sigma' \tag{95}$$

By substituting Equation 94 in the 1st rule premise, we get that

$$\Sigma; \Gamma_C; \Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : TC\,\sigma \tag{96}$$

By inversion on Equation 96 (D-con), we know that

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_1).m \mapsto \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.e \in \Sigma \tag{97}$$
$$\sigma = [\overline{\sigma}_j/\overline{a}_j]\sigma_1 \tag{98}$$
$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{99}$$
$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{\,j} \tag{100}$$
$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \tag{101}$$

By combining this result with Equation 97, through inversion (iCtx-MEnv), we know that

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.e : \forall \overline{a}_j.\overline{Q}_i \Rightarrow [\sigma_1/a]\sigma' \tag{102}$$

where $\Sigma = \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_1).m \mapsto \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.e, \Sigma_2$. By iTm-forallE and Equations 102 and 100, we have

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j : [\overline{\sigma}_j/\overline{a}_j]\overline{Q}_i \Rightarrow [\overline{\sigma}_j/\overline{a}_j][\sigma_1/a]\sigma' \tag{103}$$

By iTm-constrE and Equations 103 and 99, we have

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{d}_i : [\overline{\sigma}_j/\overline{a}_j][\sigma_1/a]\sigma' \tag{104}$$

From the 2nd rule premise we know that

$$(m : TC\,a : \sigma') \in \Gamma_C \tag{105}$$

Combining this result with Equation 101, by inversion (iCtx-clsEnv), we know

$$\Gamma_{C1}; \bullet, a \vdash_{ty} \sigma' \tag{106}$$
$$\Gamma_C = \Gamma_{C1}, \Gamma_{C2} \tag{107}$$

Therefore, $\overline{\sigma}_j$ are not free variables in σ'. Equation 103 thus simplifies to

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{d}_i : [[\overline{\sigma}_j/\overline{a}_j]\sigma_1/a]\sigma' \tag{108}$$

By applying Equation 108 to Lemma 30, in combination with Equation 101, we get

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{d}_i : [[\overline{\sigma}_j/\overline{a}_j]\sigma_1/a]\sigma'$$

Goal 95 follows by combining this result with Equation 98.

iTm-arrI $$\frac{\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \to \sigma_2}\ \text{iTm-arrI}$$

$\lambda x : \sigma_1.e$ is already a value, so impossible case.

iTm-arrE $$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \qquad \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1\,e_2 : \sigma_2}\ \text{iTm-arrE}$$

By inversion on the evaluation, we have two possible cases:

- Case iEval-app. $\dfrac{\Sigma \vdash e_1 \longrightarrow e_1'}{\Sigma \vdash e_1\, e_2 \longrightarrow e_1'\, e_2}$ iEval-app
  By induction hypothesis, we get
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1' : \sigma_1 \rightarrow \sigma_2$$
  By iTm-arrE we get
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1'\, e_2 : \sigma_2$$
- Case iEval-appAbs. $\dfrac{}{\Sigma \vdash (\lambda x : \sigma.e_1)\, e_2 \longrightarrow [e_2/x]e_1}$ iEval-appAbs
  From premise, we know
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma.e_1 : \sigma \rightarrow \sigma_2$$
  By inversion (iTm-arrI)
$$\Sigma; \Gamma_C; \Gamma, x : \sigma \vdash_{tm} e_1 : \sigma_2$$
  By substitution (Lemma 22) we get
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} [e_2/x]e_1 : \sigma_2$$

iTm-constrI $\dfrac{\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \qquad \Gamma_C; \Gamma \vdash_Q Q}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma}$ iTm-constrI

$\lambda\delta : Q.e$ is already a value, so impossible case.

iTm-constrE $\dfrac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \qquad \Sigma; \Gamma_C; \Gamma \vdash_d d : Q}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, d : \sigma}$ iTm-constrE

Similar to case iTm-arrE. The only difference lies in applying Lemma 24.

iTm-forallI $\dfrac{\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \qquad \Gamma_C; \Gamma \vdash_Q Q}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma}$ iTm-constrI

$\Lambda a.e$ is already a value, so impossible case.

iTm-forallE $\dfrac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, \sigma : [\sigma/a]\sigma'}$ iTm-forallE

Similar to case iTm-arrE. The only difference lies in applying Lemma 26.

□

Theorem 9 (Progress).
If $\Sigma; \bullet; \bullet \vdash_{tm} e : \sigma$, then either e is a value, or there exists e' such that $\Sigma \vdash e \longrightarrow e'$.

Proof. By structural induction on the typing derivation.

iTm-true $\dfrac{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} True : Bool}$ iTm-true

with $\Gamma_C = \bullet, \Gamma = \bullet$.
$True$ is a value.

iTm-false $\dfrac{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} False : Bool}$ iTm-false

with $\Gamma_C = \bullet, \Gamma = \bullet$.
$False$ is a value.

iTm-var
$$\frac{(x : \sigma) \in \Gamma \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} x : \sigma} \text{ iTm-var}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
x cannot be in an empty context. Impossible case.

iTm-let
$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \qquad \Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \textbf{let } x : \sigma_1 = e_1 \textbf{ in } e_2 : \sigma_2} \text{ iTm-let}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
By iEval-let:
$$e' = [e_1/x]e_2$$

iTm-method
$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \qquad (m : TC\,a : \sigma') \in \Gamma_C}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma'} \text{ iTm-method}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
m cannot be in an empty context. Impossible case.

iTm-arrI
$$\frac{\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \to \sigma_2} \text{ iTm-arrI}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
$\lambda x : \sigma_1.e$ is a value.

iTm-arrE
$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \qquad \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2} \text{ iTm-arrE}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
From the 1st rule premise:
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \tag{109}$$
By applying the induction hypothesis on Equation 109, we know that either:

- e_1 is a value. Because it has an arrow type, we know:
$$e_1 = (\lambda x : \sigma.e_1)\, e_2$$
By iEval-appAbs:
$$e' = [e_2/x]e_1$$
- There exists an e_1' where $\Sigma \vdash e_1 \longrightarrow e_1'$. By iEval-app:
$$e' = e_1'\, e_2$$

iTm-constrI
$$\frac{\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \qquad \Gamma_C; \Gamma \vdash_Q Q}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma} \text{ iTm-constrI}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
$\lambda\delta : Q.e$ is a value.

iTm-constrE
$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \qquad \Sigma; \Gamma_C; \Gamma \vdash_d d : Q}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, d : \sigma} \text{ iTm-constrE}$$

with $\Gamma_C = \bullet, \Gamma = \bullet$.
From the 1st rule premise:
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \tag{110}$$
By applying the induction hypothesis on Equation 110, we know that either:

- e_1 is a value. By case analysis, we know:

$$e_1 = (\lambda\delta : Q.e)$$

By iEval-DAppAbs:

$$e' = [d/\delta]e$$

- There exists an e' where $\Sigma \vdash e \longrightarrow e'$ By iEval-DApp:

$$\Sigma \vdash e\, d \longrightarrow e'\, d$$

iTm-forallI
$$\frac{\Sigma; \Gamma_C; \Gamma, a \vdash_{tm} e : \sigma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma} \text{ iTm-forallI}$$

with $\Gamma_C = \bullet$, $\Gamma = \bullet$.
$\Lambda a.e$ is a value.

iTm-forallE
$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, \sigma : [\sigma/a]\sigma'} \text{ iTm-forallE}$$

with $\Gamma_C = \bullet$, $\Gamma = \bullet$.
From the 1st rule premise:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \tag{111}$$

By applying the induction hypothesis on Equation 111, we know that either:
- e_1 is a value. By case analysis, we know:

$$e_1 = (\Lambda a.e)$$

By iEval-tyAppAbs:

$$e' = [\sigma/a]e$$

- There exists an e'_1 where $\Sigma \vdash e \longrightarrow e'$. By iEval-tyApp:

$$e' = e'_1\, \sigma$$

□

K.4 Strong Normalization

Theorem 10 (Strong Normalization).
If $\Sigma; \Gamma_C; \bullet \vdash_{tm} e : \sigma$ then all possible evaluation derivations for e terminate : $\exists v : \Sigma \vdash e \longrightarrow^* v$.

Proof. By Theorem 11 and 12, with $R^{SN} = \bullet$, $\phi^{SN} = \bullet$, $\gamma^{SN} = \bullet$, since $\Gamma = \bullet$, it follows that:

$$\exists v : \Sigma \vdash e \longrightarrow^* v$$

Furthermore, since evaluation in F_D is deterministic (Lemma 41), there is exactly 1 possible evaluation derivation. Consequently, all derivations terminate.

□

Lemma 46 (Well Typedness from Strong Normalization).
$e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, then $\Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}(\sigma)$

Proof. The goal is baked into the relation. It follows by simple induction on σ.

□

Lemma 47 (Strong Normalization preserved by forward/backward reduction).
Suppose $\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1 : R^{SN}(\sigma)$, and $\Sigma \vdash e_1 \longrightarrow e_2$, then
- *If $e_1 \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, then $e_2 \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$.*
- *If $e_2 \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, then $e_1 \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$.*

Proof. **Part 1** By induction on σ.

Bool $e_1 \in \mathcal{SN}[\![Bool]\!]^{\Sigma,\Gamma_C}_{RSN} \triangleq \Sigma;\Gamma_C;\bullet \vdash_{tm} e_1 : Bool$
$\wedge\, \exists v : \Sigma \vdash e_1 \longrightarrow^* v$

By Preservation (Theorem 8), we know that $\Sigma;\Gamma_C;\bullet \vdash_{tm} e_2 : Bool$. Because the evaluation in $F_{\mathbf{D}}$ is deterministic (Lemma 41), given $\Sigma \vdash e_1 \longrightarrow^* v$, we have $\Sigma \vdash e_2 \longrightarrow^* v$.

Type variable
$$e_1 \in \mathcal{SN}[\![a]\!]^{\Sigma,\Gamma_C}_{RSN} \triangleq \Sigma;\Gamma_C;\bullet \vdash_{tm} e_1 : {R^{SN}}_1(a) \wedge \exists v : \Sigma \vdash e_1 \longrightarrow^* v \wedge v \in {R^{SN}}_2(a)$$

Similar to Bool case.

Function
$$e_1 \in \mathcal{SN}[\![\sigma_1 \to \sigma_2]\!]^{\Sigma,\Gamma_C}_{RSN} \triangleq \Sigma;\Gamma_C;\bullet \vdash_{tm} e_1 : {R^{SN}}_1(\sigma_1 \to \sigma_2) \wedge \exists v : \Sigma \vdash e_1 \longrightarrow^* v \wedge \forall e' : e' \in \mathcal{SN}[\![\sigma_1]\!]^{\Sigma,\Gamma_C}_{RSN} \Rightarrow e_1\, e' \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{RSN}$$

By Preservation (Theorem 8), we know that $\Sigma;\Gamma_C;\bullet \vdash_{tm} e_2 : {R^{SN}}_1(\sigma_1 \to \sigma_2)$. Because the evaluation in $F_{\mathbf{D}}$ is deterministic (Lemma 41), given $\Sigma \vdash e_1 \longrightarrow^* v$, we have $\Sigma \vdash e_2 \longrightarrow^* v$. Given any $e' : e' \in \mathcal{SN}[\![\sigma_1]\!]^{\Sigma,\Gamma_C}_{RSN}$, we know that $\Sigma \vdash e_1 \longrightarrow e_2$, so $\Sigma \vdash e_1\, e' \longrightarrow e_2\, e'$. By induction hypothesis, we get $e_2\, e' \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{RSN}$.

Function over constraint $e \in \mathcal{SN}[\![Q \Rightarrow \sigma]\!]^{\Sigma,\Gamma_C}_{RSN}$

Similar to Function case.

Polymorphic type $e \in \mathcal{SN}[\![\forall a.\sigma]\!]^{\Sigma,\Gamma_C}_{RSN}$

Similar to Function case.

Part 2 Similar to Part 1.

□

Lemma 48 (Substitution for Context Interpretation).

- *If* $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma$ *then* $\forall R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, $\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$ *and* $\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$, *we have* $\Sigma;\Gamma_C;\bullet \vdash_{tm} \gamma^{SN}(\phi^{SN}(R^{SN}(e))) : R^{SN}(\sigma)$.
- *If* $\Sigma;\Gamma_C;\Gamma \vdash_d d : Q$ *then* $\forall R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, $\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$ *and* $\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$, *we have* $\Sigma;\Gamma_C;\bullet \vdash_d \gamma^{SN}(\phi^{SN}(R^{SN}(d))) : R^{SN}(Q)$.
- *If* $\Gamma_C;\Gamma \vdash_{ty} \sigma$ *then* $\forall R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, $\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$ *and* $\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$, *we have* $\Gamma_C;\bullet \vdash_{ty} R^{SN}(\sigma)$.
- *If* $\Gamma_C;\Gamma \vdash_Q Q$ *then* $\forall R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, $\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$ *and* $\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{RSN}$, *we have* $\Gamma_C;\bullet \vdash_Q R^{SN}(Q)$.

Proof. By induction on e, d σ, and Q respectively. The goal follows from Definitions 1, 2 and 3.

□

Lemma 49 (Compositionality for Strong Normalization).
Let $r = \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{RSN}$, *then* $e \in \mathcal{SN}[\![\sigma_1]\!]^{\Sigma,\Gamma_C}_{RSN,\, a\mapsto(R^{SN}(\sigma_2),r)}$ *if and only if* $e \in \mathcal{SN}[\![[\sigma_2/a]\sigma_1]\!]^{\Sigma,\Gamma_C}_{RSN}$.

Proof. By induction on σ_1.

Bool $\sigma_1 = Bool$

Since $[\sigma_2/a]Bool = Bool$, the goal follows directly.

Type variable $\sigma_1 = b$

Depending on whether b is the same variable as a, we have two cases:

- If $b = a$, then $[\sigma_2/a]a = \sigma_2$.

  Part 1: From left to right. If $e \in \mathcal{SN}[\![a]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)}$, it means that:

$$\exists v : \Sigma \vdash e \longrightarrow^* v \quad (112)$$

$$v \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \quad (113)$$

  Combining with Strong Normalization preserved by forward/backward reduction (Lemma 47), the goal is proven by Equations 112 and 113:

$$e \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \quad (114)$$

  Part 2: From right to left. From the hypothesis, we know that:

$$e \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \quad (115)$$

  We want to prove that $e \in \mathcal{SN}[\![a]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)}$. By the definition, this goal is equivalent to:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : (R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))(a) \quad (116)$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \quad (117)$$

$$v \in (R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))_2\ (a) \quad (118)$$

  Equation 116 simplifies to:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}(\sigma_2) \quad (119)$$

  By Well-Typedness from Strong Normalization (Lemma 46), Equation 115 proves 119. By Strong Normalization - Part B (Theorem 12), Equation 115 proves 117.
  We already know that $(R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))_2\ (a) = r = \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, so we simplify Equation 118 to get

$$v \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \quad (120)$$

  From Equation 115 and Strong Normalization preserved by forward/backward reduction (Lemma 47), we can prove Goal 120.
- If $b \neq a$, since $[\sigma_2/a]b = b$, the goal follows directly.

Function $\sigma_1 = \sigma_{11} \to \sigma_{12}$

Part 1: From left to right. From the hypothesis, we get:

$$e \in \mathcal{SN}[\![\sigma_{11} \to \sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)}$$

We thus know that:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : (R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))(\sigma_{11} \to \sigma_{12}) \quad (121)$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \quad (122)$$

$$\forall e' : e' \in \mathcal{SN}[\![\sigma_{11}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)} \Rightarrow e\ e' \in \mathcal{SN}[\![\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)} \quad (123)$$

Our goal is to prove that $e \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{11} \to [\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. This is equivalent to proving:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}([\sigma_2/a]\sigma_{11} \to [\sigma_2/a]\sigma_{12}) \quad (124)$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \quad (125)$$

$$\forall e' : e' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{11}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \Rightarrow e\ e' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \quad (126)$$

Goal 124 and 125 are proven directly by Equations 121 and 122. Only Goal 126 remains to be proven.
Given $e' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{11}]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, the induction hypothesis tells us that $e' \in \mathcal{SN}[\![\sigma_{11}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)}$. In combination with equation Equation 123, we get

$$e\ e' \in \mathcal{SN}[\![\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a\mapsto(R^{SN}(\sigma_2),r)} \quad (127)$$

By induction hypothesis, we get:

$$e\ e' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \quad (128)$$

The goal has been proven.

Part 2: From right to left. Similar to Part 1.

Function over constraints $\sigma_1 = Q \Rightarrow \sigma_{12}$

Part 1: From left to right. We know from the hypothesis that:

$$e \in \mathcal{SN}[\![Q \Rightarrow \sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (R^{SN}(\sigma_2), r)}$$

It follows that:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : (R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))(Q \Rightarrow \sigma_{12}) \tag{129}$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \tag{130}$$

$$\forall d : \Sigma; \Gamma_C; \bullet \vdash_d d : (R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))(Q) \Rightarrow e\, d \in \mathcal{SN}[\![\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (R^{SN}(\sigma_2), r)} \tag{131}$$

Our goal is to prove that $e \in \mathcal{SN}[\![[\sigma_2/a]Q \Rightarrow [\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. This is equivalent to proving:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}([\sigma_2/a]Q \Rightarrow [\sigma_2/a]\sigma_{12}) \tag{132}$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \tag{133}$$

$$\forall d' : \Sigma; \Gamma_C; \bullet \vdash_d d' : {R^{SN}}_1([\sigma_2/a]Q) \Rightarrow e\, d' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{134}$$

Goals 132 and 133 are proven directly by Equations 129 and 130. Only Goal 134 remains to be proven. Given $\Sigma; \Gamma_C; \bullet \vdash_d d' : {R^{SN}}_1([\sigma_2/a]Q)$, in combination with Equation 131, we get that:

$$e\, d' \in \mathcal{SN}[\![\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (R^{SN}(\sigma_2), r)} \tag{135}$$

By induction hypothesis, we get:

$$e\, d' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{136}$$

The goal has been proven.

Part 2: From right to left. Similar to Part 1.

Polymorphic types $\sigma_1 = \forall b.\sigma_{12}$

Part 1: From left to right. We know from the hypothesis that:

$$e \in \mathcal{SN}[\![\forall b.\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (R^{SN}(\sigma_2), r)}$$

This implies that:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : (R^{SN}, a \mapsto (R^{SN}(\sigma_2), r))(\forall b.\sigma_{12}) \tag{137}$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \tag{138}$$

$$\forall \sigma', r = \mathcal{SN}[\![\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}} : e\, \sigma' \in \mathcal{SN}[\![\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (R^{SN}(\sigma_2), r)} \tag{139}$$

Our goal is to prove that $e \in \mathcal{SN}[\![\forall b.[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. This is equivalent to proving:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e : R^{SN}(\forall b.[\sigma_2/a]\sigma_{12}) \tag{140}$$

$$\exists v : \Sigma \vdash e \longrightarrow^* v \tag{141}$$

$$\forall \sigma', r = \mathcal{SN}[\![\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}} \Rightarrow e\, \sigma' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{142}$$

Goals 140 and 141 are directly proven by Equations 137 and 138. Only Goal 142 remains to be proven. Given σ' and $r = \mathcal{SN}[\![\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, by feeding it to Equation 139, we get that:

$$e\, \sigma' \in \mathcal{SN}[\![\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}, a \mapsto (R^{SN}(\sigma_2), r)} \tag{143}$$

By induction hypothesis, we get:

$$e\, \sigma' \in \mathcal{SN}[\![[\sigma_2/a]\sigma_{12}]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{144}$$

The goal has been proven.

Part 2: From right to left. Similar to Part 1.

□

Corollary 1 (Compositionality for Strong Normalization (Context Interpretation)). *Suppose* $R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, *then* $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$ *if and only if* $e \in \mathcal{SN}[\![R^{SN}(\sigma)]\!]^{\Sigma,\Gamma_C}_{\bullet}$.

Proof. The choices from $R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$ always satisfy the precondition of Compositionality for Strong Normalization (Lemma 49). Therefore, we can do induction on Γ and apply Compositionality for Strong Normalization (Lemma 49), in combination with the induction hypothesis, to prove the goal. □

Theorem 11 (Strong Normalization - Part A).
If $\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma$ *then* $\forall R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, $\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$ *and* $\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, *it holds that* $\gamma^{SN}(\phi^{SN}(R^{SN}(e))) \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$.

Proof. By induction on the first hypothesis of the theorem.

iTm-true

$$\frac{\vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} True : Bool}\ \text{iTm-true}$$

We know $\gamma^{SN}(\phi^{SN}(R^{SN}(True))) = True$. So the goal is $True \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. The goal follows directly since $\Sigma;\Gamma_C;\bullet \vdash_{tm} True : Bool$ and $\Sigma \vdash True \longrightarrow^* True$.

iTm-false

$$\frac{\vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} False : Bool}\ \text{iTm-false}$$

Similar to the iTm-true case.

iTm-var

$$\frac{(x : \sigma) \in \Gamma \qquad \vdash_{ctx} \Sigma;\Gamma_C;\Gamma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} x : \sigma}\ \text{iTm-var}$$

We know that $\phi^{SN} \in \mathcal{G}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$ and $(x : \sigma) \in \Gamma$, so we know that $x \mapsto e \in \phi^{SN}$ for some e with $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. Since $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, we know from the definition of the relation that e does not contain any free variables in Γ. Consequently, $\gamma^{SN}(\phi^{SN}(R^{SN}(e))) = e$. Therefore, $\gamma^{SN}(\phi^{SN}(R^{SN}(x))) = e$. Now our goal becomes $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, which we already know.

iTm-let

$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e_1 : \sigma_1 \qquad \Sigma;\Gamma_C;\Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \qquad \Gamma_C;\Gamma \vdash_{ty} \sigma_1}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \textbf{let } x : \sigma_1 = e_1 \textbf{ in } e_2 : \sigma_2}\ \text{iTm-let}$$

By induction hypothesis, we know

$$\gamma^{SN}(\phi^{SN}(R^{SN}(e_1))) \in \mathcal{SN}[\![\sigma_1]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{145}$$

$$\gamma^{SN}(\phi^{SN}{}_2(R^{SN}(e_2))) \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{146}$$

Given Equation 145, we can choose $\phi^{SN}{}_2 = \phi^{SN}, x \mapsto \gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))$. Equation 146 thus reduces to:

$$\gamma^{SN}(\phi^{SN}, x \mapsto \gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))(R^{SN}(e_2))) \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{147}$$

Simplifying Equation 147 results in:

$$[\gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))/x](\gamma^{SN}(\phi^{SN}(R^{SN}(e_2)))) \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{148}$$

Our goal is $\gamma^{SN}(\phi^{SN}(R^{SN}(\textbf{let } x : \sigma_1 = e_1 \textbf{ in } e_2))) \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$.
We know that

$$\gamma^{SN}(\phi^{SN}(R^{SN}(\textbf{let } x : \sigma_1 = e_1 \textbf{ in } e_2)))$$
$$= \textbf{let } x : \sigma_1 = (\gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))) \textbf{ in } (\gamma^{SN}(\phi^{SN}(R^{SN}(e_2)))) \tag{149}$$

$$\Sigma \vdash \textbf{let } x : \sigma_1 = (\gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))) \textbf{ in } (\gamma^{SN}(\phi^{SN}(R^{SN}(e_2))))$$
$$\longrightarrow [\gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))/x](\gamma^{SN}(\phi^{SN}(R^{SN}(e_2)))) \tag{150}$$

Consequently, by Substitution for Context Interpretation (Lemma 48), we know that

$$\Sigma;\Gamma_C;\bullet \vdash_{tm} \textbf{let } x : \sigma_1 = (\gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))) \textbf{ in } (\gamma^{SN}(\phi^{SN}(R^{SN}(e_2)))) : R^{SN}(\sigma_2) \tag{151}$$

By Equations 148, 150 and 151, in combination with Strong Normalization preserved by forward/backward reduction (Lemma 47), we get that

$$\textbf{let } x : \sigma_1 = (\gamma^{SN}(\phi^{SN}(R^{SN}(e_1)))) \textbf{ in } (\gamma^{SN}(\phi^{SN}(R^{SN}(e_2)))) \in \mathcal{SN}[\![\sigma_2]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{152}$$

The goal follows from Equations 149 and 152.

iTm-method

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \qquad (m : TC\,a : \sigma') \in \Gamma_C}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma'}\ \text{iTm-method}$$

By inversion on the dictionary typing, we have two cases:

- $\dfrac{(\delta : Q) \in \Gamma \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_d \delta : Q}\ \text{D-var}$ with $Q = TC\,\sigma$.
  Given $\gamma^{SN} \in \mathcal{H}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$, we know that there exists a dv in γ^{SN} such that

$$\Sigma; \Gamma_C; \bullet \vdash_d dv : TC\,R^{SN}(\sigma). \tag{153}$$

  Without loss of generality, suppose

$$dv = D\,\overline{\sigma}_j\,\overline{dv}_j. \tag{154}$$

  Therefore

$$\gamma^{SN}(\phi^{SN}(R^{SN}(d.m))) = (D\,\overline{\sigma}_j\,\overline{dv}_j).m. \tag{155}$$

  Now our goal is to prove that:

$$(D\,\overline{\sigma}_j\,\overline{dv}_j).m \in \mathcal{SN}[\![[\sigma/a]\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}}. \tag{156}$$

  Substituting Equation 154 in Equation 153 results in:

$$\Sigma; \Gamma_C; \bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv}_j : TC\,R^{SN}(\sigma). \tag{157}$$

  By inversion, we have

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_1).m \mapsto \Lambda \overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e \in \Sigma \tag{158}$$

$$\overline{\Sigma; \Gamma_C; \bullet \vdash_d dv_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{159}$$

$$R^{SN}(\sigma) = [\overline{\sigma}_j/\overline{a}_j]\sigma_1 \tag{160}$$

$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \tag{161}$$

$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i}^{\,i} \tag{162}$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{\,j} \tag{163}$$

$$\Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e : [\sigma_1/a]\sigma' \tag{164}$$

$$\textit{where } \Sigma = \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_1).m \mapsto \Lambda \overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e, \Sigma_2$$

  By evaluation rule (iEval), together with Equation 158, we know that:

$$\Sigma \vdash (D\,\overline{\sigma}_j\,\overline{dv}_j).m \longrightarrow (\Lambda \overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e)\,\overline{\sigma}_j\,\overline{dv}_j \tag{165}$$

  By Strong Normalization preserved by forward/backward reduction (Lemma 47), Goal 156 becomes:

$$e\,\overline{\sigma}_j\,\overline{dv}_j \in \mathcal{SN}[\![[\sigma/a]\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}}. \tag{166}$$

  By applying weakening Lemma 32 on Equation 162, we get:

$$\overline{\Gamma_C; \Gamma \vdash_Q Q_i}^{\,i} \tag{167}$$

$$\tag{168}$$

  By applying rules iTm-forallI and iTm-constrI (in combination with Equation 168) on Equation 164, we get that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda \overline{a}_j.\lambda\overline{\delta} : \overline{Q}_i.e : \forall \overline{a}_j.\overline{Q}_i \Rightarrow [\sigma_1/a]\sigma' \tag{169}$$

  By applying rules iTm-forallE (in combination with Equation 163) and iTm-constrE (in combination with Equation 159) on Equation 169, we get that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{dv}_i : [\overline{\sigma}_j/\overline{a}_j][\sigma_1/a]\sigma' \tag{170}$$

From iCtx-clsEnv, we know that σ' only contains one free variable a. We can thus simplify Equation 170 to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{dv_i} : [[\overline{\sigma}_j/\overline{a}_j]\sigma_1/a]\sigma' \tag{171}$$

And by substituting Equation 160:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{dv_i} : [R^{SN}(\sigma)/a]\sigma' \tag{172}$$

Again, since σ' contains only one free variable a, we can further rewrite this result to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e\,\overline{\sigma}_j\,\overline{dv_i} : R^{SN}([\sigma/a]\sigma') \tag{173}$$

By induction hypothesis on Equation 173, we get:

$$e\,\overline{\sigma}_j\,\overline{dv_i} \in \mathcal{SN}[\![R^{SN}([\sigma/a]\sigma')]\!]_{\bullet}^{\Sigma',\Gamma_C} \tag{174}$$

By weakening over Σ' (Lemma 34), we get:

$$e\,\overline{\sigma}_j\,\overline{dv_i} \in \mathcal{SN}[\![R^{SN}([\sigma/a]\sigma')]\!]_{\bullet}^{\Sigma,\Gamma_C} \tag{175}$$

Because we know that $R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, by Compositionality for Strong Normalization (Context Interpretation) (Corollary 1), Equation 175 proves our Goal 166.

- $$\dfrac{\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i}^{\,i} \quad \overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{\,j} \quad \begin{array}{c}\Sigma = \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e, \Sigma_2 \\ \Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e : [\sigma_q/a]\sigma_m\end{array} \quad \begin{array}{c}\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \\ \overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i}\end{array}}{\Sigma; \Gamma_C; \Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q}\ \text{D-con}$$

The goal to be proven is the following:

$$\gamma^{SN}(\phi^{SN}(R^{SN}(d.m))) \in \mathcal{SN}[\![[\sigma/a]\sigma']\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{176}$$

$$\textit{where } \sigma = [\overline{\sigma}_j/\overline{a}_j]\sigma_q \tag{177}$$

From the rule premise, we know that:

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e \in \Sigma \tag{178}$$

$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i}^{\,i} \tag{179}$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{\,j} \tag{180}$$

$$\Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e : [\sigma_q/a]\sigma_m \tag{181}$$

$$\textit{where } \Sigma = \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e, \Sigma_2$$

By evaluation rule (iEval), we know that:

$$\Sigma \vdash (D\,\overline{\sigma}_j\,\overline{d}_i).m \longrightarrow (\Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e)\,\overline{\sigma}_j\,\overline{d}_i \tag{182}$$

By applying the substitutions, we can verify that:

$$\Sigma \vdash \gamma^{SN}(\phi^{SN}(R^{SN}((D\,\overline{\sigma}_j\,\overline{d}_i).m))) \longrightarrow (\Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e)\,R^{SN}(\overline{\sigma}_j)\,R^{SN}(\gamma^{SN}(\overline{d}_i))$$

By Strong Normalization preserved by forward/backward reduction (Lemma 47), our goal becomes:

$$(\Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e)\,R^{SN}(\overline{\sigma}_j)\,R^{SN}(\gamma^{SN}(\overline{d}_i)) \in \mathcal{SN}[\![[\sigma/a]\sigma']\!]_{R^{SN}}^{\Sigma,\Gamma_C}. \tag{183}$$

By applying rules iTm-constrI (in combination with Equation 179) and iTm-forallI, Equation 181 reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e : \forall\overline{a}_j.\overline{Q}_i \Rightarrow [\sigma_q/a]\sigma_m \tag{184}$$

By applying this to iTm-forallE, in combination with Equation 180, we get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e)\,R^{SN}(\overline{\sigma}_j) : [R^{SN}(\overline{\sigma}_j)/\overline{a}_j]\overline{Q}_i \Rightarrow [R^{SN}(\overline{\sigma}_j)/\overline{a}_j][\sigma_q/a]\sigma_m$$

Through weakening Lemma 30, we know this is equivalent to:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e)\,R^{SN}(\overline{\sigma}_j) : [R^{SN}(\overline{\sigma}_j)/\overline{a}_j]\overline{Q}_i \Rightarrow [R^{SN}(\overline{\sigma}_j)/\overline{a}_j][\sigma_q/a]\sigma_m \tag{185}$$

The 6$^{\text{th}}$ rule premise tells us that:

$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i}$$

From Equation 179, we know that $\overline{Q}_j$ only contains free variables $\overline{a}_j$. By applying the substition, we can thus verify that:

$$\overline{\Sigma; \Gamma_C; \bullet \vdash_d R^{SN}(\gamma^{SN}(d_i)) : [R^{SN}(\overline{\sigma}_j)/\overline{a}_j]Q_i}^{\,i}$$

Therefore, by ITm-constrE, Equation 185 is equivalent to:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e)\, R^{SN}(\overline{\sigma}_j)\, R^{SN}(\gamma^{SN}(\overline{d}_i)) : [R^{SN}(\overline{\sigma}_j)/\overline{a}_j][\sigma_q/a]\sigma_m \tag{186}$$

From Lemma 43, in combination with Equation 181, we know that $[\sigma_q/a]\sigma_m$ only contains free variables $\overline{a}_j$. we can thus rewrite Equation 186 to:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e)\, R^{SN}(\overline{\sigma}_j)\, R^{SN}(\gamma^{SN}(\overline{d}_i)) : R^{SN}([\overline{\sigma}_j/\overline{a}_j][\sigma_q/a]\sigma_m) \tag{187}$$

From the environment well-formedness we know that $\sigma' = \sigma_m$ and that σ' only contains free variable a. We can thus rewrite Equation 187 to:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e)\, R^{SN}(\overline{\sigma}_j)\, R^{SN}(\gamma^{SN}(\overline{d}_i)) : R^{SN}([[\overline{\sigma}_j/\overline{a}_j]\sigma_2/a]\sigma') \tag{188}$$

By substituting Equation 177, we have:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e)\, R^{SN}(\overline{\sigma}_j)\, R^{SN}(\gamma^{SN}(\overline{d}_i)) : R^{SN}([\sigma/a]\sigma') \tag{189}$$

By induction hypothesis on Equation 189, we get:

$$(\Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e)\, R^{SN}(\overline{\sigma}_j)\, R^{SN}(\gamma^{SN}(\overline{d}_i)) \in \mathcal{SN}[\![R^{SN}([\sigma/a]\sigma')]\!]_{\bullet}^{\Sigma,\Gamma_C}. \tag{190}$$

Because we know that $R^{SN} \in \mathcal{F}^{SN}[\![\Gamma]\!]^{\Sigma,\Gamma_C}$, by Compositionality for Strong Normalization (Context Interpretation) (Corollary 1), Equation 190 proves our Goal 183.

ITm-arrI
$$\frac{\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1 . e : \sigma_1 \to \sigma_2}\ \text{ITm-arrI}$$

Because $\gamma^{SN}(\phi^{SN}(R^{SN}((\lambda x : \sigma_1 . e)))) = \lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))$, our goal is to show that:

$$\lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \in \mathcal{SN}[\![\sigma_1 \to \sigma_2]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{191}$$

By definition, we need to prove the following goals:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} \lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) : R^{SN}{}_1(\sigma_1 \to \sigma_2) \tag{192}$$

$$\exists v : \Sigma \vdash \lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \longrightarrow^* v \tag{193}$$

$$\forall e' : e' \in \mathcal{SN}[\![\sigma_1]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \Rightarrow (\lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\, e' \in \mathcal{SN}[\![\sigma_2]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{194}$$

By Substitution for Context Interpretation (Lemma 48), we can easily prove Equation 192.
Furthermore, $\lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))$ is already a value, which proves Equation 193. Now given

$$\forall e' : e' \in \mathcal{SN}[\![\sigma_1]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{195}$$

We need to show

$$(\lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\, e' \in \mathcal{SN}[\![\sigma_2]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{196}$$

Let $\phi^{SN'} = \phi^{SN}, x \mapsto e'$. By induction hypothesis, we have

$$\gamma^{SN}(\phi^{SN}, x \mapsto e'(R^{SN}(e))) \in \mathcal{SN}[\![\sigma_2]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{197}$$

We know that:

$$\begin{aligned} &(\lambda x : R^{SN}(\sigma_1).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\, e' \\ &\longrightarrow [e'/x](\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \\ &= \gamma^{SN}(\phi^{SN}, x \mapsto e'(R^{SN}(e))) \end{aligned}$$

Consequently, by Strong Normalization preserved by forward/backward reduction(Lemma 47), Equation 197 proves 196.

ITm-arrE
$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \qquad \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2}\ \text{ITm-arrE}$$

By induction hypothesis, we have:

$$\gamma^{SN}(\phi^{SN}(R^{SN}(e_1))) \in \mathcal{SN}[\![\sigma_1 \to \sigma_2]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{198}$$

$$\gamma^{SN}(\phi^{SN}(R^{SN}(e_2))) \in \mathcal{SN}[\![\sigma_1]\!]_{R^{SN}}^{\Sigma,\Gamma_C} \tag{199}$$

By the definition of Equation 198, applying Equation 199 results in:

$$(\gamma^{SN}(\phi^{SN}(R^{SN}(e_1))))\, (\gamma^{SN}(\phi^{SN}(R^{SN}(e_2)))) \in \mathcal{SN}[\![\sigma_2]\!]_{R^{SN}}^{\Sigma,\Gamma_C}$$

which is exactly our goal since

$$\gamma^{SN}(\phi^{SN}(R^{SN}((e_1\, e_2)))) = (\gamma^{SN}(\phi^{SN}(R^{SN}(e_1))))\,(\gamma^{SN}(\phi^{SN}(R^{SN}(e_2))))$$

iTm-constrI
$$\frac{\Sigma;\Gamma_C;\Gamma,\delta : Q \vdash_{tm} e : \sigma \qquad \Gamma_C;\Gamma \vdash_Q Q}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma}\ \text{iTm-constrI}$$

Because $\gamma^{SN}(\phi^{SN}(R^{SN}((\lambda\delta : Q.e)))) = \lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))$, our goal is to show that:

$$\lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \in \mathcal{SN}[\![Q \Rightarrow \sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{200}$$

By definition, we need to prove the following goals:

$$\Sigma;\Gamma_C;\bullet \vdash_{tm} \lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) : R^{SN}{}_1(Q \Rightarrow \sigma) \tag{201}$$
$$\exists v : \Sigma \vdash \lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \longrightarrow^* v \tag{202}$$
$$\forall d : \Sigma;\Gamma_C;\bullet \vdash_d d : R^{SN}{}_1(Q) \Rightarrow (\lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\, d \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{203}$$

By Substitution for Context Interpretation (Lemma 48), we can easily prove Equation 201.
Furthermore, $\lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))$ is already a value, which proves Equation 202. Now given

$$\forall d : \Sigma;\Gamma_C;\bullet \vdash_d d : R^{SN}{}_1(Q) \tag{204}$$

We need to show

$$(\lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\, d \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{205}$$

Let $\gamma^{SN\prime} = \gamma^{SN}, \delta \mapsto d$. By induction hypothesis, we have

$$\gamma^{SN}, \delta \mapsto d(\phi^{SN}(R^{SN}(e))) \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{206}$$

We know that:

$$\begin{aligned}
&(\lambda\delta : R^{SN}(Q).(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\, d\\
&\longrightarrow [d/\delta](\gamma^{SN}(\phi^{SN}(R^{SN}(e))))\\
&= \gamma^{SN}, \delta \mapsto d(\phi^{SN}(R^{SN}(e)))
\end{aligned}$$

Consequently, by Strong Normalization preserved by forward/backward reduction(Lemma 47), Equation 206 proves 205.

iTm-constrE
$$\frac{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : Q \Rightarrow \sigma \qquad \Sigma;\Gamma_C;\Gamma \vdash_d d : Q}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} e\, d : \sigma}\ \text{iTm-constrE}$$

By induction hypothesis, we have

$$\gamma^{SN}(\phi^{SN}(R^{SN}(e))) \in \mathcal{SN}[\![Q \Rightarrow \sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{207}$$

By Substitution for Context Interpretation (Lemma 48), we know that:

$$\Sigma;\Gamma_C;\bullet \vdash_d \gamma^{SN}(\phi^{SN}(R^{SN}(d))) : R^{SN}(Q) \tag{208}$$

By the definition of Equation 207, applying Equation 208 results in:

$$(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))\,(\gamma^{SN}(\phi^{SN}(R^{SN}(d)))) \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$$

which is exactly our goal since

$$\gamma^{SN}(\phi^{SN}(R^{SN}((e\, d)))) = (\gamma^{SN}(\phi^{SN}(R^{SN}(e))))\,(\gamma^{SN}(\phi^{SN}(R^{SN}(d))))$$

iTm-forallI
$$\frac{\Sigma;\Gamma_C;\Gamma, a \vdash_{tm} e : \sigma}{\Sigma;\Gamma_C;\Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma}\ \text{iTm-forallI}$$

Because $\gamma^{SN}(\phi^{SN}(R^{SN}((\Lambda a.e)))) = \Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))$, our goal is to show that:

$$\Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \in \mathcal{SN}[\![\forall a.\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{209}$$

By definition, we need to prove the following goals:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} \Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) : R^{SN}{}_1(\forall a.\sigma) \tag{210}$$

$$\exists v : \Sigma \vdash \Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \longrightarrow^* v \tag{211}$$

$$\forall \sigma', r = \mathcal{SN}[\![\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}} \Rightarrow (\Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\,\sigma' \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN},a\mapsto(\sigma',r)} \tag{212}$$

By Substitution for Context Interpretation (Lemma 48), we can easily prove Equation 210.
Furthermore, $\Sigma \vdash \Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \longrightarrow^* v$ is already a value, which proves Equation 211. Now given

$$\forall \sigma', r = \mathcal{SN}[\![\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{213}$$

We need to show

$$(\Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\,\sigma' \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN},a\mapsto(\sigma',r)} \tag{214}$$

Let $R^{SN\prime} = R^{SN}, a \mapsto (\sigma', r)$. By induction hypothesis, we have:

$$\gamma^{SN}(\phi^{SN}(R^{SN}, a \mapsto (\sigma', r)(e))) \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN},a\mapsto(\sigma',r)} \tag{215}$$

We know that:

$$\begin{aligned} &(\Lambda a.(\gamma^{SN}(\phi^{SN}(R^{SN}(e)))))\,\sigma' \\ \longrightarrow\ &[\sigma'/a](\gamma^{SN}(\phi^{SN}(R^{SN}(e)))) \\ =\ &\gamma^{SN}(\phi^{SN}(R^{SN}, a \mapsto (\sigma', r)(e))) \end{aligned}$$

Consequently, by Strong Normalization preserved by forward/backward reduction(Lemma 47), Equation 215 proves 214.

ITM-FORALLE

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\,\sigma : [\sigma/a]\sigma'}\ \text{ITM-FORALLE}$$

By induction hypothesis, we have:

$$\gamma^{SN}(\phi^{SN}(R^{SN}(e))) \in \mathcal{SN}[\![\forall a.\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN}} \tag{216}$$

By Substitution for Context Interpretation (Lemma 48), we know:

$$\Gamma_C; \bullet \vdash_{ty} R^{SN}(\sigma) \tag{217}$$

Choose $r = \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$. By the definition of Equation 216, applying Equation 217 results in:

$$(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))\,(R^{SN}(\sigma)) \in \mathcal{SN}[\![\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN},a\mapsto(R^{SN}(\sigma),r)}$$

By Compositionality for Strong Normalization (Lemma 49), we get:

$$(\gamma^{SN}(\phi^{SN}(R^{SN}(e))))\,(R^{SN}(\sigma)) \in \mathcal{SN}[\![[\sigma/a]\sigma']\!]^{\Sigma,\Gamma_C}_{R^{SN},a\mapsto(\sigma,r)}$$

which is exactly our goal since

$$\gamma^{SN}(\phi^{SN}(R^{SN}((e\,\sigma)))) = (\gamma^{SN}(\phi^{SN}(R^{SN}(e))))\,(R^{SN}(\sigma))$$

□

THEOREM 12 (STRONG NORMALIZATION - PART B).
If $e \in \mathcal{SN}[\![\sigma]\!]^{\Sigma,\Gamma_C}_{R^{SN}}$ *then* $\exists v : \Sigma \vdash e \longrightarrow^* v$.

PROOF. This goal is baked into the relation. It follows by straightforward induction on σ.

□

L ELABORATION EQUIVALENCE THEOREMS

Theorem 13 (Equivalence - Environments). *If* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $\vdash^{M}_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$.

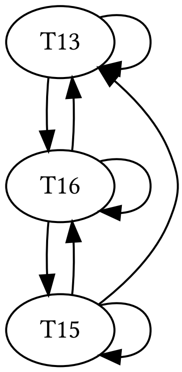

Fig. 15. Dependency graph for Theorems 13, 15 and 16

Proof. By induction on the environment well-formedness relation. This theorem is mutually proven with Theorems 15 and 16 (Figure 15). Note that at the dependencies between Theorem 13 and 16 and between Theorem 15 and 16, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Consequenty, the size of P is strictly decreasing at every cycle and the induction remains well-founded.

sCtxT-empty $\vdash_{ctx} \bullet; \bullet; \bullet \rightsquigarrow \bullet$

The goal follows directly from sCtx-empty and Ctx-Empty.

sCtxT-clsEnv $\vdash_{ctx} \bullet; \Gamma_C, m : \overline{TC_i\ a} \Rightarrow TC\ a : \forall \overline{a}_j . \overline{TC_i\ a}^i \Rightarrow \tau; \bullet \rightsquigarrow \bullet$

The goal to be proven is the following:

$$\vdash^{M}_{ctx} \bullet; \Gamma_C, m : \overline{TC_i\ a} \Rightarrow TC\ a : \forall \overline{a}_j . \overline{TC_i\ a}^i \Rightarrow \tau; \bullet \rightsquigarrow \Sigma; \Gamma_C; \Gamma \tag{218}$$

$$\Gamma_C; \Gamma \rightsquigarrow \bullet \tag{219}$$

From the rule premise we get that:

$$\Gamma_C; \bullet, a \vdash_{ty} \forall \overline{a}_j . \overline{TC_i\ a}^i \Rightarrow \tau \rightsquigarrow \sigma \tag{220}$$

$$\overline{\Gamma_C; \bullet, a \vdash_Q TC_i\ a \rightsquigarrow \sigma_i}^i \tag{221}$$

$$\vdash_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet \tag{222}$$

By applying the induction hypothesis to Equation 222, we get:

$$\vdash^{M}_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C'; \bullet \tag{223}$$

$$\Gamma_C'; \bullet \rightsquigarrow \bullet \tag{224}$$

From sCtxT-tyEnvTy, sCtx-tyEnvTy and Ctx-TVar, in combination with Equation 222, 223 and 224, respectively, we get:

$$\vdash_{ctx} \bullet; \Gamma_C; \bullet, a \rightsquigarrow \bullet, a$$

$$\vdash^{M}_{ctx} \bullet; \Gamma_C; \bullet, a \rightsquigarrow \bullet; \Gamma_C'; \bullet, a$$

$$\Gamma_C'; \bullet, a \rightsquigarrow \bullet, a$$

Applying type and constraint equivalence (Theorem 14) to Equations 220 and 221, together with these results, gives us:

$$\Gamma_C; \bullet, a \vdash_{ty}^{M} \forall \overline{a}_j.\overline{TC_i\, a}^i \Rightarrow \tau \rightsquigarrow \sigma \tag{225}$$
$$\Gamma_C{}'; \bullet, a \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{226}$$
$$\overline{\Gamma_C; \bullet, a \vdash_{Q}^{M} TC_i\, a \rightsquigarrow Q_i}^i \tag{227}$$
$$\overline{\Gamma_C{}'; \bullet, a \vdash_{Q} Q_i \rightsquigarrow \sigma_i}^i \tag{228}$$

Goal 218 follows from sCtx-clsEnv, in combination with Equations 223, 225 and 227, with $\Sigma = \bullet$, $\Gamma_C = \Gamma_C{}', m : TC\, a : \sigma$ and $\Gamma = \bullet$. Consequently, Goal 219 follows from Ctx-Empty.

sCtxT-tyEnvTm $\vdash_{ctx} \bullet; \Gamma_C; \Gamma, x : \sigma \rightsquigarrow \Gamma, x : \sigma$

The goal to be proven is the following:

$$\vdash_{ctx}^{M} \bullet; \Gamma_C; \Gamma, x : \sigma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \tag{229}$$
$$\Gamma_C; \Gamma \rightsquigarrow \Gamma, x : \sigma \tag{230}$$

From the rule premise we get that:

$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{231}$$
$$x \notin \mathbf{dom}(\Gamma) \tag{232}$$
$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{233}$$

By applying the induction hypothesis to Equation 233, we get:

$$\vdash_{ctx}^{M} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma' \tag{234}$$
$$\Gamma_C; \Gamma' \rightsquigarrow \Gamma' \tag{235}$$

We know from type equivalence (Theorem 14), in combination with Equations 231, 234 and 235, that:

$$\Gamma_C; \Gamma \vdash_{ty}^{M} \sigma \rightsquigarrow \sigma \tag{236}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{237}$$

Goal 229 follows from sCtx-tyEnvTm, in combination with Equations 232, 234 and 236, with $\Sigma = \bullet$ and $\Gamma = \Gamma', x : \sigma$. Consequently, Goal 230 follows from Ctx-Var, in combination with Equations 235 and 237, with $\Gamma = \Gamma'$.

sCtxT-tyEnvTy $\vdash_{ctx} \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \Gamma, a$

Similar to the sCtxT-tyEnvTm case.

sCtxT-tyEnvD $\vdash_{ctx} \bullet; \Gamma_C; \Gamma, \delta : TC\, \tau \rightsquigarrow \Gamma, \delta : \sigma$

Similar to the sCtxT-tyEnvTm case.

sCtxT-pgmInst $\vdash_{ctx} P, (D : \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\, \tau).m \mapsto \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}'_h : e; \Gamma_C; \Gamma \rightsquigarrow \Gamma$

The goal to be proven is the following:

$$\vdash_{ctx}^{M} P, (D : \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\, \tau).m \mapsto \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}'_h : e; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \tag{238}$$
$$\Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{239}$$

From the rule premise we get that:

$$\textbf{unambig}(\forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau) \tag{240}$$

$$\Gamma_C;\bullet \vdash_C \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau \rightsquigarrow \forall \overline{b}_j.\overline{\sigma}_i \rightarrow [\sigma/a]\{m : \forall \overline{a}_k.\overline{\sigma}'_h \rightarrow \sigma'\} \tag{241}$$

$$(m : \overline{Q}'_m \Rightarrow TC\ a : \forall \overline{a}_k.\overline{Q}'_h \Rightarrow \tau') \in \Gamma_C \tag{242}$$

$$\Gamma_C;\bullet, a \vdash_{ty} \forall \overline{a}_k.\overline{Q}'_h \Rightarrow \tau' \rightsquigarrow \forall \overline{a}_k.\overline{\sigma}'_h \rightarrow \sigma' \tag{243}$$

$$\Gamma_C;\bullet, \overline{b}_j \vdash_{ty} \tau \rightsquigarrow \sigma \tag{244}$$

$$P;\Gamma_C;\bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}'_h \vdash_{tm} e \Leftarrow [\tau/a]\tau' \rightsquigarrow e \tag{245}$$

$$D \notin \textbf{dom}(P) \tag{246}$$

$$(D' : \forall \overline{b}'_k.\overline{Q}''_h \Rightarrow TC\,\tau'').m' \mapsto \Gamma' : e' \notin P\textbf{where}[\overline{\tau}_j/\overline{b}_j]\tau = [\overline{\tau}'_k/\overline{b}'_k]\tau'' \tag{247}$$

$$\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma \tag{248}$$

By applying the induction hypothesis to Equation 248, we get:

$$\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma';\Gamma_C;\Gamma \tag{249}$$

$$\Gamma_C;\Gamma \rightsquigarrow \Gamma \tag{250}$$

Goal 239 follows directly from Equation 250. From type and constraint equivalence (Theorem 14, the required assumptions follow straightforwardly from sCtxT-tyEnvTy, sCtx-tyEnvTy and Ctx-TVar, in combination with Equations 248, 249 and 250), together with Equations 241, 243 and 244, we know:

$$\Gamma_C;\bullet \vdash^M_C \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\tau \rightsquigarrow \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\sigma \tag{251}$$

$$\Gamma_C;\bullet, a \vdash^M_{ty} \forall \overline{a}_k.\overline{Q}'_h \Rightarrow \tau' \rightsquigarrow \forall \overline{a}_k.\overline{Q}'_h \Rightarrow \sigma' \tag{252}$$

$$\Gamma_C;\bullet, a \vdash_{ty} \forall \overline{a}_k.\overline{Q}'_h \Rightarrow \sigma' \rightsquigarrow \forall \overline{a}_k.\overline{\sigma}'_h \rightarrow \sigma' \tag{253}$$

$$\Gamma_C;\bullet, \overline{b}_j \vdash^M_{ty} \tau \rightsquigarrow \sigma \tag{254}$$

$$\Gamma_C;\bullet, \overline{b}_j \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{255}$$

Similarly, from expression equivalence (Theorem 16, the environment well-formedness assumption is constructed straightforwardly), together with Equation 245, we get:

$$P;\Gamma_C;\bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}'_h \vdash^M_{tm} e \Leftarrow [\tau/a]\tau' \rightsquigarrow e \tag{256}$$

Goal 238 follows from sCtx-pgmInst, in combination with Equations 240, 251, 242, 256, 252, 246, 247 and 249, and with $\Sigma = \Sigma', (D : \forall \overline{b}_j.\overline{Q}_i \Rightarrow TC\,\sigma').m \mapsto \Lambda\overline{b}_j.\lambda\overline{\delta}_i : \overline{Q}_i.\Lambda\overline{a}_k.\lambda\overline{\delta}_h : [\sigma/a]\overline{Q}'_h.e$.

□

Theorem 14 (Equivalence - Types and Constraints).

- *If* $\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$ *and* $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *then* $\Gamma_C;\Gamma \vdash^M_{ty} \sigma \rightsquigarrow \sigma$ *and* $\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$.
- *If* $\Gamma_C;\Gamma \vdash_Q Q \rightsquigarrow \sigma$ *and* $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *then* $\Gamma_C;\Gamma \vdash^M_Q Q \rightsquigarrow Q$ *and* $\Gamma_C;\Gamma \vdash_Q Q \rightsquigarrow \sigma$.
- *If* $\Gamma_C;\Gamma \vdash_C C \rightsquigarrow \sigma$ *and* $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma$ *and* $\Gamma_C;\Gamma \rightsquigarrow \Gamma$ *then* $\Gamma_C;\Gamma \vdash^M_C C \rightsquigarrow C$ *and* $\Gamma_C;\Gamma \vdash_C C$.

Proof. By induction on the size of the type σ, class constraint Q or constraint C.

Part 1 By case analysis on the type well-formedness derivation.

sTyT-bool $\Gamma_C;\Gamma \vdash_{ty} Bool \rightsquigarrow Bool$

The goal to be proven is the following:

$$\Gamma_C;\Gamma \vdash^M_{ty} Bool \rightsquigarrow \sigma \tag{257}$$

$$\Gamma_C;\Gamma \vdash_{ty} \sigma \rightsquigarrow Bool \tag{258}$$

Goals 257 and 258 follow directly from sTy-bool and iTy-bool respectively, with $\sigma = Bool$.

sTyT-var $\Gamma_C; \Gamma \vdash_{ty} a \rightsquigarrow a$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash^M_{ty} a \rightsquigarrow \sigma \tag{259}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow a \tag{260}$$

From the rule premise we get that:

$$a \in \Gamma \tag{261}$$

By applying Lemmas 16 and 42 to Equation 261, we get:

$$a \in \Gamma \tag{262}$$
$$a \in \Gamma \tag{263}$$

Goal 259 and 260 follow directly from sTy-var and iTy-var respectively, in combination with Equations 262 and 263, with $\sigma = a$.

sTyT-arrow $\Gamma_C; \Gamma \vdash_{ty} \tau_1 \to \tau_2 \rightsquigarrow \sigma_1 \to \sigma_2$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_1 \to \tau_2 \rightsquigarrow \sigma \tag{264}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma_1 \to \sigma_2 \tag{265}$$

From the rule premise we get that:

$$\Gamma_C; \Gamma \vdash_{ty} \tau_1 \rightsquigarrow \sigma_1 \tag{266}$$
$$\Gamma_C; \Gamma \vdash_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{267}$$

By applying the induction hypothesis on Equations 266 and 267, we get:

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma_1 \tag{268}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \tag{269}$$
$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{270}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2 \tag{271}$$

Goals 264 and 265 follow directly from sTy-arrow and iTy-arrow respectively, in combination with Equations 268, 269, 270 and 271, with $\sigma = \sigma_1 \to \sigma_2$.

sTyT-qual $\Gamma_C; \Gamma \vdash_{ty} Q \Rightarrow \rho \rightsquigarrow \sigma_1 \to \sigma_2$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash^M_{ty} Q \Rightarrow \rho \rightsquigarrow \sigma \tag{272}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma_1 \to \sigma_2 \tag{273}$$

From the rule premise we get that:

$$\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_1 \tag{274}$$
$$\Gamma_C; \Gamma \vdash_{ty} \rho \rightsquigarrow \sigma_2 \tag{275}$$

By applying the induction hypothesis on Equation 275, we get:

$$\Gamma_C; \Gamma \vdash^M_{ty} \rho \rightsquigarrow \sigma_2 \tag{276}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2 \tag{277}$$

By applying Part 2 of this theorem on Equation 274, we get:

$$\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q \tag{278}$$
$$\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_1 \tag{279}$$

Goals 272 and 273 follow directly from sTy-qual and iTy-qual respectively, in combination with Equations 276, 277, 278 and 279, with $\sigma = Q \Rightarrow \sigma_2$.

sTyT-scheme $\Gamma_C; \Gamma \vdash_{ty} \forall a.\sigma \rightsquigarrow \forall a.\sigma$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash^M_{ty} \forall a.\sigma \rightsquigarrow \sigma \tag{280}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \forall a.\sigma \tag{281}$$

From the rule premise we get that:

$$a \notin \Gamma \tag{282}$$
$$\Gamma_C; \Gamma, a \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{283}$$

By repeated case analysis on the 2nd hypothesis (sCtxT-pgmInst), we get:

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{284}$$

From sCtxT-tyEnvTy, in combination with Equations 284 and 282, we know that:

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \Gamma, a \tag{285}$$

Similarly, we get from sCtx-tyEnvTy and ctx-TVar that:

$$\vdash^M_{ctx} \bullet; \Gamma_C; \Gamma, a \rightsquigarrow \bullet; \Gamma_C; \Gamma, a \tag{286}$$
$$\Gamma_C; \Gamma, a \rightsquigarrow \Gamma, a \tag{287}$$

By applying the induction hypothesis on Equation 283, together with Equations 285, 286 and 287, we get:

$$\Gamma_C; \Gamma, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma' \tag{288}$$
$$\Gamma_C; \Gamma, a \vdash_{ty} \sigma' \rightsquigarrow \sigma \tag{289}$$

Goals 280 and 281 follow directly from sTy-scheme and iTy-scheme respectively, in combination with Equations 282, 288 and 289, with $\sigma = \forall a.\sigma'$.

Part 2 By case analysis on the class constraint well-formedness derivation.

sQT-TC $\Gamma_C; \Gamma \vdash_Q TC\,\tau \rightsquigarrow [\sigma'/a]\{m : \sigma\}$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash^M_Q TC\,\tau \rightsquigarrow Q \tag{290}$$
$$\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow [\sigma'/a]\{m : \sigma\} \tag{291}$$

From the rule premise we get that:

$$\Gamma_C; \Gamma \vdash_{ty} \tau \rightsquigarrow \sigma' \tag{292}$$
$$\Gamma_C = \Gamma_{C1}, m : \overline{Q}_i \Rightarrow TC\,a : \sigma, \Gamma_{C2} \tag{293}$$
$$\Gamma_{C1}; \bullet, a \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{294}$$

By repeated case analysis on the 2nd hypothesis (sCtxT-pgmInst), we get:

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{295}$$

From sCtxT-tyEnvTy, together with Equation 295 and the fact that $a \notin \bullet$, we know that:

$$\vdash_{ctx} \bullet; \Gamma_{C1}; \bullet, a \rightsquigarrow \bullet, a \tag{296}$$

Similarly, we get from sCtx-tyEnvTy and ctx-TVar that:

$$\vdash^M_{ctx} \bullet; \Gamma_{C1}; \bullet, a \rightsquigarrow \bullet; \Gamma_{C1}; \bullet, a \tag{297}$$
$$\Gamma_{C1}; \bullet, a \rightsquigarrow \Gamma, a \tag{298}$$

By applying Part 1 of this theorem to Equations 292 and 294, together with Equations 296, 297 and 298, we get:

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma' \tag{299}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma' \rightsquigarrow \sigma' \tag{300}$$
$$\Gamma_{C1}; \bullet, a \vdash^M_{ty} \sigma \rightsquigarrow \sigma \tag{301}$$
$$\Gamma_{C1}; \bullet, a \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{302}$$

Goal 290 follows from sQ-TC, together with Equations 299, 293 and 301, with $Q = TC\,\sigma'$. Consequently, Goal 291 follows from iQ-TC, together with Equations 300, 293 and 302.

Part 3 By case analysis on the constraint well-formedness derivation.

sCT-ABS $\Gamma_C; \Gamma \vdash_C \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q \rightsquigarrow \forall \overline{a}_j.\overline{\sigma}_i \rightarrow \sigma$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash_C^M \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q \rightsquigarrow C \tag{303}$$

$$\Gamma_C; \Gamma \vdash_C C \tag{304}$$

From the rule premise we get that:

$$\overline{\Gamma_C; \Gamma, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma_i}^{\,i} \tag{305}$$

$$\Gamma_C; \Gamma, \overline{a}_j \vdash_Q Q \rightsquigarrow \sigma \tag{306}$$

$$\overline{a}_j \notin \Gamma \tag{307}$$

By repeated case analysis on the 2nd hypothesis (sCtxT-pgmInst), we get:

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{308}$$

By sCtxT-tyEnvTy, it follows from Equations 307 and 308 that:

$$\vdash_{ctx} \bullet; \Gamma_C; \Gamma, \overline{a}_j \rightsquigarrow \Gamma, \overline{a}_j \tag{309}$$

Similarly, we get from sCtx-tyEnvTy and ctx-TVar that:

$$\vdash_{ctx}^M \bullet; \Gamma_C; \Gamma, \overline{a}_j \rightsquigarrow \bullet; \Gamma_C; \Gamma, \overline{a}_j \tag{310}$$

$$\Gamma_C; \Gamma, \overline{a}_j \rightsquigarrow \Gamma, \overline{a}_j \tag{311}$$

By applying Part 2 of this theorem to Equations 305 and 306, together with Equations 309, 310 and 311, we get:

$$\overline{\Gamma_C; \Gamma, \overline{a}_j \vdash_Q^M Q_i \rightsquigarrow Q_i}^{\,i} \tag{312}$$

$$\overline{\Gamma_C; \Gamma, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma_i}^{\,i} \tag{313}$$

$$\Gamma_C; \Gamma, \overline{a}_j \vdash_Q^M Q \rightsquigarrow Q \tag{314}$$

$$\Gamma_C; \Gamma, \overline{a}_j \vdash_Q Q \rightsquigarrow \sigma \tag{315}$$

Goal 303 follows from sC-abs, together with Equations 312 and 314, with $C = \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q$. Consequently, Goal 304 follows from iC-abs, together with Equations 313 and 315.

□

Theorem 15 (Equivalence - Dictionaries). *If* $P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow e$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow d$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow e$ *where* $\Gamma_C; \Gamma \vdash_Q^M Q \rightsquigarrow Q$ *and* $\vdash_{ctx}^M P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$.

Proof. By induction on the entailment derivation, and mutually proven with Theorems 13 and 16 (Figure 15). Note that at the dependencies between Theorem 13 and 16 and between Theorem 15 and 16, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Consequenty, the size of P is strictly decreasing at every cycle and the induction remains well-founded.

From environment equivalence (Theorem 13), in combination with the 2nd hypothesis, we derive that:

$$\vdash_{ctx}^M P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \tag{316}$$

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{317}$$

sEntailT-local $P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow \delta$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow d \tag{318}$$
$$\Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow \delta \tag{319}$$
$$\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q \tag{320}$$

We know from the rule premise that:

$$(\delta : Q) \in \Gamma$$

Consequently, Goal 320 follows by repeated case analysis on Equation 316 (sCtx-tyEnvD). Furthermore, we know from Lemma 17 that:

$$(\delta : Q) \in \Gamma$$

Goal 319 follows by D-var. Goal 318 follows by sEntail-local, with $d = \delta$.

sEntailT-inst $P; \Gamma_C; \Gamma \vDash Q \rightsquigarrow (\Lambda \overline{a}_j . \lambda \overline{\delta_i : \sigma'_i}^i . \{m = \Lambda \overline{b}_k . \lambda \overline{\delta'_h : \sigma''_h}^h . e\}) \overline{\sigma}_j \overline{e}_i$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow d \tag{321}$$
$$\Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow (\Lambda \overline{a}_j . \lambda \overline{\delta_i : \sigma'_i}^i . \{m = \Lambda \overline{b}_k . \lambda \overline{\delta'_h : \sigma''_h}^h . e\}) \overline{\sigma}_j \overline{e}_i \tag{322}$$
$$\Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q \tag{323}$$

From the rule premise we get that:

$$P = P_1, (D : \forall \overline{a}_j . \overline{Q}'_i \Rightarrow Q').m \mapsto \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h : e, P_2 \tag{324}$$
$$Q = [\overline{\tau}_j / \overline{a}_j] Q' \tag{325}$$
$$P_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e \tag{326}$$
$$\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{327}$$
$$\overline{\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j}^j \tag{328}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q'_i \rightsquigarrow \sigma'_i}^i \tag{329}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j, \overline{b}_k \vdash_Q Q_h \rightsquigarrow \sigma''_h}^h \tag{330}$$
$$\overline{P; \Gamma_C; \Gamma \vDash [\overline{\tau}_j / \overline{a}_j] Q'_i \rightsquigarrow e_i}^i \tag{331}$$

By applying the induction hypothesis to Equation 331, we get:

$$\overline{P; \Gamma_C; \Gamma \vDash^M [\overline{\tau}_j / \overline{a}_j] Q'_i \rightsquigarrow d_i}^i \tag{332}$$
$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j / \overline{a}_j] Q'_i \rightsquigarrow e_i}^i \tag{333}$$

By repeated case analysis on the 2nd hypothesis, we know that:

$$\vdash_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet$$

From sCtxT-tyEnvTy, we get that:

$$\vdash_{ctx} \bullet; \Gamma_C; \bullet, \overline{a}_j \rightsquigarrow \bullet, \overline{a}_j$$

Similarly, we can derive from Equations 316 and 317 that:

$$\vdash^M_{ctx} \bullet; \Gamma_C; \bullet, \overline{a}_j \rightsquigarrow \bullet; \Gamma_C; \bullet, \overline{a}_j$$
$$\Gamma_C; \bullet, \overline{a}_j \rightsquigarrow \bullet, \overline{a}_j$$

It follows from type and constraint equivalence (Theorem 14), in combination with Equations 328, 329, 316 and 317 that:

$$\overline{\Gamma_C; \Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^j \tag{334}$$
$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^j \tag{335}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash^M_Q Q'_i \rightsquigarrow Q'_i}^i \tag{336}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q'_i \rightsquigarrow \sigma'_i}^i \tag{337}$$

By repeated case analysis on Equation 316 (sCtx-pgmInst), together with Equation 324, we know that:

$$\Gamma_C; \bullet \vdash_C^M \forall \overline{a}_j.\overline{Q}'_i \Rightarrow Q' \rightsquigarrow \forall \overline{a}_j.\overline{Q}'_i \Rightarrow Q' \tag{338}$$

$$\vdash_{ctx}^M \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma \tag{339}$$

By applying Lemma 4 to Equations 338 and 339, we get:

$$\Gamma_C; \Gamma \vdash_C^M \forall \overline{a}_j.\overline{Q}'_i \Rightarrow Q' \rightsquigarrow \forall \overline{a}_j.\overline{Q}'_i \Rightarrow Q' \tag{340}$$

By case analysis on Equation 338 (sC-abs), we get that:

$$\Gamma_C; \Gamma, \overline{a}_j \vdash_Q^M Q' \rightsquigarrow Q' \tag{341}$$

It follows from Lemma 1 and Equations 341 and 334 that:

$$\Gamma_C; \Gamma \vdash_Q^M [\overline{\tau}_j/\overline{a}_j]Q' \rightsquigarrow [\overline{\sigma}_j/\overline{a}_j]Q' \tag{342}$$

Goal 323 follows directly from Equation 342, since we know that $Q = [\overline{\tau}_j/\overline{a}_j]Q'$ (Equation 325).
By applying Lemma 12 to Equation 326, we get that:

$$\vdash_{ctx} P_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h \rightsquigarrow \Gamma' \tag{343}$$

From expression equivalence (Theorem 16), in combination with Equations 326 and 343, we get:

$$P_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h \vdash_{tm}^M e \Rightarrow \tau \rightsquigarrow e \tag{344}$$

$$\Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i, \overline{b}_k, \overline{\delta}_h : \overline{Q}_h \vdash_{tm} e : \sigma \rightsquigarrow e \tag{345}$$

Goal 321 follows from sEntail-inst, in combination with Equations 324, 325, 316, 334 and 332, with $d = D\,\overline{\sigma}_j\,\overline{d}_i$.
By inversion on Equation 316, in combination with Equation 324, we get:

$$\Sigma = \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}'_i \Rightarrow Q').m \mapsto \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}'_i.\Lambda \overline{b}_k.\lambda \overline{\delta}_h : \overline{Q}_h.e, \Sigma_2 \tag{346}$$

By applying preservation Theorem 7 to Equation 316, we get:

$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \tag{347}$$

Finally, Goal 322 follows from D-con, in combination with Equations 337, 335, 333, 345, 346 and 347.

□

Theorem 16 (Equivalence - Expressions).

- *If* $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$
then $P; \Gamma_C; \Gamma \vdash_{tm}^M e \Rightarrow \tau \rightsquigarrow e$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$
where $\vdash_{ctx}^M P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C; \Gamma \vdash_{ty}^M \tau \rightsquigarrow \sigma$.
- *If* $P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$
then $P; \Gamma_C; \Gamma \vdash_{tm}^M e \Leftarrow \tau \rightsquigarrow e$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$
where $\vdash_{ctx}^M P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C; \Gamma \vdash_{ty}^M \tau \rightsquigarrow \sigma$.

Proof. By induction on the lexicographic order of the tuple (size of the expression e, typing mode). Regarding typing mode, we define type checking to be larger than type inference. In each mutual dependency, we know that the tuple size decreases, meaning that the induction is well-founded.

Furthermore, this theorem is mutually proven with Theorems 13 and 15 (Figure 15). Note that at the dependencies between Theorem 13 and 16 and between Theorem 15 and 16, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Consequenty, the size of P is strictly decreasing at every cycle and the induction remains well-founded.

From environment equivalence (Theorem 13), in combination with the 2^{nd} hypothesis, we derive that:

$$\vdash_{ctx}^M P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma \tag{348}$$

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{349}$$

Consequently, by Theorem 7 we derive that

$$\vdash_{ctx} \Sigma; \Gamma_C; \Gamma \tag{350}$$

Part 1 By case analysis on the typing derivation.

sTm-infT-true $\quad P; \Gamma_C; \Gamma \vdash_{tm} True \Rightarrow Bool \rightsquigarrow True$

The goal follows by sTm-inf-true, iTm-true (in combination with Equation 350) and sTy-bool, with $e = True$.

sTm-infT-False $P;\Gamma_C;\Gamma \vdash_{tm} False \Rightarrow Bool \rightsquigarrow False$

Similar to the sTm-infT-true case.

sTm-infT-Let $P;\Gamma_C;\Gamma \vdash_{tm} \textbf{let}\ x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\ \textbf{in}\ e_2 \Rightarrow \tau_2 \rightsquigarrow \textbf{let}\ x : \forall\overline{a}_j.\overline{\sigma}_k \rightarrow \sigma = \Lambda\overline{a}_j.\lambda\,\overline{\delta_k : \sigma_k}^{k}.e_1\ \textbf{in}\ e_2$

The goal to be proven is the following:

$$P;\Gamma_C;\Gamma \vdash^{M}_{tm} \textbf{let}\ x : \forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = e_1\ \textbf{in}\ e_2 \Rightarrow \tau_2 \rightsquigarrow e \tag{351}$$

$$\Sigma;\Gamma_C;\Gamma \vdash_{tm} e : \sigma_2 \rightsquigarrow \textbf{let}\ x : \forall\overline{a}_j.\overline{\sigma}_k \rightarrow \sigma = \Lambda\overline{a}_j.\lambda\,\overline{\delta_k : \sigma_k}^{k}.e_1\ \textbf{in}\ e_2 \tag{352}$$

$$\Gamma_C;\Gamma \vdash^{M}_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{353}$$

From the rule premise we know that:

$$x \notin \textbf{dom}(\Gamma) \tag{354}$$

$$\textbf{unambig}(\forall\overline{a}_j.\overline{Q}_i \Rightarrow \tau_1) \tag{355}$$

$$\textbf{closure}(\Gamma_C;\overline{Q}_i) = \overline{Q}_k \tag{356}$$

$$\overline{\Gamma_C;\Gamma \vdash_Q Q_k \rightsquigarrow \sigma_k}^{k} \tag{357}$$

$$\Gamma_C;\Gamma \vdash_{ty} \forall\overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \forall\overline{a}_j.\overline{\sigma}_k \rightarrow \sigma \tag{358}$$

$$\overline{\delta}_k\ \textbf{fresh} \tag{359}$$

$$P;\Gamma_C;\Gamma,\overline{a}_j,\overline{\delta}_k : \overline{Q}_k \vdash_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \tag{360}$$

$$P;\Gamma_C;\Gamma,x : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \vdash_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \tag{361}$$

By applying Lemma 14 to Equation 361, we get that:

$$\Gamma_C;\Gamma,x : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \vdash^{M}_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{362}$$

It is straightforward to see from the definition of type well-formedness, that Goal 353 follows from Equation 362, since term variables in the environment are not relevant for type well-formedness.
We know from the hypothesis that $\vdash_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Gamma$. By repeated case analysis on this result (sCtxT-pgmInst), we get that $\vdash_{ctx} \bullet;\Gamma_C;\Gamma \rightsquigarrow \Gamma$. From sCtxT-tyEnvTm, sCtxT-tyEnvTy and sCtxT-tyEnvD, in combination with Equations 354, 357, 358 and 359, we know that:

$$\vdash_{ctx} P;\Gamma_C;\Gamma,\overline{a}_j,\overline{\delta}_k : \overline{Q}_k \rightsquigarrow \Gamma,\overline{a}_j,\overline{\delta_k : \sigma_k}^{k} \tag{363}$$

$$\vdash_{ctx} P;\Gamma_C;\Gamma,x : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \Gamma,x : \forall\overline{a}_j.\overline{\sigma}_k \rightarrow \sigma \tag{364}$$

Applying the induction hypothesis on Equations 360 and 361, in combination with Equations 363 and 364, results in:

$$P;\Gamma_C;\Gamma,\overline{a}_j,\overline{\delta}_k : \overline{Q}_k \vdash^{M}_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \tag{365}$$

$$\Sigma;\Gamma_C;\Gamma,\overline{a}_j,\overline{\delta}_k : \overline{Q}_k \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1 \tag{366}$$

$$P;\Gamma_C;\Gamma,x : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \tau_1 \vdash^{M}_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \tag{367}$$

$$\Sigma;\Gamma_C;\Gamma,x : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \sigma_1 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2 \tag{368}$$

From constraint equivalence (Theorem 14), in combination with Equation 357, we get:

$$\overline{\Gamma_C;\Gamma \vdash_Q Q_k \rightsquigarrow \sigma_k}^{k} \tag{369}$$

By applying iTm-forallI and iTm-constrI to Equation 366, together with Equation 369, we get:

$$\Sigma;\Gamma_C;\Gamma \vdash_{tm} \Lambda\overline{a}_j.\lambda\overline{\delta}_k : \overline{Q}_k.e_1 : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \sigma_1 \rightsquigarrow \Lambda\overline{a}_j.\lambda\,\overline{\delta_k : \sigma_k}^{k}.e_1 \tag{370}$$

From Lemma 43, in combination with Equation 370, we know that:

$$\Gamma_C;\Gamma \vdash_{ty} \forall\overline{a}_j.\overline{Q}_k \Rightarrow \sigma_1 \rightsquigarrow \forall\overline{a}_j.\overline{\sigma}_k \rightarrow \sigma \tag{371}$$

Goals 351 and 352 follow from sTm-inf-let and iTm-let respectively, with

$$e = \textbf{let}\ x : \forall\overline{a}_j.\overline{Q}_k \Rightarrow \sigma_1 = \Lambda\overline{a}_j.\lambda\overline{\delta}_k : \overline{Q}_k.e_1\ \textbf{in}\ e_2$$

sTm-infT-ArrE $P;\Gamma_C;\Gamma \vdash_{tm} e_1\,e_2 \Rightarrow \tau_2 \rightsquigarrow e_1\,e_2$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_1\, e_2 \Rightarrow \tau_2 \rightsquigarrow e \tag{372}$$

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma_2 \rightsquigarrow e_1\, e_2 \tag{373}$$

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{374}$$

From the rule premise we know that:

$$P; \Gamma_C; \Gamma \vdash_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1 \tag{375}$$

$$P; \Gamma_C; \Gamma \vdash_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2 \tag{376}$$

By applying the induction hypothesis to Equations 375 and 376, we get:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_1 \Rightarrow \tau_1 \to \tau_2 \rightsquigarrow e_1 \tag{377}$$

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \rightsquigarrow e_1 \tag{378}$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2 \tag{379}$$

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_1 \rightsquigarrow e_2 \tag{380}$$

Goals 372 and 373 follow from sTm-inf-ArrE and iTm-ArrE respectively, in combination with Equations 377, 378, 379 and 380. Goal 374 follows by applying Lemma 14 to Equation 372.

sTm-infT-Ann $P; \Gamma_C; \Gamma \vdash_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e \tag{381}$$

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e \tag{382}$$

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \tag{383}$$

From the rule premise we know that:

$$P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e \tag{384}$$

Goals 381, 382 and 383 follow by applying Part 2 of this theorem to Equation 384.

Part 2 By case analysis on the typing derivation.

sTm-checkT-Var $P; \Gamma_C; \Gamma \vdash_{tm} x \Leftarrow [\overline{\tau_j}/\overline{a_j}]\tau \rightsquigarrow x\, \overline{\sigma_j}\, \overline{e_i}$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} x \Leftarrow [\overline{\tau_j}/\overline{a_j}]\tau \rightsquigarrow e \tag{385}$$

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma' \rightsquigarrow x\, \overline{\sigma_j}\, \overline{e_i} \tag{386}$$

$$\Gamma_C; \Gamma \vdash^M_{ty} [\overline{\tau_j}/\overline{a_j}]\tau \rightsquigarrow \sigma' \tag{387}$$

From the rule premise we know that:

$$(x : \forall \overline{a_j}.\overline{Q_i} \Rightarrow \tau) \in \Gamma \tag{388}$$

$$\textbf{unambig}(\forall \overline{a_j}.\overline{Q_i} \Rightarrow \tau) \tag{389}$$

$$\overline{P; \Gamma_C; \Gamma \vDash [\overline{\tau_j}/\overline{a_j}]Q_i \rightsquigarrow e_i}^{\,i} \tag{390}$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \tag{391}$$

$$\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{392}$$

We know from Lemma 15, in combination with Equations 388 and 348, that:

$$(x : \forall \overline{a_j}.\overline{Q_i} \Rightarrow \sigma) \in \Gamma \tag{393}$$

By applying type equivalence (Theorem 14) to Equation 391, we get:

$$\overline{\Gamma_C; \Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \tag{394}$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^{\,j} \tag{395}$$

By applying dictionary equivalence (Theorem 15) to Equation 390, we get:

$$\overline{P; \Gamma_C; \Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i}^i \quad (396)$$

$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i \rightsquigarrow e_i}^i \quad (397)$$

$$\overline{\Gamma_C; \Gamma \vdash_Q^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow [\overline{\sigma}_j/\overline{a}_j]Q_i}^i \quad (398)$$

Goal 385 follows from sTm-check-var, in combination with Equations 388, 389, 396, 394 and 348, with $e = x\,\overline{\sigma}_j\,\overline{d}_i$. Goal 386 follows from iTm-var, iTm-forallE and iTm-constrE, in combination with Equations 350, 393, 395 and 397, with $\sigma' = [\overline{\sigma}_j/\overline{a}_j]\sigma$. Goal 387 follows by applying Lemma 14 to Equation 385.

sTm-checkT-meth $P; \Gamma_C; \Gamma \vdash_{tm} m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow e.m\,\overline{\sigma}_j\,\overline{e}_i$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vdash_{tm}^M m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow e \quad (399)$$

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma_0 \rightsquigarrow e.m\,\overline{\sigma}_j\,\overline{e}_i \quad (400)$$

$$\Gamma_C; \Gamma \vdash_{ty}^M [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow \sigma_0 \quad (401)$$

From the rule premise we know that:

$$(m : \overline{Q}'_k \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau') \in \Gamma_C \quad (402)$$

$$\textbf{unambig}(\forall \overline{a}_j, a.\overline{Q}_i \Rightarrow \tau') \quad (403)$$

$$P; \Gamma_C; \Gamma \vDash TC\,\tau \rightsquigarrow e \quad (404)$$

$$\Gamma_C; \Gamma \vdash_{ty} \tau \rightsquigarrow \sigma \quad (405)$$

$$\overline{P; \Gamma_C; \Gamma \vDash [\overline{\tau}_j/\overline{a}_j][\tau/a]Q_i \rightsquigarrow e_i}^i \quad (406)$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \tau_j \rightsquigarrow \sigma_j}^j \quad (407)$$

$$\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \quad (408)$$

By repeated case analysis on Equation 348 (sCtx-clsEnv), together with Equation 402, we know that:

$$(m : TC\,a : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma') \in \Gamma_C \quad (409)$$

By applying type equivalence (Theorem 14) to Equations 405 and 407, we get:

$$\Gamma_C; \Gamma \vdash_{ty}^M \tau \rightsquigarrow \sigma \quad (410)$$

$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \quad (411)$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty}^M \tau_j \rightsquigarrow \sigma_j}^j \quad (412)$$

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^j \quad (413)$$

By applying dictionary equivalence (Theorem 15) to Equations 404 and 406, we get:

$$P; \Gamma_C; \Gamma \vDash^M TC\,\tau \rightsquigarrow d \quad (414)$$

$$\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \rightsquigarrow e \quad (415)$$

$$\overline{P; \Gamma_C; \Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j][\tau/a]Q_i \rightsquigarrow d_i}^i \quad (416)$$

$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j][\sigma/a]Q_i \rightsquigarrow e_i}^i \quad (417)$$

Goal 399 follows from sTm-check-meth, in combination with Equations 402, 403, 414, 410, 416, 412 and 348, with $e = d.m\,\overline{\sigma}_j\,\overline{d}_i$. Consequently, Goal 400 follows from iTm-method, iTm-forallE and iTm-constrE, in combination with Equations 415, 409, 413 and 417, with $\sigma_0 = [\overline{\sigma}_j/\overline{a}_j][\sigma/a]\sigma'$. Goal 401 follows by applying Lemma 14 to Equation 399.

sTm-checkT-ArrI $P; \Gamma_C; \Gamma \vdash_{tm} \lambda x.e \Leftarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x : \sigma.e$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} \lambda x.e \Leftarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow e \tag{418}$$
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma_0 \rightsquigarrow \lambda x : \sigma.e \tag{419}$$
$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_1 \rightarrow \tau_2 \rightsquigarrow \sigma_0 \tag{420}$$

From the rule premise we know that:

$$x \notin \mathbf{dom}(\Gamma) \tag{421}$$
$$P; \Gamma_C; \Gamma, x : \tau_1 \vdash_{tm} e \Leftarrow \tau_2 \rightsquigarrow e \tag{422}$$
$$\Gamma_C; \Gamma \vdash_{ty} \tau_1 \rightsquigarrow \sigma \tag{423}$$

By applying type equivalence (Theorem 14) to Equation 423, we get:

$$\Gamma_C; \Gamma \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma \tag{424}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{425}$$

From sCtxT-tyEnvTm, together with the 2nd hypothesis and Equation 423, we know that:

$$\vdash_{ctx} P; \Gamma_C; \Gamma, x : \tau_1 \rightsquigarrow \Gamma, x : \sigma \tag{426}$$

By applying the induction hypothesis on Equation 422, together with Equation 426, we get:

$$P; \Gamma_C; \Gamma, x : \tau_1 \vdash^M_{tm} e \Leftarrow \tau_2 \rightsquigarrow e' \tag{427}$$
$$\Sigma; \Gamma_C; \Gamma, x : \sigma \vdash_{tm} e' : \sigma_2 \rightsquigarrow e \tag{428}$$
$$\Gamma_C; \Gamma, x : \tau_1 \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{429}$$

Goal 418 follows from sTm-check-ArrI, together with Equations 421, 427 and 424, with $e = \lambda x : \sigma.e'$. Consequently, Goal 419 follows from iTm-arrI, together with Equations 428 and 425, with $\sigma_0 = \sigma \rightarrow \sigma_2$. Goal 420 follows by applying Lemma 14 to Equation 418.

sTm-checkT-Inf $P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e$

The goal to be proven is the following:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e \tag{430}$$
$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e \tag{431}$$
$$\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \tag{432}$$

From the rule premise we know that:

$$P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e \tag{433}$$

Goals 430, 431 and 432 follow directly by applying Part 1 of this theorem to Equation 433.

□

Theorem 17 (Equivalence - Contexts).

- *If* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Gamma'$ *then* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M$ *and* $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M$ *where* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Sigma; \Gamma_C; \Gamma'$ *and* $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma' \vdash^M_{ty} \tau' \rightsquigarrow \sigma'$.
- *If* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Gamma'$ *then* $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M$ *and* $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M$ *where* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Sigma; \Gamma_C; \Gamma'$ *and* $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma' \vdash^M_{ty} \tau' \rightsquigarrow \sigma'$.

- *If* $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M$
and $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Gamma'$
then $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M$
and $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M$
where $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Sigma; \Gamma_C; \Gamma'$
and $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma' \vdash^M_{ty} \tau' \rightsquigarrow \sigma'$.
- *If* $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M$
and $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\vdash_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Gamma'$
then $M : (P; \Gamma_C; \Gamma \Leftarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M$
and $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M$
where $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma' \rightsquigarrow \Sigma; \Gamma_C; \Gamma'$
and $\Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma' \vdash^M_{ty} \tau' \rightsquigarrow \sigma'$.

Proof. By straightforward induction on the typing derivation. □

M COHERENCE THEOREMS

M.1 Compatibility Lemmas

Lemma 50 (Compatibility - Term Abstraction).

$$\frac{\Gamma_C; \Gamma, x : \sigma_1 \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma_2}{\Gamma_C; \Gamma \vdash \Sigma_1 : \lambda x : \sigma_1.e_1 \simeq_{log} \Sigma_2 : \lambda x : \sigma_1.e_2 : \sigma_1 \to \sigma_2}$$

Proof. By the definition of logical equivalence, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C} \tag{434}$$

$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$

$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \tag{435}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(\lambda x : \sigma_1.e_1))), \Sigma_2 : \gamma_2(\phi_2(R(\lambda x : \sigma_1.e_2)))) \in \mathcal{E}[\![\sigma_1 \to \sigma_2]\!]_R^{\Gamma_C}$$

By the definition of the $\mathcal{E}$ relation and the fact that term abstractions are values, the goal reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(\lambda x : \sigma_1.e_1))) : R(\sigma_1 \to \sigma_2)$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(\lambda x : \sigma_1.e_2))) : R(\sigma_1 \to \sigma_2)$$

$$(\Sigma_1 : \gamma_1(\phi_1(R(\lambda x : \sigma_1.e_1))), \Sigma_2 : \gamma_2(\phi_2(R(\lambda x : \sigma_1.e_2)))) \in \mathcal{V}[\![\sigma_1 \to \sigma_2]\!]_R^{\Gamma_C}$$

By applying the substitutions, the goal simplifies to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda x : R(\sigma_1).(\gamma_1(\phi_1(R(e_1)))) : R(\sigma_1 \to \sigma_2) \tag{436}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda x : R(\sigma_1).(\gamma_2(\phi_2(R(e_2)))) : R(\sigma_1 \to \sigma_2) \tag{437}$$

$$(\Sigma_1 : \lambda x : R(\sigma_1).(\gamma_1(\phi_1(R(e_1)))), \Sigma_2 : \lambda x : R(\sigma_1).(\gamma_2(\phi_2(R(e_2))))) \in \mathcal{V}[\![\sigma_1 \to \sigma_2]\!]_R^{\Gamma_C} \tag{438}$$

By unfolding the definition of logical equivalence in the hypothesis of the theorem, we get:

$$(\Sigma_1 : \gamma_1'(\phi_1'(R'(e_1))), \Sigma_2 : \gamma_2'(\phi_2'(R'(e_2)))) \in \mathcal{E}[\![\sigma_2]\!]_{R'}^{\Gamma_C} \tag{439}$$

for any $R' \in \mathcal{F}[\![\Gamma, x : \sigma_1]\!]^{\Gamma_C}$, $\phi' \in \mathcal{G}[\![\Gamma, x : \sigma_1]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma' \in \mathcal{H}[\![\Gamma, x : \sigma_1]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$.

By the definition of the $\mathcal{F}$-relation and from Equation 434, we have that $R \in \mathcal{F}[\![\Gamma, x : \sigma_1]\!]^{\Gamma_C}$ and we choose $R' = R$. By case analysis on ϕ', we know that $\phi' = \phi'', x \mapsto (e_3, e_4)$ for some $\phi'' \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and expressions e_3 and e_4 such that

$$(\Sigma_1 : e_3, \Sigma_2 : e_4) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C} \tag{440}$$

We choose $\phi'' = \phi$. Lastly, by the definition of the $\mathcal{H}$-relation and from Equation 435, we have that $\gamma \in \mathcal{H}[\![\Gamma, x : \sigma_1]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and we choose $\gamma' = \gamma$.

With the above mentioned choices for γ', ϕ' and R', unfolding the definition of the $\mathcal{E}$-relation in Equation 439, gives us:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1((\phi_1, x \mapsto e_3)(R(e_1))) : R(\sigma_2) \tag{441}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2((\phi_2, x \mapsto e_4)(R(e_2))) : R(\sigma_2) \tag{442}$$

$$\begin{aligned} \exists v_3, v_4 : \Sigma_1 &\vdash \gamma_1((\phi_1, x \mapsto e_3)(R(e_1))) \longrightarrow^* v_3 \\ &\wedge \Sigma_2 \vdash \gamma_2((\phi_2, x \mapsto e_4)(R(e_2))) \longrightarrow^* v_4 \\ &\wedge (\Sigma_1 : v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned} \tag{443}$$

By unfolding the definition of the $\mathcal{E}$-relation in Equation 440, we know that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e_3 : \sigma_1 \tag{444}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} e_4 : \sigma_1 \tag{445}$$

Note that neither e_3 nor e_4 contain any free variables, thus $\gamma_1(\phi_1(R(e_3))) = e_3$ and $\gamma_2(\phi_2(R(e_4))) = e_4$. Taking these equations into account, from the definition of substitution, Equations 441 and 442 are rewritten to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} [e_3/x](\gamma_1(\phi_1(R(e_1)))) : R(\sigma_2) \tag{446}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} [e_4/x](\gamma_2(\phi_2(R(e_2)))) : R(\sigma_2) \tag{447}$$

By applying the substitution Lemma 23 on Equations 446 and 447 respectively, in combination with Equations 444 and 445, we get:

$$\Sigma_1; \Gamma_C; \bullet, x : \sigma_1 \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\sigma_2) \tag{448}$$
$$\Sigma_2; \Gamma_C; \bullet, x : \sigma_1 \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(\sigma_2) \tag{449}$$

By Lemma 44, it follows from Equation 448 that $\vdash_{ctx} \Sigma_1; \Gamma_C; \bullet, x : \sigma_1$. By case analysis on this result, rule iCtx-tyEnvTm (the rule with which variable x wad inserted in the environment) tells us that:

$$\Gamma_C; \bullet \vdash_{ty} \sigma_1 \tag{450}$$

Goals 436 and 437 follow from applying the typing rule iTm-arrI on Equations 448 and 449 respectively, together with Equation 450.

It remains to show Goal 438. By unfolding the definition of the $\mathcal{V}$ relation, the goal simplifies to

$$\forall e_5\, e_6,\ \text{if } (\Sigma_1 : e_5, \Sigma_2 : e_6) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C}, \tag{451}$$
$$\text{then } (\Sigma_1 : \lambda x : R(\sigma_1).(\gamma_1(\phi_1(R(e_1))))\, e_5, \Sigma_2 : \lambda x : R(\sigma_1).(\gamma_2(\phi_2(R(e_2))))\, e_6) \in \mathcal{E}[\![\sigma_2]\!]_R^{\Gamma_C} \tag{452}$$

Then, suppose expressions e_5 and e_6, such that Equation 451 holds. By unfolding the definition of the $\mathcal{E}$ relation in Equation 451, we have

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e_5 : \sigma_1 \tag{453}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} e_6 : \sigma_1 \tag{454}$$

We also unfold the definition of the $\mathcal{E}$ relation in Goal 452, to get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\lambda x : R(\sigma_1).(\gamma_1(\phi_1(R(e_1)))))\, e_5 : \sigma_2 \tag{455}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\lambda x : R(\sigma_1).(\gamma_2(\phi_2(R(e_2)))))\, e_6 : \sigma_2 \tag{456}$$
$$\begin{aligned} \exists v_5, v_6 : \Sigma_1 &\vdash (\lambda x : R(\sigma_1).(\gamma_1(\phi_1(R(e_1)))))\, e_5 \longrightarrow^* v_5 \\ \wedge\ \Sigma_2 &\vdash (\lambda x : R(\sigma_1).(\gamma_2(\phi_2(R(e_2)))))\, e_6 \longrightarrow^* v_6 \\ \wedge\ &(\Sigma_1 : v_5, \Sigma_2 : v_6) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned} \tag{457}$$

Goals 455 and 456 follow by applying the iTm-arrE typing rule once on Equations 448 and 453 and once more on Equations 449 and 454. Note that Equations 448 and 449 have been already proven above.

By case analysis, it is easy to see that the first step of the evaluations in Goal 457 is iEval-appAbs, reducing the goal to:

$$\begin{aligned} \exists v_5, v_6 : \Sigma_1 &\vdash \gamma_1(\phi_1(R([e_5/x]e_1))) \longrightarrow^* v_5 \\ \wedge\ \Sigma_2 &\vdash \gamma_2(\phi_2(R([e_6/x]e_2))) \longrightarrow^* v_6 \\ \wedge\ &(\Sigma_1 : v_5, \Sigma_2 : v_6) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned}$$

We choose $e_3 = e_5$ and $e_4 = e_6$ in Equation 440. The goal follows by choosing $v_5 = v_3$ and $v_6 = v_4$ from Equation 443. □

Lemma 51 (Compatibility - Term Application).

$$\frac{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_1' : \sigma_1 \to \sigma_2 \qquad \Gamma_C; \Gamma \vdash \Sigma_1 : e_2 \simeq_{log} \Sigma_2 : e_2' : \sigma_1}{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1\, e_2 \simeq_{log} \Sigma_2 : e_1'\, e_2' : \sigma_2}$$

Proof. By inlining the definition of logical equivalence, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$$
$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$
$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1\, e_2))), \Sigma_2 : \gamma_2(\phi_2(R(e_1'\, e_2')))) \in \mathcal{E}[\![\sigma_2]\!]_R^{\Gamma_C}$$

By applying the definition of the $\mathcal{E}$ relation, the goal reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1\, e_2))) : R(\sigma_2) \tag{458}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_1'\, e_2'))) : R(\sigma_2) \tag{459}$$
$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash \gamma_1(\phi_1(R(e_1\, e_2))) \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash \gamma_2(\phi_2(R(e_1'\, e_2'))) \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned} \tag{460}$$

By applying the substitutions in Goals 458 and 459, they reduce to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\gamma_1(\phi_1(R(e_1))))\, (\gamma_1(\phi_1(R(e_2)))) : R(\sigma_2) \tag{461}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\gamma_2(\phi_2(R(e_1'))))\, (\gamma_2(\phi_2(R(e_2')))) : R(\sigma_2) \tag{462}$$

By inlining the definition of logical equivalence in the premise of the rule, we get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\sigma_1 \to \sigma_2) \tag{463}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_1'))) : R(\sigma_1 \to \sigma_2) \tag{464}$$

$$\exists v_3, v_4 : \Sigma_1 \vdash \gamma_1(\phi_1(R(e_1))) \longrightarrow^* v_3 \tag{465}$$
$$\wedge\, \Sigma_2 \vdash \gamma_2(\phi_2(R(e_1'))) \longrightarrow^* v_4 \tag{466}$$
$$\wedge\, (\Sigma_1 : v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![\sigma_1 \to \sigma_2]\!]_R^{\Gamma_C} \tag{467}$$

and

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_2))) : R(\sigma_1) \tag{468}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2'))) : R(\sigma_1) \tag{469}$$
$$\begin{aligned} \exists v_5, v_6 : \Sigma_1 &\vdash \gamma_1(\phi_1(R(e_2))) \longrightarrow^* v_5 \\ &\wedge \Sigma_2 \vdash \gamma_2(\phi_2(R(e_2'))) \longrightarrow^* v_6 \\ &\wedge (\Sigma_1 : v_5, \Sigma_2 : v_6) \in \mathcal{V}[\![\sigma_1]\!]_R^{\Gamma_C} \end{aligned}$$

By applying both Equation 463 and 468 and both Equation 464 and 469 respectively to iTm-arrE, Goal 461 and 462 are proven.

By applying the substitution, Goal 460 reduces to:

$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash (\gamma_1(\phi_1(R(e_1))))\, (\gamma_1(\phi_1(R(e_2)))) \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash (\gamma_2(\phi_2(R(e_1'))))\, (\gamma_2(\phi_2(R(e_2')))) \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned}$$

Through case analysis, we see that we should first (repeatedly) apply iEval-app and Equation 465 and 466 respectively. The goal reduces to:

$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash v_3\, (\gamma_1(\phi_1(R(e_2)))) \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash v_4\, (\gamma_2(\phi_2(R(e_2')))) \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned} \tag{470}$$

By the definition of the $\mathcal{V}$-relation in Equation 467, we know that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} v_3 : R(\sigma_1 \to \sigma_2)$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} v_4 : R(\sigma_1 \to \sigma_2)$$
$$\forall (\Sigma_1 : e_5, \Sigma_2 : e_6) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C} : (\Sigma_1 : v_3\, e_5, \Sigma_2 : v_4\, e_6) \in \mathcal{E}[\![\sigma_2]\!]_R^{\Gamma_C} \tag{471}$$

We choose $e_5 = \gamma_1(\phi_1(R(e_2)))$ and $e_6 = \gamma_2(\phi_2(R(e_2')))$. Goal 470 now follows from the definition of the $\mathcal{E}$-relation in Equation 471. □

Lemma 52 (Compatibility - Dictionary Abstraction).

$$\frac{\Gamma_C; \Gamma, \delta : Q \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma}{\Gamma_C; \Gamma \vdash \Sigma_1 : \lambda\delta : Q.e_1 \simeq_{log} \Sigma_2 : \lambda\delta : Q.e_2 : Q \Rightarrow \sigma}$$

Proof. By unfolding the definition of logical equivalence in the conclusion of the lemma, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C} \tag{472}$$

$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \tag{473}$$

$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(\lambda\delta : Q.e_1))), \Sigma_2 : \gamma_2(\phi_2(R(\lambda\delta : Q.e_2)))) \in \mathcal{E}[\![Q \Rightarrow \sigma]\!]_R^{\Gamma_C}$$

By applying the definition of the $\mathcal{E}$ relation (taking into account that $\lambda\delta : Q.e$ is a value) and partially applying the substitutions, the goal reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : R(Q).(\gamma_1(\phi_1(R(e_1)))) : R(Q \Rightarrow \sigma) \tag{474}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : R(Q).(\gamma_2(\phi_2(R(e_2)))) : R(Q \Rightarrow \sigma) \tag{475}$$

$$(\Sigma_1 : \lambda\delta : R(Q).(\gamma_1(\phi_1(R(e_1)))), \Sigma_2 : \lambda\delta : R(Q).(\gamma_2(\phi_2(R(e_2))))) \in \mathcal{V}[\![Q \Rightarrow \sigma]\!]_R^{\Gamma_C} \tag{476}$$

By unfolding the definition of logical equivalence in the premise of this lemma, we get:

$$(\Sigma_1 : \gamma'_1(\phi'_1(R'(e_1))), \Sigma_2 : \gamma'_2(\phi'_2(R'(e_2)))) \in \mathcal{E}[\![\sigma]\!]_{R'}^{\Gamma_C} \tag{477}$$

for any $R' \in \mathcal{F}[\![\Gamma, \delta : Q]\!]^{\Gamma_C}$, $\phi' \in \mathcal{G}[\![\Gamma, \delta : Q]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma' \in \mathcal{H}[\![\Gamma, \delta : Q]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$.

By the definition of the $\mathcal{F}$ and the $\mathcal{G}$ relations and from Equations 472 and 473, we have that $R \in \mathcal{F}[\![\Gamma, \delta : Q]\!]^{\Gamma_C}$ and $\phi \in \mathcal{G}[\![\Gamma, \delta : Q]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$. We choose $R' = R$ and $\phi' = \phi$. By case analysis on γ', we know that $\gamma' = \gamma'', \delta \mapsto (dv_1, dv_2)$ for some $\gamma'' \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and some dictionaries dv_1 and dv_2 such that

$$(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \tag{478}$$

We choose $\gamma'' = \gamma$. Then, unfolding the definition of the $\mathcal{E}$ relation in Equation 477 results in:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\gamma_1, \delta \mapsto dv_1)(\phi_1(R(e_1))) : R(\sigma) \tag{479}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\gamma_2, \delta \mapsto dv_2)(\phi_2(R(e_2))) : R(\sigma) \tag{480}$$

$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash (\gamma_1, \delta \mapsto dv_1)(\phi_1(R(e_1))) \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash (\gamma_2, \delta \mapsto dv_2)(\phi_2(R(e_2))) \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C} \end{aligned} \tag{481}$$

From the definition of the $\mathcal{V}$ relation in Equation 478, it follows that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d dv_1 : Q \tag{482}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_d dv_2 : Q \tag{483}$$

Since neither dv_1 nor dv_2 contain any free variables, we know that $\gamma_1(dv_1) = dv_1$ and $\gamma_2(dv_2) = dv_2$. Consequently, Equations 479 and 480 are equivalent to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} [dv_1/\delta](\gamma_1(\phi_1(R(e_1)))) : R(\sigma) \tag{484}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} [dv_2/\delta](\gamma_2(\phi_2(R(e_2)))) : R(\sigma) \tag{485}$$

By applying the substitution Lemma 25 on Equations 484 and 485 respectively, in combination with Equations 482 and 483, we find:

$$\Sigma_1; \Gamma_C; \bullet, \delta : Q \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\sigma) \tag{486}$$

$$\Sigma_2; \Gamma_C; \bullet, \delta : Q \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(\sigma) \tag{487}$$

By Lemma 44, it follows from Equation 486 that $\vdash_{ctx} \Sigma_1; \Gamma_C; \bullet, \delta : Q$. Consequently, we know from iCtx-tyEnvD that:

$$\Gamma_C; \bullet \vdash_Q Q \tag{488}$$

Since Q does not contain any free variables, it is straightforward to see that $R(Q) = Q$.

Consequently, goals 474 and 475 follow by applying Equation 486 and 487 respectively, in combination with Equation 488, to iTm-constrI.

By unfolding the definition of $\mathcal{V}$, Goal 476 reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : R(Q).(\gamma_1(\phi_1(R(e_1)))) : R(Q \Rightarrow \sigma) \tag{489}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : R(Q).(\gamma_2(\phi_2(R(e_2)))) : R(Q \Rightarrow \sigma) \tag{490}$$
$$\forall dv_3\, dv_4,\ \text{if } (\Sigma_1 : dv_3, \Sigma_2 : dv_4) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} : \tag{491}$$
$$\text{then } (\Sigma_1 : (\lambda\delta : R(Q).(\gamma_1(\phi_1(R(e_1)))))\, dv_3, \Sigma_2 : (\lambda\delta : R(Q).(\gamma_2(\phi_2(R(e_2)))))\, dv_4) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \tag{492}$$

Goals 489 and 490 are identical to Goals 474 and 475, which have been proven above. For the final goal, suppose dictionaries dv_3 and dv_4 such that Equation 491 holds. By the definition of the $\mathcal{V}$ relation in Equation 491, we obtain:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d dv_3 : Q \tag{493}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_d dv_4 : Q \tag{494}$$

We unfold the definition of the $\mathcal{E}$ relation in Goal 492, reducing it to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\lambda\delta : R(Q).(\gamma_1(\phi_1(R(e_1)))))\, dv_3 : R(\sigma) \tag{495}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\lambda\delta : R(Q).(\gamma_2(\phi_2(R(e_2)))))\, dv_4 : R(\sigma) \tag{496}$$
$$\begin{aligned} \exists v_3, v_4 : \Sigma_1 &\vdash (\lambda\delta : R(Q).(\gamma_1(\phi_1(R(e_1)))))\, dv_3 \longrightarrow^* v_3 \\ &\wedge \Sigma_2 \vdash (\lambda\delta : R(Q).(\gamma_2(\phi_2(R(e_2)))))\, dv_4 \longrightarrow^* v_4 \\ &\wedge (\Sigma_1 : v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C} \end{aligned} \tag{497}$$

Goals 495 and 496 follow by applying the iTm-constrE typing rule once on Equations 489 and 493 and once on Equations 490 and 494.

Through case analysis, it is straightforward to note that the first step of the evaluation paths in Equation 497 should be by rule iEval-DAppAbs. The goal reduces to:

$$\begin{aligned} \exists v_3, v_4 : \Sigma_1 &\vdash [dv_3/\delta](\gamma_1(\phi_1(R(e_1)))) \longrightarrow^* v_3 \\ &\wedge \Sigma_2 \vdash [dv_4/\delta](\gamma_2(\phi_2(R(e_2)))) \longrightarrow^* v_4 \\ &\wedge (\Sigma_1 : v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C} \end{aligned}$$

The above goal follows directly from Equation 481, by choosing $dv_1 = dv_3$, $dv_2 = dv_4$, $v_3 = v_1$ and $v_4 = v_2$.

□

Lemma 53 (Compatibility - Dictionary Application).

$$\frac{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : Q \Rightarrow \sigma \qquad \Gamma_C; \Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : Q}{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1\, d_1 \simeq_{log} \Sigma_2 : e_2\, d_2 : \sigma}$$

Proof. By inlining the definition of logical equivalence, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$$
$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1, \Sigma_2, \Gamma_C}$$
$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1, \Sigma_2, \Gamma_C}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1\, d_1))), \Sigma_2 : \gamma_2(\phi_2(R(e_2\, d_2)))) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C}$$

By unfolding the definition of the $\mathcal{E}$ relation in the goal above, and by simplifying the substitutions, the goal reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\gamma_1(\phi_1(R(e_1))))\, (\gamma_1(\phi_1(R(d_1)))) : R(\sigma) \tag{498}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\gamma_2(\phi_2(R(e_2))))\, (\gamma_2(\phi_2(R(d_2)))) : R(\sigma) \tag{499}$$
$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash (\gamma_1(\phi_1(R(e_1))))\, (\gamma_1(\phi_1(R(d_1)))) \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash (\gamma_2(\phi_2(R(e_2))))\, (\gamma_2(\phi_2(R(d_2)))) \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C} \end{aligned} \tag{500}$$

By inlining the definitions of logical equivalence and the $\mathcal{E}$ relation in the first premise of this lemma, we get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(Q \Rightarrow \sigma) \tag{501}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(Q \Rightarrow \sigma) \tag{502}$$

$$\begin{aligned} \exists v_3, v_4 : \Sigma_1 &\vdash \gamma_1(\phi_1(R(e_1))) \longrightarrow^* v_3 \\ &\wedge \Sigma_2 \vdash \gamma_2(\phi_2(R(e_2))) \longrightarrow^* v_4 \\ &\wedge (\Sigma_1 : v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![Q \Rightarrow \sigma]\!]_R^{\Gamma_C} \end{aligned} \tag{503}$$

Similarly, by unfolding the definition of logical equivalence in the second premise of the rule, we get:

$$(\Sigma_1 : \gamma_1(\phi_1(R(d_1))), \Sigma_2 : \gamma_2(\phi_2(R(d_2)))) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \tag{504}$$

From the definition of the $\mathcal{V}$ relation in Equation 504 we have:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d \gamma_1(\phi_1(R(d_1))) : R(Q) \tag{505}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_d \gamma_2(\phi_2(R(d_2))) : R(Q) \tag{506}$$

Note that, by the definition of substitution, we have $R(Q \Rightarrow \sigma) = R(Q) \Rightarrow R(\sigma)$. This allows the application of the typing rule iTm-constrE once on Equations 501 and 505 and once more on Equations 502 and 506, therefore proving Goals 498 and 499.

Through application of the iEval-DApp evaluation rule on each step of the two evaluation paths in Equation 503, Goal 500 reduces to:

$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash v_3\,(\gamma_1(\phi_1(R(d_1)))) \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash v_4\,(\gamma_2(\phi_2(R(d_2)))) \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C} \end{aligned} \tag{507}$$

Unfolding the definition of the $\mathcal{V}$ relation in Equation 503 results in:

$$\begin{gathered} \Sigma_1; \Gamma_C; \bullet \vdash_{tm} v_3 : R(Q \Rightarrow \sigma) \\ \Sigma_2; \Gamma_C; \bullet \vdash_{tm} v_4 : R(Q \Rightarrow \sigma) \\ \forall (\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} : (\Sigma_1 : v_3\, dv_1, \Sigma_2 : v_4\, dv_2) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \end{gathered} \tag{508}$$

We take $dv_1 = \gamma_1(\phi_1(R(d_1)))$ and $dv_2 = \gamma_2(\phi_2(R(d_2)))$. Goal 507 follows from the definition of the $\mathcal{E}$ relation in Equation 508. □

Lemma 54 (Compatibility - Type Abstraction).

$$\frac{\Gamma_C; \Gamma, a \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma}{\Gamma_C; \Gamma \vdash \Sigma_1 : \Lambda a.e_1 \simeq_{log} \Sigma_2 : \Lambda a.e_2 : \forall a.\sigma}$$

Proof. By unfolding the definition of logical equivalence, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C} \tag{509}$$

$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1, \Sigma_2, \Gamma_C} \tag{510}$$

$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1, \Sigma_2, \Gamma_C} \tag{511}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(\Lambda a.e_1))), \Sigma_2 : \gamma_2(\phi_2(R(\Lambda a.e_2)))) \in \mathcal{E}[\![\forall a.\sigma]\!]_R^{\Gamma_C}$$

Because $a \notin \Gamma$, from Equation 509 it follows that a is not in the domain of R. Furthermore, from Equations 510 and 511 it follows that for every mapping $x \mapsto (e_1', e_2') \in \phi$ and for every mapping $\delta \mapsto (dv_1', dv_2') \in \gamma$, we have $a \notin \mathbf{fv}(e_i')$ and $a \notin \mathbf{fv}(dv_i')$, where $i \in \{1, 2\}$. Therefore, we obtain $\gamma_i(\phi_i(R(\Lambda a.e_i))) = \Lambda a.\gamma_i(\phi_i(R(e_i)))$, for $i \in \{1, 2\}$. With these equations, the goal above reduces to

$$(\Sigma_1 : \Lambda a.\gamma_1(\phi_1(R(e_1))), \Sigma_2 : \Lambda a.\gamma_2(\phi_2(R(e_2)))) \in \mathcal{E}[\![\forall a.\sigma]\!]_R^{\Gamma_C}$$

By applying the definition of the $\mathcal{E}$ relation, taking into account that expressions of the form $\Lambda a.e$ are values, the goal reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \Lambda a.\gamma_1(\phi_1(R(e_1))) : R(\forall a.\sigma) \tag{512}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \Lambda a.\gamma_2(\phi_2(R(e_2))) : R(\forall a.\sigma) \tag{513}$$

$$(\Sigma_1 : \Lambda a.\gamma_1(\phi_1(R(e_1))), \Sigma_2 : \Lambda a.\gamma_2(\phi_2(R(e_2)))) \in \mathcal{V}[\![\forall a.\sigma]\!]_R^{\Gamma_C} \tag{514}$$

Suppose any σ' such that

$$\Gamma_C; \bullet \vdash_{ty} \sigma' \tag{515}$$

and any $r \in Rel[\sigma']$. Then, inlining the definition of the $\mathcal{V}$ relation in Goal 514, reduces it to:

$$(\Sigma_1 : (\Lambda a.\gamma_1(\phi_1(R(e_1))))\,\sigma', \Sigma_2 : (\Lambda a.\gamma_2(\phi_2(R(e_2))))\,\sigma') \in \mathcal{E}[\![\sigma]\!]^{\Gamma_C}_{R, a\mapsto(\sigma', r)} \tag{516}$$

Unfolding the definition of logical equivalence in the premise of this lemma, gives us:

$$(\Sigma_1 : \gamma_1'(\phi_1'(R'(e_1))), \Sigma_2 : \gamma_2'(\phi_2'(R'(e_2)))) \in \mathcal{E}[\![\sigma]\!]^{\Gamma_C}_{R'} \tag{517}$$

for any $R' \in \mathcal{F}[\![\Gamma, a]\!]^{\Gamma_C}$, $\phi' \in \mathcal{G}[\![\Gamma, a]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{R'}$ and $\gamma' \in \mathcal{H}[\![\Gamma, a]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{R'}$.

By the definition of the $\mathcal{F}$ relation, we know that $R' = R'', a \mapsto (\sigma'', r')$ for some $R'' \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$ and σ'' such that $\Gamma_C; \bullet \vdash_{ty} \sigma''$ and $r' \in Rel[\sigma'']$. We choose $R'' = R$, $\sigma'' = \sigma'$ and $r' = r$. By the definition of the $\mathcal{G}$ and $\mathcal{H}$ relations and from Equations 510 and 511, we have that $\phi \in \mathcal{G}[\![\Gamma, a]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{R}$ and $\gamma \in \mathcal{H}[\![\Gamma, a]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{R}$. Then, we choose $\phi' = \phi$ and $\gamma' = \gamma$.

Unfolding the definition of the $\mathcal{E}$ relation in Equation 517, results in:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R, a \mapsto (\sigma', r)(e_1))) : (R, a \mapsto (\sigma', r))(\sigma) \tag{518}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R, a \mapsto (\sigma', r)(e_2))) : (R, a \mapsto (\sigma', r))(\sigma) \tag{519}$$

$$\begin{aligned}\exists v_3, v_4 : \Sigma_1 &\vdash \gamma_1(\phi_1(R, a \mapsto (\sigma', r)(e_1))) \longrightarrow^* v_3\\ &\wedge \Sigma_2 \vdash \gamma_2(\phi_2(R, a \mapsto (\sigma', r)(e_2))) \longrightarrow^* v_4\\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]^{\Gamma_C}_{(R, a\mapsto(\sigma', r))}\end{aligned} \tag{520}$$

By the definition of substitution, and because σ' has no free variables, it follows that $(R, a \mapsto (\sigma', r))(\sigma) = [\sigma'/a](R(\sigma))$. Furthermode, because $a \notin \Gamma$, from Equations 510 and 511 it follows that $\gamma_i(\phi_i((R, a \mapsto (\sigma', r))(e))) = [\sigma'/a](\gamma_i(\phi_i(R(e))))$, for any expression e and $i \in \{1, 2\}$. Taking these equalities into account, by applying Equations 518 and 519 to reverse substitution Lemma 27 gives us:

$$\Sigma_1; \Gamma_C; \bullet, a \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\sigma) \tag{521}$$

$$\Sigma_2; \Gamma_C; \bullet, a \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(\sigma) \tag{522}$$

Because a is not in the domain of R, we have $R(\forall a.\sigma) = \forall a.R(\sigma)$. Hence, Goals 512 and 513 follow by passing Equations 521 and 522 to iTm-forallI, respectively.

Unfolding the definition of the $\mathcal{E}$ relation in Goal 516 and since $(R, a \mapsto (\sigma', r))(\sigma) = [\sigma'/a](R(\sigma))$, the goal reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\Lambda a.\gamma_1(\phi_1(R(e_1))))\,\sigma' : [\sigma'/a](R(\sigma)) \tag{523}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\Lambda a.\gamma_2(\phi_2(R(e_2))))\,\sigma' : [\sigma'/a](R(\sigma)) \tag{524}$$

$$\begin{aligned}\exists v_5, v_6 : \Sigma_1 &\vdash (\Lambda a.\gamma_1(\phi_1(R(e_1))))\,\sigma' \longrightarrow^* v_5\\ &\wedge \Sigma_2 \vdash (\Lambda a.\gamma_2(\phi_2(R(e_2))))\,\sigma' \longrightarrow^* v_6\\ &\wedge (\Sigma_1 : v_5, \Sigma_2 : v_6) \in \mathcal{V}[\![\sigma]\!]^{\Gamma_C}_{R, a\mapsto(\sigma', r)}\end{aligned} \tag{525}$$

Goals 523 and 524 follow by applying Goals 512 and 513 (which have previously been proven) to iTm-forallE, respectively, together with Equation 515. The first step of both evaluation paths in Equation 525 can only be taken by appropriate instantiations of rule iEval-tyAppAbs. With this, Goal 525 can be further reduced to

$$\begin{aligned}\exists v_5, v_6 : \Sigma_1 &\vdash [\sigma'/a](\gamma_1(\phi_1(R(e_1)))) \longrightarrow^* v_5\\ &\wedge \Sigma_2 \vdash [\sigma'/a](\gamma_2(\phi_2(R(e_2)))) \longrightarrow^* v_6\\ &\wedge (\Sigma_1 : v_5, \Sigma_2 : v_6) \in \mathcal{V}[\![\sigma]\!]^{\Gamma_C}_{R, a\mapsto(\sigma', r)}\end{aligned}$$

which follows from Equation 520 by choosing $v_5 = v_3$ and $v_6 = v_4$.

□

Lemma 55 (Compatibility - Type Application).

$$\frac{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \forall a.\sigma' \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma}{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1\,\sigma \simeq_{log} \Sigma_2 : e_2\,\sigma : [\sigma/a]\sigma'}$$

Proof. By inlining the definition of logical equivalence, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$$
$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$
$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$

Note that, by the definition of the $\mathcal{F}$ relation, a is not in the domain of R, since $a \notin \Gamma$. The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1\,\sigma))), \Sigma_2 : \gamma_2(\phi_2(R(e_2\,\sigma)))) \in \mathcal{E}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C} \tag{526}$$

From the definition of substitution we have that

$$\gamma_i(\phi_i(R(e_i\,\sigma))) = \gamma_i(\phi_i(R(e_i)))\,R(\sigma),\ \text{for } i \in \{1,2\}$$
$$\text{and } R([\sigma/a]\sigma') = [R(\sigma)/a]R(\sigma')$$

Taking into account these equalities and by unfolding the definition of the $\mathcal{E}$ relation, Goal 526 reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1)))\,R(\sigma) : [R(\sigma)/a]R(\sigma') \tag{527}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2)))\,R(\sigma) : [R(\sigma)/a]R(\sigma') \tag{528}$$
$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash (\gamma_1(\phi_1(R(e_1))))\,R(\sigma) \longrightarrow^* v_1 \\ &\land \Sigma_2 \vdash (\gamma_2(\phi_2(R(e_2))))\,R(\sigma) \longrightarrow^* v_2 \\ &\land (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C} \end{aligned} \tag{529}$$

By inlining the definition of logical equivalence in the first premise of this lemma, we get

$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1))), \Sigma_2 : \gamma_2(\phi_2(R(e_2)))) \in \mathcal{E}[\![\forall a.\sigma']\!]_R^{\Gamma_C}$$

Unfolding the definition of the $\mathcal{E}$ relation results in:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\forall a.\sigma') \tag{530}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(\forall a.\sigma') \tag{531}$$
$$\begin{aligned} \exists v_3, v_4 : \Sigma_1 &\vdash \gamma_1(\phi_1(R(e_1))) \longrightarrow^* v_3 \\ &\land \Sigma_2 \vdash \gamma_2(\phi_2(R(e_2))) \longrightarrow^* v_4 \\ &\land (\Sigma_1 : v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![\forall a.\sigma']\!]_R^{\Gamma_C} \end{aligned} \tag{532}$$

Starting from the second premise of this lemma, by sequentially applying Lemma 20 with the substitutions of R on σ, it follows that $\Gamma_C; \Gamma' \vdash_{ty} R(\sigma)$, where Γ' only contains term variables. Then, starting from this result, by sequentually applying Lemma 38, we obtain

$$\Gamma_C; \bullet \vdash_{ty} R(\sigma) \tag{533}$$

Since a is not in the domain of R, we have $R(\forall a.\sigma) = \forall a.R(\sigma)$. Consequently, Goals 527 and 528 follow by instantiating rule iTm-forallE with Equations 530 and 531, respectively, together with Equation 533.

The definition of the $\mathcal{V}$ relation in Equation 532 tells us that:

$$\begin{aligned} &\forall \sigma'', r \in Rel[\sigma''] : \Gamma_C; \bullet \vdash_{ty} \sigma'' \\ &\qquad \Rightarrow (\Sigma_1 : v_3\,\sigma'', \Sigma_2 : v_4\,\sigma'') \in \mathcal{E}[\![\sigma']\!]_{R,a\mapsto(\sigma'',r)}^{\Gamma_C} \end{aligned} \tag{534}$$

By repeatedly applying iEval-tyApp on each step of both evaluation paths in Equation 532, Goal 529 reduces to:

$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash v_3\,R(\sigma) \longrightarrow^* v_1 \\ &\land \Sigma_2 \vdash v_4\,R(\sigma) \longrightarrow^* v_2 \\ &\land (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C} \end{aligned}$$

which follows directly from 534 by choosing $\sigma'' = \sigma$ and unfolding the definition of the $\mathcal{E}$ relation.

□

Lemma 56 (Compatibility - Let Binding).

$$\frac{\Gamma_C; \Gamma \vdash \Sigma_1 : e_1' \simeq_{log} \Sigma_2 : e_2' : \sigma_1 \qquad \Gamma_C; \Gamma, x : \sigma_1 \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma_2}{\Gamma_C; \Gamma \vdash \Sigma_1 : \textbf{\textit{let}}\ x : \sigma_1 = e_1'\ \textbf{\textit{in}}\ e_1 \simeq_{log} \Sigma_2 : \textbf{\textit{let}}\ x : \sigma_1 = e_2'\ \textbf{\textit{in}}\ e_2 : \sigma_2}$$

Proof. By inlining the definition of logical equivalence, suppose we have:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$$
$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \tag{535}$$
$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(\mathbf{let}\ x : \sigma_1 = e_1'\ \mathbf{in}\ e_1))), \Sigma_2 : \gamma_2(\phi_2(R(\mathbf{let}\ x : \sigma_1 = e_2'\ \mathbf{in}\ e_2)))) \in \mathcal{E}[\![\sigma_2]\!]_R^{\Gamma_C} \tag{536}$$

From the definition of substitution, it follows that

$$\gamma_i(\phi_i(R(\mathbf{let}\ x : \sigma_1 = e_i'\ \mathbf{in}\ e_i))) = \mathbf{let}\ x : R(\sigma_1) = (\gamma_i(\phi_i(R(e_i'))))\ \mathbf{in}\ (\gamma_i(\phi_i(R(e_i)))),\ \text{for } i \in \{1,2\}$$

Taking into account this equality, by applying the definition of the $\mathcal{E}$ relation, Goal 536 reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \mathbf{let}\ x : R(\sigma_1) = (\gamma_1(\phi_1(R(e_1'))))\ \mathbf{in}\ (\gamma_1(\phi_1(R(e_1)))) : R(\sigma_2) \tag{537}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \mathbf{let}\ x : R(\sigma_1) = (\gamma_2(\phi_2(R(e_2'))))\ \mathbf{in}\ (\gamma_2(\phi_2(R(e_2)))) : R(\sigma_2) \tag{538}$$
$$\begin{aligned}\exists v_1, v_2 : \Sigma_1 &\vdash \mathbf{let}\ x : R(\sigma_1) = (\gamma_1(\phi_1(R(e_1'))))\ \mathbf{in}\ (\gamma_1(\phi_1(R(e_1)))) \longrightarrow^* v_1\\ \wedge\, \Sigma_2 &\vdash \mathbf{let}\ x : R(\sigma_1) = (\gamma_2(\phi_2(R(e_2'))))\ \mathbf{in}\ (\gamma_2(\phi_2(R(e_2)))) \longrightarrow^* v_2\\ \wedge\, (\Sigma_1 &: v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C}\end{aligned} \tag{539}$$

By inlining the definition of logical equivalence in the two hypotheses of this lemma, we get:

$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1'))), \Sigma_2 : \gamma_2(\phi_2(R(e_2')))) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C} \tag{540}$$
$$(\Sigma_1 : \gamma_1'(\phi_1'(R'(e_1))), \Sigma_2 : \gamma_2'(\phi_2'(R'(e_2)))) \in \mathcal{E}[\![\sigma_2]\!]_{R'}^{\Gamma_C} \tag{541}$$

for any $R' \in \mathcal{F}[\![\Gamma, x : \sigma_1]\!]^{\Gamma_C}$, $\phi' \in \mathcal{G}[\![\Gamma, x : \sigma_1]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma' \in \mathcal{H}[\![\Gamma, x : \sigma_1]\!]_{R'}^{\Sigma_1,\Sigma_2,\Gamma_C}$. Note that in Equation 540 we have already chosen the substitutions R, ϕ and γ from Equation 535. By the definition of $\mathcal{F}$, we obtain $R \in \mathcal{F}[\![\Gamma, x : \sigma_1]\!]^{\Gamma_C}$. Therefore, a valid choice for R' is R. From the definitions of $\mathcal{G}$ and $\mathcal{H}$, it must hold that $\phi' = \phi'', x \mapsto (e_3, e_4)$ and $\gamma' = \gamma''$ where $\phi'' \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$, $\gamma'' \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $(\Sigma_1 : e_3, \Sigma_2 : e_4) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C}$. We choose $\phi'' = \phi$ and $\gamma'' = \gamma$. It remains to instantiate e_3 and e_4 with concrete choices. For reasons of presentation, we defer this choice to the end of the proof.

From the definition of the $\mathcal{E}$ relation in Equations 541, we get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1((\phi_1, x \mapsto e_3)(R(e_1))) : R(\sigma_2) \tag{542}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2((\phi_2, x \mapsto e_4)(R(e_2))) : R(\sigma_2) \tag{543}$$
$$\begin{aligned}\exists v_5, v_6 : \Sigma_1 &\vdash \gamma_1((\phi_1, x \mapsto e_3)(R(e_1))) \longrightarrow^* v_5\\ \wedge\, \Sigma_2 &\vdash \gamma_2((\phi_2, x \mapsto e_4)(R(e_2))) \longrightarrow^* v_6\\ \wedge\, (\Sigma_1 &: v_5, \Sigma_2 : v_6) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C}\end{aligned} \tag{544}$$

Similarly, from Equation 540 and from $(\Sigma_1 : e_3, \Sigma_2 : e_4) \in \mathcal{E}[\![\sigma_1]\!]_R^{\Gamma_C}$, we get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1'))) : R(\sigma_1) \tag{545}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2'))) : R(\sigma_1) \tag{546}$$

and

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e_3 : R(\sigma_1) \tag{547}$$
$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} e_4 : R(\sigma_1) \tag{548}$$

Note that from Equations 547 and 548 it is evident that expressions e_3 and e_4 contain no free variables. Therefore,

$$\begin{aligned}&\gamma_1((\phi_1, x \mapsto e_3)(R(e_1)))\\ &= \gamma_1([e_3/x](\phi_1(R(e_1)))) && \text{(by definition)}\\ &= [\gamma_1(e_3)/x](\gamma_1(\phi_1(R(e_1)))) && \text{(distributivity property)}\\ &= [e_3/x](\gamma_1(\phi_1(R(e_1)))) && \text{(no free variables in } e_3\text{)}\end{aligned}$$

and similarly, $\gamma_2((\phi_2, x \mapsto e_4)(R(e_2))) = [e_4/x]\gamma_2(\phi_2(R(e_2)))$. Taking these equalities into account, we can apply the reverse substitution Lemma 23 on Equations 542 and 543, in combination with Equations 547 and 548, respectively, to obtain:

$$\Sigma_1; \Gamma_C; \bullet, x : R(\sigma_1) \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\sigma_2) \tag{549}$$
$$\Sigma_2; \Gamma_C; \bullet, x : R(\sigma_1) \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(\sigma_2) \tag{550}$$

Combining Lemma 44 with Equation 549, yields $\vdash_{ctx}\ \Sigma_1; \Gamma_C; \bullet, x : R(\sigma_1)$ and, by case analysis on this environment well-formedness judgment, it follows that $\Gamma_C; \bullet \vdash_{ty} R(\sigma_1)$. Using this, Goals 537 and 538 follow by applying both Equations 545 and 549 and both Equations 546 and 550 to iTm-let, respectively.

By case analysis, the first step of both evaluation paths in Equation 539 must be appropriate instantiations of rule iEval-let, according to which,

$$\Sigma_i \vdash \textbf{let}\ x : R(\sigma_1) = (\gamma_i(\phi_i(R(e_i')))) \textbf{ in } (\gamma_i(\phi_i(R(e_i)))) \longrightarrow [\gamma_i(\phi_i(R(e_i')))/x](\gamma_i(\phi_i(R(e_i))))$$

for each $i \in \{1, 2\}$. This simplifies Goal 539 to:

$$\begin{aligned} \exists v_1, v_2 : \Sigma_1 &\vdash [\gamma_1(\phi_1(R(e_1')))/x](\gamma_1(\phi_1(R(e_1)))) \longrightarrow^* v_1 \\ \wedge\ \Sigma_2 &\vdash [\gamma_2(\phi_2(R(e_2')))/x](\gamma_2(\phi_2(R(e_2)))) \longrightarrow^* v_2 \\ \wedge\ (\Sigma_1 &: v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma_2]\!]_R^{\Gamma_C} \end{aligned}$$

The goal follows from Equation 544. Because of Equation 540, we can choose $e_3 = \gamma_1(\phi_1(R(e_1')))$ and $e_4 = \gamma_2(\phi_2(R(e_2')))$. □

Lemma 57 (Compatibility - Method).

$$\frac{\Gamma_C; \Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : TC\ \sigma \qquad (m : TC\ a : \sigma') \in \Gamma_C \qquad \Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2}{\Gamma_C; \Gamma \vdash \Sigma_1 : d_1.m \simeq_{log} \Sigma_2 : d_2.m : [\sigma/a]\sigma'}$$

Proof. By inlining the definition of logical equivalence, suppose we have:

$$\begin{aligned} R &\in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C} \\ \phi &\in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \\ \gamma &\in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \end{aligned} \tag{551}$$

The goal to be proven is the following:

$$(\Sigma_1 : \gamma_1(\phi_1(R(d_1.m))), \Sigma_2 : \gamma_2(\phi_2(R(d_2.m)))) \in \mathcal{E}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C}$$

By applying the definition of the $\mathcal{E}$ relation, it reduces to:

$$\begin{aligned} &\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(d_1.m))) : R([\sigma/a]\sigma') \\ &\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(d_2.m))) : R([\sigma/a]\sigma') \\ &\exists v_1, v_2 : \Sigma_1 \vdash \gamma_1(\phi_1(R(d_1.m))) \longrightarrow^* v_1 \\ &\quad \wedge\ \Sigma_2 \vdash \gamma_2(\phi_2(R(d_2.m))) \longrightarrow^* v_2 \\ &\quad \wedge\ (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C} \end{aligned}$$

By applying the substitutions, and because $R([\sigma/a]\sigma') = [R(\sigma)/a]R(\sigma')$, the goal further reduces to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} (\gamma_1(\phi_1(R(d_1)))).m : [R(\sigma)/a]R(\sigma') \tag{552}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} (\gamma_2(\phi_2(R(d_2)))).m : [R(\sigma)/a]R(\sigma') \tag{553}$$

$$\begin{aligned} &\exists v_1, v_2 : \Sigma_1 \vdash (\gamma_1(\phi_1(R(d_1)))).m \longrightarrow^* v_1 \\ &\quad \wedge\ \Sigma_2 \vdash (\gamma_2(\phi_2(R(d_2)))).m \longrightarrow^* v_2 \\ &\quad \wedge\ (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C} \end{aligned} \tag{554}$$

By inlining the definition of logical equivalence in the first hypothesis of this lemma and choosing R, ϕ and γ from Equation 551, we get:

$$(\Sigma_1 : \gamma_1(\phi_1(R(d_1))), \Sigma_2 : \gamma_2(\phi_2(R(d_2)))) \in \mathcal{V}[\![TC\ \sigma]\!]_R^{\Gamma_C}$$

Then, from the definition of $\mathcal{V}$, we get:

$$\gamma_1(\phi_1(R(d_1))) = D\,\overline{\sigma}_j\,\overline{dv_1}_i$$
$$\gamma_2(\phi_2(R(d_2))) = D\,\overline{\sigma}_j\,\overline{dv_2}_i$$
$$(D : \forall\overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto e_1' \in \Sigma_1 \tag{555}$$
$$\Sigma_1; \Gamma_C; \bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv_1}_i : TC\,R(\sigma) \tag{556}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_d D\,\overline{\sigma}_j\,\overline{dv_2}_i : TC\,R(\sigma) \tag{557}$$
$$\overline{(\Sigma_1 : dv_{1\,i}, \Sigma_2 : dv_{2\,i}) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j]Q_i]\!]_R^{\Gamma_C}}^{\,i} \tag{558}$$

for some $\overline{\sigma}_j$, $\overline{dv_1}_i$, $\overline{dv_2}_i$, $\overline{Q}_i$, e' and σ_q such that $\sigma = [\overline{\sigma}_j/\overline{a}_j]\sigma_q$.

Lemma 45, applied on Equations 556 and 557, yields:

$$\vdash_{ctx} \Sigma_1; \Gamma_C; \bullet \tag{559}$$
$$\vdash_{ctx} \Sigma_2; \Gamma_C; \bullet \tag{560}$$

Also, from the second premise of this lemma's rule, there are Γ_{C1} and Γ_{C2} such that $\Gamma_C = \Gamma_{C1}, m : TC\,a : \sigma', \Gamma_{C2}$. Then, from Lemma 39, we get $\Gamma_{C1}; \bullet, a \vdash_{ty} \sigma'$, which means that the only free variable appearing in σ' is the fresh variable a. Then,

$$R(\sigma') = \sigma' \tag{561}$$

Also, since the dictionary $D\,\overline{\sigma}_j\,\overline{dv_1}_i$ is closed (it is the result of applying the closing substitutions R, ϕ_1 and γ_1 on dictionary d_1), types $\overline{\sigma}_j$ can not contain any free variables. Hence, $R(\overline{\sigma}_j) = \overline{\sigma}_j$. In addition, from the last conclusion of Lemma 40, supplied with $\vdash_{ctx} \Sigma_1; \Gamma_C; \bullet$ and Equation 555, we have that $\Gamma_C; \bullet, \overline{a}_j \vdash_{ty} \sigma_q$. Because the type variables $\overline{a}_j$ are not in Γ, they are not in the domain of R, thus $R(\sigma_q) = \sigma_q$. Then,

$$\begin{aligned} R(\sigma) &= R([\overline{\sigma}_j/\overline{a}_j]\sigma_q) \\ &= [R(\overline{\sigma}_j)/\overline{a}_j]R(\sigma_q) \\ &= [\overline{\sigma}_j/\overline{a}_j]\sigma_q = \sigma \end{aligned} \tag{562}$$

Equations 556 and 557 can only stand as conclusions of dictionary typing rule D-con. After rewriting them with Equation 562 we can invert them and get their premises. Also because environment Σ_1 can only contain a unique entry for each dictionary type (in this case, the one shown in Equation 555), we finally conclude

$$\overline{\Gamma_C; \bullet \vdash_{ty} \sigma_j}^{\,j} \tag{563}$$
$$\overline{\Sigma_1; \Gamma_C; \bullet \vdash_d dv_{1\,i} : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{564}$$
$$\overline{\Sigma_2; \Gamma_C; \bullet \vdash_d dv_{2\,i} : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{565}$$
$$\Sigma_1'; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e_1 : [\sigma_q/a]\sigma' \tag{566}$$
$$\Sigma_2'; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e_2 : [\sigma_q/a]\sigma' \tag{567}$$
$$\textit{where}\ \Sigma_1 = \Sigma_1', (D : \forall\overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto e_1', \Sigma_1'' \tag{568}$$
$$\textit{and}\ e_1' = \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e_1 \tag{569}$$
$$\textit{and}\ \Sigma_2 = \Sigma_2', (D : \forall\overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto e_2', \Sigma_2'' \tag{570}$$
$$\textit{and}\ e_2' = \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e_2 \tag{571}$$

With Equations 561 and 562, Goals 552 and 553 follow by using the second premise of the theorem's rule and Equations 556 and 557, respectively, in iTm-method.

Using Equations 568 and 570 in iEval-method, results in:

$$\Sigma_1 \vdash (D\,\overline{\sigma}_j\,\overline{dv_1}_i).m \longrightarrow e_1'\,\overline{\sigma}_j\,\overline{dv_1}_i$$
$$\Sigma_2 \vdash (D\,\overline{\sigma}_j\,\overline{dv_2}_i).m \longrightarrow e_2'\,\overline{\sigma}_j\,\overline{dv_2}_i$$

This reduces Goal 554 to:

$$\begin{aligned} &\exists v_1, v_2 : \Sigma_1 \vdash e_1'\,\overline{\sigma}_j\,\overline{dv_1}_i \longrightarrow^* v_1 \\ &\wedge \Sigma_2 \vdash e_2'\,\overline{\sigma}_j\,\overline{dv_2}_i \longrightarrow^* v_2 \\ &\wedge (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![[\sigma/a]\sigma']\!]_R^{\Gamma_C} \end{aligned} \tag{572}$$

From the definition of logical equivalence in the theorem's third hypothesis, together with Equations 568 and 570, we get that:

$$\Gamma_C; \bullet \vdash \Sigma_1' : e_1' \simeq_{log} \Sigma_2' : e_2' : \forall\overline{a}_j.[\sigma_q/a]\sigma' \tag{573}$$

Repeatedly applying compatibility Lemma 55 to Equations 573 and 563, results in:

$$\Gamma_C; \bullet \vdash \Sigma_1' : e_1' \, \overline{\sigma}_j \simeq_{log} \Sigma_2' : e_2' \, \overline{\sigma}_j : [\overline{\sigma}_j/\overline{a}_j]\overline{Q}_i \Rightarrow [\overline{\sigma}_j/\overline{a}_j][\sigma_q/a]\sigma' \tag{574}$$

By applying weakening Lemma 33 on this result, in combination with Equations 559 and 560, we get:

$$\Gamma_C; \bullet \vdash \Sigma_1 : e_1' \, \overline{\sigma}_j \simeq_{log} \Sigma_2 : e_2' \, \overline{\sigma}_j : [\overline{\sigma}_j/\overline{a}_j]\overline{Q}_i \Rightarrow [\overline{\sigma}_j/\overline{a}_j][\sigma_q/a]\sigma' \tag{575}$$

From the definition of logical equivalence and Equation 558, we can derive that:

$$\overline{\Gamma_C; \bullet \vdash \Sigma_1 : dv_{1\,i} \simeq_{log} \Sigma_2 : dv_{2\,i} : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i}$$

Repeatedly applying compatibility Lemma 53 on Equations 575, together with the above equation, results in:

$$\Gamma_C; \bullet \vdash \Sigma_1 : e_1' \, \overline{\sigma}_j \, \overline{dv_{1\,i}} \simeq_{log} \Sigma_2 : e_2' \, \overline{\sigma}_j \, \overline{dv_{2\,i}} : [\overline{\sigma}_j/\overline{a}_j][\sigma_q/a]\sigma'$$

Since expressions $e_1' \, \overline{\sigma}_j \, \overline{dv_{1\,i}}$ and $e_2' \, \overline{\sigma}_j \, \overline{dv_{2\,i}}$ are closed, by the definition of the logical relation, applying any substitutions $R \in \mathcal{F}[\![\bullet]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\bullet]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma \in \mathcal{H}[\![\bullet]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ on both expressions should result in two terms that are related by the $\mathcal{E}$ relation. By case analysis on R, ϕ and γ, only the empty substitutions are valid choices, returning exactly the same expressions. Taking this into account, we have:

$$(\Sigma_1 : e_1' \, \overline{\sigma}_j \, \overline{dv_{1\,i}}, \Sigma_2 : e_2' \, \overline{\sigma}_j \, \overline{dv_{2\,i}}) \in \mathcal{E}[\![[\overline{\sigma}_j/\overline{a}_j][\sigma_q/a]\sigma']\!]_R^{\Gamma_C}$$

In turn, unfolding the definition of the $\mathcal{E}$ relation results in:

$$\begin{aligned} \exists v_3, v_4 : \Sigma_1 &\vdash e_1' \, \overline{\sigma}_j \, \overline{dv_{1\,i}} \longrightarrow^* v_3 \\ \wedge\, \Sigma_2 &\vdash e_2' \, \overline{\sigma}_j \, \overline{dv_{2\,i}} \longrightarrow^* v_4 \\ \wedge\, (\Sigma_1 &: v_3, \Sigma_2 : v_4) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j][\sigma/a]\sigma']\!]_R^{\Gamma_C} \end{aligned} \tag{576}$$

Goal 572 follows from Equation 576 by noting that $[\overline{\sigma}_j/\overline{a}_j][\sigma/a]\sigma' = [[\overline{\sigma}_j/\overline{a}_j]\sigma/a][\overline{\sigma}_j/\overline{a}_j]\sigma' = [[\overline{\sigma}_j/\overline{a}_j]\sigma/a]\sigma'$ and taking $v_3 = v_1$ and $v_4 = v_2$.

□

M.2 Helper Theorems

Theorem 18 (Congruence - Expressions).
If $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$
and $M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \Gamma' \Rightarrow \sigma')$ *and* $M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \Gamma' \Rightarrow \sigma')$
and $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma' \Rightarrow \sigma')$
then $\Gamma_C; \Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma'$.

Proof. The goal follows directly from the definition of logical equivalence for contexts.

□

Theorem 19 ($F_{\{\}}$ Context Preserved by Elaboration).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$ *and* $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M$ *then* $\Sigma; \Gamma_C; \Gamma' \vdash_{tm} M[e] : \sigma' \rightsquigarrow M[e]$.

Proof. By structural induction on the typing derivation of M.

iM-Empty $[\bullet] : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \rightsquigarrow [\bullet]$

We need to show that:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$$

which follows immediately from the first hypothesis of the theorem.

iM-Abs $\lambda x : \sigma_1.M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_1 \to \sigma') \rightsquigarrow \lambda x : \sigma_1.M'$

We need to show the following.

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} \lambda x : \sigma_1.M'[e] : \sigma_1 \to \sigma' \rightsquigarrow \lambda x : \sigma_1.M'[e]$$

From the premises of rule iM-Abs, we obtain:

$$M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma', x : \sigma_1 \Rightarrow \sigma') \rightsquigarrow M' \tag{577}$$

$$\Gamma_C; \Gamma' \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \tag{578}$$

From the induction hypothesis applied on Equation 577, it follows that:

$$\Sigma; \Gamma_C; \Gamma', x : \sigma_1 \vdash_{tm} M'[e] : \sigma' \rightsquigarrow M'[e] \tag{579}$$

The goal follows from iTm-arrI, in combination with Equations 578 and 579.

iM-appL $M'\, e_2 : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M'\, e_2$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} M'[e]\, e_2 : \sigma' \rightsquigarrow M'[e]\, e_2$$

From the premises of rule iM-appL, we obtain:

$$M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_1 \to \sigma') \rightsquigarrow M' \tag{580}$$

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} e_2 : \sigma_1 \rightsquigarrow e_2 \tag{581}$$

From the induction hypothesis applied on Equation 580, it follows that:

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} M'[e] : \sigma_1 \to \sigma' \rightsquigarrow M'[e] \tag{582}$$

The goal follows from iTm-arrE, in combination with Equations 581 and 582.

iM-appR $e_1\, M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow e_1\, M'$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} e_1\, M'[e] : \sigma' \rightsquigarrow e_1\, M'[e]$$

From the premises of rule iM-appR, we obtain:

$$M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma_1) \rightsquigarrow M' \tag{583}$$

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} e_1 : \sigma_1 \to \sigma' \rightsquigarrow e_1 \tag{584}$$

From the induction hypothesis applied on Equation 583, it follows that:

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} M'[e] : \sigma_1 \rightsquigarrow M'[e] \tag{585}$$

The goal follows from iTm-arrE, in combination with Equations 584 and 585.

iM-dictAbs $\lambda\delta : Q.M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow Q \Rightarrow \sigma') \rightsquigarrow \lambda\delta : \sigma.M'$

The goal to be proven is the following:

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} \lambda\delta : Q.M'[e] : Q \Rightarrow \sigma' \rightsquigarrow \lambda\delta : \sigma.M'[e]$$

From the premises of rule iM-dictAbs, we obtain:

$$M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma', \delta : Q \Rightarrow \sigma') \rightsquigarrow M' \tag{586}$$

$$\Gamma_C; \Gamma' \vdash_Q Q \rightsquigarrow \sigma \tag{587}$$

From the induction hypothesis applied on Equation 586, it follows that:

$$\Sigma; \Gamma_C; \Gamma', \delta : Q \vdash_{tm} M'[e] : \sigma' \rightsquigarrow M'[e] \tag{588}$$

The goal follows from iTm-constrI, in combination with Equations 587 and 588.

iM-dictApp $M'\, d : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M'\, e$

The goal to be proven is the following.

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} M'[e]\, d : \sigma' \rightsquigarrow M'[e]\, e'$$

From the premises of rule iM-dictApp, we obtain:

$$M' : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow Q \Rightarrow \sigma') \rightsquigarrow M' \tag{589}$$

$$\Sigma; \Gamma_C; \Gamma' \vdash_d d : Q \rightsquigarrow e' \tag{590}$$

From the induction hypothesis applied on Equation 589, it follows that:

$$\Sigma; \Gamma_C; \Gamma' \vdash_{tm} M'[e] : Q \Rightarrow \sigma' \rightsquigarrow M'[e] \tag{591}$$

The goal follows from ITM-CONSTRE, in combination with Equations 590 and 591.

IM-TYABS $\Lambda a.M' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \forall a.\sigma') \rightsquigarrow \Lambda a.M'$

The goal to be proven is the following:

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} \Lambda a.M'[e] : \forall a.\sigma' \rightsquigarrow \Lambda a.M'[e]$$

From the premises of rule IM-TYABS, we obtain:

$$M' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma', a \Rightarrow \sigma') \rightsquigarrow M'$$

Applying the induction hypothesis on the above context typing, yields:

$$\Sigma;\Gamma_C;\Gamma', a \vdash_{tm} M'[e] : \sigma' \rightsquigarrow M'[e]$$

Using this result with rule ITM-FORALLI, we reach the goal.

IM-TYAPP $M'\ \sigma'' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow [\sigma''/a]\sigma') \rightsquigarrow M'\ \sigma''$

The goal to be proven is the following.

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} M'[e]\ \sigma'' : [\sigma''/a]\sigma' \rightsquigarrow M'[e]\ \sigma''$$

From the premises of rule IM-TYAPP, we obtain:

$$M' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \forall a.\sigma') \rightsquigarrow M' \quad (592)$$

$$\Gamma_C;\Gamma' \vdash_{ty} \sigma'' \rightsquigarrow \sigma'' \quad (593)$$

From the induction hypothesis applied on Equation 592, we have that:

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} M'[e] : \forall a.\sigma' \rightsquigarrow M'[e] \quad (594)$$

The goal follows from ITM-FORALLE, in combination with Equations 593 and 594.

IM-LETL $\mathbf{let}\ x : \sigma_1 = M'\ \mathbf{in}\ e_2 : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \sigma') \rightsquigarrow \mathbf{let}\ x : \sigma_1 = M'\ \mathbf{in}\ e_2$

The goal to be proven is the following:

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} \mathbf{let}\ x : \sigma_1 = M'[e]\ \mathbf{in}\ e_2 : \sigma' \rightsquigarrow \mathbf{let}\ x : \sigma_1 = M'[e]\ \mathbf{in}\ e_2$$

From the premises of rule IM-LETL, we obtain:

$$M' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \sigma_1) \rightsquigarrow M' \quad (595)$$

$$\Sigma;\Gamma_C;\Gamma', x : \sigma_1 \vdash_{tm} e_2 : \sigma' \rightsquigarrow e_2 \quad (596)$$

$$\Gamma_C;\Gamma' \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \quad (597)$$

From the induction hypothesis applied on Equation 595, we have that:

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} M'[e] : \sigma_1 \rightsquigarrow M'[e] \quad (598)$$

The goal follows from ITM-LET, in combination with Equations 596, 597 and 598.

IM-LETR $\mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ M' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma' \Rightarrow \sigma') \rightsquigarrow \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ M'$

The goal to be proven is the following:

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ M'[e] : \sigma' \rightsquigarrow \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ M'[e]$$

From the rule premise:

$$M' : (\Sigma;\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Sigma;\Gamma_C;\Gamma', x : \sigma_1 \Rightarrow \sigma') \rightsquigarrow M' \quad (599)$$

$$\Sigma;\Gamma_C;\Gamma' \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1 \quad (600)$$

$$\Gamma_C;\Gamma' \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \quad (601)$$

From the induction hypothesis applied on Equation 599, we have that:

$$\Sigma;\Gamma_C;\Gamma', x : \sigma_1 \vdash_{tm} M'[e] : \sigma' \rightsquigarrow M'[e] \quad (602)$$

The goal follows from ITM-LET, in combination with Equations 600, 601 and 602.

□

Theorem 20 (Logical Equivalence Preserved by Forward/Backward Reduction). *Given* $\Sigma_1 \vdash e_1 \longrightarrow e_1'$ *and* $\Sigma_2 \vdash e_2 \longrightarrow e_2'$*,*

- *If* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$*, then* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1' \simeq_{log} \Sigma_2 : e_2' : \sigma$*.*
- *If* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1' \simeq_{log} \Sigma_2 : e_2' : \sigma$ *and* $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma$ *and* $\Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma$*, then* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$*.*

Proof. **Part 1** By unfolding the definition of logical relation, we get:

$$R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C} \tag{603}$$
$$\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \tag{604}$$
$$\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C} \tag{605}$$
$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1))), \Sigma_2 : \gamma_2(\phi_2(R(e_2)))) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \tag{606}$$

Unfolding the definition of the closed expression relation in 606 results in:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1))) : R(\sigma) \tag{607}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2))) : R(\sigma) \tag{608}$$
$$\exists v_1, v_2, \Sigma_1 \vdash \gamma_1(\phi_1(R(e_1))) \longrightarrow^* v_1, \tag{609}$$
$$\Sigma_2 \vdash \gamma_2(\phi_2(R(e_2))) \longrightarrow^* v_2, \tag{610}$$
$$(\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![\sigma]\!]_R^{\Gamma_C} \tag{611}$$

By induction on e_1, it is easy to verify that

$$\Sigma_1 \vdash \gamma_1(\phi_1(R(e_1))) \longrightarrow \gamma_1(\phi_1(R(e_1'))) \tag{612}$$

By preservation (Theorem 8) we have:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} \gamma_1(\phi_1(R(e_1'))) : R(\sigma) \tag{613}$$

Because the evaluation in F_D is deterministic (Lemma 41), we know that:

$$\Sigma_1 \vdash \gamma_1(\phi_1(R(e_1'))) \longrightarrow^* v_1 \tag{614}$$

Similarly:

$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} \gamma_2(\phi_2(R(e_2'))) : R(\sigma) \tag{615}$$
$$\Sigma_2 \vdash \gamma_2(\phi_2(R(e_2'))) \longrightarrow^* v_2 \tag{616}$$

Combining those equations, results in:

$$(\Sigma_1 : \gamma_1(\phi_1(R(e_1'))), \Sigma_2 : \gamma_2(\phi_2(R(e_2')))) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \tag{617}$$

The goal follows from the definition of logical equivalence.

Part 2 Similar to Part 1.

□

Theorem 21 (Dictionary Reflexivity). *If* $\Sigma_1; \Gamma_C; \Gamma \vdash_d d : Q$ *and* $\Sigma_2; \Gamma_C; \Gamma \vdash_d d : Q$ *and* $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$*, then* $\Gamma_C; \Gamma \vdash \Sigma_1 : d \simeq_{log} \Sigma_2 : d : Q$*.*

Proof. Proof by structural induction on the dictionary d and consequently, since F_D dictionary typing is syntax directed, on both typing derivations.

$d = \delta$ **(D-var)** $\quad \Sigma_1; \Gamma_C; \Gamma \vdash_d \delta : Q \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_d \delta : Q$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \delta \simeq_{log} \Sigma_2 : \delta : Q$$

By unfolding the definition of logical equivalence, the goal reduces to:

$$(\Sigma_1 : \gamma_1(\phi_1(R(\delta))), \Sigma_2 : \gamma_2(\phi_2(R(\delta)))) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C}$$

where $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$.

From the given we know that $(\delta : Q) \in \Gamma$. Because of this, it follows from the definition of $\mathcal{H}$ that:

$$\gamma_1(\phi_1(R(\delta))) = dv_1$$
$$\gamma_2(\phi_2(R(\delta))) = dv_2$$
$$(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C}$$

$\boxed{d = D\,\overline{\sigma}_j\,\overline{d}_i \text{ (D-CON)}}$ $\quad \Sigma_1; \Gamma_C; \Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : D\,\overline{\sigma}_j\,\overline{d}_i \simeq_{log} \Sigma_2 : D\,\overline{\sigma}_j\,\overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q$$

Unfolding the definition of logical equivalence in the goal results in:

$$(\Sigma_1 : \gamma_1(\phi_1(R(D\,\overline{\sigma}_j\,\overline{d}_i))), \Sigma_2 : \gamma_2(\phi_2(R(D\,\overline{\sigma}_j\,\overline{d}_i)))) \in \mathcal{V}[\![TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q]\!]_R^{\Gamma_C}$$

where $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$.
By the definition of the $\mathcal{V}$ relation and by distributivity of substitution over application, it suffices to show that:

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto e \in \Sigma_1 \tag{618}$$
$$\overline{(\Sigma_1 : \gamma_1(\phi_1(R(d_i))), \Sigma_2 : \gamma_2(\phi_2(R(d_i)))) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j]Q_i]\!]_R^{\Gamma_C}}^{\,i} \tag{619}$$
$$\Sigma_1; \Gamma_C; \bullet \vdash_d \gamma_1(\phi_1(R(D\,\overline{\sigma}_j\,\overline{d}_i))) : R(TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q) \tag{620}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_d \gamma_2(\phi_2(R(D\,\overline{\sigma}_j\,\overline{d}_i))) : R(TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q) \tag{621}$$

From the premises of the two D-CON rules, we know that:

$$\vdash_{ctx} \Sigma_1; \Gamma_C; \Gamma \tag{622}$$
$$\vdash_{ctx} \Sigma_2; \Gamma_C; \Gamma \tag{623}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i}^{\,i} \tag{624}$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q'_i}^{\,i} \tag{625}$$
$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{\,j} \tag{626}$$
$$\overline{\Sigma_1; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i} \tag{627}$$
$$\overline{\Sigma_2; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q'_i}^{\,i} \tag{628}$$
$$\Sigma'_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e_1 : [\overline{\sigma}_j/\overline{a}_j]\sigma_m \tag{629}$$
$$\Sigma'_2; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}'_i \vdash_{tm} e_2 : [\overline{\sigma}_j/\overline{a}_j]\sigma_m \tag{630}$$
$$\textit{where } \Sigma_1 = \Sigma'_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e_1, \Sigma''_1 \tag{631}$$
$$\textit{and } \Sigma_2 = \Sigma'_2, (D : \forall \overline{a}_j.\overline{Q}'_i \Rightarrow TC'\,\sigma'_q).m' \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}'_i.e_2, \Sigma''_2 \tag{632}$$

From the definition of logical equivalence in the third hypothesis of the theorem, we know that $\overline{Q}_i = \overline{Q}'_i$, $TC = TC'$, $\sigma_q = \sigma'_q$, $m = m'$.
Goal 618 follows directly from Equation 631 by setting $e = \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}'_i.e_2$.
By applying the induction hypothesis on Equations 627 and 628, we obtain:

$$\overline{\Gamma_C; \Gamma \vdash \Sigma_1 : d_i \simeq_{log} \Sigma_2 : d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{\,i}$$

Unfolding the definition of logical equivalence in the above, we get:

$$\overline{(\Sigma_1 : dv_{i1}, \Sigma_2 : dv_{i2}) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j]Q_i]\!]_R^{\Gamma_C}}^{\,i} \tag{633}$$

where $dv_{i1} = \gamma_1(\phi_1(R(d_i)))$ and $dv_{i2} = \gamma_2(\phi_2(R(d_i)))$. This proves Goal 619.
By applying Lemma 20 in Equation 626, there is an environment Γ' (the resulting environment after applying on Γ all type variable substitutions of R) such that

$$\overline{\Gamma_C; \Gamma' \vdash_{ty} R(\sigma_j)}^{\,j}$$

From Lemma 20 and since the domain of R contains all type variables of Γ, it is evident that all type variables are eliminated in Γ'. By consecutive applications of Lemma 38 we have that

$$\overline{\Gamma_C; \bullet \vdash_{ty} R(\sigma_j)}^{j} \tag{634}$$

Furthermore, from the definiton of the $\mathcal{V}$ relation in Equation 633, we obtain:

$$\overline{\Sigma_1; \Gamma_C; \bullet \vdash_d dv_{i1} : R([\overline{\sigma}_j/\overline{a}_j]Q_i)}^{i}$$
$$\overline{\Sigma_2; \Gamma_C; \bullet \vdash_d dv_{i2} : R([\overline{\sigma}_j/\overline{a}_j]Q_i)}^{i}$$

From Equation 624, we know that Q_i only contains free variables $\overline{a}_j$. It follows that the above equations can be simplified to:

$$\overline{\Sigma_1; \Gamma_C; \bullet \vdash_d dv_{i1} : [R(\overline{\sigma}_j)/\overline{a}_j]Q_i}^{i} \tag{635}$$
$$\overline{\Sigma_2; \Gamma_C; \bullet \vdash_d dv_{i2} : [R(\overline{\sigma}_j)/\overline{a}_j]Q_i}^{i} \tag{636}$$

Applying the substitutions in Goals 620 and 621, reduces them to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d D\, R(\overline{\sigma}_j)\, \overline{dv}_{i1} : TC\, R([\overline{\sigma}_j/\overline{a}_j]\sigma_q)$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_d D\, R(\overline{\sigma}_j)\, \overline{dv}_{i2} : TC\, R([\overline{\sigma}_j/\overline{a}_j]\sigma_q)$$

Similarly to Q_i, it follows from iCtx-MEnv that σ_q only contains free variables $\overline{a}_j$. The above goals thus simplify to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d D\, R(\overline{\sigma}_j)\, \overline{dv}_{i1} : TC\, [R(\overline{\sigma}_j)/\overline{a}_j]\sigma_q \tag{637}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_d D\, R(\overline{\sigma}_j)\, \overline{dv}_{i2} : TC\, [R(\overline{\sigma}_j)/\overline{a}_j]\sigma_q \tag{638}$$

Goal 637 follows from D-con, in combination with Equations 631, 622, 624, 634, 629 and 635. Goal 638 follows from from D-con, in combination with Equations 632, 623, 624, 634, 630 and 636.

□

Theorem 22 (Expression Reflexivity).
If $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$ *and* $\Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e : \sigma$ *and* $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$*, then* $\Gamma_C; \Gamma \vdash \Sigma_1 : e \simeq_{log} \Sigma_2 : e : \sigma$.

Proof. The proof proceeds by induction on e and consequently, since F_D term typing is syntax directed, on both typing derivations.

$e = True$ **(iTm-true)** $\quad \Sigma_1; \Gamma_C; \Gamma \vdash_{tm} True : Bool \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} True : Bool$

The goal to prove is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : True \simeq_{log} \Sigma_2 : True : Bool$$

Unfolding the definition of logical equivalence in the above, results in the following goal:

$$(\Sigma_1 : \gamma_1(\phi_1(R(True))), \Sigma_2 : \gamma_2(\phi_2(R(True)))) \in \mathcal{E}[\![Bool]\!]^{\Gamma_C}_R \tag{639}$$

where $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_R$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_R$. However, since $True$ does not contain any free variables, we know that $\gamma_1(\phi_1(R(True))) = \gamma_2(\phi_2(R(True))) = True$. Similarly, it follows that $R(Bool) = Bool$.
Unfolding the definition of the $\mathcal{E}$ relation in Goal 639, reduces the goal to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} True : Bool \tag{640}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} True : Bool \tag{641}$$
$$\exists v_1, v_2 : \Sigma_1 \vdash True \longrightarrow^* v_1$$
$$\wedge\, \Sigma_2 \vdash True \longrightarrow^* v_2$$
$$\wedge\, (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![Bool]\!]^{\Gamma_C}_R$$

Goals 640 and 641 are satisified from the first and second hypotheses of the theorem. We set $v_1 = v_2 = True$ and since $True$ is a value, the term reductions above hold. Then, the last goal follows directly from the definition of the $\mathcal{V}$ relation, according to which the following holds trivially.

$$(\Sigma_1 : True, \Sigma_2 : True) \in \mathcal{V}[\![Bool]\!]^{\Gamma_C}_R$$

$e = False$ **(iTm-false)** $\quad \Sigma_1; \Gamma_C; \Gamma \vdash_{tm} False : Bool \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} False : Bool$

The proof is similar to the iTm-true case.

$e = x$ **(iTm-var)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} x : \sigma \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} x : \sigma$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : x \simeq_{log} \Sigma_2 : x : \sigma$$

By unfolding the definition of logical equivalence in the above, we have

$$(\Sigma_1 : \gamma_1(\phi_1(R(x))), \Sigma_2 : \gamma_2(\phi_2(R(x)))) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C}$$

for any $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$. From the definition of the $\mathcal{G}$ relation, we know that:

$$\gamma_1(\phi_1(R(x))) = e_1$$

$$\gamma_2(\phi_2(R(x))) = e_2$$

$$(\Sigma_1 : e_1, \Sigma_2 : e_2) \in \mathcal{E}[\![\sigma]\!]_R^{\Gamma_C} \tag{642}$$

The goal follows directly from Equation 642.

$e = \textbf{let}\ x : \sigma_1 = e_1\ \textbf{in}\ e_2$ **(iTm-let)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} \textbf{let}\ x : \sigma_1 = e_1\ \textbf{in}\ e_2 : \sigma_2 \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} \textbf{let}\ x : \sigma_1 = e_1\ \textbf{in}\ e_2 : \sigma_2$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \textbf{let}\ x : \sigma_1 = e_1\ \textbf{in}\ e_2 \simeq_{log} \Sigma_2 : \textbf{let}\ x : \sigma_1 = e_1\ \textbf{in}\ e_2 : \sigma_2$$

By applying the induction hypothesis in the premises of the two iTm-let rules, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_1 : \sigma_1$$

$$\Gamma_C; \Gamma, x : \sigma_1 \vdash \Sigma_1 : e_2 \simeq_{log} \Sigma_2 : e_2 : \sigma_2$$

The goal follows directly by passing the above two Equations to compatibility Lemma 56.

$e = d.m$ **(iTm-method)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma' \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma'$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : d.m \simeq_{log} \Sigma_2 : d.m : [\sigma/a]\sigma'$$

From the premises of rule iTm-method we have that

$$\Sigma_1; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \tag{643}$$

$$\Sigma_2; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \tag{644}$$

$$(m : TC\,a : \sigma') \in \Gamma_C \tag{645}$$

Applying the Dictionary Reflexivity (Theorem 21) to Equations 643 and 644, in combination with the theorem's third hypothesis, results in:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : d \simeq_{log} \Sigma_2 : d : TC\,\sigma \tag{646}$$

The goal follows directly from compatibility Lemma 57 and Equations 645 and 646, in combination with the third hypothesis.

$e = \lambda x : \sigma_1.e'$ **(iTm-arrI)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e' : \sigma_1 \rightarrow \sigma_2 \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e' : \sigma_1 \rightarrow \sigma_2$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \lambda x : \sigma_1.e' \simeq_{log} \Sigma_2 : \lambda x : \sigma_1.e' : \sigma_1 \rightarrow \sigma_2$$

By applying the induction hypothesis to the premises of the two iTm-arrI rules, we get:

$$\Gamma_C; \Gamma, x : \sigma_1 \vdash \Sigma_1 : e' \simeq_{log} \Sigma_2 : e' : \sigma_2$$

The goal follows by applying the above to compatibility Lemma 50.

$e = e_1\, e_2$ **(iTm-arrE)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2 \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1\, e_2 \simeq_{log} \Sigma_2 : e_1\, e_2 : \sigma_2$$

By applying the induction hypothesis to the premises of the two iTm-arrE rules, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_1 : \sigma_1 \rightarrow \sigma_2$$
$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_2 \simeq_{log} \Sigma_2 : e_2 : \sigma_1$$

The goal follows by passing the above two equations to compatibility Lemma 51.

$e = \lambda\delta : Q.e'$ **(iTm-constrI)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e' : Q \Rightarrow \sigma \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e' : Q \Rightarrow \sigma$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \lambda\delta : Q.e' \simeq_{log} \Sigma_2 : \lambda\delta : Q.e' : Q \Rightarrow \sigma$$

By applying the induction hypothesis to the premises of the two iTm-constrI rules, we get:

$$\Gamma_C; \Gamma, \delta : Q \vdash \Sigma_1 : e' \simeq_{log} \Sigma_2 : e' : \sigma$$

The goal follows directly by passing the above equation to compatibility Lemma 52.

$e = e'\, d$ **(iTm-constrE)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e'\, d : \sigma \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e'\, d : \sigma$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e'\, d \simeq_{log} \Sigma_2 : e'\, d : \sigma$$

By applying the induction hypothesis to the premises of the two iTm-constrE rules, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e' \simeq_{log} \Sigma_2 : e' : Q \Rightarrow \sigma \tag{647}$$

Furthermore, applying Dictionary Reflexivity (Theorem 21) in the premises of the two iTm-constrE rules, in combination with the theorem's third hypothesis, results in:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : d \simeq_{log} \Sigma_2 : d : Q \tag{648}$$

The goal follows from compatibility Lemma 53 and Equations 647 and 648.

$e = \Lambda a.e'$ **(iTm-forallI)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} \Lambda a.e' : \forall a.\sigma \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} \Lambda a.e' : \forall a.\sigma$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \Lambda a.e' \simeq_{log} \Sigma_2 : \Lambda a.e' : \forall a.\sigma$$

By applying the induction hypothesis to the premises of the two iTm-forallI rules, we get:

$$\Gamma_C; \Gamma, a \vdash \Sigma_1 : e' \simeq_{log} \Sigma_2 : e' : \sigma_1$$

The goal follows directly by applying the above equation to compatibility Lemma 54.

$e = e'\, \sigma$ **(iTm-forallE)** $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e'\, \sigma : [\sigma/a]\sigma' \wedge \Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e'\, \sigma : [\sigma/a]\sigma'$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e'\, \sigma \simeq_{log} \Sigma_2 : e'\, \sigma : [\sigma/a]\sigma'$$

From the premises of iTm-forallE, we obtain

$$\Gamma_C; \Gamma \vdash_{ty} \sigma \tag{649}$$

By applying the induction hypothesis to the premises of the two iTm-forallE rules, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1' \simeq_{log} \Sigma_2 : e_1' : \sigma_1 \tag{650}$$

The goal follows from compatibility Lemma 55 and Equations 650 and 649.

□

Theorem 23 (Context Reflexivity).
Suppose $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_1;\Gamma_C;\Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma' \rightsquigarrow \Sigma_1;\Gamma_C;\Gamma'$
and $\vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma_2;\Gamma_C;\Gamma$ *and* $\vdash^M_{ctx} P;\Gamma_C;\Gamma' \rightsquigarrow \Sigma_2;\Gamma_C;\Gamma'$
and $\Gamma_C;\Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma$ *and* $\Gamma_C;\Gamma' \vdash^M_{ty} \tau' \rightsquigarrow \sigma'$,

- *If* $M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M_1$
  and $M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M_2$
  then $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\Gamma' \Rightarrow \sigma')$.
- *If* $M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M_1$
  and $M : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M_2$
  then $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\Gamma' \Rightarrow \sigma')$.
- *If* $M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M_1$
  and $M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau') \rightsquigarrow M_2$
  then $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\Gamma' \Rightarrow \sigma')$.
- *If* $M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M_1$
  and $M : (P;\Gamma_C;\Gamma \Leftarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau') \rightsquigarrow M_2$
  then $\Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\Gamma' \Rightarrow \sigma')$.

Proof. The theorem is stated in a nested fashion, where all common hypotheses are introduced in the outer statement. Each of the four inner statements extends the outer statement with two context-typing hypotheses, and sets the conclusion of the theorem, which is identical for each of the four cases.

Suppose expressions e_1 and e_2 such that

$$\Gamma_C;\Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma \tag{651}$$

Then, by unfolding the defintion of logical equivalence in the goal of all four sub-statements, it suffices to show that

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma' \tag{652}$$

We assume all hypotheses of the outer statement and we proceed by mutual induction on the first hypothesis of the nested statements. Note that context typing derivations are of finite size, thus mutual induction over them is safe.

Part 1

sM-inf-inf-empty $\quad [\bullet] : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma \Rightarrow \tau) \rightsquigarrow [\bullet]$

By case analysis on the second hypothesis of the nested statement, its last context typing rule must be sM-inf-inf-empty as well. Therefore, the first and second hypotheses of the nested statement become

$$[\bullet] : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma \Rightarrow \tau) \rightsquigarrow [\bullet]$$
$$[\bullet] : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma \Rightarrow \tau) \rightsquigarrow [\bullet]$$

and we have $M_1 = M_2 = [\bullet]$. Thus, Goal 652 becomes

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$$

The above logical equivalence follows directly from Equation 651.

sM-inf-inf-appL $\quad M'\, e_2' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M_1'\, e_{21}$

By case analysis on the second hypothesis of the nested statement, its last context typing rule must be sM-inf-inf-appL as well. Therefore, the first and second hypotheses of the nested statement become

$$M'\, e_2' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M_1'\, e_{21}$$
$$M'\, e_2' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M_2'\, e_{22}$$

and Goal 652 becomes

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : M_1'[e_1]\, e_{21} \simeq_{log} \Sigma_2 : M_2'[e_2]\, e_{22} : \sigma_2 \tag{653}$$

From the premises of the two sM-inf-inf-appL rules, we obtain:

$$M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_1 \to \tau_2) \rightsquigarrow M_1' \tag{654}$$
$$M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_1' \to \tau_2) \rightsquigarrow M_2' \tag{655}$$
$$P;\Gamma_C;\Gamma' \vdash^M_{tm} e_2' \Leftarrow \tau_1 \rightsquigarrow e_{21} \tag{656}$$
$$P;\Gamma_C;\Gamma' \vdash^M_{tm} e_2' \Leftarrow \tau_1' \rightsquigarrow e_{22} \tag{657}$$

By applying Lemma 7 to Equations 654 and 655, we know that $\tau_1' \to \tau_2 = \tau_1 \to \tau_2$.

Applying the induction hypothesis on Equations 654 and 655, yields:

$$\Sigma_1 : M'_1 \simeq_{log} \Sigma_2 : M'_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\Gamma' \Rightarrow \sigma_1 \rightarrow \sigma_2)$$

By unfolding the definition of logical equivalence in the above equation and applying it on Equation 651, we get:

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : M'_1[e_1] \simeq_{log} \Sigma_2 : M'_2[e_2] : \sigma_1 \rightarrow \sigma_2 \quad (658)$$

By applying Expression Coherence Theorem A (Theorem 28) on Equations 656 and 657 (and on the second and fourth hypotheses of the outer statement), we get:

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : e_{21} \simeq_{log} \Sigma_2 : e_{22} : \sigma_1 \quad (659)$$

Goal 653 follows from compatibility of term applications (Lemma 51, together with Equations 658 and 659).

sM-inf-inf-appR $e'_1\, M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow e_{11}\, M'_1$

By case analysis on the second hypothesis of the nested statement, its last context typing rule must be sM-inf-inf-appR as well. Therefore, the first and second hypotheses of the nested statement become

$$e'_1\, M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow e_{11}\, M'_1$$
$$e'_1\, M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow e_{12}\, M'_2$$

and we need to show that

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : e_{11}\, M'_1[e_1] \simeq_{log} \Sigma_2 : e_{12}\, M'_2[e_2] : \sigma_2 \quad (660)$$

From the premises of the two sM-inf-inf-appR rules, we obtain:

$$M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau_1) \rightsquigarrow M'_1 \quad (661)$$
$$M' : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Leftarrow \tau'_1) \rightsquigarrow M'_2 \quad (662)$$
$$P;\Gamma_C;\Gamma' \vdash^M_{tm} e'_1 \Rightarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow e_{11} \quad (663)$$
$$P;\Gamma_C;\Gamma' \vdash^M_{tm} e'_1 \Rightarrow \tau'_1 \rightarrow \tau_2 \rightsquigarrow e_{12} \quad (664)$$

By Lemma 7, we know that $\tau'_1 = \tau_1$.
By applying Part 2 of this theorem to Equations 661 and 662, we get:

$$\Sigma_1 : M'_1 \simeq_{log} \Sigma_2 : M'_2 : (\Gamma_C;\Gamma \Rightarrow \sigma) \mapsto (\Gamma_C;\Gamma' \Rightarrow \sigma_1)$$

By unfolding the definition of logical equivalence in the above and applying it on Equation 651, we get:

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : M'_1[e_1] \simeq_{log} \Sigma_2 : M'_2[e_2] : \sigma_1 \quad (665)$$

By applying Expression Coherence Theorem A (Theorem 28) to Equations 663 and 664 (and on the second and fourth hypotheses of the outer statement), we get:

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : e_{11} \simeq_{log} \Sigma_2 : e_{12} : \sigma_1 \rightarrow \sigma_2 \quad (666)$$

Goal 660 follows from compatibility of term applications (Lemma 51, together with Equations 665 and 666).

sM-inf-inf-letL $\mathbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M'\ \mathbf{in}\ e'_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M_1$

where $M_1 = \mathbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.M'_1\ \mathbf{in}\ e_{21}$.
By case analysis on the second hypothesis of the nested statement, its last context typing rule must be sM-inf-inf-letL as well. Therefore, the first and second hypotheses of the nested statement become

$$\mathbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M'\ \mathbf{in}\ e'_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M_1$$
$$\mathbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau_1 = M'\ \mathbf{in}\ e'_2 : (P;\Gamma_C;\Gamma \Rightarrow \tau) \mapsto (P;\Gamma_C;\Gamma' \Rightarrow \tau_2) \rightsquigarrow M_2$$

where $M_2 = \mathbf{let}\ x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j.\lambda \overline{\delta}_i : \overline{Q}_i.M'_2\ \mathbf{in}\ e_{22}$.
Goal 652 becomes

$$\Gamma_C;\Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma_2 \quad (667)$$

From the premises of the two sM-INF-INF-LETL rules, we obtain:

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \Leftarrow \tau_1) \rightsquigarrow M'_1 \tag{668}$$
$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \Leftarrow \tau_1) \rightsquigarrow M'_2 \tag{669}$$
$$P; \Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 \vdash^M_{tm} e'_2 \Rightarrow \tau_2 \rightsquigarrow e_{21} \tag{670}$$
$$P; \Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 \vdash^M_{tm} e'_2 \Rightarrow \tau_2 \rightsquigarrow e_{22} \tag{671}$$
$$\Gamma_C; \Gamma' \vdash^M_{ty} \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1 \tag{672}$$
$$\overline{\delta}_i \textbf{ fresh} \tag{673}$$
$$x \notin \textbf{dom}(\Gamma') \tag{674}$$

Through repeated case analysis on Equation 672 (sTY-SCHEME and sTY-QUAL), we know that

$$\overline{a}_j \notin \Gamma' \qquad \overline{\frac{}{\Gamma_C; \Gamma', \overline{a}_j \vdash^M_Q Q_i \rightsquigarrow Q_i}}^{\,i}$$
$$\Gamma_C; \Gamma' \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma_1 \tag{675}$$

By repeated case analysis on the second hypothesis, we get that

$$\vdash^M_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet \tag{676}$$

By sCTX-TYENVTY and sCTX-TYENVD, in combination with these results and Equation 673, we obtain $\vdash^M_{ctx} \bullet; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \rightsquigarrow \bullet; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i$. Finally, Lemma 6, together with this result and the second and fourth hypothesis, teaches us that:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \tag{677}$$
$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \tag{678}$$

By applying Part 2 of this theorem to Equations 668 and 669, in combination with Equations 675, 677 and 678 and the first, third and fifth hypothesis, we get:

$$\Sigma_1 : M'_1 \simeq_{log} \Sigma_2 : M'_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \Rightarrow \sigma_1)$$

By unfolding the definition of logical equivalence in the above and applying it on Equation 651, we get:

$$\Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash \Sigma_1 : M'_1[e_1] \simeq_{log} \Sigma_2 : M'_2[e_2] : \sigma_1 \tag{679}$$

By repeatedly applying compatibility Lemmas 52 and 54, we get:

$$\Gamma_C; \Gamma' \vdash \Sigma_1 : \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . M'_1[e_1] \simeq_{log} \Sigma_2 : \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . M'_2[e_2] : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \tag{680}$$

It follows from sCTX-TYENVTM, in combination with Equations 676, 674 and 672, that
$\vdash^M_{ctx} \bullet; \Gamma_C; \bullet, x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \bullet; \Gamma_C; \bullet, x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1$. Similarly to before, by applying Lemma 6 to this result, together with the second and fourth hypothesis, we get:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1 \tag{681}$$
$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1 \tag{682}$$

By applying Expression Coherence Theorem A (Theorem 28) to Equations 670 and 671, together with Equations 681 and 682, we get:

$$\Gamma_C; \Gamma', x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1 \vdash \Sigma_1 : e_{21} \simeq_{log} \Sigma_2 : e_{22} : \sigma_2 \tag{683}$$

Goal 667 follows from compatibility of let expressions (Lemma 56, together with Equations 679 and 683).

sM-INF-INF-LETR $\textbf{let } x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 = e'_1 \textbf{ in } M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M_1$

where $M_1 = \textbf{let } x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e_{11} \textbf{ in } M'_1$.
By case analysis on the second hypothesis of the nested statement, its last context typing rule must be sM-INF-INF-LETR as well. Therefore, the first and second hypotheses of the nested statement become

$$\textbf{let } x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 = e'_1 \textbf{ in } M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M_1$$
$$\textbf{let } x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \tau_1 = e'_1 \textbf{ in } M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau_2) \rightsquigarrow M_2$$

where $M_2 = \textbf{let } x : \forall \overline{a}_j. \overline{Q}_i \Rightarrow \sigma_1 = \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e_{12} \textbf{ in } M'_2$.

Goal 652 becomes

$$\Gamma_C; \Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma_2 \tag{684}$$

From the premises of the two sM-INF-INF-LETR rules, we obtain:

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 \Rightarrow \tau_2) \rightsquigarrow M_1' \tag{685}$$

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 \Rightarrow \tau_2) \rightsquigarrow M_2' \tag{686}$$

$$P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_{11} \tag{687}$$

$$P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_{12} \tag{688}$$

$$\Gamma_C; \Gamma' \vdash^M_{ty} \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \tag{689}$$

$$\overline{\delta}_i \ \textbf{fresh} \tag{690}$$

$$x \notin \textbf{dom}(\Gamma') \tag{691}$$

By repeated case analysis on the second hypothesis, we get that

$$\vdash^M_{ctx} \bullet; \Gamma_C; \bullet \rightsquigarrow \bullet; \Gamma_C; \bullet \tag{692}$$

By sCtx-tyEnvTm, in combination with this result and Equations 689 and 691, we obtain $\vdash^M_{ctx} \bullet; \Gamma_C; \bullet, x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \bullet; \Gamma_C; \bullet, x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1$. Lemma 6, together with this result and the second and fourth hypothesis, teaches us that:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \tag{693}$$

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \tag{694}$$

By applying the induction hypothesis to Equations 685 and 686, in combination with Equations 693 and 694, we get:

$$\Sigma_1 : M_1' \simeq_{log} \Sigma_2 : M_2' : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \Rightarrow \sigma_2)$$

By unfolding the definition of logical equivalence in the above and then applying it on Equation 651, we get:

$$\Gamma_C; \Gamma', x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \vdash \Sigma_1 : M_1'[e_1] \simeq_{log} \Sigma_2 : M_2'[e_2] : \sigma_2 \tag{695}$$

Through repeated case analysis on Equation 689 (sTy-scheme and sTy-qual), we know that

$$\overline{a}_j \notin \Gamma'$$

$$\overline{\Gamma_C; \Gamma', \overline{a}_j \vdash^M_Q Q_i \rightsquigarrow Q_i}^{\,i}$$

$$\Gamma_C; \Gamma' \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma_1 \tag{696}$$

By sCtx-tyEnvTy and sCtx-tyEnvD, in combination with these results and Equations 690 and 692, we obtain $\vdash^M_{ctx} \bullet; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \rightsquigarrow \bullet; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i$. Lemma 6, together with this result and the second and fourth hypothesis, teaches us that:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \tag{697}$$

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \tag{698}$$

By applying Expression Coherence Theorem A (Theorem 28) to Equations 687 and 688, in combination with Equations 697 and 698, we get:

$$\Gamma_C; \Gamma', \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash \Sigma_1 : e_{11} \simeq_{log} \Sigma_2 : e_{12} : \sigma_1 \tag{699}$$

By repeatedly applying compatibility Lemmas 52 and 54, we get:

$$\Gamma_C; \Gamma' \vdash \Sigma_1 : \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e_{11} \simeq_{log} \Sigma_2 : \Lambda \overline{a}_j . \lambda \overline{\delta}_i : \overline{Q}_i . e_{12} : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma_1 \tag{700}$$

Goal 684 follows from Lemma 56, together with Equations 695 and 700.

sM-INF-INF-ANN $M' :: \tau' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M_1$

By case analysis, we know the final step in the second derivation has to be sM-INF-INF-ANN as well. This means that:

$$M' :: \tau' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M_1 \tag{701}$$

$$M' :: \tau' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M_2 \tag{702}$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma' \tag{703}$$

From the rule premise we know that:

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M_1 \tag{704}$$

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M_2 \tag{705}$$

Goal 703 follows directly from Part 2 of this theorem, in combination with Equations 704 and 705.

Part 2 By case analysis on the first typing derivation.

sM-inf-check-abs $\lambda x.M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_1 \to \tau_2) \rightsquigarrow \lambda x : \sigma_1.M_1'$

By case analysis, we know that the final step in the second derivation has to be either sM-inf-check-abs or sM-inf-check-inf. Note however that no matching inference rules exist. The final step in the second derivation thus has to be sM-inf-check-abs as well. This means that:

$$\lambda x.M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_1 \to \tau_2) \rightsquigarrow \lambda x : \sigma_1.M_1' \tag{706}$$

$$\lambda x.M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau_1 \to \tau_2) \rightsquigarrow \lambda x : \sigma_1.M_2' \tag{707}$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma' \vdash \Sigma_1 : \lambda x : \sigma_1.M_1'[e_1] \simeq_{log} \Sigma_2 : \lambda x : \sigma_1.M_2'[e_2] : \sigma_1 \to \sigma_2 \tag{708}$$

From the rule premise we know that:

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma', x : \tau_1 \Leftarrow \tau_2) \rightsquigarrow M_1' \tag{709}$$

$$M' : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma', x : \tau_1 \Leftarrow \tau_2) \rightsquigarrow M_2' \tag{710}$$

$$\Gamma_C; \Gamma' \vdash_{ty}^{M} \tau_1 \rightsquigarrow \sigma_1 \tag{711}$$

We know from sCtx-tyEnvTm, in combination with Equation 711 and the 4th and 6th hypothesis that:

$$\vdash_{ctx}^{M} P; \Gamma_C; \Gamma', x : \tau_1 \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma', x : \sigma_1 \tag{712}$$

$$\vdash_{ctx}^{M} P; \Gamma_C; \Gamma', x : \tau_1 \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma', x : \sigma_1 \tag{713}$$

By applying the induction hypothesis to Equations 709 and 710, in combination with Equations 712 and 713, we get:

$$\Sigma_1 : M_1' \simeq_{log} \Sigma_2 : M_2' : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \Gamma', x : \sigma_1 \Rightarrow \sigma_2) \tag{714}$$

By unfolding the definition of logical equivalence in Equation 714, we get:

$$\forall e_1', e_2' : \Gamma_C; \Gamma \vdash \Sigma_1 : e_1' \simeq_{log} \Sigma_2 : e_2' : \sigma \tag{715}$$

$$\Rightarrow \Gamma_C; \Gamma', x : \sigma_1 \vdash \Sigma_1 : M_1'[e_1'] \simeq_{log} \Sigma_2 : M_2'[e_2'] : \sigma_2 \tag{716}$$

This result, together with Equation 651, tells us that:

$$\Gamma_C; \Gamma', x : \sigma_1 \vdash \Sigma_1 : M_1'[e_1] \simeq_{log} \Sigma_2 : M_2'[e_2] : \sigma_2 \tag{717}$$

Goal 708 follows from Lemma 50, together with Equation 717.

sM-inf-check-inf $M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M_1$

By case analysis, we know that the final step in the second derivation has to be either sM-inf-check-abs or sM-inf-check-inf. Note however that in the case of sM-inf-check-abs, M would have to be of the form $M = \lambda x.M'$. In this case, no matching inference rules exist, meaning that this is an impossible case. Consequently, the final step in the second derivation can only be sM-inf-check-inf. This means that:

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M_1 \tag{718}$$

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Leftarrow \tau') \rightsquigarrow M_2 \tag{719}$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma' \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : \sigma' \tag{720}$$

From the rule premise we know that:

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M_1 \tag{721}$$

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \Gamma' \Rightarrow \tau') \rightsquigarrow M_2 \tag{722}$$

Goal 720 follows directly by applying Part 1 of this theorem to Equations 721 and 722.

Part 3 By case analysis on the first typing derivation.
Similar to Part 1.

Part 4 By case analysis on the first typing derivation.
Similar to Part 2.

□

Theorem 24 (Value Relation for Dictionary Values). *If* $\Sigma_1; \Gamma_C; \bullet \vdash_d dv_1 : Q$ *and* $\Sigma_2; \Gamma_C; \bullet \vdash_d dv_2 : Q$ *and* $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$
then $(\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_{\bullet}$.

Proof. By induction on the size of dv_1 and dv_2.
From the definition of dictionary values we know that

$$dv_1 = D\,\overline{\sigma}_j\,\overline{dv}_i$$

$$dv_2 = D'\,\overline{\sigma}'_h\,\overline{dv}'_k$$

For some D, D', $\overline{\sigma}_j$, $\overline{\sigma}'_h$, $\overline{dv}_i$ and $\overline{dv}'_k$.
By case analysis on both the 1st and 2nd hypothesis (D-con), we know that:

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto e \in \Sigma_1 \text{ where } Q = [\overline{\sigma}_j/\overline{a}_j]Q' \tag{723}$$

$$\overline{\Gamma_C; \bullet \vdash_{ty} \sigma_j}^{j} \tag{724}$$

$$\overline{\Sigma_1; \Gamma_C; \bullet \vdash_d dv_i : [\overline{\sigma}_j/\overline{a}_j]Q_i}^{i} \tag{725}$$

$$\Sigma_{11}; \Gamma_C; \bullet \vdash_{tm} e : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma \text{ where } \Sigma_1 = \Sigma_{11}, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto e, \Sigma_{12} \tag{726}$$

$$\vdash_{ctx} \Sigma_1; \Gamma_C; \bullet \tag{727}$$

$$(D' : \forall \overline{b}_h.\overline{Q}'_k \Rightarrow Q'').m' \mapsto e' \in \Sigma_2 \text{ where } Q = [\overline{\sigma}'_h/\overline{b}_h]Q'' \tag{728}$$

$$\overline{\Gamma_C; \bullet \vdash_{ty} \sigma'_h}^{h} \tag{729}$$

$$\overline{\Sigma_2; \Gamma_C; \bullet \vdash_d dv'_k : [\overline{\sigma}'_h/\overline{b}_h]Q'_k}^{k} \tag{730}$$

$$\Sigma_{21}; \Gamma_C; \bullet \vdash_{tm} e' : \forall \overline{b}_h.\overline{Q}'_k \Rightarrow \sigma' \text{ where } \Sigma_2 = \Sigma_{21}, (D' : \forall \overline{b}_h.\overline{Q}'_k \Rightarrow Q'').m' \mapsto e', \Sigma_{22} \tag{731}$$

$$\vdash_{ctx} \Sigma_2; \Gamma_C; \bullet \tag{732}$$

By case analysis on Equations 727 and 732 (iCtx-MEnv) and the definition of logical equivalence in the 3rd hypothesis, it follows from Equations 723 and 728 that:

$$D = D'$$

$$\overline{a}_j = \overline{b}_h$$

$$\overline{Q}_i = \overline{Q}'_k$$

$$Q' = Q''$$

$$m = m'$$

$$\Gamma_C \vdash \Sigma_{11} \simeq_{log} \Sigma_{21}$$

$$\Gamma_C \vdash \Sigma_{12} \simeq_{log} \Sigma_{22}$$

$$\Gamma_C; \bullet \vdash \Sigma_{11} : e \simeq_{log} \Sigma_{21} : e' : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma$$

Consequently, we also know that $j = h$ and $i = k$.
Furthermore, rule iCtx-MEnv also tells us that **unambig**$(\forall \overline{a}_j.\overline{Q}_i \Rightarrow Q')$. The definition of unambiguity thus gives us $\overline{a}_j \in \mathbf{fv}(Q')$. This, in combination with Equations 723 and 728 tells us that $\overline{\sigma}_j = \overline{\sigma}'_h$.

Unfolding the definition of the $\mathcal{V}$ relation, reduces the goal to be proven to:

$$\overline{(\Sigma_1 : dv_i, \Sigma_2 : dv'_k) \in \mathcal{V}[\![[\overline{\sigma}_j/\overline{a}_j]Q_i]\!]^{\Gamma_C}_{\bullet}}^{i} \tag{733}$$

$$\Sigma_1; \Gamma_C; \bullet \vdash_d D\, \overline{\sigma}_j\, \overline{dv}_i : Q \tag{734}$$

$$\Sigma_2; \Gamma_C; \bullet \vdash_d D\, \overline{\sigma}_j\, \overline{dv}'_k : Q \tag{735}$$

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto e \in \Sigma_1 \textit{ where } Q = [\overline{\sigma}_j/\overline{a}_j]Q' \tag{736}$$

Goals 734 and 735 are given by the hypothesis. Goal 736 follows directly from Equation 723. Finally, Goal 733 follows by applying the induction hypothesis on Equations 725 and 730.

□

Theorem 25 (Environment Equivalence Preservation). *If* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ *and* $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$ *then* $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$.

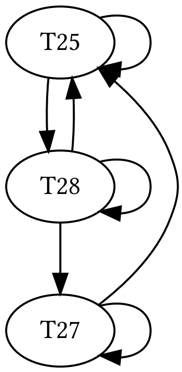

Fig. 16. Dependency graph for Theorems 25, 27 and 28

Proof. By structural induction on P and mutually proven with Theorems 27 and 28 (see Figure 16). Note that at the dependency between Theorem 25 and 28, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Because of this, the size of P is strictly decreasing in every cycle. Consequently, the induction remains well-founded.

$P = \bullet$

By case analysis on the 1st and 2nd hypothesis:

$$\Sigma_1 = \Sigma_2 = \bullet$$

The goal follows from ctxLog-empty.

$P = P', (D : C).m \mapsto \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}_h : e$

By case analysis on the 1st and 2nd hypothesis (sCtx-pgmInst):

$$\Sigma_1 = \Sigma'_1, (D : C).m \mapsto \Lambda\overline{b}_j.\lambda\overline{\delta}_i : \overline{Q}_i.\Lambda\overline{a}_k.\lambda\overline{\delta}_h : [\sigma/a]\overline{Q}_h.e_1$$

$$\Sigma_2 = \Sigma'_2, (D : C').m \mapsto \Lambda\overline{b}_j.\lambda\overline{\delta}_i : \overline{Q}'_i.\Lambda\overline{a}_k.\lambda\overline{\delta}_h : [\sigma/a]\overline{Q}'_h.e_2$$

$$P'; \Gamma_C; \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}_h \vdash^M_{tm} e \Leftarrow [\tau/a]\tau' \rightsquigarrow e_1 \tag{737}$$

$$P'; \Gamma_C; \bullet, \overline{b}_j, \overline{\delta}_i : \overline{Q}_i, \overline{a}_k, \overline{\delta}_h : [\tau/a]\overline{Q}_h \vdash^M_{tm} e \Leftarrow [\tau/a]\tau' \rightsquigarrow e_2 \tag{738}$$

$$\vdash^M_{ctx} P'; \Gamma_C; \Gamma \rightsquigarrow \Sigma'_1; \Gamma_C; \Gamma \tag{739}$$

$$\vdash^M_{ctx} P'; \Gamma_C; \Gamma \rightsquigarrow \Sigma'_2; \Gamma_C; \Gamma \tag{740}$$

Since the elaboration from $\lambda_{\mathbf{TC}}$ constraints to $F_{\mathbf{D}}$ constraints is entirely deterministic (Lemma 10), we know that $C' = C$, $\overline{Q}'_i = \overline{Q}_i$, $\overline{Q}'_h = \overline{Q}_h$ and consequently that $[\sigma/a]\overline{Q}'_h = [\sigma/a]\overline{Q}_h$.
From the induction hypothesis, together with Equations 739 and 740, we get that:

$$\Gamma_C \vdash \Sigma'_1 \simeq_{log} \Sigma'_2 \tag{741}$$

From Expression Coherence Theorem A (Theorem 28), in combination with Equations 737, 738, 739 and 740, we know:

$$\Gamma_C; \Gamma \vdash \Sigma_1' : e_1 \simeq_{log} \Sigma_2' : e_2 : [\sigma/a]\sigma' \tag{742}$$

where $\Gamma_C; \Gamma \vdash_{ty}^{M} [\tau/a]\tau' \rightsquigarrow [\sigma/a]\sigma'$.
The goal follows from ctxLog-cons, together with Equations 741 and 742.

□

Theorem 26 (Contextual Equivalence in F_D Implies Contextual Equivalence in λ_{TC}). *If* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$
and $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ *and* $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$
and $\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma$
and there exists an τ *such that* $\Gamma_C; \Gamma \vdash_{ty}^{M} \tau \rightsquigarrow \sigma$
then $P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$.

Proof. By unfolding the definition of contextual equivalence, the goal becomes:

$$\forall M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1 \tag{743}$$

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2 \tag{744}$$

$$\textit{then } M_1[e_1] \simeq M_2[e_2] \tag{745}$$

We thus assume Equations 743 and 744 and prove Equation 745.
By unfolding the definition of contextual equivalence in the first hypothesis, we get that:

$$\begin{aligned} &\forall M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1 \\ &\forall M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2 \\ &\textit{if } \Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \\ &\textit{then } M_1[e_1] \simeq M_2[e_2] \end{aligned} \tag{746}$$

By applying Lemma 18 to the fourth hypothesis, we know that:

$$\vdash_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \bullet$$

By applying context equivalence (Theorem 17) on Equations 743 and 744, together with this result and hypotheses 2, 3 and 4, we know that:

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1' \tag{747}$$

$$M_1' : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1' \tag{748}$$

$$M : (P; \Gamma_C; \Gamma \Rightarrow \tau) \mapsto (P; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2' \tag{749}$$

$$M_2' : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2' \tag{750}$$

Similarly, by applying Lemma 19 to the first and second hypothesis, we get:

$$\vdash_{ctx}^{M} P; \Gamma_C; \bullet \rightsquigarrow \Sigma_1; \Gamma_C; \bullet$$

$$\vdash_{ctx}^{M} P; \Gamma_C; \bullet \rightsquigarrow \Sigma_1; \Gamma_C; \bullet$$

By applying Theorem 23 to Equations 747 and 749, together with this result, hypotheses 2, 3 and 5, and sTy-bool, we know that:

$$\Sigma_1 : M_1' \simeq_{log} \Sigma_2 : M_2' : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \tag{751}$$

We take $M_1 = M_1'$ and $M_2 = M_2'$. Consequently, since F_D context elaboration is deterministic (Theorem 38), we get that $M_1 = M_1'$ and $M_2 = M_2'$. Goal 745 follows from Equations 746, 748, 750 and 751.

□

M.3 Partial Coherence Theorems

Theorem 27 (Coherence - Dictionaries - Part A). *If* $P; \Gamma_C; \Gamma \vDash^{M} Q \rightsquigarrow d_1$ *and* $P; \Gamma_C; \Gamma \vDash^{M} Q \rightsquigarrow d_2$
and $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ *and* $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$
then $\Gamma_C; \Gamma \vdash \Sigma_1 : d_1 \simeq_{log} \Sigma_2 : d_2 : Q$
where $\Gamma_C; \Gamma \vdash_{Q}^{M} Q \rightsquigarrow Q$.

PROOF. By induction on the first constraint entailment derivation. This theorem is mutually proven with Theorems 25 and 28 (see Figure 16). Note that at the dependency between Theorem 25 and 28, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Because of this, the size of P is strictly decreasing in every cycle. Consequently, the induction remains well-founded.

sENTAIL-INST

$$\frac{Q = [\overline{\tau}_j/\overline{a}_j]Q' \quad \vdash^M_{ctx} P;\Gamma_C;\Gamma \rightsquigarrow \Sigma;\Gamma_C;\Gamma \quad \begin{array}{c}P = P_1,(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto \bullet,\overline{a}_j,\overline{\delta}_i : \overline{Q}_i,\overline{b}_k,\overline{\delta}_h : \overline{Q}_h : e, P_2 \\ \overline{\Gamma_C;\Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \quad \overline{P;\Gamma_C;\Gamma \vDash^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i}^{\,i}\end{array}}{P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow D\,\overline{\sigma}_j\,\overline{d}_i} \text{ sENTAIL-INST}$$

The final step in the second derivation can be either sENTAIL-INST or sENTAIL-LOCAL:

- sENTAIL-INST: This means that:

$$P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow D\,\overline{\sigma}_j\,\overline{d}_i \tag{752}$$

$$P;\Gamma_C;\Gamma \vDash^M Q \rightsquigarrow D'\,\overline{\sigma}'_j\,\overline{d}'_i \tag{753}$$

The goal to be proven is the following:

$$\Gamma_C;\Gamma \vdash \Sigma_1 : D\,\overline{\sigma}_j\,\overline{d}_i \simeq_{log} \Sigma_2 : D'\,\overline{\sigma}'_j\,\overline{d}'_i : Q \tag{754}$$

$$\textit{where } \Gamma_C;\Gamma \vdash^M_Q Q \rightsquigarrow Q \tag{755}$$

By repeated case analysis on the 3rd hypothesis (sCTX-PGMINST), together with the fact that

$$(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q').m \mapsto \bullet,\overline{a}_j,\overline{\delta}_i : \overline{Q}_i,\overline{b}_k,\overline{\delta}_h : \overline{Q}_h : e \in P$$

(1st rule premise (sENTAIL-INST)), we know that:

$$\Gamma_C;\bullet \vdash^M_C \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q' \rightsquigarrow \forall \overline{a}_j.\overline{Q}_i \Rightarrow Q' \tag{756}$$

Consequently, by case analysis (sQ-TC) on this result, we get that:

$$\Gamma_C;\bullet,\overline{a}_j \vdash^M_Q Q' \rightsquigarrow Q' \tag{757}$$

By applying Lemma 1 to this result, in combination with the 4th rule premise (sENTAIL-INST), we get that:

$$\Gamma_C;\bullet \vdash^M_Q [\overline{\tau}_j/\overline{a}_j]Q' \rightsquigarrow [\overline{\sigma}_j/\overline{a}_j]Q' \tag{758}$$

Goal 755 follows directly from Equation 758, since we know that $Q = [\overline{\tau}_j/\overline{a}_j]Q'$ (2nd rule premise (sENTAIL-INST)). By unfolding the definition of logical equivalence in Goal 754, the goal reduces to:

$$(\Sigma_1 : \gamma_1(\phi_1(R(D\,\overline{\sigma}_j\,\overline{d}_i))), \Sigma_2 : \gamma_2(\phi_2(R(D'\,\overline{\sigma}'_j\,\overline{d}'_i)))) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_R \tag{759}$$

for any $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_R$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_R$.
By simplifying the substitutions (note that ϕ only substitutes term variables, which has no impact on the types or dictionaries), Goal 759 reduces to:

$$(\Sigma_1 : D\,R(\overline{\sigma}_j)\,(\gamma_1(\bullet(R(\overline{d}_i)))), \Sigma_2 : D'\,R(\overline{\sigma}'_j)\,(\gamma_2(\bullet(R(\overline{d}'_i))))) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_R \tag{760}$$

By applying preservation Theorem 6 on Equations 752 and 753 respectively, we get:

$$\Sigma_1;\Gamma_C;\Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : Q \tag{761}$$

$$\Sigma_2;\Gamma_C;\Gamma \vdash_d D'\,\overline{\sigma}'_j\,\overline{d}'_i : Q \tag{762}$$

By repeatedly applying substitution lemmas 37, 28 and 29, using R and γ, Equations 761 and 762 simplify to:

$$\Sigma_1;\Gamma_C;\bullet \vdash_d D\,R(\overline{\sigma}_j)\,(\gamma_1(\bullet(R(\overline{d}_i)))) : R(Q) \tag{763}$$

$$\Sigma_2;\Gamma_C;\bullet \vdash_d D'\,R(\overline{\sigma}'_j)\,(\gamma_2(\bullet(R(\overline{d}'_i)))) : R(Q) \tag{764}$$

Furthermore, by applying preservation Theorem 7 on the 3rd and 4th hypothesis, we know that $\vdash_{ctx} \Sigma_1;\Gamma_C;\Gamma$ and $\vdash_{ctx} \Sigma_2;\Gamma_C;\Gamma$. By Theorem 25, we know that $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$. Consequently, by applying Theorem 24 on Equations 763 and 764 we get:

$$(\Sigma_1 : D\,R(\overline{\sigma}_j)\,(\gamma_1(\bullet(R(\overline{d}_i)))), \Sigma_2 : D'\,R(\overline{\sigma}'_j)\,(\gamma_2(\bullet(R(\overline{d}'_i))))) \in \mathcal{V}[\![R(Q)]\!]^{\Gamma_C}_\bullet \tag{765}$$

Goal 760 follows from Equation 765 by Lemma 35.

- sEntail-local: This means that:

$$P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow D\,\overline{\sigma}_j\,\overline{d}_i \tag{766}$$
$$P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow \delta \tag{767}$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : D\,\overline{\sigma}_j\,\overline{d}_i \simeq_{log} \Sigma_2 : \delta : Q \tag{768}$$
$$\textit{where } \Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q \tag{769}$$

Goal 769 follows similarly to Goal 755 in the previous part of this proof.
The 1st rule premise (sEntail-local) tells us that:

$$(\delta : Q) \in \Gamma \tag{770}$$

Unfolding the definition of logical equivalence reduces Goal 768 to:

$$(\Sigma_1 : \gamma_1(\phi_1(R(D\,\overline{\sigma}_j\,\overline{d}_i))), \Sigma_2 : \gamma_2(\phi_2(R(\delta)))) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_R \tag{771}$$

for any $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_R$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_R$.
By simplifying the substitutions (note that ϕ only substitutes term variables, which has no impact on the types or dictionaries), Goal 771 reduces to:

$$(\Sigma_1 : D\,R(\overline{\sigma}_j)\,(\gamma_1(\bullet(R(\overline{d}_i)))), \Sigma_2 : \gamma_2(\phi_2(R(\delta)))) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_R \tag{772}$$

The definition of $\mathcal{H}$, together with Equation 770, combined with the fact that $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ and $\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$, tells us that:

$$\delta \mapsto (dv_1, dv_2) \in \gamma \textit{ where } (\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]^{\Gamma_C}_R \tag{773}$$

We thus know that:

$$\gamma_2(\phi_2(R(\delta))) = dv_2$$

By unfolding the definition of the $\mathcal{V}$ relation in Equation 773, we get:

$$\Sigma_2; \Gamma_C; \bullet \vdash_d dv_2 : R(Q) \tag{774}$$

Preservation Theorem 7, applied to the 3rd and 4th hypothesis, tells us that:

$$\vdash_{ctx} \Sigma_1; \Gamma_C; \Gamma$$
$$\vdash_{ctx} \Sigma_2; \Gamma_C; \Gamma$$

Applying preservation Theorem 6 to Equation 766 results in:

$$\Sigma_1; \Gamma_C; \Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : Q \tag{775}$$

By repeatedly applying substitution lemmas 37, 28 and 29, using R and γ, Equation 775 simplifies to:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d D\,R(\overline{\sigma}_j)\,(\gamma_1(\bullet(R(\overline{d}_i)))) : R(Q) \tag{776}$$

From Theorem 25, we know that $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$. By applying Theorem 24 to Equations 774 and 776 we get:

$$(\Sigma_1 : D\,R(\overline{\sigma}_j)\,(\gamma_1(\bullet(R(\overline{d}_i)))), \Sigma_2 : dv_2) \in \mathcal{V}[\![R(Q)]\!]^{\Gamma_C}_\bullet \tag{777}$$

Goal 772 follows from Equation 777, in combination with Lemma 35.

sEntail-local

$$\frac{(\delta : Q) \in \Gamma \qquad \vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma; \Gamma_C; \Gamma}{P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow \delta}\ \textsc{sEntail-local}$$

The final step in the derivation can be either sEntail-inst or sEntail-local:
- sEntail-inst: This proof case is identical to the 2nd part of the previous case.
- sEntail-local: This means that:

$$P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow \delta$$
$$P; \Gamma_C; \Gamma \vDash^M Q \rightsquigarrow \delta'$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \delta \simeq_{log} \Sigma_2 : \delta' : Q \tag{778}$$
$$\textit{where } \Gamma_C; \Gamma \vdash^M_Q Q \rightsquigarrow Q \tag{779}$$

The rule premise tells us that:

$$(\delta : Q) \in \Gamma \tag{780}$$
$$(\delta' : Q) \in \Gamma \tag{781}$$

Goal 779 follows by repeated case analysis (sCtx-tyEnvD) on the 3rd hypothesis, together with Equation 780. Unfolding the definition of logical equivalence reduces Goal 778 to:

$$(\Sigma_1 : \gamma_1(\phi_1(R(\delta))), \Sigma_2 : \gamma_2(\phi_2(R(\delta')))) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \tag{782}$$

for any $R \in \mathcal{F}[\![\Gamma]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$ and $\gamma \in \mathcal{H}[\![\Gamma]\!]_R^{\Sigma_1,\Sigma_2,\Gamma_C}$.
The definition of $\mathcal{H}$, together with Equations 780 and 781, combined with the fact that $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ and $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$, tells us that:

$$\delta \mapsto (dv_1, dv_2) \in \gamma \textit{ where } (\Sigma_1 : dv_1, \Sigma_2 : dv_2) \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \tag{783}$$
$$\delta' \mapsto (dv_1', dv_2') \in \gamma \textit{ where } (\Sigma_1 : dv_1', \Sigma_2 : dv_2') \in \mathcal{V}[\![Q]\!]_R^{\Gamma_C} \tag{784}$$

We thus know that:

$$\gamma_1(\phi_1(R(\delta))) = dv_1$$
$$\gamma_2(\phi_2(R(\delta'))) = dv_2'$$

From the definition of the $\mathcal{V}$ relation in Equations 783 and 784 it follows that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_d dv_1 : R(Q) \tag{785}$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_d dv_2' : R(Q) \tag{786}$$

Applying preservation Theorem 7 to our hypothesis that $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ and $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$ gives us $\vdash_{ctx} \Sigma_1; \Gamma_C; \Gamma$ and $\vdash_{ctx} \Sigma_2; \Gamma_C; \Gamma$, respectively.
From Theorem 25, we get that $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$. Consequently, by applying Theorem 24 on Equations 785 and 786 we get:

$$(\Sigma_1 : dv_1, \Sigma_2 : dv_2') \in \mathcal{V}[\![R(Q)]\!]_\bullet^{\Gamma_C} \tag{787}$$

Goal 782 follows from Equation 787 by Lemma 35.

□

Theorem 28 (Coherence - Expressions - Part A).

- *If* $P; \Gamma_C; \Gamma \vdash_{tm}^{M} e \Rightarrow \tau \rightsquigarrow e_1$ *and* $P; \Gamma_C; \Gamma \vdash_{tm}^{M} e \Rightarrow \tau \rightsquigarrow e_2$ *and* $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ *and* $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$ *then* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$ *where* $\Gamma_C; \Gamma \vdash_{ty}^{M} \tau \rightsquigarrow \sigma$.
- *If* $P; \Gamma_C; \Gamma \vdash_{tm}^{M} e \Leftarrow \tau \rightsquigarrow e_1$ *and* $P; \Gamma_C; \Gamma \vdash_{tm}^{M} e \Leftarrow \tau \rightsquigarrow e_2$ *and* $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma$ *and* $\vdash_{ctx}^{M} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma$ *then* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$ *where* $\Gamma_C; \Gamma \vdash_{ty}^{M} \tau \rightsquigarrow \sigma$.

Proof. By mutual induction on the first typing derivation. This theorem is mutually proven with Theorems 25 and 27 (see Figure 16). Note that at the dependency between Theorem 25 and 28, the size of P is strictly decreasing, whereas P remains constant at every other dependency. Because of this, the size of P is strictly decreasing in every cycle. Consequently, the induction remains well-founded.
By applying Lemma 14 to the 1st hypothesis, we get that:

$$\Gamma_C; \Gamma \vdash_{ty}^{M} \tau \rightsquigarrow \sigma \tag{788}$$

Part 1

sTm-inf-true $P; \Gamma_C; \Gamma \vdash_{tm}^{M} True \Rightarrow Bool \rightsquigarrow True$

Through case analysis, it is straightforward to see that the final step in the second derivation is sTm-inf-true as well. This means that $e_1 = e_2 = True$. From Theorem 25, we know that $\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2$. The goal follows from reflexivity Theorem 22.

sTm-inf-false $P; \Gamma_C; \Gamma \vdash_{tm}^{M} True \Rightarrow Bool \rightsquigarrow True$

The proof is identical to the sTm-inf-true case.

sTm-inf-let $P; \Gamma_C; \Gamma \vdash^M_{tm} \textbf{let } x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 = e_1 \textbf{ in } e_2 \Rightarrow \tau_2 \rightsquigarrow \textbf{let } x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma = \Lambda \overline{a}_j . \lambda \overline{\delta}_k : \overline{Q}_k . e_1 \textbf{ in } e_2$

Through case analysis, we know that the final step in the second derivation has to be sTm-inf-let as well. This means that:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} \textbf{let } x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 = e_1 \textbf{ in } e_2 \Rightarrow \tau_2 \rightsquigarrow e_3$$
$$\textit{where } e_3 = \textbf{let } x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma = \Lambda \overline{a}_j . \lambda \overline{\delta}_k : \overline{Q}_k . e_1 \textbf{ in } e_2$$
$$P; \Gamma_C; \Gamma \vdash^M_{tm} \textbf{let } x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \tau_1 = e_1 \textbf{ in } e_2 \Rightarrow \tau_2 \rightsquigarrow e_4$$
$$\textit{where } e_4 = \textbf{let } x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma = \Lambda \overline{a}_j . \lambda \overline{\delta}_k : \overline{Q}_k . e'_1 \textbf{ in } e'_2$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_3 \simeq_{log} \Sigma_2 : e_4 : \sigma_2 \textit{ where } \Gamma_C; \Gamma \vdash^M_{ty} \sigma_2 \rightsquigarrow \sigma_2 \tag{789}$$

The rule premise tells us that:

$$P; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e_1 \tag{790}$$
$$P; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau_1 \vdash^M_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e_2 \tag{791}$$
$$P; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k \vdash^M_{tm} e_1 \Leftarrow \tau_1 \rightsquigarrow e'_1 \tag{792}$$
$$P; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau_1 \vdash^M_{tm} e_2 \Rightarrow \tau_2 \rightsquigarrow e'_2 \tag{793}$$
$$\textit{where } \textbf{closure}(\Gamma_C; \overline{Q}_i) = \overline{Q}_k$$

From Lemma 13, together with Equations 790, 791, 792 and 793, and through repeated case analysis on the results to discover the contents of the elaborated environments, we get that:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k$$
$$\vdash^M_{ctx} P; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma$$
$$\vdash^M_{ctx} P; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k$$
$$\vdash^M_{ctx} P; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \tau_1 \rightsquigarrow \Sigma_2; \Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma$$

By applying the induction hypothesis to Equations 791 and 793, we get:

$$\Gamma_C; \Gamma, x : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma \vdash \Sigma_1 : e_2 \simeq_{log} \Sigma_2 : e'_2 : \sigma_2 \tag{794}$$

Furthermore, applying Part 2 of this lemma to Equations 790 and 792 results in:

$$\Gamma_C; \Gamma, \overline{a}_j, \overline{\delta}_k : \overline{Q}_k \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e'_1 : \sigma \tag{795}$$

Applying compatibility Lemma 56, together with Equation 794, reduces Goal 789 to:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \Lambda \overline{a}_j . \lambda \overline{\delta}_k : \overline{Q}_k . e_1 \simeq_{log} \Sigma_2 : \Lambda \overline{a}_j . \lambda \overline{\delta}_k : \overline{Q}_k . e'_1 : \forall \overline{a}_j . \overline{Q}_k \Rightarrow \sigma \tag{796}$$

Combining compatibility Lemma 52 with Equation 795 gives us:

$$\Gamma_C; \Gamma, \overline{a}_j \vdash \Sigma_1 : \lambda \overline{\delta}_k : \overline{Q}_k . e_1 \simeq_{log} \Sigma_2 : \lambda \overline{\delta}_k : \overline{Q}_k . e'_1 : \overline{Q}_k \Rightarrow \sigma \tag{797}$$

Goal 796 follows directly by applying Equation 797 to compatibility Lemma 54.

sTm-inf-ArrE $P; \Gamma_C; \Gamma \vdash^M_{tm} e_1\, e_2 \Rightarrow \tau_2 \rightsquigarrow e_1\, e_2$

Through case analysis, we see that the final step in the second derivation can only be sTm-inf-ArrE. This means that:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_1\, e_2 \Rightarrow \tau_2 \rightsquigarrow e_1\, e_2$$
$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_1\, e_2 \Rightarrow \tau_2 \rightsquigarrow e'_1\, e'_2$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1\, e_2 \simeq_{log} \Sigma_2 : e'_1\, e'_2 : \sigma_2 \textit{ where } \Gamma_C; \Gamma \vdash^M_{ty} \tau_2 \rightsquigarrow \sigma_2 \tag{798}$$

The rule premise tells us that:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_1 \Rightarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow e_1 \quad (799)$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2 \quad (800)$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_1 \Rightarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow e_1' \quad (801)$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e_2 \Leftarrow \tau_1 \rightsquigarrow e_2' \quad (802)$$

Applying the induction hypothesis on Equations 799 and 801 results in:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_1' : \sigma_1 \rightarrow \sigma_2 \textit{ where } \Gamma_C; \Gamma \vdash^M_{ty} \tau_1 \rightarrow \tau_2 \rightsquigarrow \sigma_1 \rightarrow \sigma_2 \quad (803)$$

By applying Part 2 of this lemma to Equations 800 and 802 we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_2 \simeq_{log} \Sigma_2 : e_2' : \sigma_1 \textit{ where } \Gamma_C; \Gamma \vdash^M_{ty} \tau_1 \rightsquigarrow \sigma_1 \quad (804)$$

Goal 798 follows by applying Equations 803 and 804 to compatibility Lemma 51.

sTm-inf-Ann $P; \Gamma_C; \Gamma \vdash^M_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e$

Through case analysis we know that the final step in the second derivation is sTm-inf-Ann. This means that:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e :: \tau \Rightarrow \tau \rightsquigarrow e'$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e \simeq_{log} \Sigma_2 : e' : \sigma \textit{ where } \Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma \quad (805)$$

From the rule premise we know:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e \quad (806)$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e' \quad (807)$$

The goal follows directly from Part 2 of this lemma, applied to Equation 806 and 807.

Part 2

sTm-check-var $P; \Gamma_C; \Gamma \vdash^M_{tm} x \Leftarrow [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow x\,\overline{\sigma}_j\,\overline{d}_i$

Through case analysis, we see that the final step in the second typing derivation can either be sTm-check-var or sTm-check-Inf. However, noting that no matching inference rules exist, we conclude that the final derivation step has to be sTm-check-var. This means that:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} x \Leftarrow [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow x\,\overline{\sigma}_j\,\overline{d}_i$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} x \Leftarrow [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow x\,\overline{\sigma}_j'\,\overline{d}_i'$$

The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : x\,\overline{\sigma}_j\,\overline{d}_i \simeq_{log} \Sigma_2 : x\,\overline{\sigma}_j'\,\overline{d}_i' : \sigma' \textit{ where } \Gamma_C; \Gamma \vdash^M_{ty} [\overline{\tau}_j/\overline{a}_j]\tau \rightsquigarrow \sigma' \quad (808)$$

By inversion on Equation 788, we know that $\sigma' = [\overline{\sigma}_j/\overline{a}_j]\sigma$.
The rule premise tells us that:

$$(x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \tau) \in \Gamma \quad (809)$$

$$\overline{P; \Gamma_C; \Gamma \models^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i}^{\,i} \quad (810)$$

$$\overline{\Gamma_C; \Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j}^{\,j} \quad (811)$$

$$\overline{P; \Gamma_C; \Gamma \models^M [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow d_i'}^{\,i} \quad (812)$$

$$\overline{\Gamma_C; \Gamma \vdash^M_{ty} \tau_j \rightsquigarrow \sigma_j'}^{\,j} \quad (813)$$

Since type elaboration is completely deterministic (Lemma 9), we know that $\overline{\sigma}_j = \overline{\sigma}_j'$.
From Lemma 15, combined with the 3rd hypothesis and Equation 809, we know that:

$$(x : \forall \overline{a}_j.\overline{Q}_i \Rightarrow \sigma) \in \Gamma \quad (814)$$

By applying Theorem 7 to the 3rd and 4th hypothesis, we get:

$$\vdash_{ctx} \Sigma_1; \Gamma_C; \Gamma \quad (815)$$

$$\vdash_{ctx} \Sigma_2; \Gamma_C; \Gamma \quad (816)$$

Applying Equations 814, 815 and 816 to iTm-var, results in:

$$\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma \quad (817)$$

$$\Sigma_2; \Gamma_C; \Gamma \vdash_{tm} x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma \quad (818)$$

From Theorem 25, we know that:

$$\Gamma_C \vdash \Sigma_1 \simeq_{log} \Sigma_2 \quad (819)$$

From reflexivity Theorem 22, applied on Equations 817, 818 and 819, we know that:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : x \simeq_{log} \Sigma_2 : x : \forall \overline{a}_j . \overline{Q}_i \Rightarrow \sigma \quad (820)$$

From repeated case analysis on the 3rd hypothesis (sCtx-pgmInst), we get:

$$\vdash^{M}_{ctx} \bullet; \Gamma_C; \Gamma \rightsquigarrow \bullet; \Gamma_C; \Gamma \quad (821)$$

By applying Theorem 5 to Equations 811 and 821, we know that:

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j}^{j} \quad (822)$$

Applying compatibility Lemma 55 j times to Equations 820 and 822 results in:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : x\,\overline{\sigma}_j \simeq_{log} \Sigma_2 : x\,\overline{\sigma}_j : [\overline{\sigma}_j/\overline{a}_j]\overline{Q}_i \Rightarrow [\overline{\sigma}_j/\overline{a}_j]\sigma \quad (823)$$

Applying Theorem 27 to Equations 810 and 812 gives us:

$$\overline{\Gamma_C; \Gamma \vdash \Sigma_1 : d_i \simeq_{log} \Sigma_2 : d'_i : Q'_i}^{i} \quad (824)$$

$$\textit{where}\ \overline{\Gamma_C; \Gamma \vdash^{M}_{Q} [\overline{\tau}_j/\overline{a}_j]Q_i \rightsquigarrow Q'_i}^{i} \quad (825)$$

By inversion on Equation 825, we know that $\overline{Q'_i = [\overline{\sigma}_j/\overline{a}_j]Q_i}^{i}$.
Goal 808 follows by repeatedly applying compatibility Lemma 53 to Equation 823, combined with Equation 824.

sTm-check-meth $P; \Gamma_C; \Gamma \vdash^{M}_{tm} m \Leftarrow [\overline{\tau}_j/\overline{a}_j][\tau/a]\tau' \rightsquigarrow d.m\,\overline{\sigma}_j\,\overline{d}_i$

The proof is similar to the sTm-check-var case.

sTm-check-ArrI $P; \Gamma_C; \Gamma \vdash^{M}_{tm} \lambda x.e \Leftarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x : \sigma_1.e$

Through case analysis, it is straightforward to note that the final step in the second typing derivation can be either sTm-check-ArrI or sTm-check-Inf. In the latter case however, no matching inference rules exist. The sTm-check-ArrI case is the only remaining possibility. This means that:

$$P; \Gamma_C; \Gamma \vdash^{M}_{tm} \lambda x.e \Leftarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x : \sigma_1.e$$

$$P; \Gamma_C; \Gamma \vdash^{M}_{tm} \lambda x.e \Leftarrow \tau_1 \rightarrow \tau_2 \rightsquigarrow \lambda x : \sigma'_1.e'$$

From the 3rd rule premise we know that:

$$\Gamma_C; \Gamma \vdash^{M}_{ty} \tau_1 \rightsquigarrow \sigma_1$$

$$\Gamma_C; \Gamma \vdash^{M}_{ty} \tau_1 \rightsquigarrow \sigma'_1$$

Since type elaboration is entirely deterministic (Lemma 9), it is straightforward to note that $\sigma_1 = \sigma'_1$.
The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : \lambda x : \sigma_1.e \simeq_{log} \Sigma_2 : \lambda x : \sigma_1.e' : \sigma' \ \textit{where}\ \Gamma_C; \Gamma \vdash^{M}_{ty} \tau_1 \rightarrow \tau_2 \rightsquigarrow \sigma' \quad (826)$$

By inversion on Equation 788, we know that $\sigma' = \sigma_1 \rightarrow \sigma_2$.
From the rule premise we know:

$$P; \Gamma_C; \Gamma, x : \tau_1 \vdash^{M}_{tm} e \Leftarrow \tau_2 \rightsquigarrow e \quad (827)$$

$$P; \Gamma_C; \Gamma, x : \tau_1 \vdash^{M}_{tm} e \Leftarrow \tau_2 \rightsquigarrow e' \quad (828)$$

By applying Equation 827 to Lemma 13, and through repeated case analysis on the result to discover the contents of the elaborated environments, we know that:

$$\vdash^{M}_{ctx} P; \Gamma_C; \Gamma, x : \tau_1 \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma, x : \sigma_1$$

Applying the induction hypothesis on Equations 827 and 828 results in:

$$\Gamma_C; \Gamma, x : \sigma_1 \vdash \Sigma_1 : e \simeq_{log} \Sigma_2 : e' : \sigma_2 \textit{ where } \Gamma_C; \Gamma, x : \tau_1 \vdash^{M}_{ty} \tau_2 \rightsquigarrow \sigma_2 \quad (829)$$

Goal 826 follows directly from compatibility Lemma 50.

sTm-check-Inf $P; \Gamma_C; \Gamma \vdash^{M}_{tm} e \Leftarrow \tau \rightsquigarrow e$

Through case analysis, we note that the final step in the second typing derivation can either be sTm-check-var, sTm-check-meth, sTm-check-ArrI or sTm-check-Inf. In the first 3 cases, the proof is symmetrical to the corresponding proof cases described above. We proceed with the last case:
The goal to be proven is the following:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e \simeq_{log} \Sigma_2 : e' : \sigma \textit{ where } \Gamma_C; \Gamma \vdash^{M}_{ty} \tau \rightsquigarrow \sigma \quad (830)$$

where we know that:

$$P; \Gamma_C; \Gamma \vdash^{M}_{tm} e \Leftarrow \tau \rightsquigarrow e \quad (831)$$

$$P; \Gamma_C; \Gamma \vdash^{M}_{tm} e \Leftarrow \tau \rightsquigarrow e' \quad (832)$$

The rule premise tells us that:

$$P; \Gamma_C; \Gamma \vdash^{M}_{tm} e \Rightarrow \tau \rightsquigarrow e \quad (833)$$

$$P; \Gamma_C; \Gamma \vdash^{M}_{tm} e \Rightarrow \tau \rightsquigarrow e' \quad (834)$$

The goal follows directly by applying Part 1 of this lemma to Equations 833 and 834. Part 1 can be applied on e, even though the term size did not decrease, because inference is defined to be smaller than type checking in our proof by induction.

□

Theorem 29 (Coherence - Expressions - Part B).
If $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma$ *then* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$.

Proof. By unfolding the definition of contextual equivalence, the goal becomes:

$$\forall M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \bullet \Rightarrow Bool)$$

$$\forall M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool)$$

$$\textit{if } \Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \quad (835)$$

$$\textit{then } \Sigma_1 : M_1[e_1] \simeq \Sigma_2 : M_2[e_2] \quad (836)$$

We select any M_1 and M_2 such that Equation 835 holds, and thus need to prove Goal 836.
From the congruence Theorem 18 and the 1[st] hypothesis, we know that:

$$\Gamma_C; \bullet \vdash \Sigma_1 : M_1[e_1] \simeq_{log} \Sigma_2 : M_2[e_2] : Bool$$

By applying the definition of logical equivalence, we get:

$$(\Sigma_1 : \gamma_1(\phi_1(R(M_1[e_1]))), \Sigma_2 : \gamma_2(\phi_2(R(M_2[e_2])))) \in \mathcal{E}[\![Bool]\!]^{\Gamma_C}_{R} \quad (837)$$

for any $R \in \mathcal{F}[\![\bullet]\!]^{\Gamma_C}$, $\phi \in \mathcal{G}[\![\bullet]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{R}$ and $\gamma \in \mathcal{H}[\![\bullet]\!]^{\Sigma_1,\Sigma_2,\Gamma_C}_{R}$.
However, from the definition of $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$, it follows that $R = \bullet$, $\phi = \bullet$ and $\gamma = \bullet$.
Equation 837 thus simplifies to:

$$(\Sigma_1 : M_1[e_1], \Sigma_2 : M_2[e_2]) \in \mathcal{E}[\![Bool]\!]^{\Gamma_C}_{\bullet}$$

Unfolding the definition of the $\mathcal{E}$ relation, tells us that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} M[e_1] : Bool$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} M[e_2] : Bool$$
$$\exists v_1, v_2 : \Sigma_1 \vdash M[e_1] \longrightarrow^* v_1$$
$$\land\ \Sigma_2 \vdash M[e_2] \longrightarrow^* v_2$$
$$\land\ (\Sigma_1 : v_1, \Sigma_2 : v_2) \in \mathcal{V}[\![Bool]\!]_{\bullet}^{\Gamma_C}$$

From the definition of $\mathcal{V}$, we know that either $v_1 = v_2 = True$ or $v_1 = v_2 = False$. The goal follows immediately.

□

Theorem 30 (Coherence - Expressions - Part C). *If* $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$
and $\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma \rightsquigarrow e_1$ *and* $\Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma \rightsquigarrow e_2$
and $\Gamma_C; \Gamma \rightsquigarrow \Gamma$
then $\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma$.

Proof. By unfolding the definition of contextual equivalence, the goal becomes:

$$\forall M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_1; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1$$
$$\forall M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2$$
$$\textit{if}\ \Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \quad (838)$$
$$\textit{then}\ M_1[e_1] \simeq M_2[e_2] \quad (839)$$

We select any M_1 and M_2 such that Equation 838 holds, and thus need to prove Goal 839.
By unfolding the definition of contextual equivalence in the 1st hypothesis, we get that:

$$\forall M_1 : (\Sigma_1; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_1$$
$$\forall M_2 : (\Sigma_2; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma_2; \Gamma_C; \bullet \Rightarrow Bool) \rightsquigarrow M_2$$
$$\textit{if}\ \Sigma_1 : M_1 \simeq_{log} \Sigma_2 : M_2 : (\Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Gamma_C; \bullet \Rightarrow Bool) \quad (840)$$
$$\textit{then}\ \Sigma_1 : M_1[e_1] \simeq \Sigma_2 : M_2[e_2] \quad (841)$$

By applying Theorem 19 to M_1 and M_2, together with the 2nd and 3rd hypothesis, we get:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} M_1[e_1] : Bool \rightsquigarrow M_1[e_1]$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} M_2[e_2] : Bool \rightsquigarrow M_2[e_2]$$

From the definition of kleene equivalence, Equation 841 reduces to:

$$\exists v : \Sigma_1 \vdash M_1[e_1] \longrightarrow^* v \land \Sigma_2 \vdash M_2[e_2] \longrightarrow^* v$$

Finally, Lemma 40 applied to these results, tells us that:

$$\Sigma_1; \Gamma_C; \bullet \vdash_{tm} v : Bool \rightsquigarrow v_1$$
$$\Sigma_2; \Gamma_C; \bullet \vdash_{tm} v : Bool \rightsquigarrow v_2$$
$$M_1[e_1] \longrightarrow^* v_1$$
$$M_2[e_2] \longrightarrow^* v_2$$

Goal 839 follows from the definition of kleene equivalence since either $v_1 = v_2 = True$ or $v_1 = v_2 = False$.

□

M.4 Main Coherence Theorems

Theorem 31 (Coherence). *If* $\bullet; \bullet \vdash_{pgm} pgm : \tau; P_1; \Gamma_{C1} \rightsquigarrow e_1$ *and* $\bullet; \bullet \vdash_{pgm} pgm : \tau; P_2; \Gamma_{C2} \rightsquigarrow e_2$ *then* $\Gamma_{C1} = \Gamma_{C2}$, $P_1 = P_2$ *and* $P_1; \Gamma_{C1}; \bullet \vdash e_1 \simeq_{ctx} e_2 : \tau$.

Proof. Since we know from sCtxT-empty that

$$\vdash_{ctx} \bullet; \bullet; \bullet \rightsquigarrow \bullet$$

the goal follows directly from the Program Coherence Theorem (Theorem 33).

□

Theorem 32 (Coherence - Expressions).

- *If* $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e_1$ *and* $P; \Gamma_C; \Gamma \vdash_{tm} e \Rightarrow \tau \rightsquigarrow e_2$ *then* $P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$.
- *If* $P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e_1$ *and* $P; \Gamma_C; \Gamma \vdash_{tm} e \Leftarrow \tau \rightsquigarrow e_2$ *then* $P; \Gamma_C; \Gamma \vdash e_1 \simeq_{ctx} e_2 : \tau$.

Proof. By applying Lemma 12 to the 1st hypothesis, we know that:

$$\vdash_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Gamma \quad (842)$$

Part 1 From environment equivalence (Theorem 13), we get that:

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_1; \Gamma_C; \Gamma \quad (843)$$

$$\vdash^M_{ctx} P; \Gamma_C; \Gamma \rightsquigarrow \Sigma_2; \Gamma_C{}'; \Gamma' \quad (844)$$

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma \quad (845)$$

Since class and typing environment elaboration is entirely deterministic (Lemma 11), it is easy to see that $\Gamma_C{}' = \Gamma_C$ and $\Gamma' = \Gamma$.

We know from expression equivalence (Theorem 16) that:

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e_1 \quad (846)$$

$$\Sigma_1; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1 \quad (847)$$

$$P; \Gamma_C; \Gamma \vdash^M_{tm} e \Leftarrow \tau \rightsquigarrow e_2 \quad (848)$$

$$\Sigma_2; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2 \quad (849)$$

$$\textit{where } \Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma_1 \quad (850)$$

$$\textit{and } \Gamma_C; \Gamma \vdash^M_{ty} \tau \rightsquigarrow \sigma_2$$

Since type elaboration is entirely deterministic (Lemma 9), it is easy to see that $\sigma = \sigma_1 = \sigma_2$.

By applying Expression Coherence Theorem A (Theorem 28) to Equations 843, 844, 846 and 848, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{log} \Sigma_2 : e_2 : \sigma \quad (851)$$

By applying Expression Coherence Theorem B (Theorem 29) to Equation 851, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma \quad (852)$$

By applying Expression Coherence Theorem C (Theorem 30) to Equations 845, 847, 849 and 852, we get:

$$\Gamma_C; \Gamma \vdash \Sigma_1 : e_1 \simeq_{ctx} \Sigma_2 : e_2 : \sigma \quad (853)$$

The goal follows directly from Theorem 26, together with Equations 853, 842, 843, 844 and 850.

Part 2 Similar to Part 1.

□

Theorem 33 (Coherence - Programs).
If $P; \Gamma_C \vdash_{pgm} pgm : \tau; P_1; \Gamma_{C1} \rightsquigarrow e_1$,
$P; \Gamma_C \vdash_{pgm} pgm : \tau; P_2; \Gamma_{C2} \rightsquigarrow e_2$,
$\vdash_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \bullet$,
then $\Gamma_{C1} = \Gamma_{C2}$, $P_1 = P_2$
and $P, P_1; \Gamma_C, \Gamma_{C1}; \bullet \vdash e_1 \simeq_{ctx} e_2 : \tau$.

Proof. By structural induction on pgm.

$pgm = cls; pgm'$

By case analysis on the program typing derivations (sPgmT-cls):

$$\Gamma_C \vdash_{cls} cls : \Gamma_{C1}' \quad (854)$$
$$\Gamma_C \vdash_{cls} cls : \Gamma_{C2}' \quad (855)$$
$$P; \Gamma_C, \Gamma_{C1}' \vdash_{pgm} pgm' : \tau; P_1; \Gamma_{C1}'' \rightsquigarrow e_1$$
$$P; \Gamma_C, \Gamma_{C2}' \vdash_{pgm} pgm' : \tau; P_2; \Gamma_{C2}'' \rightsquigarrow e_2$$
$$\Gamma_{C1} = \Gamma_{C1}', \Gamma_{C1}''$$
$$\Gamma_{C2} = \Gamma_{C2}', \Gamma_{C2}''$$

Since class typing is entirely deterministic, we know that $\Gamma_{C1}' = \Gamma_{C2}'$. From Theorem 3, in combination with Equations 854 and 855, we know that:

$$\vdash_{ctx} P; \Gamma_C, \Gamma_{C1}'; \bullet \rightsquigarrow \bullet$$
$$\vdash_{ctx} P; \Gamma_C, \Gamma_{C2}'; \bullet \rightsquigarrow \bullet$$

The goal follows from the induction hypothesis.

$pgm = inst; pgm'$

By case analysis on the program typing derivations (sPgmT-inst):

$$P; \Gamma_C \vdash_{inst} inst : P_{11} \quad (856)$$
$$P; \Gamma_C \vdash_{inst} inst : P_{21} \quad (857)$$
$$P, P_{11}; \Gamma_C \vdash_{pgm} pgm' : \tau; P_{12}; \Gamma_{C1} \rightsquigarrow e_1 \quad (858)$$
$$P, P_{21}; \Gamma_C \vdash_{pgm} pgm' : \tau; P_{22}; \Gamma_{C2} \rightsquigarrow e_2 \quad (859)$$
$$P_1 = P_{11}, P_{12}$$
$$P_2 = P_{21}, P_{22}$$

The goal to be proven is the following:

$$\Gamma_{C1} = \Gamma_{C2} \quad (860)$$
$$P_{11}, P_{12} = P_{21}, P_{22} \quad (861)$$
$$P, P_{11}, P_{12}; \Gamma_C, \Gamma_{C1}; \bullet \vdash e_1 \simeq_{ctx} e_2 : \tau \quad (862)$$

By case analysis on Equations 856 and 857 (sInstT-inst), we know that:

$$inst = \textbf{instance } \overline{Q}_p \Rightarrow TC\,\tau \textbf{ where } \{m = e\}$$
$$(m : \overline{Q}_i' \Rightarrow TC\ a : \forall \overline{a}_j.\overline{Q}_h' \Rightarrow \tau_1) \in \Gamma_C \quad (863)$$
$$\overline{b}_k = \textbf{fv}(\tau')$$
$$\textbf{closure}(\Gamma_C; \overline{Q}_p) = \overline{Q}_{q1}$$
$$\textbf{closure}(\Gamma_C; \overline{Q}_p) = \overline{Q}_{q2}$$
$$\textbf{unambig}(\forall \overline{b}_k.\overline{Q}_{q1} \Rightarrow TC\,\tau') \quad (864)$$
$$\textbf{unambig}(\forall \overline{b}_k.\overline{Q}_{q2} \Rightarrow TC\,\tau')$$
$$\overline{P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_{q1} : \overline{Q}_{q1} \models [\tau'/a]Q_i' \rightsquigarrow e_{1\,i}}^{\,i}$$
$$\overline{P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_{q2} : \overline{Q}_{q2} \models [\tau'/a]Q_i' \rightsquigarrow e_{2\,i}}^{\,i}$$
$$D \textbf{ fresh} \quad (865)$$
$$\overline{\delta}_{q1} \textbf{ fresh}$$
$$\overline{\delta}_{q2} \textbf{ fresh}$$
$$\overline{\delta}_h' \textbf{ fresh}$$

$$P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_{q1} : \overline{Q}_{q1}, \overline{a}_j, \overline{\delta}'_h : [\tau'/a]\overline{Q}'_h \vdash_{tm} e \Leftarrow [\tau'/a]\tau_1 \rightsquigarrow e'_1 \quad (866)$$

$$P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_{q2} : \overline{Q}_{q2}, \overline{a}_j, \overline{\delta}'_h : [\tau'/a]\overline{Q}'_h \vdash_{tm} e \Leftarrow [\tau'/a]\tau_1 \rightsquigarrow e'_2 \quad (867)$$

$$P_{11} = (D : \forall \overline{b}_k.\overline{Q}_{q1} \Rightarrow TC\,\tau').m \mapsto \bullet, \overline{b}_k, \overline{\delta}_{q1} : \overline{Q}_{q1}, \overline{a}_j, \overline{\delta}'_h : [\tau'/a]\overline{Q}'_h : e_1 \quad (868)$$

$$P_{21} = (D : \forall \overline{b}_k.\overline{Q}_{q2} \Rightarrow TC\,\tau').m \mapsto \bullet, \overline{b}_k, \overline{\delta}_{q2} : \overline{Q}_{q2}, \overline{a}_j, \overline{\delta}'_h : [\tau'/a]\overline{Q}'_h : e_2 \quad (869)$$

$$\Gamma_C; \bullet, \overline{b}_k \vdash_{ty} \tau' \rightsquigarrow \sigma' \quad (870)$$

$$\overline{\Gamma_C; \bullet, \overline{b}_k \vdash_Q Q_{q1} \rightsquigarrow \sigma_{q1}}^{q} \quad (871)$$

$$\overline{\Gamma_C; \bullet, \overline{b}_k \vdash_Q Q_{q2} \rightsquigarrow \sigma_{q2}}^{q} \quad (872)$$

$$(D' : \forall \overline{b}'_s.\overline{Q}'_n \Rightarrow TC\,\tau_2).m' \mapsto \Gamma' : e' \notin P \text{ where } [\overline{\tau}'_s/\overline{b}'_s]\tau_2 = [\overline{\tau}'_k/\overline{b}_k]\tau' \quad (873)$$

Note that since instance typing is entirely deterministic, $P_{11} = P_{21}$. Similarly, since the closure over the superclass relation is deterministic, we know that $\overline{Q}_{q1} = \overline{Q}_{q2}$. Note that we assume that the fresh variables are identical in both program typing derivations.
We can derive from sCtxT-pgmInst that

$$\vdash_{ctx} P, P_{11}; \Gamma_C; \bullet \rightsquigarrow \bullet \quad (874)$$

$$\vdash_{ctx} P, P_{21}; \Gamma_C; \bullet \rightsquigarrow \bullet \quad (875)$$

assuming we can show that:

$$\textbf{unambig}(\forall \overline{b}_k.\overline{Q}_{q1} \Rightarrow TC\,\tau') \quad (876)$$

$$\Gamma_C; \bullet \vdash_C \forall \overline{b}_k.\overline{Q}_{q1} \Rightarrow TC\,\tau' \rightsquigarrow \forall \overline{b}_k.\overline{\sigma}_{q1} \rightarrow [\sigma'/a]\{m : \forall \overline{a}_j.\overline{\sigma}'_h \rightarrow \sigma_1\} \quad (877)$$

$$(m : \overline{Q}'_i \Rightarrow TC\,a : \forall \overline{a}_j.\overline{Q}'_h \Rightarrow \tau_1) \in \Gamma_C \quad (878)$$

$$\Gamma_C; \bullet, a \vdash_{ty} \forall \overline{a}_j.\overline{Q}'_h \Rightarrow \tau_1 \rightsquigarrow \forall \overline{a}_j.\overline{\sigma}'_h \rightarrow \sigma_1 \quad (879)$$

$$\Gamma_C; \bullet, \overline{b}_k \vdash_{ty} \tau' \rightsquigarrow \sigma' \quad (880)$$

$$P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_{q1} : \overline{Q}_{q1}, \overline{a}_j, \overline{\delta}'_h : [\tau'/a]\overline{Q}'_h \vdash_{tm} e \Leftarrow [\tau'/a]\tau_1 \rightsquigarrow e'_1 \quad (881)$$

$$P; \Gamma_C; \bullet, \overline{b}_k, \overline{\delta}_{q2} : \overline{Q}_{q2}, \overline{a}_j, \overline{\delta}'_h : [\tau'/a]\overline{Q}'_h \vdash_{tm} e \Leftarrow [\tau'/a]\tau_1 \rightsquigarrow e'_2 \quad (882)$$

$$D \notin \textbf{dom}(P) \quad (883)$$

$$(D' : \forall \overline{b}'_k.\overline{Q}''_h \Rightarrow TC\,\tau'').m' \mapsto \Gamma' : e' \notin P \text{ where } [\overline{\tau}_k/\overline{b}_k]\tau' = [\overline{\tau}'_k/\overline{b}'_k]\tau'' \quad (884)$$

$$\vdash_{ctx} P; \Gamma_C; \bullet \rightsquigarrow \bullet \quad (885)$$

Goals 876, 878, 880, 881, 882, 883 and 884 follow directly from Equations 864, 863, 870, 866, 867, 865 and 873 respectively. Goal 885 follows directly from the 3rd hypothesis. Goal 879 follows by applying case analysis on Equation 885 (sCtxT-clsEnv), in combination with Equation 878. Goal 877 follows from sCT-Abs and sQT-TC, in combination with Equations 878, 879, 870 and 871.
Finally, Goals 860, 861 and 862 follow by applying the induction hypothesis on Equations 858 and 859, in combination with Equations 874 and 875.

$\boxed{pgm = e}$

The goal follows directly from coherence Theorem 32.

□

N F_D-TO-$F_{\{\}}$ THEOREMS

N.1 Lemmas

Lemma 58 (Dictionary Elaboration Uniqueness).
If $\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_1$ *and* $\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_2$*, then* $\sigma_1 = \sigma_2$.

N.1.1 *Determinism / Uniqueness.*

Proof. By straightforward induction on the well-formedness derivation. □

Lemma 59 (Type Elaboration Uniqueness).
If $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma_1$ *and* $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma_2$*, then* $\sigma_1 = \sigma_2$.

Proof. By straightforward induction on the well-formedness derivation. □

Lemma 60 (Context Elaboration Uniqueness).
If $\Gamma_C; \Gamma \rightsquigarrow \Gamma_1$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma_2$ *then* $\Gamma_1 = \Gamma_2$.

Proof. By straightforward induction on the well-formedness derivation. □

Lemma 61 (Determinism of Evaluation).
If $e \longrightarrow e_1$ *and* $e \longrightarrow e_2$ *then* $e_1 = e_2$.

Proof. By straightforward induction on both evaluation derivations. □

Lemma 62 (Dictionary Variable Elaboration Soundness).
If $\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $\Gamma \vdash_{ty} \sigma$.

N.1.2 *Soundness.*

Proof. By straightforward induction on the dictionary typing derivation. □

Lemma 63 (Type Elaboration Soundness).
If $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$ *and* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *then* $\Gamma \vdash_{ty} \sigma$.

Proof. By straightforward induction on the type well-formedness derivation. □

Lemma 64 (Term Variable Elaboration Soundness).
If $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$ *and* $(x : \sigma) \in \Gamma$*, then there are unique* Γ *and* σ *such that* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$ *and* $(x : \sigma) \in \Gamma$.

Proof. By straightforward induction on the environment well-formedness derivation. □

Lemma 65 (Dictionary Variable in Environment Elaboration Soundness).
If $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$ *and* $(\delta : TC\,\sigma) \in \Gamma$*, then there are unique* Γ *and* σ *such that* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\vdash_{ctx} \Gamma$ *and* $\Gamma_C; \Gamma \vdash_Q TC\,\sigma \rightsquigarrow \sigma$ *and* $(\delta : \sigma) \in \Gamma$.

Proof. By straightforward induction on the environment well-formedness derivation. ◻

Lemma 66 (Environment Elaboration Soundness).
If $\vdash_{ctx} \Sigma; \Gamma_C; \Gamma$*, then there is a unique* Γ *such that* $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ *and* $\vdash_{ctx} \Gamma$*.*

Proof. By straightforward induction on the environment well-formedness derivation. ◻

Lemma 67 (Canonical Forms for Functions).
If $\Gamma \vdash_{tm} v : \sigma_1 \rightarrow \sigma_2$ *for some value* v*, then* v *is of the form* $\lambda x : \sigma_1.e$*, for some* x *and* e*.*

N.1.3 Canonical Forms Lemmas.

Proof. By straightforward induction on the typing derivation. ◻

Lemma 68 (Canonical Forms for Type Abstractions).
If $\Gamma \vdash_{tm} v : \forall a.\sigma$ *for some value* v*, then* v *is of the form* $\Lambda a.e$*, for some* e*.*

Proof. By straightforward induction on the typing derivation. ◻

Lemma 69 (Distribution of tEval-app).
If $\bullet \vdash_{tm} e : \sigma_1 \rightarrow \sigma_2$ *and* $\bullet \vdash_{tm} e' : \sigma_1 \rightarrow \sigma_2$ *and* $\bullet \vdash_{tm} e_1 : \sigma_1$ *and* $e \longrightarrow^* e'$*, then* $e\, e_1 \longrightarrow^* e'\, e_1$*.*

N.1.4 Evaluation Lemmas.

Proof. The goal follows from Canonical Forms Lemma 67, together with the well-known strong normalization of System F with records. ◻

Lemma 70 (Distribution of tEval-Tapp).
If $\bullet \vdash_{tm} e : \forall a.\sigma'$ *and* $\bullet \vdash_{tm} e' : \forall a.\sigma'$ *and* $\bullet \vdash_{ty} \sigma$ *and* $e \longrightarrow^* e'$*, then* $e\, \sigma \longrightarrow^* e'\, \sigma$*.*

Proof. The goal follows from Canonical Forms Lemma 68, together with the well-known strong normalization of System F with records. ◻

N.2 Soundness

Theorem 34 (Term Elaboration Soundness).

If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e$ (886)

then, there are unique Γ *and* σ *such that*

$\Gamma_C; \Gamma \rightsquigarrow \Gamma$ (887)

and $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$ (888)

and $\Gamma \vdash_{tm} e : \sigma$ (889)

Proof. This theorem is proved mutually with Theorem 35. The proof follows structural induction on Hypothesis 886 of the theorem.

Case iTm-true
$$\frac{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} True : Bool \rightsquigarrow True}\ \text{iTm-true}$$

We need to show that there are unique Γ and σ such that $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ and $\Gamma_C; \Gamma \vdash_{ty} Bool \rightsquigarrow \sigma$ and $\Gamma \vdash_{tm} True : \sigma$. Obviously, σ can only be equal to $Bool$. By Lemma 66 applied on the premise of rule iTm-true, there is a unique Γ such that $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ and $\vdash_{ctx} \Gamma$. We use the latter result to instantiate rule tTm-True, which concludes with $\Gamma \vdash_{tm} True : Bool$.

Case iTm-false

$$\frac{\vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} False : Bool \rightsquigarrow False}\ \text{iTm-false}$$

Similar to case iTm-true.

Case iTm-var

$$\frac{(x : \sigma) \in \Gamma \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} x : \sigma \rightsquigarrow x}\ \text{iTm-var}$$

By Lemma 66, there is a unique Γ such that

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma$$
$$\text{and } \vdash_{ctx} \Gamma \tag{890}$$

With Lemma 64 applied on the two premises of rule iTm-var, we also find a unique σ such that $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$ and $(x : \sigma) \in \Gamma$. Then, by rule tTm-Var applied on the latter result and on Equation 890, we reach the goal.

Case iTm-let

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1 \qquad \Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2 \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2 : \sigma_2 \rightsquigarrow \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2}\ \text{iTm-let}$$

The induction hypothesis for the first premise of rule iTm-let is

$$\text{There are unique } \Gamma,\ \sigma_1,\ \text{such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{891}$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \tag{892}$$
$$\text{and } \Gamma \vdash_{tm} e_1 : \sigma_1 \tag{893}$$

The induction hypothesis for the second premise of rule iTm-let is

$$\text{There are unique } \Gamma',\ \sigma_2,\ \text{such}$$
$$\text{that } \Gamma_C; \Gamma, x : \sigma_1 \rightsquigarrow \Gamma' \tag{894}$$
$$\text{and } \Gamma_C; \Gamma, x : \sigma_1 \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2 \tag{895}$$
$$\text{and } \Gamma' \vdash_{tm} e_2 : \sigma_2 \tag{896}$$

By inversion on Equation 894, there are Γ_0 and σ_1' such that

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma_0 \tag{897}$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1' \tag{898}$$
$$\text{and } \Gamma' = \Gamma_0, x : \sigma_1'$$

By uniqueness (Lemma 60 on Equations 891 and 897 and Lemma 59 on Equations 892 and 898), we get $\Gamma_0 = \Gamma$ and $\sigma_1' = \sigma_1$. This refines Equation 896 into

$$\Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \tag{899}$$

By applying Lemma 63 on Equations 892 and 891, we get

$$\Gamma \vdash_{ty} \sigma_1 \tag{900}$$

By applying rule tTm-Let on Equations 893, 899 and 900, we get

$$\Gamma \vdash_{tm} \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2 : \sigma_2 \tag{901}$$

It remains to show that $\Gamma_C; \Gamma \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2$. This is easily derived from Equation 895, since the existence of variable x in the context does not affect the well-formedness nor the translation of σ_2.

Case iTm-method

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma \rightsquigarrow e \qquad (m : TC\,a : \sigma') \in \Gamma_C}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma/a]\sigma' \rightsquigarrow e.m}\ \text{iTm-method}$$

Applying Theorem 35 to the first premise of rule ɪTᴍ-ᴍᴇᴛʜᴏᴅ results in

$$\text{There are unique } \Gamma \text{ and } \sigma_1 \text{ such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{902}$$
$$\text{and } \Gamma_C; \Gamma \vdash_Q TC\,\sigma \rightsquigarrow \sigma_1 \tag{903}$$
$$\text{and } \Gamma \vdash_{tm} e : \sigma_1 \tag{904}$$

By inversion on Equation 903, we get

$$\Gamma_C = \Gamma_{C1}, m' : TC\,a' : \sigma_m, \Gamma_{C2} \tag{905}$$
$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{906}$$
$$\Gamma_{C1}; \bullet, a' \vdash_{ty} \sigma_m \rightsquigarrow \sigma_m \tag{907}$$

for some m', a', σ_m, σ and σ_m. However, each dictionary TC corresponds to a unique entry in the class environment Γ_C. By this uniqueness, we get $m' = m$, $a = a'$, $\sigma_m = \sigma'$. Then, $\sigma_1 = [\sigma/a]\{m : \sigma_m\}$, and Equations 903 and 904 become

$$\Gamma_C; \Gamma \vdash_Q TC\,\sigma \rightsquigarrow [\sigma/a]\{m : \sigma_m\}$$
$$\text{and } \Gamma \vdash_{tm} e : [\sigma/a]\{m : \sigma_m\} \tag{908}$$

Lemma 20 applied on Equations 906 and 907, results in

$$\Gamma_C; \Gamma \vdash_{ty} [\sigma/a]\sigma' \rightsquigarrow [\sigma/a]\sigma_m$$

and rule ᴛTᴍ-Pʀᴏᴊ instantiated with Equation 908 gives $\Gamma \vdash_{tm} e.m : [\sigma/a]\sigma_m$.

Case ɪTᴍ-ᴀʀʀI

$$\frac{\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \rightsquigarrow e \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \rightarrow \sigma_2 \rightsquigarrow \lambda x : \sigma_1.e}\ \text{ɪTᴍ-ᴀʀʀI}$$

The induction hypothesis from the first premise of rule ɪTᴍ-ᴀʀʀI is

$$\text{There are unique } \Gamma' \text{ and } \sigma_1 \text{ such}$$
$$\text{that } \Gamma_C; \Gamma, x : \sigma_1 \rightsquigarrow \Gamma'$$
$$\text{and } \Gamma_C; \Gamma, x : \sigma_1 \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2 \tag{909}$$
$$\text{and } \Gamma' \vdash_{tm} e : \sigma_2 \tag{910}$$

Similarly to the ɪTᴍ-Lᴇᴛ case, Γ' is of the form $\Gamma, x : \sigma_1$, where

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{911}$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \tag{912}$$

In addition, from Lemma 63 applied on Equation 911 and the second premise of rule ɪTᴍ-ᴀʀʀI, we obtain $\Gamma \vdash_{ty} \sigma_1$. By instantiating the premises of rule ᴛTᴍ-Aʙs with the above result and with Equation 910, we get $\Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \rightarrow \sigma_2$, where $\Gamma_C; \Gamma \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2$. The latter holds from Equation 909, where x has been removed from the context (it does not affect the well-formedness of type σ_2). This, in combination with Equation 912 in rule ɪTʏ-Aʀʀ, gives $\Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightarrow \sigma_2 \rightsquigarrow \sigma_2 \rightarrow \sigma_2$.

Case ɪTᴍ-ᴀʀʀE

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \rightarrow \sigma_2 \rightsquigarrow e_1 \qquad \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_2 : \sigma_1 \rightsquigarrow e_2}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2 \rightsquigarrow e_1\, e_2}\ \text{ɪTᴍ-ᴀʀʀE}$$

The induction hypothesis from the first premise of rule ɪTᴍ-ᴀʀʀE is

$$\text{There are unique } \Gamma \text{ and } \sigma \text{ such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{913}$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightarrow \sigma_2 \rightsquigarrow \sigma \tag{914}$$
$$\text{and } \Gamma \vdash_{tm} e_1 : \sigma \tag{915}$$

By inversion on Equation 914, σ can only be of the form $\sigma_1 \rightarrow \sigma_2$ for the σ_1 and σ_2, uniquely determined by equations

$$\Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1 \tag{916}$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_2 \rightsquigarrow \sigma_2$$

. The induction hypothesis from the second premise of rule iTm-arrE is

$$\text{There are unique } \Gamma' \text{ and } \sigma_1' \text{ such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma' \quad (917)$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1' \quad (918)$$
$$\text{and } \Gamma \vdash_{tm} e_2 : \sigma_1' \quad (919)$$

By uniqueness (Lemma 60) on Equations 913 and 917, it must hold that $\Gamma' = \Gamma$. Also, uniqueness (Lemma 59) on Equations 916 and 918, gives $\sigma_1' = \sigma_1$.
Combining Equations 915 and 919, (rewritten with the equalities holding so far) in rule tTm-App, we get $\Gamma \vdash_{tm} e_1\, e_2 : \sigma_2$.

Case iTm-constrI

$$\frac{\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e \qquad \Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma \rightsquigarrow \lambda\delta : \sigma.e}\ \text{iTm-constrI}$$

The induction hypothesis from the first premise of rule iTm-constrI is

$$\text{There are unique } \Gamma' \text{ and } \sigma \text{ such}$$
$$\text{that } \Gamma_C; \Gamma, \delta : Q \rightsquigarrow \Gamma'$$
$$\text{and } \Gamma_C; \Gamma, \delta : Q \vdash_{ty} \sigma \rightsquigarrow \sigma \quad (920)$$
$$\text{and } \Gamma' \vdash_{tm} e : \sigma \quad (921)$$

Similarly to the iTm-Let case, Γ' is of the form $\Gamma, \delta : \sigma_q$, where

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma \quad (922)$$
$$\text{and } \Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_q \quad (923)$$

In addition, from Lemma 62 applied on Equation 922 and the second premise of rule iTm-constrI, we obtain $\Gamma \vdash_{ty} \sigma_1$. By instantiating the premises of rule tTm-Abs with the above result and with Equation 921, we get $\Gamma \vdash_{tm} \lambda x : \sigma_q.e : \sigma_q \rightarrow \sigma$, where $\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$. The latter holds from Equation 920, where x has been removed from the context (it does not affect the well-formedness of type σ_2). This, in combination with Equation 923 in rule iTy-Qual, gives $\Gamma_C; \Gamma \vdash_{ty} Q \Rightarrow \sigma \rightsquigarrow \sigma_q \rightarrow \sigma$.

Case iTm-constrE

$$\frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \rightsquigarrow e_1 \qquad \Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow e_2}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, d : \sigma \rightsquigarrow e_1\, e_2}\ \text{iTm-constrE}$$

The induction hypothesis from the first premise of rule iTm-constrE is

$$\text{There are unique } \Gamma \text{ and } \sigma' \text{ such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma \quad (924)$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} Q \Rightarrow \sigma \rightsquigarrow \sigma' \quad (925)$$
$$\text{and } \Gamma \vdash_{tm} e_1 : \sigma \quad (926)$$

By inversion on Equation 925, σ' can only be of the form $\sigma_q \rightarrow \sigma$ for the σ_q and σ, uniquely determined by equations

$$\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_q \quad (927)$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma$$

Theorem 35, applied to the second premise of rule iTm-constrE, is

$$\text{There are unique } \Gamma' \text{ and } \sigma_0 \text{ such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma' \quad (928)$$
$$\text{and } \Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma_0 \quad (929)$$
$$\text{and } \Gamma' \vdash_{tm} e_2 : \sigma_0 \quad (930)$$

By uniqueness (Lemma 60) on Equations 924 and 928, it must hold that $\Gamma' = \Gamma$. Also, uniqueness (Lemma 58) on Equations 927 and 929, gives $\sigma_0 = \sigma_q$.
Combining Equations 926 and 930, (rewritten with the above-mentioned equalities) in rule tTm-App, we get $\Gamma \vdash_{tm} e_1\, e_2 : \sigma$.

$$\textbf{Case iTm-forallI} \quad \frac{\Sigma; \Gamma_C; \Gamma, a \vdash_{tm} e : \sigma \rightsquigarrow e}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma \rightsquigarrow \Lambda a.e} \text{iTm-forallI}$$

The induction hypothesis for the premise of rule iTm-forallI is the following.

$$\text{There are unique } \Gamma' \text{ and } \sigma \text{ such}$$
$$\text{that } \Gamma_C; \Gamma, a \rightsquigarrow \Gamma' \tag{931}$$
$$\text{and } \Gamma_C; \Gamma, a \vdash_{ty} \sigma \rightsquigarrow \sigma \tag{932}$$
$$\text{and } \Gamma' \vdash_{tm} e : \sigma \tag{933}$$

By inversion on Equation 931, Γ' can only be of the form Γ, a, where Γ uniquely determined by

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma$$

Then, Equation 933 becomes

$$\Gamma, a \vdash_{tm} e : \sigma$$

Using this to instantiate rule tTm-Tabs, we get

$$\Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma$$

where, by rule iTy-scheme on Equation 932, it follows that $\Gamma_C; \Gamma \vdash_{ty} \forall a.\sigma \rightsquigarrow \forall a.\sigma$.

$$\textbf{Case iTm-forallE} \quad \frac{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \rightsquigarrow e \qquad \Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\,\sigma : [\sigma/a]\sigma' \rightsquigarrow e\,\sigma} \text{iTm-forallE}$$

The induction hypothesis from the first premise of rule iTm-forallE is as follows.

$$\text{There are unique } \Gamma \text{ and } \sigma_0 \text{ such}$$
$$\text{that } \Gamma_C; \Gamma \rightsquigarrow \Gamma \tag{934}$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \forall a.\sigma' \rightsquigarrow \sigma_0 \tag{935}$$
$$\text{and } \Gamma \vdash_{tm} e : \sigma_0 \tag{936}$$

By inversion on Equation 935, σ_0 can only be of the form $\forall a.\sigma'$, where σ' is uniquely determined by equation

$$\Gamma_C; \Gamma, a \vdash_{ty} \sigma' \rightsquigarrow \sigma' \tag{937}$$

Lemma 20, applied on the second premise of rule iTm-forallE and on Equation 937, results in $\Gamma_C; \Gamma \vdash_{ty} [\sigma/a]\sigma' \rightsquigarrow [\sigma/a]\sigma'$.
Also, Lemma 63 applied on the second premise of rule iTm-forallE and on Equation 934 results in $\Gamma \vdash_{ty} \sigma$. We use the latter, together with Equation 936, to instantiate rule tTm-Tapp, which concludes $\Gamma \vdash_{tm} e : [\sigma/a]\sigma'$.

□

Theorem 35 (Dictionary Elaboration Soundness).

If $\Sigma; \Gamma_C; \Gamma \vdash_d \delta : Q \rightsquigarrow e$ (938)

then, there are unique Γ *and* σ *such that*

$\Gamma_C; \Gamma \rightsquigarrow \Gamma$ (939)

and $\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma$ (940)

and $\Gamma \vdash_{tm} e : \sigma$ (941)

Proof. This theorem is proved mutually with Theorem 34. The proof follows structural induction on the first hypothesis.

$$\textbf{Case D-var} \quad \frac{(\delta : Q) \in \Gamma \qquad \vdash_{ctx} \Sigma; \Gamma_C; \Gamma}{\Sigma; \Gamma_C; \Gamma \vdash_d \delta : Q \rightsquigarrow \delta} \text{D-var}$$

By Lemma 65, there exist Γ and σ such that $\Gamma_C; \Gamma \rightsquigarrow \Gamma$ and $\Gamma_C; \Gamma \vdash_Q TC\,\sigma \rightsquigarrow \sigma$ and $(\delta : \sigma) \in \Gamma$. This satisfies Equations 939 and 940 of the theorem.
By instantiating rule tTm-Var on $(\delta : \sigma) \in \Gamma$, we get $\Gamma \vdash_{tm} \delta : \sigma$, which satisfies Equation 941 of the theorem.

Case D-con

$$\frac{\Sigma = \Sigma_1, (D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e, \Sigma_2 \quad \vdash_{ctx} \Sigma;\Gamma_C;\Gamma \quad \overline{\Gamma_C;\bullet,\overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma'_i}^{\,i} \quad \overline{\Gamma_C;\Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^{\,j} \quad \Sigma_1;\Gamma_C;\bullet,\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \vdash_{tm} e : [\sigma_q/a]\sigma_m \rightsquigarrow e \quad \overline{\Sigma;\Gamma_C;\Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i \rightsquigarrow e_i}^{\,i}}{\Sigma;\Gamma_C;\Gamma \vdash_d D\,\overline{\sigma}_j\,\overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q \rightsquigarrow (\Lambda\overline{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.\{m = e\})\,\overline{\sigma}_j\,\overline{e}_i}\ \text{D-con}$$

We need to show that there exist Γ_0 and σ such that $\Gamma_C;\Gamma \rightsquigarrow \Gamma_0$ and

$$\Gamma_C;\Gamma \vdash_Q TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q \rightsquigarrow \sigma \tag{942}$$

$$\Gamma_0 \vdash_{tm} (\Lambda a_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.\{m = e\})\,\overline{\sigma}_j\,\overline{e}_i : \sigma \tag{943}$$

From the 2nd premise of rule D-con, applied on Lemma 66, there is a unique Γ such that

$$\Gamma_C;\Gamma \rightsquigarrow \Gamma \tag{944}$$

$$\vdash_{ctx} \Gamma$$

We thus take $\Gamma_0 = \Gamma$.
Next, we will try to refine type σ. By Lemma 40 applied on the first premise of rule D-con, there are a, σ_0 and $\overline{\sigma}_i$ such that

$$(m : TC\,a : \sigma_m) \in \Gamma_C \tag{945}$$

$$\text{and } \overline{\Gamma_C;\bullet,\overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma''_i}^{\,i} \tag{946}$$

$$\text{and } \Gamma_C;\bullet,\overline{a}_j \vdash_Q TC\,\sigma_q \rightsquigarrow \sigma_0 \tag{947}$$

Since Equation 947 holds, the premises of the only rule that applies, namely rule iQ-TC, must hold as well.

$$\Gamma_C;\bullet,\overline{a}_j \vdash_{ty} \sigma_q \rightsquigarrow \sigma_q \tag{948}$$

$$\Gamma_C = \Gamma_{C1}, m' : TC\,a' : \sigma'_m, \Gamma_{C2} \tag{949}$$

$$\Gamma_{C1};\bullet,a \vdash_{ty} \sigma'_m \rightsquigarrow \sigma_m \tag{950}$$

for some m', a', σ'_m, σ_q and σ_m, where $\sigma_0 = [\sigma_q/a']\{m' : \sigma_m\}$. By uniqueness on Equations 945 and 949, we get $m' = m$, $a = a'$, $\sigma'_m = \sigma_m$. Then, $\sigma_0 = [\sigma_q/a]\{m : \sigma_m\} = \{m : [\sigma_q/a]\sigma_m\}$. By Lemma 21 applied on Equation 948 and the 4th set of premises of rule D-con, Goal 942 is satisfied with $\sigma = \{m : [\,\overline{\sigma_j/a_j}^{\,j}\,][\sigma_q/a]\sigma_m\}$.
Then, Goal 943 becomes

$$\Gamma \vdash_{tm} (\Lambda a_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.\{m = e\})\,\overline{\sigma}_j\,\overline{e}_i : \{m : [\,\overline{\sigma_j/a_j}^{\,j}\,][\sigma_q/a]\sigma_m\} \tag{951}$$

For that, it suffices to show that

$$\Gamma, \overline{a}_j, \overline{\delta_i : \sigma'_i}^{\,i} \vdash_{tm} e : [\sigma_q/a]\sigma_m \tag{952}$$

$$\overline{\Gamma \vdash_{ty} \sigma_j}^{\,j} \tag{953}$$

$$\text{and } \overline{\Gamma \vdash_{tm} e_i : [\,\overline{\sigma_j/a_j}^{\,j}\,]\sigma'_i}^{\,i} \tag{954}$$

because, then we can use rules tTm-Rec, tTm-TApp and tTm-App to reach Equation 951.

For Goal 953. Passing the 4th set of premises of rule D-con in Lemma 62, we get $\overline{\Gamma \vdash_{ty} \sigma_j}^{\,j}$, satisfying Equation 953.

For Goal 952. Theorem 34, applied to the 5th premise of rule D-con, is:

There are Γ' and σ_0 such that

$$\Gamma_C;\bullet,\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \rightsquigarrow \Gamma'$$

$$\text{and } \Gamma_C;\bullet,\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \vdash_{ty} [\sigma_q/a]\sigma_m \rightsquigarrow \sigma_0 \tag{955}$$

$$\text{and } \Gamma' \vdash_{tm} e : \sigma_0. \tag{956}$$

It is easy to verify that $\Gamma' = \bullet, \overline{a}_j, \overline{\delta_i : \sigma'_i}^{\,i}$. From Lemma 31 on Equation 950, we get the weakened equation

$$\Gamma_C;\bullet,a \vdash_{ty} \sigma_m \rightsquigarrow \sigma_m$$

where Γ_C is specified in Equation 949. Using this result together with Equation 948 in Lemma 20, we get $\Gamma_C;\bullet,\overline{a}_j \vdash_{ty} [\sigma_q/a]\sigma_m \rightsquigarrow [\sigma_q/a]\sigma_m$, which we weaken as

$$\Gamma_C;\bullet,\overline{a}_j,\overline{\delta}_i : \overline{Q}_i \vdash_{ty} [\sigma_q/a]\sigma_m \rightsquigarrow [\sigma_q/a]\sigma_m$$

Then, by uniqueness on the latter and on Equation 955, we have $\sigma_0 = [\sigma_q/a]\sigma_m$, and Equation 956 becomes

$$\bullet, \bar{a}_j, \overline{\delta_i : \sigma'_i}^{\,i} \vdash_{tm} e : [\sigma_q/a]\sigma_m$$

Prefixing the typing environment of the above with Γ, Goal 952 is satisfied.

For Goal 954. The 6th set of premises of rule D-CON induces, for each i, the following induction hypothesis.

$$\text{There are } \Gamma_i \text{ and } \sigma''_i \text{ such that}$$

$$\Gamma_C; \Gamma \rightsquigarrow \Gamma_i \tag{957}$$

$$\Gamma_C; \Gamma \vdash_Q [\bar{\sigma}_j/\bar{a}_j]Q_i \rightsquigarrow \sigma''_i \tag{958}$$

$$\text{and } \Gamma_i \vdash_{tm} e_i : \sigma''_i \tag{959}$$

From Equations 944 and 957 in Lemma 60, we have $\Gamma_i = \Gamma$. For each equation in the 3rd set of premises of rule D-CON, we appply Lemma 21 multiple times with the 4th set of premises of rule D-CON, to get

$$\overline{\Gamma_C; \Gamma \vdash_Q [\bar{\sigma}_j/\bar{a}_j]Q_i \rightsquigarrow [\overline{\sigma_j/a_j}^{\,j}]\sigma'_i}^{\,i}$$

Then $\sigma''_i = [\overline{\sigma_j/a_j}^{\,j}]\sigma'_i$, for all i, and Equation 959 satisfies Goal 954.

□

N.3 Determinism

THEOREM 36 (DETERMINISTIC DICTIONARY ELABORATION). *If* $\Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow e_1$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_d d : Q \rightsquigarrow e_2$, *then* $e_1 = e_2$.

PROOF. This theorem is proved mutually with Theorem 37. The proof follows structural induction on both hypotheses.

Case D-VAR

The first and second hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_d \delta : Q \rightsquigarrow \delta_1$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_d \delta : Q \rightsquigarrow \delta_2$$

From the convention regarding namespace translations, explained in Section D.5, it follows directly that $\delta_1 = \delta_2 = \delta$.

Case D-CON

The first and second hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_d D\,\bar{\sigma}_j\,\bar{d}_i : TC\,[\bar{\sigma}_j/\bar{a}_j]\sigma_q \rightsquigarrow (\Lambda\bar{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.\{m = e\})\,\bar{\sigma}_j\,\bar{e}_i \tag{960}$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_d D\,\bar{\sigma}_j\,\bar{d}_i : TC\,[\bar{\sigma}_j/\bar{a}_j]\sigma_q \rightsquigarrow (\Lambda\bar{a}_j.\lambda\,\overline{\delta'_i : \sigma''_i}^{\,i}.\{m' = e'\})\,\bar{\sigma}'_j\,\bar{e}'_i \tag{961}$$

The 5th premise of the two instantiations of rule D-CON, are

$$\Sigma_1; \Gamma_C; \bullet, \bar{a}_j, \bar{\delta}_i : \bar{Q}_i \vdash_{tm} e : [\sigma_q/a]\sigma_m \rightsquigarrow e \tag{962}$$

$$\text{and } \Sigma_1; \Gamma_C; \bullet, \bar{a}_j, \bar{\delta}'_i : \bar{Q}'_i \vdash_{tm} e : [\sigma_q/a]\sigma_m \rightsquigarrow e' \tag{963}$$

and the 1st premise of the two D-CON rules are

$$(D : \forall\bar{a}_j.\bar{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\bar{a}_j.\lambda\bar{\delta}_i : \bar{Q}_i.e \in \Sigma$$

$$(D : \forall\bar{a}_j.\bar{Q}'_i \Rightarrow TC\,\sigma_q).m' \mapsto \Lambda\bar{a}_j.\lambda\bar{\delta}'_i : \bar{Q}'_i.e \in \Sigma$$

However, a valid method-implementations environment, like Σ in this case, contains a unique entry for each constructor D. From the two above premises and this uniqueness property, we have that

$$\bar{Q}'_i = \bar{Q}_i \text{ and } \bar{\delta}'_i = \bar{\delta}_i, \text{ for all } i \quad \text{and } m' = m \tag{964}$$

By applying these equations to Equations 962 and 963, their typing environment becomes identical. We can, now, use Theorem 37 on these two equations, from which we get $e = e'$. Also, from the namespace-translation convention, we get $m' = m$ and $\overline{\delta}'_i = \overline{\delta}_i$.

If we rewrite Equations 960 and 961 with the equations obtained so far, we have

$$\Sigma; \Gamma_C; \Gamma \vdash_d D\, \overline{\sigma}_j\, \overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q \rightsquigarrow (\Lambda \overline{a}_j.\lambda\, \overline{\delta_i : \sigma'_i}^{\,i}.\{m = e\})\, \overline{\sigma}_j\, \overline{e}_i \tag{965}$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_d D\, \overline{\sigma}_j\, \overline{d}_i : TC\,[\overline{\sigma}_j/\overline{a}_j]\sigma_q \rightsquigarrow (\Lambda \overline{a}_j.\lambda\, \overline{\delta_i : \sigma''_i}^{\,i}.\{m = e\})\, \overline{\sigma}'_j\, \overline{e}'_i \tag{966}$$

From Equations 964 and the 3rd set of premises of the two D-con instantiations in Equations 965 and 966, we get

$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma'_i}^{\,i}$$

$$\text{and } \overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma''_i}^{\,i}$$

By passing the two above in Lemma 58, we get $\sigma''_i = \sigma'_i$, for all i.

From the 4th set of premises of the two D-con instantiations in Equations 965 and 966, we get

$$\overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^{\,j} \tag{967}$$

$$\text{and } \overline{\Gamma_C; \Gamma \vdash_{ty} \sigma_j \rightsquigarrow \sigma'_j}^{\,j} \tag{968}$$

By passing the two above in Lemma 59, we get $\sigma'_j = \sigma_j$, for all j.

From the 6th set of premises of the two D-con instantiations in Equations 965 and 966, we get

$$\overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i \rightsquigarrow e_i}^{\,i} \tag{969}$$

$$\text{and } \overline{\Sigma; \Gamma_C; \Gamma \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i \rightsquigarrow e'_i}^{\,i} \tag{970}$$

From the induction hypothesis for the two above equations, we have $e'_i = e_i$, for all i.

Rewritting the derivations in Equations 965 and 966 with the new equations obtained, we have

$$(\Lambda \overline{a}_j.\lambda\, \overline{\delta_i : \sigma''_i}^{\,i}.\{m = e\})\, \overline{\sigma}'_j\, \overline{e}'_i = (\Lambda \overline{a}_j.\lambda\, \overline{\delta_i : \sigma'_i}^{\,i}.\{m = e\})\, \overline{\sigma}_j\, \overline{e}_i$$

□

Theorem 37 (Deterministic Term Elaboration).
If $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e_1$ *and* $\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \sigma \rightsquigarrow e_2$*, then* $e_1 = e_2$.

Proof. This theorem is proved mutually with Theorem 36. The proof follows structural induction on both hypotheses.

Case iTm-true

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} True : Bool \rightsquigarrow e_1$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} True : Bool \rightsquigarrow e_2$$

From rule iTm-true, it must hold that $e_1 = e_2 = True$.

Case iTm-false

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} False : Bool \rightsquigarrow e_1$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} False : Bool \rightsquigarrow e_2$$

From rule iTm-false, it must hold that $e_1 = e_2 = False$.

Case iTm-var

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} x : \sigma \rightsquigarrow x_1$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} x : \sigma \rightsquigarrow x_2$$

From rule iTm-var, it must hold that $x_1 = x_2 = x$, where x is a target-term-variable with the same identifier as x.

Case iTm-let

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \textbf{let } x : \sigma_1 = e_1 \textbf{ in } e_2 : \sigma_2 \rightsquigarrow \textbf{let } x_0 : \sigma_1 = e_1 \textbf{ in } e_2 \quad (971)$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} \textbf{let } x : \sigma_1 = e_1 \textbf{ in } e_2 : \sigma_2 \rightsquigarrow \textbf{let } x_0' : \sigma_1' = e_1' \textbf{ in } e_2' \quad (972)$$

From our convention regarding translation of identifiers, we have

$$x_0 = x_0' = x \quad (973)$$

where x is a target-term variable with the same identifier as x.
The first premise of the two iTm-let instantiations above, are

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \rightsquigarrow e_1'$$

By induction hypothesis, we get

$$e_1 = e_1' \quad (974)$$

The 3rd premise of the two instantiations in Equations 971 and 972 of rule iTm-let are:

$$\Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1'$$

By uniqueness (Lemma 59), we get

$$\sigma_1 = \sigma_1' \quad (975)$$

The 2nd premise of the two instantiations in Equations 971 and 972 of rule iTm-let are:

$$\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2$$
$$\text{and } \Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e_2 : \sigma_2 \rightsquigarrow e_2'$$

By induction hypothesis, we get

$$e_2 = e_2' \quad (976)$$

From Equations 973, 974, 975 and 976, we obtain

$$\textbf{let } x_0 : \sigma_1 = e_1 \textbf{ in } e_2 = \textbf{let } x_0' : \sigma_1' = e_1' \textbf{ in } e_2'$$

Case iTm-method

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma_q/a]\sigma_m \rightsquigarrow e.m_0 \quad (977)$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} d.m : [\sigma_q/a]\sigma_m \rightsquigarrow e'.m_0' \quad (978)$$

By our convention for dictionary labels, we have

$$m_0 = m_0' = m$$

where m is a record field with the same identifier as class method m.
The first premise of the two iTm-method instantiations in Equations 977 and 978 are:

$$\Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma_q \rightsquigarrow e$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_d d : TC\,\sigma_q \rightsquigarrow e'$$

By Theorem 36, we have

$$e = e'$$

Then, $e.m_0 = e'.m_0'$.

Case iTm-arrI

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \to \sigma_2 \rightsquigarrow \lambda x_0 : \sigma_1.e \tag{979}$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda x : \sigma_1.e : \sigma_1 \to \sigma_2 \rightsquigarrow \lambda x_0' : \sigma_1'.e' \tag{980}$$

From our identifiers' translation, it is implied that

$$x_0 = x_0' = x \tag{981}$$

where x is the target-term variable with the same identifier as x.
The 2nd premise of the two ITm-ArrI instantiations above are:

$$\Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1$$
$$\text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma_1 \rightsquigarrow \sigma_1'$$

By uniqueness (Lemma 59), we have

$$\sigma_1 = \sigma_1' \tag{982}$$

The first premise of the two ITm-ArrI instantiations in Equations 979 and 980 are:

$$\Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \rightsquigarrow e$$
$$\text{and } \Sigma; \Gamma_C; \Gamma, x : \sigma_1 \vdash_{tm} e : \sigma_2 \rightsquigarrow e'$$

By the induction hypothesis, we get

$$e = e' \tag{983}$$

From Equations 981, 982 and 983, we obtain

$$\lambda x_0 : \sigma_1.e = \lambda x_0' : \sigma_1'.e'$$

Case ITm-ArrE

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2 \rightsquigarrow e_1\, e_2 \tag{984}$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1\, e_2 : \sigma_2 \rightsquigarrow e_1'\, e_2' \tag{985}$$

The first premise of the above two instantiated ITm-ArrE rules are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \rightsquigarrow e_1$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \rightsquigarrow e_1'$$

By induction hypothesis, we have

$$e_1 = e_1' \tag{986}$$

Similarly, from the second premise of the two ITm-ArrE rules, and the induction hypothesis, we get

$$e_2 = e_2' \tag{987}$$

Then, we obtain

$$e_1\, e_2 = e_1'\, e_2'$$

by Equations 986 and 987.

Case ITm-ConstrI

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma \rightsquigarrow \lambda\delta_0 : \sigma.e \tag{988}$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma \rightsquigarrow \lambda\delta_0' : \sigma'.e' \tag{989}$$

From our identifiers' translation, it is implied that

$$\delta_0 = \delta_0' = \delta \tag{990}$$

where δ is the target-term variable with the same identifier as the dictionary variable δ.
The 2nd premise of the two ITm-ConstrI instantiations above are:

$$\Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma$$
$$\text{and } \Gamma_C; \Gamma \vdash_Q Q \rightsquigarrow \sigma'$$

By uniqueness (Lemma 58), we have

$$\sigma = \sigma' \tag{991}$$

The first premise of the two iTm-constrI instantiations in Equations 988 and 989 are:

$$\Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e$$
$$\text{and } \Sigma; \Gamma_C; \Gamma, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e'$$

By the induction hypothesis, we get

$$e = e' \tag{992}$$

From Equations 990, 991 and 992, we obtain

$$\lambda\delta_0 : \sigma.e = \lambda\delta_0' : \sigma'.e'$$

Case iTm-constrE

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, d : \sigma \rightsquigarrow e_1\, e_2 \tag{993}$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e\, d : \sigma \rightsquigarrow e_1'\, e_2' \tag{994}$$

The first premise of the above two instantiated iTm-constrE rules are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \rightsquigarrow e_1$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e : Q \Rightarrow \sigma \rightsquigarrow e_1'$$

By induction hypothesis, we have

$$e_1 = e_1' \tag{995}$$

Similarly, from the second premise of the two iTm-constrE rules, and the induction hypothesis, we get

$$e_2 = e_2' \tag{996}$$

Then, we obtain

$$e_1\, e_2 = e_1'\, e_2'$$

by Equations 995 and 996.

Case iTm-forallI

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma \rightsquigarrow \Lambda a_0.e$$
$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} \Lambda a.e : \forall a.\sigma \rightsquigarrow \Lambda a_0'.e'$$

It is implied by our identifiers' translation convention, that

$$a_0 = a_0' = a \tag{997}$$

where a is the target-type variable with the same identifier as the a.
The premise of the two iTm-forallI instantiations above are:

$$\Sigma; \Gamma_C; \Gamma, a \vdash_{tm} e : \sigma \rightsquigarrow e$$
$$\text{and } \Sigma; \Gamma_C; \Gamma, a \vdash_{tm} e : \sigma \rightsquigarrow e'$$

By the induction hypothesis, we have

$$e = e' \tag{998}$$

Equations 997 and 998 result in

$$\Lambda a_0.e = \Lambda a_0'.e'$$

Case iTm-forallE

The two hypotheses of the theorem are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e\,\sigma : [\sigma/a]\sigma' \rightsquigarrow e\,\sigma \tag{999}$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e\,\sigma : [\sigma/a]\sigma' \rightsquigarrow e'\,\sigma' \tag{1000}$$

The first premise of the above two instantiations of rule iTm-forallE are:

$$\Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \rightsquigarrow e$$

$$\text{and } \Sigma; \Gamma_C; \Gamma \vdash_{tm} e : \forall a.\sigma' \rightsquigarrow e'$$

From the induction hypothesis, we get

$$e = e' \tag{1001}$$

The second premise of the two instantiations of rule iTm-forallE in Equations 999 and 1000 are:

$$\Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma \text{and } \Gamma_C; \Gamma \vdash_{ty} \sigma \rightsquigarrow \sigma'$$

Applying Lemma 59 on these equations gives

$$\sigma = \sigma' \tag{1002}$$

From Equations 1001 and 1002 we have

$$e\,\sigma = e'\,\sigma'$$

□

Theorem 38 (Deterministic Context Elaboration).
If $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M_1$ *and* $M : (\Sigma; \Gamma_C; \Gamma \Rightarrow \sigma) \mapsto (\Sigma; \Gamma_C; \Gamma' \Rightarrow \sigma') \rightsquigarrow M_2$, *then* $M_1 = M_2$.

Proof. This proof proceeds by straightforward structural induction on both hypotheses, in combination with Lemmas 58 and 59 and Theorems 36 and 37.

□

N.4 Semantic Preservation

Theorem 39 (Semantic Preservation).
If $\Sigma \vdash e \longrightarrow e'$

$$\textit{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e : \sigma \rightsquigarrow e \tag{1003}$$

$$\textit{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e' : \sigma \rightsquigarrow e' \tag{1004}$$

for some σ, e *and* e',
then, there is an e_h *such that*
$e \longrightarrow^* e_h$ *and* $e' \longrightarrow^* e_h$.

Proof. This proof proceeds by induction on the first hypothesis.

Case iEval-app $\dfrac{\Sigma \vdash e_1 \longrightarrow e_1'}{\Sigma \vdash e_1\,e_2 \longrightarrow e_1'\,e_2}$ iEval-app

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1\,e_2 : \sigma_2 \rightsquigarrow e_1\,e_2$$

$$\text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e_1'\,e_2 : \sigma_2 \rightsquigarrow e_1'\,e_2$$

for some σ_2, e_1, e_1' and e_2.
The last rule of these derivations must be instances of iTm-arrE. For Hypothesis 1003, we have:

$$\frac{\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1 : \sigma_1 \to \sigma_2 \rightsquigarrow e_1 \qquad \Sigma; \Gamma_C; \bullet \vdash_{tm} e_2 : \sigma_1 \rightsquigarrow e_2}{\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1\,e_2 : \sigma_2 \rightsquigarrow e_1\,e_2}\ \text{iTm-arrE} \tag{1005}$$

and for Hypothesis 1004, we have:

$$\frac{\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1' : \sigma_1' \to \sigma_2 \rightsquigarrow e_1' \qquad \Sigma; \Gamma_C; \bullet \vdash_{tm} e_2 : \sigma_1' \rightsquigarrow e_2}{\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1'\,e_2 : \sigma_2 \rightsquigarrow e_1'\,e_2}\ \text{iTm-arrE} \tag{1006}$$

From the second premise of the two above rules and by uniqueness (Lemma 59), we get $\sigma_1 = \sigma_1'$.

The induction hypothesis is:

$$\begin{array}{r} \text{If } \Sigma; \Gamma_C; \bullet \vdash_{tm} e_1 : \sigma_h \rightsquigarrow e_p \\ \text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e_1' : \sigma_h \rightsquigarrow e_q \\ \text{for some } \sigma_h,\ e_p \text{ and } e_q, \\ \text{then, there is an } e' \text{ such that} \\ e_p \longrightarrow^* e' \text{ and } e_q \longrightarrow e' \end{array}$$

Then, an appropriate choice for e_h is $e'\, e_2$, since from Lemma 69, we have $e_1\, e_2 \longrightarrow^* e'\, e_2$ and $e_1'\, e_2 \longrightarrow^* e'\, e_2$.

Case iEval-appAbs $\dfrac{}{\Sigma \vdash (\lambda x : \sigma.e_1)\, e_2 \longrightarrow [e_2/x]e_1}$ iEval-appAbs

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\lambda x : \sigma.e_1)\, e_2 : \sigma' \rightsquigarrow e_0 \tag{1007}$$

$$\text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} [e_2/x]e_1 : \sigma' \rightsquigarrow e_0' \tag{1008}$$

for some σ', e_0 and e_0'. We need to show that there exists an e_h such that $e_0 \longrightarrow^* e_h$ and $e_0' \longrightarrow^* e_h$. We do this by showing that $e_0 \longrightarrow^* e_0'$.
By inversion, the last part of Derivation 1007 must be an instance of iTm-arrI directly followed by iTm-arrE, as shown below.

$$\dfrac{\dfrac{\Sigma; \Gamma_C; \bullet, x : \sigma \vdash_{tm} e_1 : \sigma' \rightsquigarrow e_1 \qquad \Gamma_C; \bullet \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \bullet \vdash_{tm} \lambda x : \sigma.e_1 : \sigma \rightarrow \sigma' \rightsquigarrow \lambda x : \sigma.e_1} \text{iTm-arrI} \qquad \Sigma; \Gamma_C; \bullet \vdash_{tm} e_2 : \sigma \rightsquigarrow e_2}{\Sigma; \Gamma_C; \bullet \vdash_{tm} (\lambda x : \sigma.e_1)\, e_2 : \sigma' \rightsquigarrow (\lambda x : \sigma.e_1)\, e_2} \text{iTm-arrE}$$

where $e_0 = (\lambda x : \sigma.e_1)\, e_2$. From the above equation, we can use premises

$$\begin{array}{r} \Sigma; \Gamma_C; \bullet, x : \sigma \vdash_{tm} e_1 : \sigma' \rightsquigarrow e_1 \\ \text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e_2 : \sigma \rightsquigarrow e_2 \end{array}$$

in Lemma 22, to obtain

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} [e_2/x]e_1 : \sigma' \rightsquigarrow [e_1/x]e_2$$

Then, by uniqueness (Theorem 37 on the latter and on Equation 1008), we have $e_0' = [e_1/x]e_2$.
We set $e_h = [e_1/x]e_2$, since $(\lambda x : \sigma.e_1)\, e_2 \longrightarrow^* [e_1/x]e_2$, by evaluation rule tEval-AppAbs, and $[e_1/x]e_2 \longrightarrow^* [e_1/x]e_2$, by reflexivity of $\longrightarrow^*$.

Case iEval-tyApp $\dfrac{\Sigma \vdash e \longrightarrow e'}{\Sigma \vdash e\, \sigma \longrightarrow e'\, \sigma}$ iEval-tyApp

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\begin{array}{r} \Sigma; \Gamma_C; \bullet \vdash_{tm} e\, \sigma : [\sigma/a]\sigma_p \rightsquigarrow e\, \sigma \\ \text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e'\, \sigma : [\sigma/a']\sigma_q \rightsquigarrow e'\, \sigma \end{array}$$

for some a, a', σ_p, σ_q, e, e' and σ.
The last rule of both derivations above must be instances of iTm-forallE. For the first, we have:

$$\dfrac{\Sigma; \Gamma_C; \bullet \vdash_{tm} e : \forall a.\sigma_p \rightsquigarrow e \qquad \Gamma_C; \bullet \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \bullet \vdash_{tm} e\, \sigma : [\sigma/a]\sigma_p \rightsquigarrow e\, \sigma} \text{iTm-forallE} \tag{1009}$$

and for the second:

$$\dfrac{\Sigma; \Gamma_C; \bullet \vdash_{tm} e' : \forall a'.\sigma_q \rightsquigarrow e' \qquad \Gamma_C; \bullet \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \bullet \vdash_{tm} e\, \sigma : [\sigma/a']\sigma_q \rightsquigarrow e'\, \sigma} \text{iTm-forallE} \tag{1010}$$

By applying Theorem 8 on the first premise of Equation 1009 and on the premise of rule iEval-tyApp, we have that $\forall a.\sigma_p = \forall a'.\sigma_q$.

The induction hypothesis is:

$$\begin{array}{r}\text{If } \Sigma;\Gamma_C;\bullet \vdash_{tm} e_1 : \sigma_h \rightsquigarrow e_p \\ \text{and } \Sigma;\Gamma_C;\bullet \vdash_{tm} e_1' : \sigma_h \rightsquigarrow e_q \\ \text{for some } \sigma_h,\ e_p \text{ and } e_q, \\ \text{then, there is an } e_h' \text{ such that} \\ e_p \longrightarrow^* e_h' \text{ and } e_q \longrightarrow e_h'\end{array}$$

For $\sigma_h = \forall a.\sigma_p$, $e_p = e$ and $e_q = e'$, the two conditions are fulfilled by the first premise of Equations 1009 and 1010. We choose $e_h = e_h'\ \sigma$, because from Lemma 70, we have $e\ \sigma \longrightarrow^* e_h'\ \sigma$ and $e'\ \sigma \longrightarrow^* e_h'\ \sigma$.

Case iEval-tyAppAbs $\dfrac{}{\Sigma \vdash (\Lambda a.e)\ \sigma \longrightarrow [\sigma/a]e}$ iEval-tyAppAbs

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\Sigma;\Gamma_C;\bullet \vdash_{tm} (\Lambda a.e)\ \sigma : \sigma_0 \rightsquigarrow e_0 \tag{1011}$$

$$\text{and } \Sigma;\Gamma_C;\bullet \vdash_{tm} [\sigma/a]e : \sigma_0 \rightsquigarrow e_0' \tag{1012}$$

for some σ_0, e_0 and e_0'. We need to show that there exists an e_h such that $e_0 \longrightarrow^* e_h$ and $e_0' \longrightarrow^* e_h$. We do this by showing that $e_0 \longrightarrow^* e_0'$.
By inversion, the last part of Derivation 1011 must be an instance of iTm-forallI directly followed by iTm-forallE, as shown below.

$$\dfrac{\dfrac{\Sigma;\Gamma_C;\bullet,a \vdash_{tm} e : \sigma' \rightsquigarrow e}{\Sigma;\Gamma_C;\bullet \vdash_{tm} \Lambda a.e : \forall a.\sigma' \rightsquigarrow \Lambda a.e}\ \text{iTm-forallI} \qquad \Gamma_C;\bullet \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma;\Gamma_C;\bullet \vdash_{tm} (\Lambda a.e)\ \sigma : [\sigma/a]\sigma' \rightsquigarrow (\Lambda a.e)\ \sigma}\ \text{iTm-forallE}$$

where $\sigma_0 = [\sigma/a]\sigma'$ and $e_0 = (\Lambda a.e)\ \sigma$. From the above equation, we can use premises

$$\begin{array}{r}\Sigma;\Gamma_C;\bullet,a \vdash_{tm} e : \sigma' \rightsquigarrow e \\ \text{and } \Gamma_C;\bullet \vdash_{ty} \sigma \rightsquigarrow \sigma\end{array}$$

in Lemma 26, to obtain

$$\Sigma;\Gamma_C;\bullet \vdash_{tm} [\sigma/a]e : [\sigma/a]\sigma' \rightsquigarrow [\sigma/a]e$$

Then, by uniqueness (Theorem 37 on the latter and on Equation 1012), we have $e_0' = [\sigma/a]e$.
We set $e_h = [\sigma/a]e$, since $(\Lambda a.e)\ \sigma \longrightarrow^* [\sigma/a]e$, by evaluation rule tEval-TAppAbs, and $[\sigma/a]e \longrightarrow^* [\sigma/a]e$, by reflexivity of $\longrightarrow^*$.

Case iEval-DApp $\dfrac{\Sigma \vdash e \longrightarrow e'}{\Sigma \vdash e\ d \longrightarrow e'\ d}$ iEval-DApp

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\begin{array}{r}\Sigma;\Gamma_C;\bullet \vdash_{tm} e\ d : \sigma \rightsquigarrow e_1\ e_2 \\ \text{and } \Sigma;\Gamma_C;\bullet \vdash_{tm} e'\ d : \sigma \rightsquigarrow e_1'\ e_2\end{array}$$

for some e_1, e_1' and e_2.
The last rule of these derivations must be instances of iTm-constrE. For Hypothesis 1003, we have:

$$\dfrac{\Sigma;\Gamma_C;\bullet \vdash_{tm} e : Q \Rightarrow \sigma \rightsquigarrow e_1 \qquad \Sigma;\Gamma_C;\bullet \vdash_d d : Q \rightsquigarrow e_2}{\Sigma;\Gamma_C;\bullet \vdash_{tm} e\ d : \sigma \rightsquigarrow e_1\ e_2}\ \text{iTm-constrE} \tag{1013}$$

and for Hypothesis 1004, we have:

$$\dfrac{\Sigma;\Gamma_C;\bullet \vdash_{tm} e' : Q' \Rightarrow \sigma \rightsquigarrow e_1' \qquad \Sigma;\Gamma_C;\bullet \vdash_d d : Q' \rightsquigarrow e_2}{\Sigma;\Gamma_C;\bullet \vdash_{tm} e_1'\ d : \sigma \rightsquigarrow e_1'\ e_2}\ \text{iTm-constrE} \tag{1014}$$

From the second premise of the two above rules and by uniqueness, we get $Q = Q'$.

The induction hypothesis is:

$$\begin{aligned} &\text{If } \Sigma; \Gamma_C; \bullet \vdash_{tm} e : \sigma_h \rightsquigarrow e_p \\ &\text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e' : \sigma_h \rightsquigarrow e_q \\ &\text{for some } \sigma_h,\ e_p \text{ and } e_q, \\ &\text{then, there is an } e'_h \text{ such that} \\ &e_p \longrightarrow^* e'_h \text{ and } e_q \longrightarrow e'_h \end{aligned}$$

Then, the conditions of the above hold from the first premise of Derivations 1013 and 1014, where $\sigma_h = Q \Rightarrow \sigma$, $e_p = e_1$ and $e_q = e'_1$.
Then, an appropriate choice for e_h is $e'_h\, e_2$, since from Lemma 69, we have $e_1\, e_2 \longrightarrow^* e'_h\, e_2$ and $e'_1\, e_2 \longrightarrow^* e'_h\, e_2$.

Case iEval-DAppAbs $\dfrac{}{\Sigma \vdash (\lambda\delta : Q.e)\, d \longrightarrow [d/\delta]e}$ iEval-DAppAbs

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (\lambda\delta : Q.e)\, d : \sigma \rightsquigarrow e_0 \tag{1015}$$
$$\text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} [d/\delta]e : \sigma \rightsquigarrow e'_0 \tag{1016}$$

for some σ', e_0 and e'_0. We need to show that there exists an e_h such that $e_0 \longrightarrow^* e_h$ and $e'_0 \longrightarrow^* e_h$. We do this by showing that $e_0 \longrightarrow^* e'_0$.
By inversion, the last part of Derivation 1015 must be an instance of iTm-constrI directly followed by iTm-constrE, as shown below.

$$\dfrac{\dfrac{\Sigma; \Gamma_C; \bullet, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e_1 \qquad \Gamma_C; \bullet \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \bullet \vdash_{tm} \lambda\delta : Q.e : Q \Rightarrow \sigma \rightsquigarrow \lambda\delta : \sigma.e_1}\ \text{iTm-constrI} \qquad \Sigma; \Gamma_C; \bullet \vdash_d d : Q \rightsquigarrow e_2}{\Sigma; \Gamma_C; \bullet \vdash_{tm} (\lambda\delta : Q.e)\, d : \sigma \rightsquigarrow (\lambda\delta : \sigma.e_1)\, e_2}\ \text{iTm-constrE}$$

where $e_0 = (\lambda\delta : \sigma.e_1)\, e_2$. From the above equation, we can use premises

$$\begin{aligned} &\Sigma; \Gamma_C; \bullet, \delta : Q \vdash_{tm} e : \sigma \rightsquigarrow e_1 \\ &\text{and } \Sigma; \Gamma_C; \bullet \vdash_d d : Q \rightsquigarrow e_2 \end{aligned}$$

in Lemma 24, to obtain

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} [d/\delta]e : \sigma \rightsquigarrow [e_1/\delta]e_2$$

Then, by uniqueness (Theorem 36 on the latter and on Equation 1016), we have $e'_0 = [e_1/\delta]e_2$.
We set $e_h = [e_1/\delta]e_2$, since $(\lambda\delta : \sigma.e_1)\, e_2 \longrightarrow^* [e_1/\delta]e_2$, by evaluation rule tEval-AppAbs, and $[e_1/\delta]e_2 \longrightarrow^* [e_1/\delta]e_2$, by reflexivity of $\longrightarrow^*$.

Case iEval-method $\dfrac{(D : \forall \overline{a}_j.\overline{Q}_i \Rightarrow TC\, \sigma_q).m \mapsto e \in \Sigma}{\Sigma \vdash (D\, \overline{\sigma}_j\, \overline{d}_i).m \longrightarrow e\, \overline{\sigma}_j\, \overline{d}_i}$ iEval-method

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} (D\, \overline{\sigma}_j\, \overline{d}_i).m : \sigma_0 \rightsquigarrow e_0 \tag{1017}$$
$$\text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} e\, \overline{\sigma}_j\, \overline{d}_i : \sigma_0 \rightsquigarrow e'_0 \tag{1018}$$

for some σ_0, e_0 and e'_0. We need to show that there is a e_h such that $e_0 \longrightarrow^* e_h$ and $e'_0 \longrightarrow^* e_h$.

By inversion on Equation 1017, we have $e_0 = ((\Lambda\overline{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.\{m = e_m\})\,\overline{\sigma}_j\,\overline{e}_i).m$ and $\sigma_0 = [[\overline{\sigma}_j/\overline{a}_j]\sigma_q/a]\sigma_m$, for some σ_m, e_m, a, σ'_i, e_m, σ_j and e_i, such that

$$(m : TC\,a : \sigma_m) \in \Gamma_C$$
$$\Gamma_C; \bullet, a \vdash_{ty} \sigma_m \rightsquigarrow \sigma_m$$
$$(D : \forall\overline{a}_j.\overline{Q}_i \Rightarrow TC\,\sigma_q).m \mapsto \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e_m \in \Sigma$$
$$\Sigma_1; \Gamma_C; \bullet, \overline{a}_j, \overline{\delta}_i : \overline{Q}_i \vdash_{tm} e_m : [\sigma_q/a]\sigma_m \rightsquigarrow e_m$$
$$\overline{\Gamma_C; \bullet, \overline{a}_j \vdash_Q Q_i \rightsquigarrow \sigma'_i}^{\,i}$$
$$\overline{\Gamma_C; \bullet \vdash_{ty} \sigma_j \rightsquigarrow \sigma_j}^{\,j}$$
$$\overline{\Sigma; \Gamma_C; \bullet \vdash_d d_i : [\overline{\sigma}_j/\overline{a}_j]Q_i \rightsquigarrow e_i}^{\,i}$$

Because Σ contains a unique method implementation per class instance, we also have

$$e = \Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e_m \tag{1019}$$

Then, term $e\,\overline{\sigma}_j\,\overline{d}_i$, equal to $(\Lambda\overline{a}_j.\lambda\overline{\delta}_i : \overline{Q}_i.e_m)\,\overline{\sigma}_j\,\overline{d}_i$, is deterministcally elaborated to $(\Lambda\overline{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.e_m)\,\overline{\sigma}_j\,\overline{e}_i$.
Indeed, $e_h = [\,\overline{e_i/x_i}^{\,i}\,][\,\overline{\sigma_j/a_j}^{\,j}\,]e_m$ is an appropriate choice, since

$$\begin{aligned}
e_0 &= ((\Lambda\overline{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.\{m = e_m\})\,\overline{\sigma}_j\,\overline{e}_i).m \\
&\longrightarrow (\lambda\,\overline{\delta_i : [\,\overline{\sigma_j/a_j}^{\,j}\,]\sigma'_i}^{\,i}.\{m = [\,\overline{\sigma_j/a_j}^{\,j}\,]e_m\}).m \\
&\longrightarrow \{m = [\,\overline{e_i/\delta_i}^{\,i}\,][\,\overline{\sigma_j/a_j}^{\,j}\,]e_m\}.m \\
&\longrightarrow [\,\overline{e_i/\delta_i}^{\,i}\,][\,\overline{\sigma_j/a_j}^{\,j}\,]e_m
\end{aligned}$$

and

$$\begin{aligned}
e'_0 &= (\Lambda\overline{a}_j.\lambda\,\overline{\delta_i : \sigma'_i}^{\,i}.e_m)\,\overline{\sigma}_j\,\overline{e}_i \\
&\longrightarrow \lambda\,\overline{\delta_i : [\,\overline{\sigma_j/a_j}^{\,j}\,]\sigma'_i}^{\,i}.[\,\overline{\sigma_j/a_j}^{\,j}\,]e_m \\
&\longrightarrow [\,\overline{e_i/\delta_i}^{\,i}\,][\,\overline{\sigma_j/a_j}^{\,j}\,]e_m
\end{aligned}$$

Case iEval-let $\dfrac{}{\Sigma \vdash \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2 \longrightarrow [e_1/x]e_2}$ iEval-let

Hypotheses 1003 and 1004 of the theorem, adapted to this case, are:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e : \sigma' \rightsquigarrow e_0 \tag{1020}$$
$$\text{and } \Sigma; \Gamma_C; \bullet \vdash_{tm} [e_1/x]e_2 : \sigma' \rightsquigarrow e'_0 \tag{1021}$$

for some σ', e_0 and e'_0. We need to show that there exists an e_h such that $e_0 \longrightarrow^* e_h$ and $e'_0 \longrightarrow^* e_h$. We do this by showing that $e_0 \longrightarrow^* e'_0$.
By inversion, the last rule used for Derivation 1020 must be an instance of iTm-let.

$$\frac{\Sigma; \Gamma_C; \bullet \vdash_{tm} e_1 : \sigma \rightsquigarrow e_1 \qquad \Sigma; \Gamma_C; \bullet, x : \sigma \vdash_{tm} e_2 : \sigma' \rightsquigarrow e_2 \qquad \Gamma_C; \bullet \vdash_{ty} \sigma \rightsquigarrow \sigma}{\Sigma; \Gamma_C; \bullet \vdash_{tm} \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e : \sigma' \rightsquigarrow \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2}\ \text{iTm-constrI}$$

where $e_0 = \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2$. From the above equation, we can use the first two premises in Lemma 22, to obtain

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} [e_1/x]e_2 : \sigma' \rightsquigarrow [e_1/x]e_2$$

Then, by uniqueness (Theorem 37 on the latter and on Equation 1021), we have $e'_0 = [e_1/x]e_2$.
We set $e_h = [e_1/x]e_2$, since $(\lambda x : \sigma.e_1)\,e_2 \longrightarrow^* [e_1/x]e_2$, by evaluation rule tEval-AppAbs, and $[e_1/x]e_2 \longrightarrow^* [e_1/x]e_2$, by reflexivity of $\longrightarrow^*$.

□

Lemma 71 ($F_{\{\}}$ Preservation of Values).
If $\Sigma; \Gamma_C; \bullet \vdash_{tm} v : \sigma \rightsquigarrow e$ *then* e *is a value.*

Proof. By straightforward case analysis on the typing derivation.

Theorem 40 (Value Semantic Preservation).
If $\Sigma; \Gamma_C; \bullet \vdash_{tm} e : \sigma \rightsquigarrow e$ *and* $\Sigma \vdash e \longrightarrow^* v$ *then* $\Sigma; \Gamma_C; \bullet \vdash_{tm} v : \sigma \rightsquigarrow v$ *and* $e \simeq v$.

Proof. From the Preservation Theorem 8 and Progress Theorem 9, in combination with the hypothesis, we know that:

$$\Sigma; \Gamma_C; \bullet \vdash_{tm} v : \sigma \rightsquigarrow e'$$

Lemma 71 teaches us that e' is some value v.

The goal follows by repeatedly applying Theorem 39, in combination with the fact that evaluation in both $\boldsymbol{F}_{\mathbf{D}}$ and $\boldsymbol{F}_{\{\}}$ is deterministic (Lemmas 41 and 61).

□