CIRT: Calculus of Inductive Refinement Types

Realizability Semantics of Refinement Type Theory

ALESSIO FERRARINI, IMDEA Software Institute, Spain and Universidad Politécnica de Madrid, Spain
G. A. KAVVOS, University of Bristol, United Kingdom
NIKI VAZOU, IMDEA Software Institute, Spain

Refinement types are a powerful mechanism for statically verifying properties of programs. They have been successfully integrated into practical languages such as Haskell, Scala, and Rust. However, their metatheory is not yet fully understood. In this paper we present the first semantic model for a refinement type system that supports the full feature combination of subtyping, SMT-style equality, impredicative polymorphism, and (stratified) inductive types. Interactions between these features have led to inconsistencies in the past, and existing formalizations of refinement type systems omit at least one of them. We present a realizability model based on partial equivalence relations (PERs) over a Scott domain, which overcomes these obstacles. We use our model to prove the consistency of a core refinement type system and to give a semantic account of the annotation-independence property of refinement types. Our semantics uses two interpretations: a refined one, which interprets types and their refinements as PERs; and an unrefined one, consisting of complete uniform PERs with a bottom element (CUPERs), which models computational behaviour independently of refinements. We extend the model to positive and stratified inductive types, showing that both are consistent in the presence of impredicativity.

1 Introduction

Refinement types [41] extend a host language with logical predicates that constrain the values inhabiting types. For instance, the type $\{x{:}\mathsf{Int} \mid x \neq 0\}$ *refines* the type of integers, allowing us to statically forbid division by zero. Such systems have been integrated into practical languages, including Haskell (Liquid Haskell) [67], Scala (Stainless) [34], and Rust (Flux) [43]. Unlike most foundational type theories, refinement type systems were developed primarily as engineering artifacts: their design was driven by the goal of verifying existing programs with minimal disruption to the underlying language, often accepting a degree of "soundiness" [46] in return for practicability.

Despite their practical success, the metatheory of refinement types remains only partially understood. Over the last decade practical refinement type systems have accumulated several features that interact in non-trivial ways at the semantic level, and potentially lead to inconsistencies if not handled with care. For example, *1) subtyping* combined with *2) unrefined equality* (*i.e.* an SMT-style equality that we describe in Section 2.1.2) and function extensionality has led to inconsistencies in both Liquid Haskell and F* [49]. Additionally, *3) impredicative polymorphism* combined with some classes of *4) non-strictly positive inductive* definitions as supported in Liquid Haskell can be shown to be inconsistent in standard type-theoretic settings [21].

The desire to prevent inconsistency has led to several formalizations of refinement type systems. For example, Swamy et al. [64] prove normalization for a model of F*, but their calculus has no impredicative polymorphism. Hamza et al. [34] prove normalization for a model of Stainless which lacks subtyping. Ghalayini and Krishnaswami [30] construct a model for a system with explicit refinement types, but do not consider inductive types. Finally, Borkowski et al. [14] prove type soundness for a model of Liquid Haskell without higher-order unrefined equalities. As a result, *there is currently no single semantic account that combines the four key features of refinement types.* If we truly want to understand refinement types and prove their soundness, we need a semantic model that supports all of these features simultaneously.

Authors' Contact Information: Alessio Ferrarini, IMDEA Software Institute, Madrid, Spain and Universidad Politécnica de Madrid, Madrid, Spain, alessio.ferrarini@imdea.org; G. A. Kavvos, University of Bristol, Bristol, United Kingdom, alex.kavvos@bristol.ac.uk; Niki Vazou, IMDEA Software Institute, Madrid, Spain, niki.vazou@imdea.org.

Nevertheless, discovering such a semantic model is challenging. The set-theoretic semantics given by Ghalayini and Krishnaswami [30], which interprets each refined type as a subset of the interpretation of its erased type, does not extend to impredicative polymorphism, which has no set-theoretic model in classical set theory [57]; subtyping and unrefined equality further constrain how a model can interpret refined types, as we discuss in Section 2.3. Second, modeling the non-strictly-positive inductive types of Liquid Haskell is not straightforward, as they do not have a direct analogue in type theory.

In this paper we present a model based on *partial equivalence relations (PERs)* over a Scott domain. In broad strokes, our model follows much of the standard structure of PER models: types are interpreted as PERs over the domain and terms as well-defined elements of these PERs [48]. Thus, the semantic counterpart of typing is membership in the domain of the PER, and equality is interpreted as the PER itself. The subtyping relation is interpreted as inclusions between PERs, so no cast operations are needed. We will show that the notion of *unrefined equality* used in refinements may also be modelled by the unrefined PER interpreting a type. Finally, the PERs offer enough structure to model impredicative polymorphism in the usual manner, viz. as set-theoretic intersection over the set of all PERs.

The realizability model that we designed interprets the key features of refinement type systems and is deliberately *elementary* so that it can serve as a reference model against which candidate features are checked *before* they are implemented. Concretely, our contributions are:

- *A model of refinement types as they are, not as they are convenient to model.* We isolate four features of practical refinement type systems (implicit subtyping, SMT-style unrefined equality, impredicative polymorphism, and non-standard inductive types) whose interaction is precisely what has caused soundness bugs in real systems, and we model their *combination* without simplifying any of them away. No existing formalization supports all four (Section 2.2).
- *Modeling unrefined equality.* Every type receives a refined interpretation as a PER, included in an unrefined interpretation as a CUPER over a Scott domain (Sections 4 to 6). This model explains how the SMT-style equality interacts with higher-order refined terms. So, our model does not merely avoid the extensionality inconsistencies of Liquid Haskell and $F^\star$, it *explains* them: they validate refined equalities that cannot coexist with unrefined equality (Section 2.1.2).
- *Consistency of positive and stratified inductive types under impredicativity* (Section 7 and corollary 2). The model shows that positive and stratified types are consistent in a refined setting even with impredicative polymorphism. Our techniques are not specific to refinement types: they can be adapted to any type theory that admits a realizability model, for example MLTT. Ours is moreover the first *denotational* semantics of stratified indexed types, previously accounted for only by normalization or step-indexing arguments [34, 40]. The interpretation of an inductive type is an initial algebra: folding and unfolding form an isomorphism, and the eliminator may expose the recursive arguments, in contrast to Mendler-style *weakly* initial algebras. So, both positive and stratified types share a single, uniform introduction and elimination discipline (Section 7). The message is constructive: positive and stratified indexed types are a straightforward addition to refinement systems.

The rest of the paper is structured as follows. In Section 2 we present the key features of the system, compare them with existing formalizations, and explain why they require a semantic treatment beyond set-theoretic models. Sections 5 to 7 develop the formal construction of the model CIRT and prove consistency. Finally, we present related works and conclude.

Every theorem in this paper is accompanied by a detailed proof in the supplementary material.

2 Refinement types: Syntax & Semantics

We start (Section 2.1) by presenting the key ingredients of practical refinement type systems that are relevant for the semantic development of this paper. Next (Section 2.2), we describe how these features are formalized in the literature and summarize their respective limitations. Finally (Section 2.3), we derive from these features the requirements that guide the design of our model.

2.1 Core Features of refinement type systems

A refinement type $\{x{:}\tau \mid \phi\}$ consists of a base type τ, a proposition ϕ that constrains the values inhabiting τ as the ones satisfying ϕ. The language of propositions ϕ includes boolean-valued expressions of the host language, equality predicates over terms (including equality that may not be decidable), and closure under standard logical connectives. Refinements are not restricted to base types, but may also appear throughout the type structure, for example in (dependent) function types, polymorphic types, and inductive types.

A key aspect of refinement type systems is that refinements act as an *annotation layer* over an existing language: they enrich static checking while leaving the operational semantics unchanged. This design enables deployment without runtime overhead, and further ensures that the behaviour of a program does not depend on refinement annotations. We call the independence of program behavior from verification annotations the *annotation-independence* property.

Refinement type systems are extended with several orthogonal mechanisms that interact in non-trivial ways at the semantic level. Next, we present the four main ingredients that appear in practical systems and that are relevant for the semantic issues studied in this paper.

2.1.1 Subtyping. A refinement that arbitrarily constrains the elements of a type may be naturally regarded as a *subtype* of another type; for example, positive integers are integers. This allows a term to be used at a less-refined or unrefined type without requiring explicit casts. Semantically, subtyping expresses inclusion between the meanings of types: every value inhabiting a subtype must also inhabit its supertype. Importantly, subtyping is *implicit*, meaning that, as presented in the rule Tm-Sub below, no type casts are required to use a term at a supertype. Function types, in turn, satisfy the standard contravariant/covariant subtyping rule Sub-Fun [18, 19].

$$\frac{\Gamma \vdash t{:}\tau \quad \Gamma \vdash \tau \preceq \sigma}{\Gamma \vdash t{:}\sigma}\ \textsc{Tm-Sub} \qquad \frac{\Gamma \vdash \sigma \preceq \tau \quad \Gamma \vdash \rho \preceq \pi}{\Gamma \vdash \tau \to \rho \preceq \sigma \to \pi}\ \textsc{Sub-Fun}$$

2.1.2 Unrefined equality. Refinements often use equality as an unrefined relation on terms, including terms of function type. This reflects the view of the SMT solver, which operates on a representation of these unrefined terms, and derives equalities without refined type information [68].

This notion of *unrefined equality* is the cause of many subtle soundness questions. For example, function extensionality was introduced in both Liquid Haskell [66] and F* [49] and later shown to be inconsistent. The inconsistency arises from the interaction between unrefined equality and subtyping: in a typed setting, equality is indexed by type and extensionality can be formulated so as to be harmless; with unrefined equality, the same axiom has greater force and Vazou and Greenberg [66] showed that admitting it leads to a proof of false.

2.1.3 Parametric Polymorphism. Most refinement type systems are built on top of languages that have parametric polymorphism, such as Haskell or Scala. We model such languages through impredicative type quantification à la System F [31, 58]. Thus, a quantified type has the same kind as the types over which it quantifies. For example, the type $\forall x{:}\mathsf{Ty}.\,\mathsf{Int}$ itself has kind Ty, and may therefore be instantiated with arbitrary types of kind Ty, including itself. This contrasts with

predicative polymorphism, where the quantified type lives at a strictly higher level than the types it quantifies over, in a stratified hierarchy [44].

Refinement types should interact smoothly with polymorphism: it should be possible to refine type variables, instantiate them with arbitrary types, and quantify over indexed type families.

2.1.4 Non-standard Inductive Types. To preserve consistency, general recursion must be excluded from a system of refinement types. However, simply replacing it with some flavour of structural recursion does not suffice, as it is well-known that recursive type definitions can also be used to encode general recursion [36, 54]. The canonical counterexample is:

```
data Bad where MkBad :: (Bad → Void) → Bad
```

where `Void` is the empty type. This type enables encoding self-application and yields a contradiction:

```
bad :: Bad → Void              worst :: Void
bad (MkBad f) = f (MkBad f)    worst = bad (MkBad bad)
```

This term is a typed instance of Curry's paradox; that admitting such negative datatypes breaks normalisation—and hence consistency—in the second-order λ-calculus is due to Mendler [53]. While this code is totally valid Haskell, it is rightfully rejected by Liquid Haskell, which aims to construct a consistent refinement type system. However, strict positivity is too strong a restriction: positive types and stratified negative types could be admitted without compromising consistency.

Positive types allow the inductive type to occur negatively, provided that each negative occurrence is itself under a negative function arrow; a canonical example is Hofmann's routine type [12, 39]:

```
data Rou where Over :: Rou; Next :: ((Rou → [Nat]) → [Nat]) → Rou
```

While it is not immediately obvious, the class of positive types can also lead to inconsistencies in dependently typed systems built on CIC [21]. In contrast, our denotational model proves that they are harmless in a refinement type system.

Stratified types, introduced by Jacob-Rao et al. [40] and also used by Hamza et al. [34], are indexed inductive types whose recursive occurrences appear at 'strictly smaller indices' both in positive and negative positions, ensuring well-foundedness. As an example, consider the shallow embedding of the simply-typed lambda calculus (STLC) in Liquid Haskell (from [25, 26]) where the data types `Typ` and `Val`, respectively, embed types and values of STLC:

```
data Typ = TIota | TArr Typ Typ
data Val where {-@ stratified Val @-}
{-@ VIota :: Int → Ix Val TIota @-}
    VIota :: Int →    Val
{-@ VArr :: t:Typ → u:Typ → (Ix Val t → Ix Val u) → Ix Val (TArr t u) @-}
    VArr ::   Typ →   Typ → (   Val   →    Val  ) →    Val
```

The type `Typ` contains the base type (`TIota`) and the function type (`TArr`). The family `Val` is indexed by such types; intuitively, `Val t` contains exactly the values of object-language type `t`. The `Ix` annotation is Liquid Haskell's notation for attaching such an index to a datatype.

Ignoring the refinement annotations, `Val` can be viewed as the solution of the recursive domain equation of the untyped λ-calculus, which is sufficiently expressive to encode general recursion. The refinement annotations, however, equip values with a type index. A value indexed by `TIota` must be constructed using `VIota`, whereas a value indexed by `TArr t u` must be constructed using `VArr`. The recursive occurrences of `Val` appearing in the argument of `VArr` are indexed by `t` and `u`, which are strict subcomponents of the result index `TArr t u`. Consequently, recursive occurrences only appear at smaller indices, allowing the family `Val` to be interpreted by induction on the structure of the index rather than as an arbitrary recursive domain equation. The `stratified Val` annotation

enables the stratification checker to verify the well-formedness of the type: as explained in section 8, it checks that the indices of the recursive occurrences are indeed strictly smaller than the index of the result type, and accepts the definition of `Val`.

2.2 Existing formalizations

We give a high-level overview of existing formalizations of refinement type systems, and the extent to which they model the features described in Section 2.1. Table 1 summarizes the four formalizations that we discuss: $\mu F^\star$ [64] that models $F^\star$, System FR [34] that models Stainless, λ_{RF} [14] that models Liquid Haskell, and explicit refinement types [30] that models a system with explicit refinement types. For each system we indicate the extent to which it supports the four features of refinement types, and the semantics used to model them.

Subtyping. Many formal models of refinement types, such as explicit refinement types and System FR, do not include explicit subtyping. Instead they rely on a *reflection* rule:

$$\frac{\Gamma \vdash t{:}\tau \quad \Gamma \vdash \phi(t)\ ok}{\Gamma \vdash t{:}\{x{:}\tau \mid \phi\}}\ \textsc{Tm-Refl}$$

Subtyping is then *implicitly* possible, as whenever ϕ implies ψ, any term $\Gamma \vdash t{:}\{x{:}\tau \mid \phi\}$ is also a term of type $\Gamma \vdash t{:}\{x{:}\tau \mid \psi\}$. However, this is *not* equivalent to the explicit formulation. To see why, suppose we have a free variable $f : \mathsf{Int} \to \{x{:}\mathsf{Int} \mid x \geq 0\}$ which we wish to use as a function of type $\{x{:}\mathsf{Int} \mid x \geq 0\} \to \mathsf{Int}$. This cannot be derived from reflection alone, as refinement information cannot flow through type formers. The eta-expansion $\lambda x.\, f \cdot x$ does have this type, a technique that goes back to Barendregt et al. [11], but it is a different term from f. Explicit subtyping, as presented in section 2.1.1, is instead designed with specific rules for each type former, *e.g.* contravariance on the domain of the function.

Equality. Most systems, including those of λ_{RF} and $\mu F^\star$, restrict equality to base types, such as Int. This is sound as the refined equality coincides with the unrefined one on base types. However, it is too restrictive, as it rules out equalities between higher-order terms entirely.

In System FR equality is *reduction-based*, so that $t \equiv u$ if and only if t and u reduce to a common value. This is untyped and by consequence also unrefined, but too fine-grained. For example, reduction almost always uses weak head normal form, and natural equalities at higher-order types do not hold, *e.g.* making $\{x{:}\mathsf{Unit} \mid \lambda y.\, 1+2 \equiv \lambda y.\, 3\} \to \bot$ inhabited. Unfortunately, such equalities arise naturally in refinement types when encoding the equivalence of terminating functional programs.

In explicit refinement types

equality is the typed $\beta\eta$-equality of the underlying simply-typed terms; Ghalayini and Krishnaswami [30] already suggest moving to a semantics based on partial equivalence relations, which our model also uses.

Polymorphism. λ_{RF} allows the refinement of type variables, but they can only be instantiated with base types. Similarly, the calculus of System FR includes indexed types, but quantification ranges only over fully applied types. As the system of $\mu F^\star$ is dependent, they only have *predicative* polymorphism: $\forall x{:}\mathsf{Ty}.\, \mathsf{Int}$ is only a type at the strictly higher level than Ty, and hence it cannot be instantiated with itself. Finally, in explicit refinement types there is no polymorphism.

Inductive types. In System FR they have step-indexed inductive types, which are similar to stratified types in the sense of Jacob-Rao et al. [40], but they do not have a direct analogue of positive types. In the rest of the systems there are no inductive types at all, so no direct analogue of either positive or stratified types.

Table 1. High level comparison of the formalizations of refinement type systems of: $\mu F^\star$ [64], System FR [34], λ_{RF} [14], and explicit refinement types [30].

System	Models	Subtyping 2.1.1	Equality 2.1.2	Polymorphism 2.1.3	Inductive types 2.1.4	Construction
$\mu F^\star$	($F^\star$)	Yes	Base types	Predicative	No	Normalization
System FR	(Stainless)	No	Reduction	Yes	Step-Indexed	Normalization
λ_{RF}	(Liquid Haskell)	Yes	Base types	Yes, Limited	No	Safety
Explicit Refinement Types		No	Typed $\beta\eta$	No	No	Set model

2.3 Design requirements for a realizability model

The simplest models of type theories are set-theoretic: types are interpreted as sets and terms as elements of these sets [37]. The core features described above impose three requirements on the design of our model.

The **first** requirement is support for impredicative polymorphism. Set-theoretic models of impredicative polymorphism cannot exist because of size issues [57], at least not within classical logic [55]. So our model interprets types as PERs over a fixed domain: these form a set, so type quantification can be interpreted as an intersection over them.

The **second** requirement is that subtyping be interpreted as inclusion. It fails in models that interpret each function type as the set of all functions between the interpretations of its domain and codomain: in such a model, subtyping between function types requires *cast operations*. For example, type $\mathsf{Int} \to \mathsf{Int}$ is interpreted as the set of total functional relations $R \subseteq \mathbb{Z} \times \mathbb{Z}$. Computationally, this type is a subtype of $\{x{:}\mathsf{Int} \mid x \geq 0\} \to \mathsf{Int}$. However, the latter is interpreted as the set of total functional relations $R \subseteq \mathbb{N} \times \mathbb{Z}$. The first set is *not* a subset of the second, as it contains pairs $\langle n,\ m\rangle$ with $n < 0$. To pass from one to the other we must *cast*, discarding the offending pairs. This example exhibits the general principle of *coercive subtyping*, which interprets each subtyping judgment as such a cast inserted at every use of subsumption [19]. This is unfortunate on two counts. First, the denotation of a term comes to depend not on the term alone but on the shape of its typing derivation; this is the classical *coherence problem*, and well-definedness can be recovered only by a theorem showing that all derivations of a judgment agree [16, 59]. Second, and more seriously for us, the cast genuinely *removes* elements, so that the meaning of a term comes to depend on the refinements appearing in its type, and annotation-independence is lost. Interpreting subtyping as inclusion requires no casts [18, 28, 48]. This is the case in the subset model of refinement types [51], where a refined type is a subset of its underlying type; our model has the same structure, with PERs in place of raw sets.

The **third** requirement is that the meaning of a term not depend on the refinements in its type, so that unrefined equality can observe its behavior outside them. Unrefined equality cannot be modeled in the presence of the cast operations described above: a cast removes elements from the interpretation of a functional term, and there is no way to recover them to check whether two terms are equal on their unrefined domain.

2.4 Our approach: PERs over a Scott domain

To meet these requirements, we use a model based on *partial equivalence relations* (PERs) over a Scott domain $\mathcal{D}$. Our terms are interpreted as elements of $\mathcal{D}$ and types are interpreted as PERs over $\mathcal{D}$, so that the semantic counterpart of typing is membership in a PER rather than membership in a set, and the equality is interpreted by the relation described by the PER itself. The subtyping relation is interpreted as set inclusion between PERs, so no cast operations are needed and annotation independence holds by construction. The unrefined equality can be modeled by carefully designing

the unrefined PERs so that two terms are related by unrefined equality exactly when they are identified by the unrefined interpretation of their type. Finally, the PERs offer enough structure to model impredicative polymorphism as set-theoretic intersection over the set of all PERs.

3 Background

This section collects the domain-theoretic preliminaries on which our model rests. We follow the standard accounts of Davey and Priestley [23], Abramsky and Jung [5], Gunter [33], Amadio and Curien [8], and Streicher [63], to which we refer the reader for proofs.

Definition 1 (PER). *A partial equivalence relation (PER) over a set A is a symmetric and transitive relation over A, in other words it is a set $S \subseteq A \times A$ such that (i) if $\langle a, b\rangle \in S$ then $\langle b, a\rangle \in S$, and (ii) if $\langle a, b\rangle \in S$ and $\langle b, c\rangle \in S$ then $\langle a, c\rangle \in S$.*

Definition 2 (Domain of a PER). *Given a PER S over A, its* domain *is the set of elements of A which are related to themselves, i.e.* $\mathrm{dom}\,(S) = \{a \in A \mid \langle a, a\rangle \in S\}$.

Definition 3 (Diagonal PER). *The* diagonal PER *of any set A is given by* $\Delta\,(A) = \left\{ \langle a, a\rangle \,\middle|\, a \in A \right\}$.

We will use the notation $\mathcal{PER}\,(A)$ to denote the set of PERs over A. The PER structure will be used to model types, following a long line of work that interprets types as partial equivalence relations over a domain or a combinatory algebra [1, 7, 20]: the domain of a PER gives us the membership information, while the relation itself gives us the equality relation. While the definition of a PER does not give us reflexivity, it satisfies a weaker property that we call partial reflexivity: if $\langle a, b\rangle \in S$ then $\langle a, a\rangle \in S$.

Definition 4 (Pointed PER). *Let A be a set with a distinguished element $\bot \in A$. A* pointed PER over A *is a PER $S \in \mathcal{PER}\,(A)$ such that $\langle \bot, \bot\rangle \in S$*

The point will be used to model a nonterminating element, in the manner of Amadio [7].

Theorem 3.1 (Knaster-Tarski fixed point theorem). *Let $(L, \leq)$ be a complete lattice and $f : L \to L$ be a monotone function. Then the set of fixed points of f forms a complete lattice, and in particular there is a least fixed point* $\mathsf{lfp}\, f = \bigsqcap\{x \in L \mid f(x) \leq x\}$ *and a greatest fixed point* $\mathsf{gfp}\, f = \bigsqcup\{x \in L \mid x \leq f(x)\}$ *[65].*

Definition 5 (Scott domain). *A* Scott domain *is a complete partial order that is moreover ω-*algebraic *and* bounded-complete. *It is ω-algebraic when every element is the directed least upper bound of the countably many* compact *elements below it—an element c is compact just if $c \leq \bigsqcup X$ for a directed X implies $c \leq x$ for some $x \in X$. It is bounded-complete when every subset that has an upper bound has a least upper bound [5, 33, 61].*

Definition 6 (Chain of projections). *A* chain of projections *on a Scott domain D is an increasing family of continuous functions $(\pi_n)_{n\in\mathbb{N}}$, where each $\pi_n : D \to D$ is idempotent and below the identity, with π_0 the constant function $\bot$, and which moreover satisfies $\bigsqcup_n \pi_n = id$.*

*Given an element $d \in D$ we define the n-*th approximation of d *to be $d_n \triangleq \pi_n(d)$.*

In the rest of this section D is a Scott domain equipped with a chain of projections.

To model types we restrict attention to PERs that interact well with this order- and approximation structure, in the manner of the ideal and realizability models of MacQueen et al. [48], Abadi and Plotkin [1], and Amadio [7]. The *uniformity* condition below is the analogue for PERs of their uniform ideals.

Definition 7 (Complete, uniform, and pointed PERs).

(1) *A PER $S \subseteq D \times D$ over a domain D is* complete *whenever every directed subset $\{\langle a_i, b_i\rangle\}_{i\in I} \subseteq S$ (with respect to the product order on $D \times D$) has a least upper bound $\langle \bigsqcup_{i\in I} a_i, \bigsqcup_{i\in I} b_i\rangle$ in S.*
(2) *A PER $S \subseteq D \times D$ over a domain D is* uniform *whenever for every $\langle a, b\rangle \in S$ and every $n \in \mathbb{N}$ we have $\langle \pi_n(a), \pi_n(b)\rangle \in S$.*
(3) *In this paper, a CUPER is a pointed, complete, and uniform PER. We write $\mathcal{CUPER}(D)$ for the collection of such PERs over a domain D.*

Completeness asks that a type be closed under the directed least upper bounds of $D \times D$, so that the limits along which recursive computations converge remain within it and least fixed points can be interpreted there [7]. Uniformity asks instead that membership be stable under approximation: a related pair remains related at every stage π_n. In the presence of completeness, an element then belongs to a type exactly when all of its approximations do.

Recursive and inductive types will be interpreted as fixed points (section 7). We adapt the contractive construction of Amadio [7]: because $\bigsqcup_n \pi_n = id$, a CUPER is determined by its approximations, and a contractive operator—one that sharpens agreement by one approximation per unfolding—has a unique fixed point.

Definition 8 (Approximation of a relation). *The n-th* approximation *of a relation $A \subseteq D \times D$ is its pairwise image under the n-th projection, $A_n \triangleq \left\{ \langle \pi_n(v), \pi_n(w)\rangle \;\middle|\; \langle v, w\rangle \in A \right\}$.*

Lemma 3.2 (Restriction). *Let S be a CUPER over D and write $D_n \triangleq \pi_n(D)$. Then $S_n = S \cap (D_n \times D_n)$. Consequently each S_n is a pointed PER, and $S_j \subseteq S_n$ whenever $j \leq n$.*

Lemma 3.3 (A CUPER is determined by its approximations). *If A and B are CUPERs over D with $A_n = B_n$ for all n, then $A = B$.*

Definition 9 (Non-expansive and contractive operators). *An operator $\Phi : \mathcal{CUPER}(D) \to \mathcal{CUPER}(D)$ is* non-expansive *when $A_n = B_n$ implies $\Phi(A)_n = \Phi(B)_n$ for all n, and* contractive *when $A_n = B_n$ implies $\Phi(A)_{n+1} = \Phi(B)_{n+1}$ for all n.*

Lemma 3.4 (Contractive fixed point). *Every contractive operator $\Phi : \mathcal{CUPER}(D) \to \mathcal{CUPER}(D)$ has a unique fixed point.*

4 Overview of the model

The basis of our model is a Scott domain $\mathcal{D}$, over which we will define PERs.

Definition 10. *We take $\mathcal{D}$ to be the solution to the recursive domain equation*

$$\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}] \oplus (\mathcal{D} \times \mathcal{D}) \oplus \mathbb{N}_\perp \oplus \mathbb{1}_\perp \oplus \mathbb{2}_\perp \oplus \mathcal{L}_\perp$$

where $[\mathcal{D} \to \mathcal{D}]$ is the domain of continuous functions, $\oplus$ is the coalesced sum, $\times$ is the cartesian product, and $\mathbb{N}_\perp$, $\mathbb{1}_\perp$, $\mathbb{2}_\perp$, $\mathcal{L}_\perp$ are the flat domains of natural numbers, the unit, booleans, and labels.

As usual, this domain is obtained as the bilimit of the ω-chain $\mathcal{D}_i$ given by

$$\mathcal{D}_0 = \{\perp\} \qquad \mathcal{D}_{n+1} = [\mathcal{D}_n \to \mathcal{D}_n] \oplus (\mathcal{D}_n \times \mathcal{D}_n) \oplus \mathbb{N} \oplus \mathbb{1} \oplus \mathbb{2} \oplus \mathcal{L} \oplus \{\perp\}$$

along the embedding-projection pairs $i_{n,m} : [\mathcal{D}_n \to \mathcal{D}_m]$ and $j_{n,m} : [\mathcal{D}_m \to \mathcal{D}_n]$ satisfying $j_a \circ i_a = id$ and $i_a \circ j_a \leq id$ [61–63]. Each constructor thus carries an embedding-retraction pair, e.g. $i_\to : [[\mathcal{D} \to \mathcal{D}] \to \mathcal{D}]$ and $r_\to : [\mathcal{D} \to [\mathcal{D} \to \mathcal{D}]]$, satisfying $r_\to \circ i_\to = id$ and $i_\to \circ r_\to \leq id$, and similarly for products, $\mathbb{N}_\perp$, $\mathbb{1}_\perp$, $\mathbb{2}_\perp$, and $\mathcal{L}_\perp$. The projections of definition 6 are $\pi_n = i_n \circ j_n$, so that the n-th approximation of $d \in \mathcal{D}$ is $d_n = i_n(j_n(d))$. They form a chain of projections on $\mathcal{D}$: in particular $\bigsqcup_n \pi_n = id$, which follows from the *minimal invariance* of $\mathcal{D}$ [56], namely that id is the least fixed point of $e \mapsto F(e, e)$, for F the functor of the domain equation.

A central feature of our model is that every type is equipped with two semantics:

- The *refined semantics*, which interprets types as PERs over $\mathcal{D}$ and respects refinements.
- The *unrefined semantics*, which interprets types as CUPERs over $\mathcal{D}$, ignoring refinements.

Following Amadio [7], the distinguished bottom element will account for non-termination and general recursive types that are present in the unrefined semantics.

The two semantics will be related by inclusion: the refined interpretation of each type will be included in its unrefined interpretation. This reflects the intuition that refinements only add logical information without changing the underlying computation. Several constructions in the model, notably function types and equality predicates, rely on the interaction between these two layers.

Since types may be indexed by terms, kinds cannot be interpreted merely as syntactic classifications. Instead, kinds will be interpreted set-theoretically as collections of PER-valued functions respecting the PER structure of their arguments. Context interpretations therefore store both term-level information and type-level information together with projections connecting the refined and unrefined worlds.

5 Basic system

We start by presenting the syntax of (Section 5.1) and semantics (Section 5.2) of the basis of the CIRT calculus, which is a dependent version of System F [10, 47]. We prove the soundness of our semantics (Theorem 5.6) and consistency of the system (Corollary 2). Later we extend this core calculus with refinements (Section 6) and indexed types (Section 7). All theorems are stated once and proved for the complete calculus.

5.1 Syntax

Figure 1 presents the typing rules for the kinds, contexts, types, and terms of the basic system.

Kinds comprise just one base kind Ty, which classifies types. Their presence will become useful in section 7, where they are extended to include indexed kinds for types indexed by terms.

Contexts are built by extension with *term variables* $x :_{\mathrm{T}} \tau$ and *type variables* $x :_{\mathrm{K}} \kappa$.

Types are constructed from type variables, base types (naturals and booleans), dependent functions, and System-F style impredicative quantification. The dependent functions are only allowed to abstract over *term* variables. In this section none of our base types truly exercise dependence, but we will add refinements and indexed types in sections 6 and 7.

Terms include variables, λ-abstraction and application, type abstraction and application, and base terms for naturals and booleans. We omit the operations on base types (arithmetic, comparisons, conditionals): they can be added as constants interpreted by continuous functions that map related arguments to related results, in both the refined and the unrefined interpretation of their type (as will be required by Theorem 5.6). In the application rules, $u\,[t/x]$ denotes the capture avoiding substitution of t for x in u, as is usual in dependently-typed systems. Note that the terms are Curry-style terms, and do *not* contain type annotations.

5.2 Semantics

As mentioned in section 4, every type has two interpretations: the *unrefined semantics* $\llparenthesis - \rrparenthesis$ interprets types as CUPERs over $\mathcal{D}$, forgetting refinements. In contrast, the *refined semantics* $[\![-]\!]$ interprets types as PERs over $\mathcal{D}$, retaining the logical information needed by refinements and indexed types. Both are defined by mutual induction on typing derivations. In summary:

- Contexts are interpreted as PERs over refined or unrefined environments.
- Kinds are interpreted as dependent PERs families in $[\![-]\!]$ and as the constant CUPER family in $\llparenthesis - \rrparenthesis$.

Contexts $\boxed{\vdash \Gamma \text{ ctx}}$

$$\frac{}{\vdash \cdot \text{ ctx}}\ \textsc{Ctx-Empty} \qquad \frac{\vdash \Gamma \text{ ctx} \quad \Gamma \vdash \tau{:}\mathsf{Ty}}{\vdash \Gamma,\ x{:}_{\mathsf{T}}\tau \text{ ctx}}\ \textsc{Ctx-Ty} \qquad \frac{\vdash \Gamma \text{ ctx} \quad \Gamma \vdash \kappa{:}\mathsf{Kd}}{\vdash \Gamma,\ x{:}_{\mathsf{K}}\kappa \text{ ctx}}\ \textsc{Ctx-Kd}$$

Kinds $\boxed{\Gamma \vdash \kappa{:}\mathsf{Kd}}$

$$\frac{\vdash \Gamma \text{ ctx}}{\Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd}}\ \textsc{Kd-Type}$$

Types $\boxed{\Gamma \vdash \tau{:}\kappa}$

$$\frac{\vdash \Gamma \text{ ctx}}{\Gamma \vdash \mathsf{Nat}{:}\mathsf{Ty}}\ \textsc{Ty-Nat} \qquad \frac{\vdash \Gamma \text{ ctx}}{\Gamma \vdash \mathsf{Bool}{:}\mathsf{Ty}}\ \textsc{Ty-Bool} \qquad \frac{\vdash \Gamma \text{ ctx} \quad \Gamma \ni x{:}_{\mathsf{K}}\kappa}{\Gamma \vdash x{:}\kappa}\ \textsc{Ty-Var}$$

$$\frac{\Gamma \vdash \tau{:}\mathsf{Ty} \quad \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}}{\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}}\ \textsc{Ty-Fun} \qquad \frac{\Gamma \vdash \kappa{:}\mathsf{Kd} \quad \Gamma,\ x{:}_{\mathsf{K}}\kappa \vdash \tau{:}\mathsf{Ty}}{\Gamma \vdash \forall x{:}\kappa.\ \tau{:}\mathsf{Ty}}\ \textsc{Ty-All}$$

Terms $\boxed{\Gamma \vdash t{:}\tau}$

$$\frac{\vdash \Gamma \text{ ctx} \quad n \in \mathbb{N}}{\Gamma \vdash n{:}\mathsf{Nat}}\ \textsc{Tm-Nat} \qquad \frac{\vdash \Gamma \text{ ctx} \quad b \in \mathbb{2}}{\Gamma \vdash b{:}\mathsf{Bool}}\ \textsc{Tm-Bool} \qquad \frac{\vdash \Gamma \text{ ctx} \quad \Gamma \ni x{:}_{\mathsf{T}}\tau}{\Gamma \vdash x{:}\tau}\ \textsc{Tm-Var}$$

$$\frac{\Gamma,\ x{:}_{\mathsf{T}}\tau \vdash t{:}\sigma}{\Gamma \vdash \lambda x.\ t{:}\ (x{:}\tau) \to \sigma}\ \textsc{Tm-Lam} \qquad \frac{\Gamma \vdash t{:}\ (x{:}\tau) \to \sigma \quad \Gamma \vdash u{:}\tau}{\Gamma \vdash t \cdot u{:}\sigma\,[u/x]}\ \textsc{Tm-App}$$

$$\frac{\Gamma,\ x{:}_{\mathsf{K}}\kappa \vdash t{:}\tau}{\Gamma \vdash \Lambda x.\ t{:}\forall x{:}\kappa.\ \tau}\ \textsc{Tm-TyLam} \qquad \frac{\Gamma \vdash t{:}\forall x{:}\kappa.\ \tau \quad \Gamma \vdash \sigma{:}\kappa}{\Gamma \vdash t\ \sigma{:}\tau\,[\sigma/x]}\ \textsc{Tm-TyApp}$$

Fig. 1. Typing rules of basic CIRT.

• Types are interpreted as context-indexed PERs in $[\![-]\!]$ and context-indexed CUPERs in $\lfloor\!\lfloor -\rfloor\!\rfloor$.

Terms have only one interpretation, viz. as elements of $\mathcal{D}$. It is defined (in section 5.2.4) by induction on the raw syntax of Curry-style terms and is denoted by $(\!|-|\!)$. Concretely, for each context Γ such that t is typed, the interpretation of t has the following signature:

$$(\!|t|\!)\ (-) : (\!|\vdash \Gamma \text{ ctx}|\!) \to \mathcal{D}$$

where $(\!|\vdash \Gamma \text{ ctx}|\!)$ is the set of *runtime environments* determined by Γ, which only contain values:

$$(\!|\vdash \cdot \text{ ctx}|\!) \triangleq \mathbb{1}, \quad (\!|\vdash \Gamma,\ x{:}_{\mathsf{T}}\tau \text{ ctx}|\!) \triangleq (\!|\vdash \Gamma \text{ ctx}|\!) \times \mathcal{D}, \quad (\!|\vdash \Gamma,\ x{:}_{\mathsf{K}}\kappa \text{ ctx}|\!) \triangleq (\!|\vdash \Gamma \text{ ctx}|\!)$$

Notation: In the rest of the paper, we use the ‖ delimiter to mark the definitions of interpretations.

We keep this carrier separate because terms are Curry-style: their raw denotation is independent of typing derivations. Later we will equip runtime environments with a PER $\|\Gamma\|\ (-)$, which we use to state type soundness: for any $r, r' \in (\!|\vdash \Gamma \text{ ctx}|\!)$,

$$\Gamma \vdash t{:}\tau,\ \gamma \in \mathrm{dom}\,(\lfloor\!\lfloor \vdash \Gamma \text{ ctx} \rfloor\!\rfloor),\ \langle r, r' \rangle \in \|\Gamma\|\ (\gamma) \quad \Longrightarrow \quad \langle (\!|t|\!)\ (r),\ (\!|t|\!)\ (r') \rangle \in \lfloor\!\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\!\rfloor\ (\gamma)\,.$$

Thus the two type semantics constrain the same ‘runtime’ computations.

This distinction also appears in contexts. The unrefined context semantics keeps type-variable assignments, but does not store term-variable values. This reflects that unrefined type interpretation erases term dependency. The refined context semantics keeps term assignments, since refinements and indexed types may depend on them.

5.2.1 Contexts. In the (un)refined semantics contexts are interpreted as PERs over *environments*. The empty context is interpreted as the diagonal PER on the empty environment, i.e. the singleton:

$$\llbracket \vdash \cdot \ \mathsf{ctx} \rrbracket \triangleq \Delta(\mathbb{1}) \qquad \llparenthesis \vdash \cdot \ \mathsf{ctx} \rrparenthesis \triangleq \Delta(\mathbb{1})$$

Context extension by a term variable is interpreted as the usual dependent sum in the refined interpretation, whereas it is silently dropped in the unrefined one:

$$\begin{aligned} \llparenthesis \vdash \Gamma,\, x{:}_{\mathsf{T}}\tau \ \mathsf{ctx} \rrparenthesis &\triangleq \llparenthesis \vdash \Gamma \ \mathsf{ctx} \rrparenthesis \\ \llbracket \vdash \Gamma,\, x{:}_{\mathsf{T}}\tau \ \mathsf{ctx} \rrbracket &\triangleq \left\{ \langle \langle \gamma,\, v \rangle,\, \langle \delta,\, w \rangle \rangle \,\middle|\, \langle \gamma,\, \delta \rangle \in \llbracket \vdash \Gamma \ \mathsf{ctx} \rrbracket,\, \langle v,\, w \rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \right\} \end{aligned}$$

This distinction captures the fact that the computational behaviour of this calculus is actually not dependent at all and that the refinement types exercise dependence only on the logical level.

Finally, extending a context with a type variable in the unrefined semantics simply stores an environment compatible with the interpretation of the kind.

$$\llparenthesis \vdash \Gamma,\, x{:}_{\mathsf{K}}\kappa \ \mathsf{ctx} \rrparenthesis \triangleq \left\{ \langle \langle \gamma,\, P \rangle,\, \langle \delta,\, P \rangle \rangle \,\middle|\, \langle \gamma,\, \delta \rangle \in \llparenthesis \vdash \Gamma \ \mathsf{ctx} \rrparenthesis,\, P \in \llparenthesis \Gamma \vdash \kappa{:}\mathsf{Kd} \rrparenthesis\,(\gamma) \right\}$$

The refined semantics must also determine the compatible unrefined environment. Thus, a type-variable extension stores both a refined kind element and an unrefined over-approximation of the same type variable; the former belongs to the refined interpretation of the kind, while the latter belongs to the unrefined one.

$$\llbracket \vdash \Gamma,\, x{:}_{\mathsf{K}}\kappa \ \mathsf{ctx} \rrbracket \triangleq \left\{ \langle \langle \gamma,\, \langle P,\, P_\perp \rangle \rangle,\, \langle \delta,\, \langle P,\, P_\perp \rangle \rangle \rangle \,\middle|\, \begin{array}{l} \langle \gamma,\, \delta \rangle \in \llbracket \vdash \Gamma \ \mathsf{ctx} \rrbracket,\, P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma), \\ P_\perp \in \llparenthesis \Gamma \vdash \kappa{:}\mathsf{Kd} \rrparenthesis\,(\iota_\Gamma\,(\gamma)),\, \pi_{\Gamma;\,\kappa}\,(\gamma,\, P) \subseteq P_\perp \end{array} \right\}$$

The function $\iota_\Gamma\,(-)$ is a *context projection* operation, which maps each refined environment to its unrefined version. Given a context $\vdash \Gamma \ \mathsf{ctx}$ it is defined by induction on Γ as follows:

$$\iota_\Gamma\,(-) : \mathsf{dom}\,(\llbracket \vdash \Gamma \ \mathsf{ctx} \rrbracket) \to \mathsf{dom}\,(\llparenthesis \vdash \Gamma \ \mathsf{ctx} \rrparenthesis)$$

$$\iota.\,(\star) \triangleq \star \qquad \iota_{\Gamma,\, x{:}_{\mathsf{T}}\tau}\,(\langle \gamma,\, v \rangle) \triangleq \iota_\Gamma\,(\gamma) \qquad \iota_{\Gamma,\, x{:}_{\mathsf{K}}\kappa}\,(\langle \gamma,\, \langle P,\, P_\perp \rangle \rangle) \triangleq \langle \iota_\Gamma\,(\gamma),\, P_\perp \rangle$$

The empty context is projected to the empty context; values are removed; and a (refined, unrefined) pair of PERs is mapped to the unrefined PER. It satisfies the expected compatibility condition:

$$\langle \gamma,\, \delta \rangle \in \llbracket \vdash \Gamma \ \mathsf{ctx} \rrbracket \implies \langle \iota_\Gamma\,(\gamma),\, \iota_\Gamma\,(\delta) \rangle \in \llparenthesis \vdash \Gamma \ \mathsf{ctx} \rrparenthesis.$$

Definition 11 (Type projection). *The companion type projection $\pi_{\Gamma;\,\kappa}\,(-,\,-)$ extracts the underlying relation on $\mathcal{D}$ from a refined type interpretation, so that it can be compared with an unrefined interpretation. In the basic system there is only the kind* Ty*, so the projection is just the identity:*

$$\begin{aligned} \pi_{\Gamma;\,\kappa}\,(-,\,-) &: (\gamma : \mathsf{dom}\,(\llbracket \vdash \Gamma \ \mathsf{ctx} \rrbracket)) \to \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma) \to \mathcal{P}(\mathcal{D} \times \mathcal{D}) \\ \pi_{\Gamma;\,\mathsf{Ty}}\,(\gamma,\, P) &\triangleq P \end{aligned}$$

For the base kind Ty *the projection is the identity, so its value is in fact a PER.*

5.2.2 Kinds. Kinds are interpreted as families of PERs in the refined semantics and as constant families of CUPERs in the unrefined semantics. Since $\mathcal{D}$ is a set, the PERs over it form a set $\mathcal{PER}\,(\mathcal{D})$, which is a complete lattice under inclusion, and the CUPERs form a subset $\mathcal{CUPER}\,(\mathcal{D})$ of it.

The semantics of kinds takes as argument an environment and returns the set of all possible interpretations of types of that kind in that context:

$$\llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(-) : \mathsf{dom}\,(\llbracket \vdash \Gamma \ \mathsf{ctx} \rrbracket) \to \mathbf{Set} \qquad \llparenthesis \Gamma \vdash \kappa{:}\mathsf{Kd} \rrparenthesis\,(-) : \mathsf{dom}\,(\llparenthesis \vdash \Gamma \ \mathsf{ctx} \rrparenthesis) \to \mathbf{Set}$$

For the base kind this is just all PERs and CUPERs respectively:

$$\llbracket \Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd} \rrbracket\,(\gamma) \triangleq \mathcal{PER}\,(\mathcal{D}) \qquad \llparenthesis \Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd} \rrparenthesis\,(\gamma) \triangleq \mathcal{CUPER}\,(\mathcal{D})$$

The PERs and CUPERs are meant to be a notion of equality between realizers. We can show that this semantics respects this notion of equality, mapping related environments to related kinds:

Theorem 5.1 (Kinds respect PERs). *Let* $\vdash \Gamma\ \mathsf{ctx}$ *and* $\Gamma \vdash \kappa{:}\mathsf{Kd}$. *Then:*
(i) Refined: If $\langle \gamma, \delta \rangle \in [\![\vdash \Gamma\ \mathsf{ctx}]\!]$, *then* $[\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\ (\gamma) = [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\ (\delta)$.
(ii) Unrefined: If $\langle \gamma, \delta \rangle \in \llbracket\vdash \Gamma\ \mathsf{ctx}\rrbracket$, *then* $\llbracket\Gamma \vdash \kappa{:}\mathsf{Kd}\rrbracket\ (\gamma) = \llbracket\Gamma \vdash \kappa{:}\mathsf{Kd}\rrbracket\ (\delta)$.

One of our goals is to prove that the refined interpretation of types is included in the unrefined one, but we first define how we relate them. The type projection of definition 11 maps a refined interpretation at a kind to the relation over $\mathcal{D}$ that it determines. Since $\mathcal{CUPER}\ (\mathcal{D})$ are relations over $\mathcal{D}$, this lets us ask whether a refined interpretation is included in an unrefined one.

5.2.3 Types. Types are interpreted as dependent functions or sections over those families: they map environments to an element of the appropriate kind:

$$[\![\Gamma \vdash \tau{:}\kappa]\!]\ (-) : (\gamma : \mathsf{dom}\ ([\![\vdash \Gamma\ \mathsf{ctx}]\!])) \to [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\ (\gamma)$$
$$\llbracket\Gamma \vdash \tau{:}\kappa\rrbracket\ (-) : (\gamma : \mathsf{dom}\ (\llbracket\vdash \Gamma\ \mathsf{ctx}\rrbracket)) \to \llbracket\Gamma \vdash \kappa{:}\mathsf{Kd}\rrbracket\ (\gamma)$$

Base types are interpreted as the PERs over $\mathcal{D}$ that relate exactly the elements of $\mathcal{D}$ that correspond to the values of the type.

$$[\![\Gamma \vdash \mathsf{Nat{:}Ty}]\!]\ (\gamma) \triangleq \left\{ \langle i_{\mathbb{N}}\ (n)\,,\ i_{\mathbb{N}}\ (n)\rangle \,\middle|\, n \in \mathbb{N} \right\} \quad [\![\Gamma \vdash \mathsf{Bool{:}Ty}]\!]\ (\gamma) \triangleq \left\{ \langle i_{\mathbb{2}}\ (b)\,,\ i_{\mathbb{2}}\ (b)\rangle \,\middle|\, b \in \mathbb{2} \right\}$$

where $\mathcal{B}$ ranges over the base sets $\mathbb{N}$, $\mathbb{1}$, $\mathbb{2}$, and $\mathcal{L}$, and $i_{\mathcal{B}}\ (v)$ denotes the continuous embedding of $v \in \mathcal{B}$ into the domain $\mathcal{D}$ (per definition 10).

The unrefined interpretation extends the basic PERs by adding the pair $\langle \bot, \bot \rangle$:

$$\llbracket\Gamma \vdash \mathsf{Nat{:}Ty}\rrbracket\ (\gamma) \triangleq \left\{ \langle i_{\mathbb{N}}\ (n)\,,\ i_{\mathbb{N}}\ (n)\rangle \,\middle|\, n \in \mathbb{N} \right\} \cup \{\langle \bot, \bot \rangle\}$$
$$\llbracket\Gamma \vdash \mathsf{Bool{:}Ty}\rrbracket\ (\gamma) \triangleq \left\{ \langle i_{\mathbb{2}}\ (b)\,,\ i_{\mathbb{2}}\ (b)\ \rangle \,\middle|\, b \in \mathbb{2} \right\} \cup \{\langle \bot, \bot \rangle\}$$

Since basic types are flat domains in the domain equation that describes $\mathcal{D}$, adding the pair $\langle \bot, \bot \rangle$ is enough to obtain pointedness, completeness and uniformity.

Function types have a more interesting interpretation, as they relate functions that are 'extensionally' equal, *i.e.* map related arguments to related outputs. In the unrefined case:

$$\llbracket\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}\rrbracket\ (\gamma) \triangleq \left\{ \langle f, g\rangle \,\middle|\, \forall\, \langle v, w\rangle \in \llbracket\Gamma \vdash \tau{:}\mathsf{Ty}\rrbracket\ (\gamma)\,.\ \langle f \cdot v, g \cdot w\rangle \in \llbracket\Gamma, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}\rrbracket\ (\gamma) \right\}$$

One would naïvely expect the refined interpretation to mimic the unrefined one, replacing $\llbracket-\rrbracket$ with $[\![-]\!]$. However, this would equate functions whose domains have been restricted by refinements. If the refined function PER ignored the unrefined behavior, it would be impossible to prove that the refined interpretation is included in the unrefined one (per theorem 5.4), as it would equate functions that are not related in the unrefined interpretation. Hence, we impose in the refined interpretation the extra condition (ii) below that ensures compatibility with the unrefined interpretation:

$$[\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!]\ (\gamma)$$
$$\triangleq \left\{ \langle f, g\rangle \,\middle|\, \begin{array}{ll} \forall\ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\gamma)\,. & \langle f \cdot v, g \cdot w\rangle \in [\![\Gamma, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\ (\langle \gamma, v\rangle) \quad \text{(i)} \\ \forall \langle v_\bot, w_\bot\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\iota_\Gamma\ (\gamma))\,. & \langle f \cdot v_\bot, g \cdot w_\bot\rangle \in \llbracket\Gamma, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}\rrbracket\ (\iota_\Gamma\ (\gamma)) \quad \text{(ii)} \end{array} \right\}$$

The unrefined relatedness condition also excludes semantic functions that do not correspond to any term that is well typed in the underlying type system. For example consider a function that returns values of different types depending on the input. Concretely $f(0) = 0$ and $f(n) = \mathit{tt}$ for $n \geq 1$. f is not in the refined interpretation of the type $\{x{:}\mathsf{Nat} \mid x \geq 1\} \to \mathsf{Bool}$ even though it satisfies the refined condition because it violates the unrefined relatedness condition.

Type Quantification. Impredicative type quantification is interpreted by intersection over all candidates in the interpretation of the quantified kind. In the unrefined case, when $\kappa = \mathsf{Ty}$, it is defined to be the intersection over all CUPERs.

$$\llbracket \Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \triangleq \left\{ \langle v,\, w\rangle \,\middle|\, \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)\,.\ \langle v,\, w\rangle \in \llbracket \Gamma,\, x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\, P\rangle) \right\}$$

For example, in the unrefined interpretation, an element belongs to the polymorphic identity type $\forall x{:}\mathsf{Ty}.\, x \to x$ exactly when it behaves as an identity function (or one of its approximations $\pi_n(id)$) for every CUPER chosen as the interpretation of x.

The refined interpretation follows the all-PER reading of System F [58], but adds a compatibility constraint between the refined candidates and their unrefined over-approximations.

$$\llbracket \Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \triangleq \left\{ \langle v,\, w\rangle \,\middle|\, \begin{array}{c} \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)\,,\ \forall P_{\perp} \in \lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\,(\iota_{\Gamma}\,(\gamma)) \\ \pi_{\Gamma;\,\kappa}\,(\gamma,\, P) \subseteq P_{\perp} \Rightarrow \langle v,\, w\rangle \in \llbracket \Gamma,\, x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\, \langle P,\, P_{\perp}\rangle\rangle) \end{array} \right\}$$

Type variables are interpreted by lookup in the context interpretation; while in the refined semantics the refined component is projected:

$$\llbracket \Gamma \vdash x{:}\kappa \rrbracket\,(\gamma) \triangleq P \quad \text{where } \gamma(x) = \langle P,\, P_{\perp}\rangle \qquad \lfloor \Gamma \vdash x{:}\kappa \rfloor\,(\gamma) \triangleq P \quad \text{where } \gamma(x) = P$$

In both semantics, the interpretation of types belongs to the interpretation of their kind:

Theorem 5.2 (Type interpretation is well-formed). *Let* $\vdash \Gamma\ \mathsf{ctx}$, $\Gamma \vdash \kappa{:}\mathsf{Kd}$, $\Gamma \vdash \tau{:}\kappa$ *be well-formed context, kind, and type judgments then:*

(i) Refined: If $\gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket)$, *then* $\llbracket \Gamma \vdash \tau{:}\kappa \rrbracket\,(\gamma) \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)$.

(ii) Unrefined: If $\gamma \in \mathsf{dom}\,(\lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor)$, *then* $\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\,(\gamma) \in \lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\,(\gamma)$.

Following the same reasoning behind Theorem 5.1, the semantics respects the notion of equality, since equal contexts map to equal interpretations of types.

Theorem 5.3 (Types respect PERs). *Let* $\vdash \Gamma\ \mathsf{ctx}$, $\Gamma \vdash \kappa{:}\mathsf{Kd}$, $\Gamma \vdash \tau{:}\kappa$ *be well-formed context, kind, and type judgments, then:*

(i) Refined: If $\langle \gamma,\, \delta\rangle \in \llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket$, *then* $\llbracket \Gamma \vdash \tau{:}\kappa \rrbracket\,(\gamma) = \llbracket \Gamma \vdash \tau{:}\kappa \rrbracket\,(\delta)$.

(ii) Unrefined: If $\langle \gamma,\, \delta\rangle \in \lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor$, *then* $\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\,(\gamma) = \lfloor \Gamma \vdash \tau{:}\kappa \rfloor\,(\delta)$.

Last, we prove that indeed the refined interpretation of types is included in the unrefined one, since in general kinds are interpreted as families of PERs, where the type projection operation is used to relate the two interpretations.

Theorem 5.4 (Refined interpretation included in unrefined interpretation). *Let* $\vdash \Gamma\ \mathsf{ctx}$, $\Gamma \vdash \tau{:}\kappa$ *be well-formed context and type judgments, and* $\gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket)$, *then*

$$\pi_{\Gamma;\,\kappa}\,(\gamma,\, \llbracket \Gamma \vdash \tau{:}\kappa \rrbracket\,(\gamma)) \subseteq \lfloor \Gamma \vdash \tau{:}\kappa \rfloor\,(\iota_{\Gamma}\,(\gamma))$$

As a corollary, unrefined equality is invariant under replacement by refined-equal representatives. In other words, although the unrefined interpretation of a type may relate more pairs than the refined interpretation, the unrefined equality test is well-defined on refined equivalence classes. This follows from the inclusion above and symmetry and transitivity of the unrefined PER.

Corollary 1 (Unrefined equality respects refined equality). *Let* $\vdash \Gamma\ \mathsf{ctx}$ *be a well-formed context, let* $\Gamma \vdash \tau{:}\mathsf{Ty}$ *be a well-formed type, and let* $\gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket)$.

If $\langle v,\, v'\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$ *and* $\langle w,\, w'\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$, *then*

$$\langle v,\, w\rangle \in \lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\iota_{\Gamma}\,(\gamma)) \quad \Longleftrightarrow \quad \langle v',\, w'\rangle \in \lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\iota_{\Gamma}\,(\gamma))\,.$$

5.2.4 Terms. Terms are interpreted directly as elements of $\mathcal{D}$. The interpretation is defined on raw terms rather than on typing derivations and depends only on the assignment of term variables. Type assignments are therefore erased via a runtime projection. This exemplifies the fact that the semantics is *annotation independent* as neither typing judgment, derivation nor type annotations can influence the interpretation of terms.

To begin, we define a function that projects away all the type assignments from a context interpretation, keeping only the term assignments.

Definition 12 (Runtime Projection). *Let* $\vdash \Gamma\ \mathsf{ctx}$. *We define* runtime projection *from refined environments into runtime environments by induction on* Γ*:*

$$\rho_{(-)} : \mathsf{dom}\left(\llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket\right) \to \left(\!\left| \vdash \Gamma\ \mathsf{ctx} \right|\!\right)$$

$$\rho_{\cdot}\left(\star\right) \triangleq \star \qquad \rho_{\Gamma, x :_{\mathsf{T}} \tau}\left(\langle \gamma, v \rangle\right) \triangleq \langle \rho_{\Gamma}\left(\gamma\right), v \rangle \qquad \rho_{\Gamma, x :_{\kappa} \kappa}\left(\langle \gamma, \langle P, P_{\perp} \rangle \rangle\right) \triangleq \rho_{\Gamma}\left(\gamma\right)$$

This projection discards type assignments, as types are erased at runtime.

Literals are then interpreted as their canonical embeddings into $\mathcal{D}$.

$$\left(\!\left| n \right|\!\right)\left(\gamma\right) \triangleq i_{\mathbb{N}}\left(n\right) \qquad \left(\!\left| b \right|\!\right)\left(\gamma\right) \triangleq i_{\mathbb{2}}\left(b\right)$$

Variables are interpreted by projection from the runtime environment, λ-abstractions as the embedding of continuous functions, and application as application in $\mathcal{D}$:

$$\left(\!\left| x \right|\!\right)\left(\gamma\right) \triangleq \gamma\left(x\right) \qquad \left(\!\left| \lambda x.\, t \right|\!\right)\left(\gamma\right) \triangleq i_{\to}\left(v \mapsto \left(\!\left| t \right|\!\right)\left(\langle \gamma, v \rangle\right)\right) \qquad \left(\!\left| t \cdot u \right|\!\right)\left(\gamma\right) \triangleq \left(\!\left| t \right|\!\right)\left(\gamma\right) \cdot \left(\!\left| u \right|\!\right)\left(\gamma\right)$$

Type abstraction and application have no computational content, so they are ignored:

$$\left(\!\left| \Lambda x.\, t \right|\!\right)\left(\gamma\right) \triangleq \left(\!\left| t \right|\!\right)\left(\gamma\right) \qquad \left(\!\left| t\ \tau \right|\!\right)\left(\gamma\right) \triangleq \left(\!\left| t \right|\!\right)\left(\gamma\right)$$

As usual we prove that the semantics is PER-respecting. In particular, the interpretation of a term should not distinguish between related contexts. For the unrefined semantics, the context interpretation $\lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor$ only contains the erased type-level environment. It does not contain the runtime environment used by $\left(\!\left| - \right|\!\right)$. We therefore need to define $\|\Gamma\|\left(\gamma\right)$ as the PER on $\left(\!\left| \vdash \Gamma\ \mathsf{ctx} \right|\!\right)$ induced by the unrefined types in Γ.

Definition 13 (Unrefined runtime assignment). *Let* $\vdash \Gamma\ \mathsf{ctx}$. *The* unrefined runtime assignment *over* $\gamma \in \mathsf{dom}\left(\lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor\right)$ *is a PER on* $\left(\!\left| \vdash \Gamma\ \mathsf{ctx} \right|\!\right)$*, defined as* $\|\Gamma\|\left(\gamma\right)$*:*

$$\begin{aligned}
&\|\Gamma\|\left(-\right) && : \mathsf{dom}\left(\lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor\right) \to \mathcal{PER}\left(\left(\!\left| \vdash \Gamma\ \mathsf{ctx} \right|\!\right)\right) \\
&\|\cdot\|\left(\star\right) && \triangleq \Delta\left(\mathbb{1}\right) \\
&\|\Gamma, x :_{\mathsf{T}} \tau\|\left(\gamma\right) && \triangleq \left\{ \langle \langle r, v \rangle, \langle r', w \rangle \rangle \,\middle|\, \langle r, r' \rangle \in \|\Gamma\|\left(\gamma\right), \langle v, w \rangle \in \lfloor \Gamma \vdash \tau : \mathsf{Ty} \rfloor\left(\gamma\right) \right\} \\
&\|\Gamma, x :_{\kappa} \kappa\|\left(\langle \gamma, P \rangle\right) && \triangleq \|\Gamma\|\left(\gamma\right)
\end{aligned}$$

Related refined runtime projections are related by the unrefined runtime assignment relation.

Lemma 5.5 (Refined context runtime projection is unrefined context runtime-related). *Let* $\vdash \Gamma\ \mathsf{ctx}$. *If* $\langle \gamma, \delta \rangle \in \llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket$*, then* $\langle \rho_{\Gamma}\left(\gamma\right), \rho_{\Gamma}\left(\delta\right) \rangle \in \|\Gamma\|\left(\iota_{\Gamma}\left(\gamma\right)\right)$.

We are now ready to state that both the refined and unrefined term semantics are PER-respecting.

Theorem 5.6 (Semantics is PER-respecting (terms)). *Let* $\vdash \Gamma\ \mathsf{ctx}$ *and* $\Gamma \vdash t : \tau$. *Then:*

(i) Refined: If $\langle \gamma, \delta \rangle \in \llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket$*, then* $\langle \left(\!\left| t \right|\!\right)\left(\rho_{\Gamma}\left(\gamma\right)\right), \left(\!\left| t \right|\!\right)\left(\rho_{\Gamma}\left(\delta\right)\right) \rangle \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\left(\gamma\right)$.

(ii) Unrefined: If $\gamma \in \mathsf{dom}\left(\lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor\right)$*;* $\langle \delta, \epsilon \rangle \in \|\Gamma\|\left(\gamma\right)$*, then* $\langle \left(\!\left| t \right|\!\right)\left(\delta\right), \left(\!\left| t \right|\!\right)\left(\epsilon\right) \rangle \in \lfloor \Gamma \vdash \tau : \mathsf{Ty} \rfloor\left(\gamma\right)$.

Consistency of the syntax then is a corollary of Theorem 5.6 (i) . We represent false as the type $\forall x{:}\mathsf{Ty}.\, x$ as from Tm-TyApp it can be used to inhabit every other type.

Corollary 2 (Relative consistency). *There is no term* t *such that* $\cdot \vdash t : \forall x{:}\mathsf{Ty}.\, x$ *is derivable.*

We show that the type is not inhabited as it would imply the existence of an element of $\emptyset$.

Propositions $\boxed{\Gamma \vdash \phi \text{ prp}}$

$$\frac{\Gamma \vdash \phi \text{ prp} \quad \Gamma \vdash \psi \text{ prp}}{\Gamma \vdash \phi \wedge \psi \text{ prp}} \textsc{Prop-And} \qquad \frac{\Gamma \vdash \phi \text{ prp}}{\Gamma \vdash \neg\phi \text{ prp}} \textsc{Prop-Not} \qquad \frac{\Gamma \vdash t{:}\tau \quad \Gamma \vdash u{:}\tau}{\Gamma \vdash t \equiv_\tau u \text{ prp}} \textsc{Prop-Eq}$$

Types continued $\boxed{\Gamma \vdash \tau{:}\kappa}$

$$\frac{\Gamma \vdash \tau{:}\mathsf{Ty} \quad \Gamma,\, x{:}_\mathsf{T}\tau \vdash \phi \text{ prp}}{\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}} \textsc{Ty-Ref}$$

Terms continued $\boxed{\Gamma \vdash t{:}\tau}$

$$\frac{\Gamma \vdash t{:}\tau}{\Gamma \vdash t{:}\{x{:}\tau \mid x \equiv_\tau t\}} \textsc{Tm-Refl} \qquad \frac{\Gamma \vdash t{:}\tau \quad \Gamma \vdash \tau \preceq \sigma \quad \Gamma \vdash \sigma{:}\mathsf{Ty}}{\Gamma \vdash t{:}\sigma} \textsc{Tm-Sub}$$

Subtyping $\boxed{\Gamma \vdash \tau \preceq \sigma}$

$$\frac{}{\Gamma \vdash \tau \preceq \{x{:}\tau \mid tt \equiv_{\mathsf{Bool}} tt\}} \textsc{Sub-Top} \qquad \frac{\Gamma,\, x{:}_\mathsf{T}\tau \models \phi \implies \psi}{\Gamma \vdash \{x{:}\tau \mid \phi\} \preceq \{x{:}\tau \mid \psi\}} \textsc{Sub-Impl}$$

$$\frac{\Gamma \vdash \sigma \preceq \tau \quad \Gamma,\, x{:}_\mathsf{T}\sigma \vdash \rho \preceq \pi}{\Gamma \vdash (x{:}\tau) \to \rho \preceq (x{:}\sigma) \to \pi} \textsc{Sub-Fun} \qquad \frac{\Gamma \vdash \tau \preceq \sigma}{\Gamma \vdash \{x{:}\tau \mid \phi\} \preceq \{x{:}\sigma \mid \phi\}} \textsc{Sub-Refine}$$

$$\frac{}{\Gamma \vdash \tau \preceq \tau} \textsc{Sub-Refl} \qquad \frac{}{\Gamma \vdash \{x{:}\{y{:}\tau \mid \psi\} \mid \phi\} \preceq \{y{:}\tau \mid \psi \wedge \phi\,[y/x]\}} \textsc{Sub-Flat}$$

$$\frac{}{\Gamma \vdash \{x{:}\tau \mid \phi\} \preceq \tau} \textsc{Sub-Forget} \qquad \frac{\Gamma,\, x{:}_\kappa\kappa \vdash \tau \preceq \sigma}{\Gamma \vdash \forall x{:}\kappa.\, \tau \preceq \forall x{:}\kappa.\, \sigma} \textsc{Sub-All}$$

$$\frac{\Gamma \vdash \tau \preceq \sigma \quad \Gamma \vdash \sigma{:}\mathsf{Ty} \quad \Gamma \vdash \sigma \preceq \rho}{\Gamma \vdash \tau \preceq \rho} \textsc{Sub-Trans}$$

Fig. 2. Typing rules of refined CIRT.

6 Refinement types

In this section we extend CIRT with refinement types. The extension includes a proposition judgment together with reflection and subtyping rules and an SMT validity judgment. Section 6.1 presents the syntax of the calculus and Section 6.2 presents the semantics.

6.1 Syntax

Figure 2 extends the syntax of CIRT with refinement types. It shows only the new rules.

Propositions form the syntax of refinements. They are built from term equality (at any type) and the standard boolean connectives. Since propositions form a separate layer on top of the programming language their semantics are not required to be continuous. This permits forming equalities between terms at any type (using rule Prop-Eq). Any boolean-valued term of the language can be embedded as a proposition by equating it to tt.

This propositional logic is interpreted classically: a proposition is meant to denote either tt or ff. We can therefore define all of propositional logic from conjunction and negation, with the other connectives being defined in terms of those.

Types, of the previous section, are extended by the *refinement type* $\{x{:}\tau \mid \phi\}$, which can be thought of as the subset of elements of τ satisfying ϕ.

Terms. Refinements add two new typing rules on terms: *reflection* and *subtyping*. Reflection (via rule Tm-Refl) asserts that a term satisfies the proposition that it is equal to itself. This is the strongest proposition a term can satisfy and is therefore to increase the precision of type checking [42]. Subtyping (via rule Tm-Sub) allows us to use a term at a 'larger' type.

Subtyping and SMT. The judgment $\Gamma \models \phi$ models the discharge of propositions by an SMT solver. It is used to give meaning to the Sub-Impl rule, which, following the subtyping rules of [41], reduces subtyping to implication between propositions. In addition, subtyping is reflexive and transitive, a refinement type is a subtype of the type it refines, and any term of any type satisfies the trivial proposition. Function types are contravariant in their argument and covariant in their result, and type quantification is covariant in its body. Finally, nested refinements can get flattened by taking the conjunction of their propositions or propagate subtyping through the refinement type constructor.

6.2 Semantics

The semantics extends the core model with interpretations for propositions, refinement types, and the new typing and subtyping rules.

Propositions have semantics only at the refined level; from the unrefined point of view they are just annotations. The interpretation of a proposition maps refined environments to a boolean value:

$$\{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\ (-) : \mathsf{dom}\,([\![\vdash \Gamma\ \mathsf{ctx}]\!]) \to \mathbb{2}$$

Equality is interpreted by checking whether the unrefined interpretation equates the two terms at their type. As explained in Section 2.1.2 the notion of equality *cannot* depend on the refined interpretation of types and Corollary 1 shows that this unrefined equality test is invariant under refined-equal replacement. Taking this into account, the interpretation of equality is true when the unrefined interpretations of the two terms are related by the unrefined interpretation of their type:

$$\{\!|\Gamma \vdash t \equiv_\tau u\ \mathsf{prp}|\!\}\ (\gamma) \triangleq tt \iff \langle (\!|t|\!)\ (\rho_\Gamma\ (\gamma)), (\!|u|\!)\ (\rho_\Gamma\ (\gamma))\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\iota_\Gamma\ (\gamma))$$

The remaining connectives are interpreted as expected: conjunction returns the interpretation of the right conjunct when the left conjunct is true and ff otherwise; negation is boolean negation.

$$\{\!|\Gamma \vdash \phi \wedge \psi\ \mathsf{prp}|\!\}\ (\gamma) \triangleq \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\ (\gamma) \wedge \{\!|\Gamma \vdash \psi\ \mathsf{prp}|\!\}\ (\gamma)$$
$$\{\!|\Gamma \vdash \neg\phi\ \mathsf{prp}|\!\}\ (\gamma) \triangleq \neg\, \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\ (\gamma)$$

As before, the interpretation of propositions respects PERs.

Theorem 6.1 (PER-Respecting Propositions). *Let* $\vdash \Gamma\ \mathsf{ctx}$ *and* $\Gamma \vdash \phi\ \mathsf{prp}$. *If* $\langle \gamma, \delta\rangle \in [\![\vdash \Gamma\ \mathsf{ctx}]\!]$, *then*

$$\{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\ (\gamma) = \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\ (\delta)$$

Refinement types. Refinement types carry no meaning in the unrefined semantics:

$$[\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\ (\gamma) \triangleq [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\gamma)$$

In the refined semantics $\{x{:}\tau \mid \phi\}$ is the subset of the interpretation of τ that satisfies ϕ. By Theorem 6.1 propositions do not distinguish related environments and either component can be used:

$$[\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\ (\gamma) \triangleq \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\gamma) \,\middle|\, \{\!|\Gamma, x :_\tau \tau \vdash \phi\ \mathsf{prp}|\!\}\ (\langle \gamma, v\rangle) = tt \right\}$$

Subtyping. We keep the behaviour of the SMT judgment abstract and require the following soundness condition: if a proposition is well-formed and valid then it is true in all environments.

Assumption 1 (SMT soundness). *If* $\vdash \Gamma\ \mathsf{ctx}, \Gamma \vdash \phi\ \mathsf{prp}$, *and* $\Gamma \models \phi$, *then* $\forall \gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket)$,

$$\{\!\{\Gamma \vdash \phi\ \mathsf{prp}\}\!\}\,(\gamma) = tt$$

Subtyping judgments are interpreted in the refined semantics as proofs of subset inclusion between the two interpretations of types. Since the unrefined semantics ignores refinements a subtype is equal to its supertype with respect to the unrefined semantics:

Theorem 6.2 (Subtyping semantics). *If* $\vdash \Gamma\ \mathsf{ctx}, \Gamma \vdash \tau{:}\mathsf{Ty}, \Gamma \vdash \sigma{:}\mathsf{Ty}$, *and* $\Gamma \vdash \tau \preceq \sigma$ *then:*
(i) Refined: If $\gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma\ \mathsf{ctx} \rrbracket)$ *then* $\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \subseteq \llbracket \Gamma \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$.
(ii) Unrefined: If $\gamma \in \mathsf{dom}\,(\lfloor \vdash \Gamma\ \mathsf{ctx} \rfloor)$ *then* $\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\gamma) = \lfloor \Gamma \vdash \sigma{:}\mathsf{Ty} \rfloor\,(\gamma)$.

Terms. The interpretation of terms is unchanged since no new term formers are introduced, and the proof of Theorem 5.6 covers the new typing rules.

7 Inductive types

Finally, we extend the calculus of Section 6 with inductive types. We add kinds for inductive type constructors and formation rules for inductive types, reflecting index dependence.

The central question of our design is which recursive occurrences of an inductive type should be allowed so that its semantic interpretation remains well defined. Strict positivity is the familiar discipline from dependent type theory: a datatype such as lists is accepted because recursive occurrences appear only on the right of the function type. Refinement systems also use two broader patterns. Positive inductive types allow non-strictly-positive shapes whose overall variance is still covariant. Stratified inductive types allow recursive occurrences at smaller indices, even in negative positions.

Strictly positive inductive types admit an elegant canonical syntactic presentation in terms of W-types, which capture constructions generated by strictly positive signatures. However, positive and stratified inductive types allow more general shapes that are not reducible to standard strictly positive signatures. For this reason inductive types will be presented via a general axiom schema.

Section 7.1 presents the syntax of indexed kinds, positive and stratified inductive types, and the typing rules for introduction and elimination, in particular both positive and stratified types use a single general introduction rule and elimination schema. Section 7.2 presents the semantics of indexed kinds and inductive types and prove that the semantics of the eliminator is well defined and satisfies the semantics of the typing rules.

7.1 Syntax

Figure 3 extends the syntax of CIRT with inductive types. We only show the new rules.

Kinds. Since in our calculus inductive types are indexed (*e.g.* the `Val` type of Section 2.1.4), rule Kd-Fun extends kinds with term indexed kinds, the function space is dependent in the same way as the function space of terms.

Inductive types. There are two syntactic disciplines to form inductive types: positive as in rule Ty-IndPos and stratified as in rule Ty-IndStrat. Syntactically, an inductive type K is defined in the form $\mu\, K^{\langle \nu \rangle} \left\{ \underline{C{:}\overline{(y{:}\sigma)}^{\,j} \to K\ \overline{t}^{\,k}}_{\,l} \right\}$, where ν is either $+$ or m for positive and stratified inductive types respectively. K is a type operator taking k arguments, called the *indices* of the inductive type. Each type is defined as a collection of l constructors and each constructor is specified by a typing signature which is a, possibly nullary if $j = 0$, dependent function returning the inductive type

Kinds continued $\boxed{\Gamma \vdash \kappa{:}\mathsf{Kd}}$

$$\frac{\Gamma \vdash \tau{:}\mathsf{Ty} \quad \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd}}{\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}}\ \textsc{Kd-Fun}$$

Types continued $\boxed{\Gamma \vdash \tau{:}\kappa}$

$$\frac{\Gamma \vdash t{:}\tau \quad \Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa}{\Gamma \vdash \sigma\ t{:}\kappa\,[t/x]}\ \textsc{Ty-App} \qquad \frac{\begin{array}{c}\Gamma \vdash \overline{\tau}^{\,k} \to \mathsf{Ty}{:}\mathsf{Kd} \quad \forall l, j.\, \mathrm{pol}_K\,(\tau_{l,j}) \leq +\\ \forall l.\, \Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty} \vdash \overline{\tau_l}^{\,j} \to K\ \overline{t_l}^{\,k}{:}\mathsf{Ty}\end{array}}{\Gamma \vdash \mu K^{\langle + \rangle}\left\{\underline{C{:}\overline{\tau}^{\,j} \to K\ \overline{t}^{\,k}}_{\,l}\right\}{:}\overline{\tau}^{\,k} \to \mathsf{Ty}}\ \textsc{Ty-IndPos}$$

$$\frac{\begin{array}{c}\Gamma \vdash \overline{x{:}\tau}^{\,k} \to \mathsf{Ty}{:}\mathsf{Kd} \quad \Gamma \vdash m{:}\overline{x{:}\tau}^{\,k} \to \mathsf{Nat}\\ \forall l, j.\, \mathsf{stratifiable}_{K;\,m;\,\overline{t_l}^{\,k}}\left((\Gamma,\, K{:}_{\kappa}\overline{x{:}\tau}^{\,k} \to \mathsf{Ty},\, \overline{y{:}_{\mathsf{T}}\sigma}^{\,j-1}), \cdot, (\overline{y{:}_{\mathsf{T}}\sigma}^{\,\geq j}), \tau_{l,j}\right)\\ \forall l.\, \Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty} \vdash \overline{\tau_l}^{\,j} \to K\ \overline{t_l}^{\,k}{:}\mathsf{Ty}\end{array}}{\Gamma \vdash \mu K^{\langle m \rangle}\left\{\underline{C{:}\overline{(y{:}\sigma)}^{\,j} \to K\ \overline{t}^{\,k}}_{\,l}\right\}{:}\overline{\tau}^{\,k} \to \mathsf{Ty}}\ \textsc{Ty-IndStrat}$$

Terms continued $\boxed{\Gamma \vdash t{:}\tau}$

$$\frac{D \triangleq \mu K^{\langle \nu \rangle}\left\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i}\right\} \quad \Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty}}{\Gamma \vdash C(i)_D{:}(\overline{y{:}\sigma\,[D/K]}^{\,j}) \to D \cdot \overline{t}^{\,k}}\ \textsc{Tm-Intro}$$

$$\frac{\begin{array}{c}D \triangleq \mu K^{\langle \nu \rangle}\left\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i}\right\} \quad \Gamma,\, \overline{x{:}_{\mathsf{T}}\tau}^{\,k},\, y{:}_{\mathsf{T}}D \cdot \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty}\\ \forall i.\, \Gamma,\, \overline{z{:}_{\mathsf{T}}\sigma\,[D/K]}^{\,j},\, \overline{\mathsf{El}_{K;\,\rho}\,(z,\,\sigma)\,[D/K]}^{\,j} \vdash c_i{:}\rho\,[\overline{t}^{\,k}/\overline{x}^{\,k}]\,[C(i)_D\ \overline{z}^{\,j}/y]\end{array}}{\Gamma \vdash \mathit{elim}_D\,(\overline{c}^{\,i}) : (\overline{x{:}_{\mathsf{T}}\tau}^{\,k}) \to (y{:}D \cdot \overline{x}^{\,k}) \to \rho}\ \textsc{Tm-Elim}$$

Subtyping continued $\boxed{\Gamma \vdash \tau \preceq \sigma}$

$$\frac{\forall k.\, \Gamma \models t_k \equiv_{\tau_k} u_k}{\Gamma \vdash \tau\ \overline{t}^{\,k} \preceq \tau\ \overline{u}^{\,k}}\ \textsc{Sub-Ix}$$

Fig. 3. Typing rules of inductive CIRT. Where $\mathsf{stratifiable}_{K;\,m;\,ts}\,(\Gamma, \Delta, \Theta, \tau) \triangleq \mathsf{strat}_{\Gamma;\,\Theta;\,K;\,m;\,ts}\,(\Delta, \tau)$.

being defined fully applied to some concrete indices; the argument of the function can be any type, including types that contain the inductive type being defined. Both rules at the premises have a predicate (pol and strat) that ensures the well-formedness of the inductive type and are defined next.

Positive inductive types are those where every occurrence of the type constructor appears under an even number of function domains. Universal quantification is also treated as variance-flipping binder, since it can capture occurrences of the type constructors.

Example 1 (Hofmann's Routine). *As an example of a positive inductive type, Hofmann's Routine (from Section 2.1.4) is defined in CIRT, to be of kind* Ty*, as follows:*

$$\mathsf{Rou} \triangleq \mu\,\mathsf{R}^{\langle + \rangle}\,\{\,\mathsf{Over}{:}\mathsf{R},\, \mathsf{Next}{:}((\mathsf{R} \to \mathsf{ListNat}) \to \mathsf{ListNat}) \to \mathsf{R}\,\}\,.$$

The recursive occurrence of R *in the argument of* Next *appears to the left of two function arrows, so its polarity is positive.*

Formally, the variance of a specific type constructor in a type is captured by a polarity lattice.

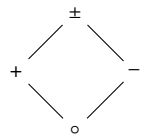


Fig. 4. Polarity lattice.

Definition 14 (Polarity lattice). *Polarities are elements of a lattice with four elements ± denoting both positive and negative occurrences, + denoting only positive occurrences, − denoting only negative occurrences, and ∘ denoting no occurrences. The Hasse diagram (Figure 4) uniquely defines the ordering (≤), join (⊔) and complement ($\overline{p}$) operations on polarities.*

The polarity of a type constructor, is next defined as an element of the polarity lattice.

Definition 15 (Polarity assignment). *The polarity of a type constructor K in a type τ or kind κ is an element of the polarity lattice, defined by structural induction on the type or kind.*

$$\begin{array}{rcl@{\qquad}rcl}
\mathrm{pol}_K(\mathcal{B}) & \triangleq & \circ & \mathrm{pol}_K((x{:}\tau) \to \sigma) & \triangleq & \overline{\mathrm{pol}_K(\tau)} \sqcup \mathrm{pol}_K(\sigma) \\
\mathrm{pol}_K(K) & \triangleq & + & \mathrm{pol}_K(\{x{:}\tau \mid \phi\}) & \triangleq & \mathrm{pol}_K(\tau) \\
\mathrm{pol}_K(\tau\ \bar{t}^{\,k}) & \triangleq & \mathrm{pol}_K(\tau) & \mathrm{pol}_K(\forall x{:}\kappa.\,\tau) & \triangleq & \overline{\mathrm{pol}_K(\kappa)} \sqcup \mathrm{pol}_K(\tau) \\
\mathrm{pol}_K(\mathsf{Ty}) & \triangleq & \circ & \mathrm{pol}_K((x{:}\tau) \to \kappa) & \triangleq & \overline{\mathrm{pol}_K(\tau)} \sqcup \mathrm{pol}_K(\kappa)
\end{array}$$

$$\mathrm{pol}_K\left(\mu L^{\langle + \rangle}\left\{\underline{C{:}(\overline{y{:}\sigma}^{\,j}) \to L \cdot \bar{t}^{\,k}}_{\,i}\right\}\right) \triangleq \bigsqcup_{i,j} \mathrm{pol}_K(\sigma_{i,j})$$

$$\mathrm{pol}_K\left(\mu L^{\langle m \rangle}\left\{\underline{C{:}(\overline{y{:}\sigma}^{\,j}) \to L \cdot \bar{t}^{\,k}}_{\,i}\right\}\right) \triangleq \begin{cases} \circ & \text{if } \forall i, j.\mathrm{pol}_K(\sigma_{i,j}) = \circ \\ \pm & \text{otherwise} \end{cases}$$

We say that the occurrence of constructor K in a type is positive if $\mathrm{pol}_K(\tau) \leq +$ and negative if $\mathrm{pol}_K(\tau) \leq -$. The definitions extend naturally to kinds.

To assign the polarity of a type in a inductive declaration, when the type is positive we can simply collect the polarities of the type constructor in each of the constructor argument types.

When the type is stratified, we require that the type constructor does not appear in any of the argument, not even in positive positions. This is because stratified types, by self reference in negative positions, can move a syntactically positive occurrence into a semantically negative position.

Stratified inductive types permit negative occurrences of the inductive type by enforcing the syntactic discipline of stratification [40]. Concretely, given a natural number metric on the indices of the inductive type we define a structural order on terms. Now each recursive occurrence of the inductive type in each of the constructor arguments must be indexed by terms that are strictly smaller than the indices of the constructor return type. This ensures that the stratified types are well-founded as they are generated bottom-up from the well-founded order on the indices.

Example 2 (Stratified indexed types). *The indexed value type from Section 2.1.4 is a stratified type operator* $\mathsf{Val} : (x{:}\mathsf{Typ}) \to \mathsf{Ty}$*, where* Typ *is the inductive type of object-language simple types. The CIRT declaration is the following:*

$$\mathsf{Val} \triangleq \mu\,\mathsf{V}^{\langle m \rangle}\,\{\,\mathsf{VIota}{:}\mathsf{Nat} \to \mathsf{V}\ \mathsf{TIota}, \mathsf{VArr}{:}(t{:}\mathsf{Typ}) \to (u{:}\mathsf{Typ}) \to (\mathsf{V}\ t \to \mathsf{V}\ u) \to \mathsf{V}\ (\mathsf{TArr}\ t\ u)\,\}$$

The function argument of VArr *contains two recursive occurrences, one of them,* $\mathsf{V}\ t$*, in a negative position, but both occurrences are indexed by the subcomponents t and u of the result index* $\mathsf{TArr}\ t\ u$*. Taking the metric m to be the size of* Typ*, both recursive indices are strictly smaller than the result index, so* Val *is well defined.*

More formally, given a natural number metric on the indices, the stratification predicate checks that all recursive occurrences of the inductive type in a type are indexed by terms that are strictly smaller than the indices of the type being defined.

Definition 16 (Stratification predicate). *The stratification of a type constructor K in a type τ or kind κ is defined as a predicate by structural induction on the type or kind.*

$$\begin{aligned}
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \mathcal{B}) &\triangleq tt \\
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \tau\ \overline{u}^{k'}) &\triangleq \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \tau) \quad \textit{where } \tau \neq K \\
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, (x{:}\tau) \to \sigma) &\triangleq \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \tau) \wedge \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,((\Delta,\ x{:}_{\tau}\tau), \sigma) \\
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \{x{:}\tau \mid \phi\}) &\triangleq \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \tau) \\
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \forall x{:}\kappa.\ \tau) &\triangleq \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \kappa) \wedge \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,((\Delta,\ x{:}_{\kappa}\kappa), \tau) \\
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \mathsf{Ty}) &\triangleq tt \\
\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, (x{:}\tau) \to \kappa) &\triangleq \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,(\Delta, \tau) \wedge \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\,((\Delta,\ x{:}_{\tau}\tau), \kappa)
\end{aligned}$$

When the type is an inductive type, we check that all the constructor argument types are stratified with respect to the metric.

$$\mathsf{strat}_{\Gamma;\Theta;K;m;ts}\left(\Delta, \mu\, L^{\langle m\rangle}\left\{\underline{C{:}(\overline{y{:}\sigma}^{\,j}) \to L\cdot\overline{t}^{\,k}}_{\,i}\right\}\right) \triangleq \bigwedge_{i,j} \mathsf{strat}_{\Gamma;\Theta;K;m;ts}\left((\Delta,\ L{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty}), \sigma_{i,j}\right)$$

The interesting case is when the type is a recursive occurrence of the type being defined:

$$\mathsf{strat}_{\Gamma;\Theta;K;m;\overline{t}^{\,k}}\,(\Delta, K\ \overline{u}^{\,j}) \triangleq \left\{\begin{array}{l} j = k, \forall \gamma \in \mathsf{dom}\left(\llbracket \vdash \Gamma\ \mathsf{ctx}\rrbracket\right), \forall\, \langle \gamma,\ \delta\rangle \in \mathsf{dom}\left(\llbracket \vdash \Gamma,\ \Delta\ \mathsf{ctx}\rrbracket\right), \forall\, \langle \gamma,\ \epsilon\rangle \in \mathsf{dom}\left(\llbracket \vdash \Gamma,\ \Theta\ \mathsf{ctx}\rrbracket\right), \\ r_{\mathbb{N}}\left(\llparenthesis m\rrparenthesis\,(\rho_{\Gamma}\,(\gamma)) \cdot \overline{\llparenthesis u\rrparenthesis\left(\rho_{\Gamma,\Delta}\left(\langle \gamma,\ \delta\rangle\right)\right)}^{\,k}\right) < r_{\mathbb{N}}\left(\llparenthesis m\rrparenthesis\,(\rho_{\Gamma}\,(\gamma)) \cdot \overline{\llparenthesis t\rrparenthesis\left(\rho_{\Gamma,\Theta}\left(\langle \gamma,\ \epsilon\rangle\right)\right)}^{\,k}\right) \end{array}\right.$$

A recursive occurrence of a type constructor K in a type τ is stratified with respect to a sequence of indices $\overline{t}^{\,k}$ if $\mathsf{strat}_{\Gamma;\Theta;K;m;\overline{t}^{\,k}}\,(\Delta, \tau)$ holds and the definition extends naturally to kinds.

While the condition requires checking the stratification predicate for all environments, in practice it can be checked syntactically by ensuring that the indices of the recursive occurrences are syntactically smaller than those of the constructor return type, just as structural termination is checked.

Introduction and elimination, are neutral with respect to the formation discipline, as they are semantically justified by the fixed point construction of inductive types. Thus, the same introduction and elimination rules apply to both positive and stratified inductive types; the formation discipline is handled entirely by the corresponding type formation rule.

The introduction rule Tm-Intro simply unfolds the inductive type in the constructor types. The elimination rule Tm-Elim instead is more complicated: it takes a target type ρ where the inductive type is eliminated, and a case c_i for each constructor K_i of the inductive type. Each case c_i has access both to the unfolding of the constructor arguments $\sigma_j\,[D/K]$ and to the inductive hypothesis for each strictly positive occurrence of the inductive type in the constructor arguments $\mathsf{El}_{K;\rho}\,(z_j,\ \sigma_j)$. Introducing the inductive hypothesis in strictly positive positions corresponds to the permitted recursive calls of structurally recursive functions over inductive types with pattern matching [2].

Definition 17 (Inductive hypothesis). *Given a type constructor K and a type τ, the inductive hypothesis $\mathsf{El}_{K;\rho}\,(u,\ \tau)$ is defined by structural induction over τ.*

$$\begin{aligned}
\mathsf{El}_{K;\rho}\,(u,\ \tau\ \overline{t}^{\,n}) &\triangleq \mathsf{Unit} & \mathsf{El}_{K;\rho}\,(u,\ K\ \overline{t}^{\,k}) &\triangleq \rho\,[\overline{t}^{\,k}/\overline{x}^{\,k}]\,[u/y] \\
\mathsf{El}_{K;\rho}\,(u,\ \mathcal{B}) &\triangleq \mathsf{Unit} & \mathsf{El}_{K;\rho}\,(u,\ x{:}\tau \to \sigma) &\triangleq x{:}\tau \to \mathsf{El}_{K;\rho}\,(u\cdot x,\ \sigma) \\
\mathsf{El}_{K;\rho}\,(u,\ \{x{:}\tau \mid \phi\}) &\triangleq \mathsf{El}_{K;\rho}\,(u,\ \tau) & \mathsf{El}_{K;\rho}\,(u,\ \forall x{:}\kappa.\ \tau) &\triangleq \forall x{:}\kappa.\ \mathsf{El}_{K;\rho}\,(u\ x,\ \tau)
\end{aligned}$$

where unit can be encoded as $\mathsf{Unit} \triangleq \{x{:}\mathsf{Bool} \mid x \equiv tt\}$ *and* $\star \triangleq tt$

Next, we exemplify the elimination rule with the types defined in examples 1 and 2.

Example 3 (Eliminating Hofmann's Routine). *For the routine type* Rou*, an eliminator into a target type* ρ *depending on* $\mathsf{r} : \mathsf{Rou}$ *has the shape* $\mathit{elim}_{\mathsf{Rou}}\ (c_{\mathsf{Over}}, c_{\mathsf{Next}}) : (\mathsf{r}{:}\mathsf{Rou}) \to \rho(\mathsf{r})$*. The two cases correspond to the two constructors:*

$$\begin{array}{lcl} c_{\mathsf{Over}} & : & \rho(\mathsf{Over}_{\mathsf{Rou}}) \\ c_{\mathsf{Next}} & : & g{:}((\mathsf{Rou} \to \mathsf{ListNat}) \to \mathsf{ListNat}) \to h_g{:}((\mathsf{Rou} \to \mathsf{ListNat}) \to \mathsf{Unit}) \to \rho(\mathsf{Next}_{\mathsf{Rou}}\ g). \end{array}$$

The recursive occurrence of R *in the constructor argument is positive but not strictly positive, so the* h_g *carries no information. Using this we can write all the structurally recursive functions (all the functions except "extract") in Hofmann [39]'s original presentation of the Routine type.*

Example 4 (Eliminating indexed values). *For the stratified type* Val*, an eliminator into a target type* $\rho(t, v)$ *depending on an index* $t : \mathsf{Typ}$ *and a value* $v : \mathsf{Val}\ t$ *has the shape* $\mathit{elim}_{\mathsf{Val}}\ (c_{\mathsf{I}}, c_{\mathsf{A}}) : (t{:}\mathsf{Typ}) \to (v{:}\mathsf{Val}\ t) \to \rho(t, v)$*. The base case proves the target for values built by* VIota *while the arrow case receives a function argument together with the induction hypothesis generated for its codomain:*

$$\begin{array}{rcl} c_{\mathsf{I}} & : & (n{:}\mathsf{Nat}) \to \mathsf{Unit} \to \rho(\mathsf{TIota}, \mathsf{VIota}_{\mathsf{Val}}\ n) \\ c_{\mathsf{A}} & : & (t{:}\mathsf{Typ}) \to (u{:}\mathsf{Typ}) \to (f{:}(\mathsf{Val}\ t \to \mathsf{Val}\ u)) \\ & \to & \mathsf{Unit} \to \mathsf{Unit} \to ((x{:}\mathsf{Val}\ t) \to \rho(u, f\ x)) \to \rho(\mathsf{TArr}\ t\ u, \mathsf{VArr}_{\mathsf{Val}}\ t\ u\ f) \end{array}$$

Thus, to eliminate a function value, the case for VArr *may use the induction hypothesis on every result* $f\ x$ *at the smaller index* u*.*

Subtyping, is extended with the rule Sub-Ix, which allows equal indices to be substituted in the type of an inductive datatype. Uniform parametric datatypes do not require any additional subtyping rules, since they can be encoded using universal and existential quantification [32]. Since the encoding is expressed entirely using existing type formers, subtyping for parametric datatypes is inherited from the subtyping rules of the basic type formers, without requiring any datatype-specific subtyping rule.

7.2 Semantics

The semantics of inductive types extends both interpretations with indexed type operators. The unrefined semantics forgets the indices and uses a guarded fixed point construction, while the refined semantics keeps the indices and justifies positive and stratified declarations by separate fixed point arguments.

7.2.1 Type operators and application. The unrefined interpretation of kinds just ignores all the index information, since the unrefined semantics ignores every term dependency:

$$\begin{array}{c} \llbracket \Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd} \rrbracket\ (\gamma) \triangleq \llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd} \rrbracket\ (\gamma) \\ \llbracket \Gamma \vdash \sigma\ t{:}\kappa\ [t/x] \rrbracket\ (\gamma) \triangleq \llbracket \Gamma \vdash \sigma{:}\ (x{:}\tau) \to \kappa \rrbracket\ (\gamma) \end{array}$$

These definitions make $\llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\ (\gamma) = \mathcal{CUPER}\ (\mathcal{D})$ for all kinds κ.

Instead, in the refined semantics, function kinds are interpreted as set-theoretic functions, which take as arguments as many elements of $\mathcal{D}$ as the arity of the kind and return an element of $\mathcal{PER}\ (\mathcal{D})$; as a side condition, the function respects the signature given by the kind and respects the equivalence relation of the domain of the function.

$$\begin{array}{l} \llbracket \Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd} \rrbracket\ (\gamma) \\ \quad \triangleq \left\{ f \;\middle|\; \begin{array}{cc} f \in \mathcal{D} \to (\mathcal{D} \to)^{\mathsf{arity}(\kappa)}\ \mathcal{PER}\ (\mathcal{D}) & \text{(i)} \\ \forall v \in \mathsf{dom}\ (\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\ (\gamma))\ \ f(v) \in \llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd} \rrbracket\ (\langle \gamma, v \rangle) & \text{(ii)} \\ \forall \langle v, w \rangle \in \left(\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma)) \cap \mathsf{dom}\ (\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\ (\gamma))^2\right)\ \ f(v) = f(w) & \text{(iii)} \end{array} \right\} \end{array}$$

where the arity of a kind is defined as

$\mathsf{arity}(\mathsf{Ty}) \triangleq 0$ and $\mathsf{arity}((x{:}\tau) \to \kappa) \triangleq 1 + \mathsf{arity}(\kappa)$.

One might wonder why the semantics of the function kind is not defined directly as functions $v \in \mathsf{dom}(\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket(\gamma)) \to \llbracket \Gamma, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd} \rrbracket(\langle \gamma, v \rangle)$. But, since set theoretically functions are defined as sets of pairs, the interpretation of kinds cannot be related by set-theoretic subsets. Conditions (i) and (ii) allow us to define the interpretation of function kinds as a direct definition, while not constraining the domain to be exactly $\mathsf{dom}(\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket(\gamma))$. Condition (iii) ensures that the interpretation of indexed types cannot distinguish between related, hence equal, terms; as inductive types can witness equalities between them. As an example, the subtyping rule Sub-Ix allows substitution of equal indices in the type of an inductive type.

Application instead can be interpreted as set-theoretic application; the side condition in the definition of the interpretation of function kinds ensures that the application is well-defined.

$$\llbracket \Gamma \vdash \sigma\ t{:}\kappa\,[t/x] \rrbracket(\gamma) \triangleq \llbracket \Gamma \vdash \sigma{:}(x{:}\tau) \to \kappa \rrbracket(\gamma)\,(\!(t)\!)\,(\rho_\Gamma(\gamma)))$$

Extending type projections. Indexed kinds are the reason type projections (Definition 11) are required, so that we can check if the refined interpretation of a type in $\llbracket \Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd} \rrbracket(\gamma)$ (which is a function) is a subset of the unrefined interpretation of a type in $\lfloor \Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd} \rfloor(\gamma)$ (which is a CUPER). Thus instead of looking directly at the function for comparison we look at its image, since kinds can be defined outside the interpretation of the argument, we cannot take the simple set-theoretic image, but we need to take the image of the function restricted to the domain of the interpretation of the domain type. More formally:

Definition 11 - Extended (Type projection). *The definition of type projection is extended to indexed kinds as follows:* $\pi_{\Gamma;\,(x:\tau)\to\kappa}(\gamma, f) \triangleq \cup\{\pi_{\Gamma,x:_{\mathsf{T}}\tau;\,\kappa}(\langle \gamma, v \rangle, f \cdot v) \mid v \in \mathsf{dom}(\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket(\gamma))\}$.

7.2.2 Unrefined inductive types. In the unrefined semantics inductive types are interpreted as fixed-point of a contractive operator on CUPERs. The operator acts on the n-th approximations (per definition 8) and the determinacy of a CUPER by its approximations (lemma 3.3) then yields its fixed point. We show that the unrefined interpretation of a type with a free type variable is either contractive (definition 9) or the identity function, which is sufficient for the existence of a fixed point.

Theorem 7.1 (Unrefined type semantics is contractive or identity). *Let* $\vdash \Gamma, x{:}_\kappa\kappa, \Delta\ \mathsf{ctx}$ *be a well-formed context and* $\Gamma, x{:}_\kappa\kappa, \Delta \vdash \tau{:}\mu$ *a well-formed type. For every context interpretation* $\langle \langle \gamma, P \rangle, \delta \rangle \in \mathsf{dom}(\lfloor \vdash \Gamma, x{:}_\kappa\kappa, \Delta\ \mathsf{ctx} \rfloor)$*, the following operator, which varies the interpretation assigned to* x*, is either contractive or the identity function:*

$$\lfloor \Gamma, x{:}_\kappa\kappa, \Delta \vdash \tau{:}\mu \rfloor(\langle \langle \gamma, - \rangle, \delta \rangle) : \mathcal{CUPER}(\mathcal{D}) \to \mathcal{CUPER}(\mathcal{D})$$

Because each contractive operator has a unique fixed point (Lemma 3.4); when the operator is the identity, every CUPER is a fixed point. In either case a fixed point exists.

The unrefined semantics of inductive types are defined as the fixed point of all the constructors:

$$\left\lfloor \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} : \overline{\tau}^{\,k} \to \mathsf{Ty} \right\rfloor(\gamma) \triangleq \mathsf{fix}_X\, \mathcal{F}_\perp(X)$$

where,

$$\mathcal{F}_\perp(X) \triangleq \bigcup_i \left\{ \left\langle \langle C_i, \overline{v}^{\,j} \rangle, \langle C_i, \overline{w}^{\,j} \rangle \right\rangle \middle| \langle v_j, w_j \rangle \in \llbracket \Gamma, K{:}_\kappa\overline{\tau}^{\,k} \to \mathsf{Ty}, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \rrbracket(\langle \gamma, X \rangle) \right\} \cup \{\langle \perp, \perp \rangle\}$$

From Theorem 7.1 each element of the union is either contractive or the identity function, and in either case non-expansive. Products and tagged unions act on approximations componentwise and are therefore non-expansive—$(A \times B)_n$ is determined by A_n and B_n—while tagging a component by a constructor is contractive, as its $(n+1)$-st approximation depends only on the n-th approximation of the component. Every branch of $\mathcal{F}_\perp$ tags its arguments by a constructor C_i, so each branch is

contractive; taking the componentwise product of the arguments and the union over constructors then preserves contractivity. Hence the $\mathcal{F}_\perp$ operator is contractive, and by Lemma 3.4 it has a unique fixed point—so the fixed point $\mathrm{fix}_X\, \mathcal{F}_\perp(X)$ defined above is well-posed.

7.2.3 Refined inductive types. The refined semantics instead uses more traditional techniques from the semantics of inductive types: for positive inductive types the type operator is monotone and the Knaster-Tarski theorem yields a least fixed point; for stratified inductive types the well-founded order over the indices yields a unique, hence least, fixed point by a bottom-up construction. These arguments discharge the new induction cases of theorem 5.2. For *least* to be meaningful we order the semantics of types by the pointwise lift of PER inclusion and the semantics of kinds by inclusion.

Definition 19 (Lattice structure of types). *Given* $\vdash \Gamma\ \mathsf{ctx}$ *and* $\Gamma \vdash \kappa{:}\mathsf{Kd}$, *we define the following order on* $P, Q \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)$*:*

- *For* $\kappa = \mathsf{Ty}$, $P \le Q$ *if* $P \subseteq Q$.
- *For* $\kappa = (x{:}\tau) \to \mu$, $P \le Q$ *if for all* $v \in \mathsf{dom}\,([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma))$, $P \cdot v \le Q \cdot v$.

This order is a complete lattice: meets are intersections, and the join of A and B is the least PER containing $A \cup B$, which exists because PERs are closed under intersection and $\mathcal{D} \times \mathcal{D}$ is the top PER; the pointwise lift of a complete lattice is again a complete lattice. Here we identify functions that agree on the domain of the interpretation of the domain type, as only that restriction is ever used.

Each inductive type is then interpreted as the least fixed point of the operator $\mathcal{F}$, which unfolds a candidate interpretation X into the union of all the possible constructor assignments.

$$\left[\!\!\left[\Gamma \vdash \mu K^{\langle\nu\rangle}\left\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i}\right\} {:}\,\overline{\tau}^{\,k} \to \mathsf{Ty}\right]\!\!\right](\gamma) \triangleq \mathrm{lfp}_X \mathcal{F}(X)$$

where,

$$D \triangleq \mu K^{\langle\nu\rangle}\left\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i}\right\}; \quad \mathsf{Kd}_D \triangleq \overline{\tau}^{\,k} \to \mathsf{Ty}$$

$$s_\sigma \triangleq [\![\Gamma,\, K{:}_\kappa \mathsf{Kd}_D,\, \overline{x{:}_\tau \sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]; \quad s_D \triangleq [\![\Gamma \vdash D{:}\mathsf{Kd}_D]\!]; \quad s_\tau \triangleq [\![\Gamma,\, \overline{x{:}_\tau \tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}]\!]$$

$$\mathcal{F}(X) \triangleq \overline{q}^{\,k} \mapsto \bigcup_i \left\{ \left\langle \left\langle C_i, \overline{v}^{\,j} \right\rangle, \left\langle C_i, \overline{w}^{\,j} \right\rangle \right\rangle \;\middle|\; \begin{array}{c} \left\langle v_j, w_j \right\rangle \in s_\sigma[\langle\langle \gamma,\, \langle X,\, s_D[\iota_\Gamma\,(\gamma)]\rangle\rangle,\, \overline{v}^{\,j-1}\rangle] \\ \left\langle q_k, (\!| t_k |\!)\left(\left\langle \rho_\Gamma\,(\gamma), \overline{v}^{\,j} \right\rangle\right)\right\rangle \in s_\tau[\iota_\Gamma\,(\gamma)] \end{array} \right\}$$

In $\mathcal{F}(X)$ the first condition relates the constructor arguments at their declared types, with the candidate X interpreting the recursive occurrences of the type; the second requires the input indices to be related to the values of the constructor's index terms. The environment used in $\mathcal{F}(X)$ is well formed only when X is included in s_D, in the sense required of type variables in contexts, so the least fixed point is taken in the sublattice of such X. This is again a complete lattice, and $\mathcal{F}$ preserves it. Importantly, $\mathcal{F}$ has a least fixed point, both for positive and stratified types, as we show next.

Positive refined inductive types. For positive inductive types the fixed point exists by the Knaster-Tarski (theorem 3.1), so it suffices to show that $\mathcal{F}$ is monotone. Since kinds may quantify over types indexed by the type being defined, we show that the semantics of kinds is (anti-)monotone; there the order is plain set-theoretic inclusion, as kinds are interpreted as sets of set-theoretic functions.

Lemma 7.2 ((Anti-)Monotonicity of types). *Let* $\Delta \triangleq \Gamma,\, K{:}_\kappa \mu,\, \overline{x{:}\sigma}^{\,i}$, *where* x_i *can bind both type and term variables, and let* $\vdash \Delta\ \mathsf{ctx}$, $\Delta \vdash \mu{:}\mathsf{Kd}$, *and* $\Delta \vdash \tau{:}\mu$ *be well-formed judgments. Given two context interpretations* $\langle\langle \gamma,\, \langle P, P_\perp\rangle\rangle, \overline{v}^{\,i}\rangle, \langle\langle \gamma,\, \langle Q, P_\perp\rangle\rangle, \overline{v}^{\,i}\rangle \in \mathsf{dom}\,([\![\vdash \Delta\ \mathsf{ctx}]\!])$ *that differ only in the assignment of* K, *with* $P \le Q$, *then*

(i) If $\mathrm{pol}_K\,(\tau) \le +$, *then* $[\![\Delta \vdash \tau{:}\mu]\!]\,(\langle\langle \gamma,\, \langle P, P_\perp\rangle\rangle, \overline{v}^{\,i}\rangle) \le [\![\Delta \vdash \tau{:}\mu]\!]\,(\langle\langle \gamma,\, \langle Q, P_\perp\rangle\rangle, \overline{v}^{\,i}\rangle)$*; and*

(ii) If $\mathrm{pol}_K\,(\tau) \le -$, *then* $[\![\Delta \vdash \tau{:}\mu]\!]\,(\langle\langle \gamma,\, \langle P, P_\perp\rangle\rangle, \overline{v}^{\,i}\rangle) \ge [\![\Delta \vdash \tau{:}\mu]\!]\,(\langle\langle \gamma,\, \langle Q, P_\perp\rangle\rangle, \overline{v}^{\,i}\rangle)$.

The semantics of kinds satisfy a similar property, with set inclusion instead of the order on types.

Since $\mathcal{F}$ only builds tuples out of the interpretations of the constructor argument types, in which K occurs positively, it is monotone by lemma 7.2, and by theorem 3.1 it has a least fixed point.

Stratified refined inductive types. For stratified inductive types the metric m over the indices justifies a bottom-up construction. The stratification condition ensures that the interpretation at an index $\overline{v}^{\,k}$ instantiates every recursive occurrence of the type constructor only at indices $\overline{w}^{\,k'}$ with $m(\overline{w}^{\,k'}) < m(\overline{v}^{\,k})$, and hence depends only on interpretations at strictly m-smaller indices. The semantics is therefore defined by well-founded recursion on the order induced by m, which yields a unique solution of the fixed-point equation, hence it is also the least.

7.2.4 Introduction and elimination. The introduction simply constructs a pair of the constructor tag and the tuple of the arguments while the elimination interpretation is the least fixed point of semantic case analysis:

$$\left(\!\!\left(C(i)_{\mu K^{\langle \nu \rangle}\left\{\underline{C:(\overline{y:\sigma}^{\,j})\to K\cdot\overline{t}^{\,k}}_{\,i}\right\}}\right)\!\!\right)(\gamma) \triangleq i_{\to}\left(\overline{v}^{\,j} \mapsto \langle C_i,\, \overline{v}^{\,j}\rangle\right)$$

$$\left(\!\!\left(elim_{\mu K^{\langle \nu \rangle}\left\{\underline{C:(\overline{y:\sigma}^{\,j})\to K\cdot\overline{t}^{\,k}}_{\,i}\right\}}\,(\overline{c}^{\,i})\right)\!\!\right)(\gamma) \triangleq i_{\to}\left(\mathsf{lfp}_r\,\overline{v}^{\,k},\, w \mapsto \begin{cases} \overline{(\!(c_i)\!)\,(\gamma)\cdot\overline{q}^{\,j}\cdot\overline{\mathsf{tmEl}_{K;r}\left(\sigma,\,\langle\gamma,\,\overline{q}^{\,j-1}\rangle,\,q_j\right)}^{\,j}} & \text{if } w = \langle C_i,\,\overline{q}^{\,j}\rangle \\ \bot & \text{otherwise} \end{cases}\right)$$

Where the semantics of the actual recursive calls follow the same pattern as the typing rules for the eliminator. Namely, the semantics is inductively defined on the type of the constructor argument by invoking r on all the strictly positive recursive occurrences of the type constructor:

Definition 20 (Semantics of recursive calls). *Given a type constructor K, a type τ, the interpretation of the motive r, an environment γ recording the interpretations of the preceding constructor arguments, and the current argument v,the semantics of the recursive calls is defined as follows:*

$$\begin{aligned}
\mathsf{tmEl}_{K;r}\,(\mathcal{B},\,\gamma,\,v) &\triangleq \star & \mathsf{tmEl}_{K;r}\,(K\cdot\overline{t}^{\,k},\,\gamma,\,v) &\triangleq r\cdot\overline{(\!(t)\!)\,(\gamma)}^{\,k}\cdot v \\
\mathsf{tmEl}_{K;r}\,(\tau\cdot\overline{t}^{\,n},\,\gamma,\,v) &\triangleq \star & \mathsf{tmEl}_{K;r}\,(\{x{:}\tau \mid \phi\},\,\gamma,\,v) &\triangleq \mathsf{tmEl}_{K;r}\,(\tau,\,\gamma,\,v) \\
\mathsf{tmEl}_{K;r}\,(\forall x{:}\kappa.\,\tau,\,\gamma,\,v) &\triangleq \mathsf{tmEl}_{K;r}\,(\tau,\,\gamma,\,v) & & \\
\mathsf{tmEl}_{K;r}\,(x{:}\tau\to\sigma,\,\gamma,\,v) &\triangleq i_{\to}\left(w \mapsto \mathsf{tmEl}_{K;r}\,(\sigma,\,\langle\gamma,\,w\rangle,\,v\cdot w)\right) & &
\end{aligned}$$

Notice that the whole construction branches over the syntax of the type in the first argument and it does not depend on the semantics.

We proved that the eliminator is PER-respecting and it realizes the type of the inductive hypotheses. For the refined semantics we prove the following stronger statement that ensures that the eliminator works for any under approximation P of the inductive type: if the realizers r and r' map related indices and P-related values to related results (hypothesis (i)) and are moreover related at the unrefined type of the eliminator (hypothesis (ii)), then the recursive-call packages computed by tmEl inhabit the corresponding inductive-hypothesis types El. The generality in P is forced on us: the refined interpretation of the inductive type is defined as a fixed point, so while proving the eliminator correct the inductive type is only available up to an approximation.

Lemma 7.3 (Refined semantics is PER-respecting (Eliminator)).
Let $D \triangleq \mu K^{\langle \nu \rangle}\left\{\underline{C:(\overline{y:\sigma}^{\,j})\to K\cdot\overline{t}^{\,k}}_{\,i}\right\}, \Delta(i) \triangleq K{:}_{\kappa}(\overline{x:\tau}^{\,k})\to\mathsf{Ty},\ \overline{y:\sigma}^{\,i}$ and $P_{\bot}(\gamma) \triangleq [\![\Gamma \vdash D{:}(\overline{x:\tau}^{\,k})\to \mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))$. Assume the following judgments are well-formed: $\vdash \Gamma,\ \Delta(j)\ \mathsf{ctx}$ and $\Gamma,\ \overline{x:\tau}^{\,k},\ y{:}_{\tau}D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty}$. Given an under approximation of D: $P \le [\![\Gamma \vdash D{:}(\overline{x:\tau}^{\,k})\to\mathsf{Ty}]\!]\,(\gamma)$, in two related context interpretations $\langle\langle\gamma,\,\langle P,\,P_{\bot}(\gamma)\rangle\rangle,\,\overline{v}^{\,j-1}\rangle,\ \langle\langle\delta,\,\langle P,\,P_{\bot}(\delta)\rangle\rangle,\,\overline{w}^{\,j-1}\rangle\rangle \in [\![\vdash \Gamma,\ \Delta(j-1)\ \mathsf{ctx}]\!]$ and r and r' related realizers for the restricted P, i.e.

(i) $\forall\ \overline{\langle t,\,t'\rangle}^{\,k} \in [\![\Gamma,\ \overline{x:\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}]\!](\langle\gamma,\,\overline{t}^{\,k-1}\rangle), \forall\,\langle v,\,w\rangle \in P\cdot\overline{t}^{\,k},\ \langle r\cdot\overline{t}^{\,k}\cdot v,\,r'\cdot\overline{t'}^{\,k}\cdot w\rangle \in [\![\Gamma,\ \overline{x:\tau}^{\,k},\ y{:}_{\tau}D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle\langle\gamma,\,\overline{t}^{\,k}\rangle,\,v\rangle)$, and

(ii) $\langle r,\, r' \rangle \in [\![\Gamma \vdash (\overline{x{:}_i\tau}^{\,k}) \to D \cdot \overline{x}^{\,k} \to \rho{:}\mathsf{Ty}]\!] (\iota_\Gamma\, (\gamma))$

and some term $\Gamma,\, \Delta(j-1) \vdash m{:}\sigma_j$*, then the restricted eliminators realize the type of the induction hypothesis over argument* m*:*

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\, (\sigma_j,\, \gamma_\rho,\, (\!|m|\!)\, (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\, (\sigma_j,\, \delta_\rho,\, (\!|m|\!)\, (\delta_\rho)) \end{array} \right\rangle \in [\![\Gamma,\, \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\, (m,\, \sigma_j)\, {:}\mathsf{Ty}]\!]\, (\langle\langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle\rangle,\, \overline{v}^{\,j-1} \rangle)$$

$$\textit{where } \gamma_\rho \triangleq \rho_{\Gamma,\Delta(j-1)}\, (\langle\langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle\rangle,\, \overline{v}^{\,j-1} \rangle) \;\; \textit{and } \delta_\rho \triangleq \rho_{\Gamma,\Delta(j-1)}\, (\langle\langle \delta,\, \langle P,\, P_\perp(\delta) \rangle\rangle,\, \overline{w}^{\,j-1} \rangle)$$

The same holds for the unrefined interpretation, but there the stronger statement is unnecessary: every unrefined type interpretation is pointed and complete, so it contains the least fixed point.

In the refined case completeness is not for free, so we argue differently, depending on the formation discipline. For positive types, write T for the refined interpretation of the inductive type, *i.e.* the least fixed point of the operator $\mathcal{F}$, and let $P \leq T$ be the restriction of T to the pairs on which the eliminator respects the typing of the motive. We show that P is a pre-fixed point of $\mathcal{F}$, so that $T \leq P$, and hence $P = T$. The inclusion $\mathcal{F}(P) \leq P$ amounts to two conditions: membership in T, which holds as $\mathcal{F}(P) \leq \mathcal{F}(T) = T$ by monotonicity of $\mathcal{F}$; and Lemma 7.3.

For stratified types $\mathcal{F}$ need not be monotone; instead we define a family of truncations $P(n)$, where $P(n)$ agrees with T at all indices of stratum smaller than n and is empty elsewhere. By Lemma 7.3 and induction on n, we show that the eliminator respects the typing of the motive on each $P(n)$.

This completes the treatment of inductive types. As for refinement types, the statements of the theorems of Section 5 are unchanged, and their proofs are extended with the cases for the rules of Figure 3. Hence soundness (Theorem 5.6) and consistency (Corollary 2) hold for the complete CIRT calculus, with refinement types, subtyping, and both positive and stratified inductive types.

8 Compiling indexed types

We implemented the positive and stratified inductive types of CIRT in Liquid Haskell and use it to compile the examples of section 2.1.4. To permit positive types, we simply relaxed the existing strictly positivity check. The encoding of stratified inductive types is more involved, as Liquid Haskell does not have a direct support of indexed inductive types. Yet, following the technique of *data propositions* [14], stratified types can compile to ordinary inductive types, by encoding the index information in the refinement logic, via an uninterpreted function.

Finally, we implemented a stratification condition that ensures that the well-formedness condition of the declaration is satisfied.

Stratification is checked when a datatype is declared as `stratified`. For example, the `Val` type of section 2.1.4 needs to be declared as `stratified`, otherwise the positivity check of Liquid Haskell would reject it. Yet, this implementation is direct: in Liquid Haskell, it required ~400 lines of code, thus we believe that it can be easily adapted to other refinement type systems.

9 Related work

Models of refinement type systems. Refinement types were introduced by Freeman and Pfenning [27] as a discipline for carving out subsets of ML datatypes. Stating properties of data by predicates over unindexed types, as refinement types do, has a long history in logic-enriched type theories [6, 29]. In Sections 2.1 and 2.2 we already discussed in detail what is and what is not captured by existing formalizations of refinement type systems. Other works, such as [15], build on [34] by reintroducing a subtyping judgment, validating its necessity. However, they establish only a partial correctness result relative to termination and do not guarantee consistency. Lepigre [45] gives a classical realizability model of a second-order language with impredicative polymorphism and an equality type on untyped terms, interpreted as observational equivalence of programs rather than by the unrefined interpretation of a type; it has neither subtyping nor

inductive types. Closest to ours in spirit is the work of Economou et al. [24], who likewise give an elementary domain-theoretic denotational semantics for a refinement type system with subtyping and use it to derive logical consistency. Their system, however, belongs to the index-refinement tradition of Dependent ML and liquid types [60]: it supports neither impredicative polymorphism nor the non-standard inductive types that are our central concern. The view of refinements as an annotation layer over an unrefined language is made precise by Melliès and Zeilberger [51], who axiomatise a refinement type system as a functor from a category of refined typing derivations down to its unrefined base. Our dual-layer model—refined PERs sitting above a layer of CUPERs that models computation—is a semantic instance of this picture: a type denotes a PER over $\mathcal{D}$ included in a CUPER over $\mathcal{D}$, a term denotes an element of $\mathcal{D}$ preserving both relations, and annotation independence is its concrete manifestation. Methodologically, our interpretation of types as PERs follows the recipe of Harper [35] for constructing type systems over an operational semantics, with the closed terms replaced by the elements of $\mathcal{D}$.

Models of System F. Interpreting types as PERs over a universal domain is a standard technique of realizability [1, 7, 17]. We build on the model of System F with general inductive types given by Amadio [7], which interprets types as CUPERs over a domain and itself descends from the ideal model of MacQueen et al. [48]. Those models solve recursive type equations by a Banach [9] contraction on a metric of relations; we adapt the same contractive argument to the minimal-invariant projections of the domain [56], avoiding the metric. Birkedal et al. [13] obtain recursive types in admissible PERs from parametricity instead.

Stratified inductive types. Jacob-Rao et al. [40] introduce stratified types as a way to circumvent the absence of large elimination in Tores. Their system omits impredicative polymorphism, subtyping, and refinement types, and instead supports general recursive types via Mendler [52]-style recursion. Their development also includes a normalization proof. Our work can be seen as extending this setting with impredicative polymorphism and refinement structure. The strategy of relaxing syntactic strict positivity in an impredicative setting goes back to Matthes [50], who replaces positivity with a monotonicity witness while retaining strong normalization for System F extended with fixed-point types. Sized types [3] index an inductive type by an ordinal that bounds the height of its elements, in order to check termination. Sizes form a dedicated kind, the operator must be monotone, and types do not depend on terms. Abel [4] notes that the size-indexed approximants remain well defined for a non-monotone operator, but does not study the metatheory of that case.

Dependent type theory. In contrast to refinement types, standard dependent type theories, such as the Calculus of Inductive Constructions enforce strict positivity requirements for inductive definitions. This restriction is fundamentally necessary in standard type theory to preserve logical consistency, particularly when combined with impredicative polymorphism, as demonstrated by the Coquand and Paulin [21] paradox. Furthermore, systems with large elimination allow for the construction of power sets from inductive types, making non-strictly positive definitions inherently dangerous. Our PER-based semantics demonstrates that, in a refinement setting without large elimination, this strict requirement can be safely relaxed.

10 Conclusion & Future Work

We presented a semantic model for refinement type systems based on partial equivalence relations over a Scott domain. The model accounts for all subtyping, unrefined equality, impredicative polymorphism, and non-standard inductive types, and comes with PER-respecting interpretations.

This realizability model shows that the core refinement type calculus is consistent, even in the presence of positive and stratified inductive types. These two inductive type schemes are currently

permitted by Liquid Haskell and we believe that the same design can guide future extensions of more expressive type systems.

Our refinements express total correctness: a term of a refined type terminates and satisfies its predicate. Refined types are therefore ordinary PERs and need not contain a bottom element, whose presence under lazy evaluation makes the refinement logic inconsistent [67]. Partial correctness and quotient types would require a different model, in both cases because equality is unrefined. Partial correctness requires the refined semantics of types to be complete [22], which is not the case in general. Quotient types [38] identify more elements than unrefined equality does, which contradicts Theorem 5.4.

Our proofs are not mechanically verified. A complete mechanization would require formalizing the underlying domain theory, the PER/CUPER constructions, and the semantic interpretation of the inductive types introduced in this work. While mechanized developments of syntactic metatheory are more common, mechanizations of semantic models at a comparable level of expressiveness remain an active research field. Developing the semantic infrastructure and carrying out the corresponding soundness proofs would constitute a substantial project in its own right.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback and suggestions. This work was funded by the Excellence Seal María de Maeztu Grant (CEX2024-001471-M funded by MICIU/AEI/10.13039/501100011033) and the ERC Grant CRETE (GA 101039196). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council; neither of which can be held responsible for them.

Disclosure of Use of AI in Research

No AI tools were used to conduct the research presented in this work.

References

[1] Martín Abadi and Gordon D. Plotkin. 1990. A PER Model of Polymorphism and Recursive Types. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science (LICS '90)* (1990-06). 355–365. doi:10.1109/LICS.1990.113761

[2] Andreas Abel. 1999. *A Semantic Analysis of Structural Recursion.* Diplomarbeit. Ludwig-Maximilians-Universität München.

[3] Andreas Abel. 2008. Semi-Continuous Sized Types and Termination. *Logical Methods in Computer Science* 4, 2 (2008). doi:10.2168/LMCS-4(2:3)2008

[4] Andreas Abel. 2012. Type-Based Termination, Inflationary Fixed-Points, and Mixed Inductive-Coinductive Types. *Electronic Proceedings in Theoretical Computer Science* 77 (2012), 1–11. arXiv:1202.3496 [cs.LO] doi:10.4204/EPTCS.77.1

[5] Samson Abramsky and Achim Jung. 1994. Domain Theory. In *Handbook of Logic in Computer Science (Vol. 3): Semantic Structures*, S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum (Eds.). Oxford University Press, 1–168.

[6] Robin Adams and Zhaohui Luo. 2010. Classical Predicative Logic-Enriched Type Theories. *Annals of Pure and Applied Logic* 161, 11 (2010), 1315–1345. doi:10.1016/j.apal.2010.04.005

[7] Roberto M. Amadio. 1991. Recursion over Realizability Structures. *Information and Computation* 91, 1 (1991), 55–85. doi:10.1016/0890-5401(91)90074-C

[8] Roberto M. Amadio and Pierre-Louis Curien. 1998. *Domains and Lambda-Calculi.* Number 46 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press. doi:10.1017/CBO9780511983504

[9] Stefan Banach. 1922. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae* 3, 1 (1922), 133–181. doi:10.4064/fm-3-1-133-181

[10] Henk Barendregt. 1991. Introduction to generalized type systems. *Journal of Functional Programming* 1, 2 (1991), 125–154. doi:10.1017/S0956796800020025

[11] Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. 1983. A Filter Lambda Model and the Completeness of Type Assignment. *Journal of Symbolic Logic* 48, 4 (1983), 931–940. jstor:2273659 doi:10.2307/2273659

[12] Ulrich Berger, Ralph Matthes, and Anton Setzer. 2019. Martin Hofmann's Case for Non-Strictly Positive Data Types. In *24th International Conference on Types for Proofs and Programs (TYPES 2018)* (2019) *(Leibniz International Proceedings*

in Informatics (LIPIcs), Vol. 130), Peter Dybjer, José Espírito Santo, and Luís Pinto (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 1:1–1:22. doi:10.4230/LIPICS.TYPES.2018.1

[13] Lars Birkedal, Rasmus Ejlers Møgelberg, and Rasmus Lerchedahl Petersen. 2007. Domain-Theoretical Models of Parametric Polymorphism. *Theoretical Computer Science* 388, 1-3 (2007), 152–172. doi:10.1016/j.tcs.2007.06.016

[14] Michael H. Borkowski, Niki Vazou, and Ranjit Jhala. 2024. Mechanizing Refinement Types. *Proc. ACM Program. Lang.* 8 (2024), 70:2099–70:2128. Issue POPL. doi:10.1145/3632912

[15] Matt Bovel, Viktor Kunčak, and Martin Odersky. 2026. *First-Class Refinement Types for Scala.* doi:10.48550/ARXIV.2605.08369

[16] Val Breazu-Tannen, Thierry Coquand, Carl A. Gunter, and Andre Scedrov. 1991. Inheritance as Implicit Coercion. *Information and Computation* 93, 1 (1991), 172–221. doi:10.1016/0890-5401(91)90055-7

[17] Kim B. Bruce and John C. Mitchell. 1992. PER Models of Subtyping, Recursive Types and Higher-Order Polymorphism. In *Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (New York, NY, USA, 1992) *(POPL '92)*. ACM, 316–327. doi:10.1145/143165.143230

[18] Luca Cardelli. 1984. A Semantics of Multiple Inheritance. In *Semantics of Data Types* (Berlin, Heidelberg, 1984) *(Lecture Notes in Computer Science, Vol. 173)*, Gilles Kahn, David B. MacQueen, and Gordon Plotkin (Eds.). Springer, 51–67. doi:10.1007/3-540-13346-1_2

[19] Luca Cardelli and Peter Wegner. 1985. On understanding types, data abstraction, and polymorphism. *Comput. Surveys* 17, 4 (1985), 471–523. doi:10.1145/6041.6042

[20] Felice Cardone. 1989. Relational Semantics for Recursive Types and Bounded Quantification. In *Automata, Languages and Programming* (Berlin, Heidelberg, 1989) *(Lecture Notes in Computer Science, Vol. 372)*, Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca (Eds.). Springer, 164–178. doi:10.1007/BFb0035759

[21] Thierry Coquand and Christine Paulin. 1990. Inductively Defined Types. In *COLOG-88: International Conference on Computer Logic* (Berlin, Heidelberg, 1990), Per Martin-Löf and Grigori Mints (Eds.). Springer, 50–66. doi:10.1007/3-540-52335-9_47

[22] Karl Crary. 1998. Admissibility of Fixpoint Induction over Partial Types. In *Automated Deduction — CADE-15* (Berlin, Heidelberg, 1998), Claude Kirchner and Hélène Kirchner (Eds.). Springer, 270–285. doi:10.1007/BFb0054265

[23] Brian A. Davey and Hilary A. Priestley. 2002. *Introduction to Lattices and Order* (2 ed.). Cambridge University Press. doi:10.1017/CBO9780511809088

[24] Dimitrios J. Economou, Neel Krishnaswami, and Jana Dunfield. 2023. Focusing on Refinement Typing. *ACM Trans. Program. Lang. Syst.* 45, 4 (2023), 22:1–22:62. doi:10.1145/3610408

[25] Alessio Ferrarini and Niki Vazou. 2025. Large Elimination and Indexed Types in Refinement Types. In *31st International Conference on Types for Proofs and Programs (TYPES 2025), Book of Abstracts* (Glasgow, Scotland, 2025). https://msp.cis.strath.ac.uk/types2025/abstracts/TYPES2025_paper33.pdf

[26] Alessio Ferrarini, Niki Vazou, and Wouter Swierstra. 2026. PLEX: Normalization for Refinement Types. *Proc. ACM Program. Lang.* 10 (2026), 140:1321–140:1347. Issue OOPSLA1. doi:10.1145/3798248

[27] Tim Freeman and Frank Pfenning. 1991. Refinement Types for ML. *SIGPLAN Not.* 26, 6 (1991), 268–277. doi:10.1145/113446.113468

[28] Alain Frisch, Giuseppe Castagna, and Véronique Benzaken. 2008. Semantic Subtyping: Dealing Set-Theoretically with Function, Union, Intersection, and Negation Types. *J. ACM* 55, 4 (2008), 19:1–19:64. doi:10.1145/1391289.1391293

[29] Nicola Gambino and Peter Aczel. 2006. The Generalised Type-Theoretic Interpretation of Constructive Set Theory. *Journal of Symbolic Logic* 71, 1 (2006), 67–103. doi:10.2178/jsl/1140641163

[30] Jad Elkhaleq Ghalayini and Neel Krishnaswami. 2023. Explicit Refinement Types. *Proc. ACM Program. Lang.* 7 (2023), 195:187–195:214. Issue ICFP. doi:10.1145/3607837

[31] Jean-Yves Girard. 1972. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur.* Thèse d'État. Université Paris VII.

[32] Jean-Yves Girard. 1989. *Proofs and Types.* Number 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

[33] Carl A. Gunter. 1992. *Semantics of Programming Languages: Structures and Techniques.* MIT Press.

[34] Jad Hamza, Nicolas Voirol, and Viktor Kunčak. 2019. System FR: Formalized Foundations for the Stainless Verifier. *Proc. ACM Program. Lang.* 3 (2019), 166:1–166:30. Issue OOPSLA. doi:10.1145/3360592

[35] Robert Harper. 1992. Constructing Type Systems over an Operational Semantics. *Journal of Symbolic Computation* 14, 1 (1992), 71–84. doi:10.1016/0747-7171(92)90026-Z

[36] Robert Harper. 2016. *Practical Foundations for Programming Languages* (2 ed.). Cambridge University Press. doi:10.1017/CBO9781316576892

[37] Leon Henkin. 1950. Completeness in the theory of types. *Journal of Symbolic Logic* 15, 2 (1950), 81–91. doi:10.2307/2266967

[38] Brandon Hewer and Graham Hutton. 2024. Quotient Haskell: Lightweight Quotient Types for All. *Proceedings of the ACM on Programming Languages* 8, POPL (2024), 27:785–27:815. doi:10.1145/3632869

[39] Martin Hofmann. 1993. *Non Strictly Positive Datatypes in System F.* https://www.seas.upenn.edu/~sweirich/types/archive/1993/msg00027.html

[40] Rohan Jacob-Rao, Brigitte Pientka, and David Thibodeau. 2018. Index-Stratified Types. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)* (2018) *(Leibniz International Proceedings in Informatics (LIPIcs), Vol. 108)*, Hélène Kirchner (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 19:1–19:17. doi:10.4230/LIPICS.FSCD.2018.19

[41] Ranjit Jhala and Niki Vazou. 2020. *Refinement Types: A Tutorial.* arXiv:2010.07763 [cs] doi:10.48550/arXiv.2010.07763

[42] Kenneth Knowles and Cormac Flanagan. 2010. Hybrid Type Checking. *ACM Trans. Program. Lang. Syst.* 32, 2 (2010), 6:1–6:34. doi:10.1145/1667048.1667051

[43] Nico Lehmann, Adam T. Geller, Niki Vazou, and Ranjit Jhala. 2023. Flux: Liquid Types for Rust. *Proc. ACM Program. Lang.* 7 (2023), 169:1533–169:1557. Issue PLDI. doi:10.1145/3591283

[44] Daniel Leivant. 1991. Finitely stratified polymorphism. *Information and Computation* 93, 1 (1991), 93–113. doi:10.1016/0890-5401(91)90053-5

[45] Rodolphe Lepigre. 2016. A Classical Realizability Model for a Semantical Value Restriction. In *Programming Languages and Systems: 25th European Symposium on Programming, ESOP 2016* (Berlin, Heidelberg, 2016) *(Lecture Notes in Computer Science, Vol. 9632)*, Peter Thiemann (Ed.). Springer, 476–502. doi:10.1007/978-3-662-49498-1_19

[46] Benjamin Livshits, Manu Sridharan, Yannis Smaragdakis, Ondřej Lhoták, J. Nelson Amaral, Bor-Yuh Evan Chang, Samuel Z. Guyer, Uday P. Khedker, Anders Møller, and Dimitrios Vardoulakis. 2015. In Defense of Soundiness: A Manifesto. *Commun. ACM* 58, 2 (2015), 44–46. doi:10.1145/2644805

[47] Giuseppe Longo and Eugenio Moggi. 1991. Constructive natural deduction and its ω-set interpretation. *Mathematical Structures in Computer Science* 1, 2 (1991), 215–254. doi:10.1017/S0960129500001298

[48] David MacQueen, Gordon Plotkin, and Ravi Sethi. 1986. An Ideal Model for Recursive Polymorphic Types. *Information and Control* 71, 1-2 (1986), 95–130. doi:10.1016/S0019-9958(86)80019-5

[49] Kenji Maillard. 2018. *Incompatibilities between Functional Extensionality, Subtyping, and Untyped Equality · Issue #1542 · FStarLang/FStar.* GitHub. https://github.com/FStarLang/FStar/issues/1542

[50] Ralph Matthes. 1999. Monotone (Co)Inductive Types and Positive Fixed-Point Types. *RAIRO - Theoretical Informatics and Applications* 33, 4-5 (1999), 309–328. doi:10.1051/ita:1999120

[51] Paul-André Melliès and Noam Zeilberger. 2015. Functors Are Type Refinement Systems. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (2015) *(POPL '15)*. Association for Computing Machinery, 3–16. doi:10.1145/2676726.2676970

[52] Nax Paul Mendler. 1991. Inductive Types and Type Constraints in the Second-Order Lambda Calculus. *Annals of Pure and Applied Logic* 51, 1-2 (1991), 159–172. doi:10.1016/0168-0072(91)90069-X

[53] Nax Paul Mendler. 1987. Recursive Types and Type Constraints in Second-Order Lambda Calculus. In *Proceedings of the Symposium on Logic in Computer Science (LICS '87)* (Ithaca, NY, USA, 1987). IEEE Computer Society Press, 30–36.

[54] Benjamin C. Pierce. 2002. *Types and Programming Languages.* MIT Press.

[55] Andrew M. Pitts. 1987. Polymorphism is set theoretic, constructively. In *Category Theory and Computer Science*, David H. Pitt, Axel Poigné, and David E. Rydeheard (Eds.). Vol. 283. Springer, Berlin, Heidelberg, 12–39. doi:10.1007/3-540-18508-9_18

[56] Andrew M. Pitts. 1996. Relational Properties of Domains. *Information and Computation* 127, 2 (1996), 66–90. doi:10.1006/inco.1996.0052

[57] John C. Reynolds. 1984. Polymorphism Is Not Set-Theoretic. In *Semantics of Data Types* (Berlin, Heidelberg, 1984), Gilles Kahn, David B. MacQueen, and Gordon Plotkin (Eds.). Springer, 145–156. doi:10.1007/3-540-13346-1_7

[58] John C. Reynolds. 1984. Types, Abstraction, and Parametric Polymorphism. In *Information Processing 83: Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983* (Amsterdam, 1984), R. E. A Mason (Ed.). Elsevier Science Publishers B. V. (North-Holland), 513–523.

[59] John C. Reynolds. 1980. Using Category Theory to Design Implicit Conversions and Generic Operators. In *Semantics-Directed Compiler Generation* (Berlin, Heidelberg, 1980) *(Lecture Notes in Computer Science, Vol. 94)*, Neil D. Jones (Ed.). Springer, 211–258. doi:10.1007/3-540-10250-7_24

[60] Patrick M. Rondon, Ming Kawaguchi, and Ranjit Jhala. 2008. Liquid Types. *SIGPLAN Not.* 43, 6 (2008), 159–169. doi:10.1145/1379022.1375602

[61] Dana S. Scott. 1976. Data Types as Lattices. *SIAM J. Comput.* 5, 3 (1976), 522–587. doi:10.1137/0205037

[62] Michael B. Smyth and Gordon D. Plotkin. 1982. The Category-Theoretic Solution of Recursive Domain Equations. *SIAM J. Comput.* 11, 4 (1982), 761–783. doi:10.1137/0211062

[63] Thomas Streicher. 2006. *Domain-Theoretic Foundations of Functional Programming.* World Scientific. doi:10.1142/6284

[64] Nikhil Swamy, Cătălin Hriţcu, Chantal Keller, Aseem Rastogi, Antoine Delignat-Lavaud, Simon Forest, Karthikeyan Bhargavan, Cédric Fournet, Pierre-Yves Strub, Markulf Kohlweiss, Jean-Karim Zinzindohoue, and Santiago Zanella-Béguelin. 2016. Dependent Types and Multi-Monadic Effects in F*. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (New York, NY, USA, 2016-01-11) *(POPL '16)*. Association for Computing Machinery, 256–270. doi:10.1145/2837614.2837655

[65] Alfred Tarski. 1955. A Lattice-Theoretical Fixpoint Theorem and Its Applications. *Pacific J. Math.* 5, 2 (1955), 285–309. doi:10.2140/pjm.1955.5.285

[66] Niki Vazou and Michael Greenberg. 2022. How to Safely Use Extensionality in Liquid Haskell. In *Proceedings of the 15th ACM SIGPLAN International Haskell Symposium* (New York, NY, USA, 2022-09-06) *(Haskell 2022)*. Association for Computing Machinery, 13–26. doi:10.1145/3546189.3549919

[67] Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, and Simon Peyton Jones. 2014. Refinement Types for Haskell. *SIGPLAN Not.* 49, 9 (2014), 269–282. doi:10.1145/2692915.2628161

[68] Niki Vazou, Anish Tondwalkar, Vikraman Choudhury, Ryan G. Scott, Ryan R. Newton, Philip Wadler, and Ranjit Jhala. 2017. Refinement Reflection: Complete Verification with SMT. *Proc. ACM Program. Lang.* 2 (2017), 53:1–53:31. Issue POPL. doi:10.1145/3158141

A Compilation examples

The example of simply typed lambda calculus values (Section 2.1.4) can be compiled to the following non indexed type

```
data Val where
{-@ VIota :: Int → {v:Val | prop v = TIota} @-}
    VIota :: Int → Val
{-@ VArr :: t:Typ → u:Typ → ({v: Val | prop v = t} → {v: Val | prop v = u})
         → {v: Val | prop v = TArr t u} @-}
    VArr :: Typ → Typ → (Val → Val) → Val
```

B Elimination examples

B.1 Negative inductive types are still inconsistent

If the type declaration of Bad (Section 2.1.4) were allowed, where $\bot$ is an empty type:

$$\mathsf{Bad} : \mathsf{Ty} \triangleq \mu\, \mathsf{B}^{\langle + \rangle}\, \{\, \mathsf{MkBad}{:}(\mathsf{B} \to \bot) \to \mathsf{B}\,\}\,.$$

The eliminator for Bad would have type

$$\mathit{elim}_{\mathsf{Bad}}\ (c_{\mathsf{MkBad}}) : (b{:}\mathsf{Bad}) \to \rho(b), \text{where}$$

$$c_{\mathsf{MkBad}} : (f{:}\mathsf{Bad} \to \bot) \to (\mathsf{Bad} \to \mathsf{Unit}) \to \rho(\mathsf{MkBad}\ f)$$

We can show that Bad must be empty: instantiating the motive at the constant type $\rho(b) \triangleq \mathsf{Bad} \to \bot$, the eliminator has type $\mathsf{Bad} \to \mathsf{Bad} \to \bot$, and diagonalizing gives

$$\mathit{notBad} : \mathsf{Bad} \to \bot \triangleq \lambda\, b.\ \mathit{elim}_{\mathsf{Bad}}\ (\lambda\, f\ g.\ f)\ \ b\ b$$

But, at the same time, show that it is inhabited

$$\mathit{bad} : \mathsf{Bad} \triangleq \mathsf{MkBad}\ \mathit{notBad}$$

Hence, deriving a contradiction.

$$\mathit{contra} : \bot \triangleq \mathit{notBad}\ \mathit{bad}$$

The inconsistency arises because the eliminator exposes recursive occurrences of Bad directly. Mendler-style recursion, by contrast, hides those occurrences, preventing such self-application arguments and preserving normalization even for datatypes that are not positive.

B.2 Using Hofmann's routine

We can use Hofmann's routine type that we defined in CIRT in Example 3 to define the structural recursive definition of the `unfold` and `br` function from [12]:

$$\mathit{unfold} : \mathsf{Rou} \to (\mathsf{Rou} \to \mathsf{ListNat}) \to \mathsf{ListNat} \triangleq \mathit{elim}_{\mathsf{Rou}}\ (\lambda\, k.\ k\ \mathsf{Over}, \lambda\, f\ g.\ f)$$

$$\mathsf{TreeNat} : \mathsf{Ty} \triangleq \mu\, \mathsf{T}^{\langle + \rangle}\, \{\, \mathsf{Leaf}{:}\mathsf{Nat} \to \mathsf{T}, \mathsf{Node}{:}\mathsf{T} \to \mathsf{Nat} \to \mathsf{T} \to \mathsf{T}\,\},$$

$$\mathit{br} : \mathsf{TreeNat} \to \mathsf{Rou} \to \mathsf{Rou}$$

$$\triangleq \mathit{elim}_{\mathsf{TreeNat}} \left(\begin{array}{l} \lambda\, n\ r.\ \lambda\, c.\ \mathsf{Next}(\lambda\, k.\ \mathsf{Cons}\ n\ (\mathit{unfold}\ c\ k)), \\ \lambda\, \mathit{tl}\ n\ \mathit{tr}\ r_1\ r_2\ r_3.\ \lambda\, c.\ \mathsf{Next}(\lambda\, k.\ \mathsf{Cons}\ n\ (\mathit{unfold}\ c\ (k \circ r_1 \circ r_3))) \end{array} \right)$$

where the motive is the constant type $\mathsf{Rou} \to \mathsf{Rou}$, so that the induction hypotheses $r_1, r_3 : \mathsf{Rou} \to \mathsf{Rou}$ for the two subtrees can be composed with the continuation k ($r_2 : \mathsf{Unit}$ is the trivial hypothesis for the Nat argument).

The `extract` function is non structural, but by allowing non-structural recursion in CIRT, we can define it as follows:

$$extract : \mathsf{Rou} \to \mathsf{ListNat} \triangleq elim_{\mathsf{Rou}}\,(\mathsf{Nil}, \lambda\, f\; g.\; f\; extract)$$

Finally, we can define `breadthfirst` and complete Hofmann's program:

$$breadthfirst : \mathsf{TreeNat} \to \mathsf{ListNat} \triangleq \lambda\, t.\; extract\;\; (br\; t\;\; \mathsf{Over})$$

C Weakening, substitution and narrowing

We use the notation $\Gamma \vdash J$ to mean any of the judgments $\Gamma \vdash \kappa{:}\mathsf{Kd}$, $\Gamma \vdash \tau{:}\kappa$, $\Gamma \vdash t{:}\tau$, $\Gamma \vdash \phi\ \mathsf{prp}$, $\Gamma \vdash \tau \preceq \sigma$.

As usual, we have weakening and substitution lemmas for the syntactic judgments, which are proved by induction on the derivation. The weakening lemma is used in the proof of the substitution lemma.

Lemma C.1 (Weakening). *Let $\Gamma,\ x{:}_\tau\tau,\ \Delta$ (resp. $\Gamma,\ x{:}_\kappa\kappa,\ \Delta$) be a well-formed context, with x fresh. If $\Gamma,\ \Delta \vdash J$ is derivable, then so is the corresponding judgment over the enlarged context $\Gamma,\ x{:}_\tau\tau,\ \Delta \vdash J$ (resp. $\Gamma,\ x{:}_\kappa\kappa,\ \Delta \vdash J$).*

Proof. By induction on the derivation. Every rule is closed under inserting an unused binding: extend Δ every time a rule introduces a binder. □

Lemma C.2 (Syntactic substitution). *Let $\Gamma \vdash t{:}\tau$ (resp. $\Gamma \vdash \tau{:}\kappa$). If the judgment $\Gamma,\ x{:}_\tau\tau,\ \Delta \vdash J$ (resp. $\Gamma,\ x{:}_\kappa\kappa,\ \Delta \vdash J$) is derivable, then so is the judgment obtained by applying the substitution $[t/x]$ (resp. $[\tau/x]$) to all its components, $\Gamma,\ \Delta\,[t/x] \vdash J\,[t/x]$ (resp. $\Gamma,\ \Delta\,[\tau/x] \vdash J\,[\tau/x]$).*

Proof. By induction on the derivation. In the variable case for x the use of x is replaced by the assumed derivation of $\Gamma \vdash t{:}\tau$ (resp. $\Gamma \vdash \tau{:}\kappa$), weakened by Lemma C.1. The remaining rules commute with the substitution, which distributes over every former and passes under binders. □

Terms are interpreted by induction on their raw syntax, so substitution of a term commutes with the interpretation independently of any typing derivation.

Lemma C.3 (Term substitution). *For all terms t, u, every variable x and every runtime assignment γ defined on the free variables of t and of u other than x,*

$$(\!|u\,[t/x]|\!)\,(\gamma) = (\!|u|\!)\,(\langle \gamma,\ (\!|t|\!)\,(\gamma)\rangle)\,.$$

Proof. By induction on the raw syntax of u. □

The four semantic substitution lemmas below all assert that extending an environment by the denotation of the substituted term or type coincides with first performing the substitution syntactically.

Lemma C.4 (Semantic substitution: term into a kind, refined). *If $\vdash \Gamma,\ x{:}_\tau\sigma,\ \Delta\ \mathsf{ctx}$ and $\Gamma,\ x{:}_\tau\sigma,\ \Delta \vdash \kappa{:}\mathsf{Kd}$, then for every $\langle\langle \gamma,\ (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle,\ \delta\rangle \in \mathsf{dom}\,([\![\vdash \Gamma,\ x{:}_\tau\sigma,\ \Delta\ \mathsf{ctx}]\!])$,*

$$[\![\Gamma,\ x{:}_\tau\sigma,\ \Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\langle\langle \gamma,\ (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle,\ \delta\rangle) = [\![\Gamma,\ \Delta\,[t/x] \vdash \kappa\,[t/x]\ {:}\mathsf{Kd}]\!]\,(\langle \gamma,\ \delta\rangle)\,.$$

Lemma C.5 (Semantic substitution: term into a type, refined). *If $\vdash \Gamma,\ x{:}_\tau\sigma,\ \Delta\ \mathsf{ctx}$ and $\Gamma,\ x{:}_\tau\sigma,\ \Delta \vdash \rho{:}\kappa$, then for every $\langle\langle \gamma,\ (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle,\ \delta\rangle \in \mathsf{dom}\,([\![\vdash \Gamma,\ x{:}_\tau\sigma,\ \Delta\ \mathsf{ctx}]\!])$,*

$$[\![\Gamma,\ x{:}_\tau\sigma,\ \Delta \vdash \rho{:}\kappa]\!]\,(\langle\langle \gamma,\ (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle,\ \delta\rangle) = [\![\Gamma,\ \Delta\,[t/x] \vdash \rho\,[t/x]\ {:}\kappa\,[t/x]]\!]\,(\langle \gamma,\ \delta\rangle)\,.$$

Lemma C.6 (Semantic substitution: type into a kind, refined). *If* $\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx *and* $\Gamma, x{:}_\kappa \kappa, \Delta \vdash \sigma{:}\mathsf{Kd}$, *then for every* $\langle\langle \gamma, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma), \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \delta\rangle \in \mathrm{dom}([\![\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx$]\!])$,

$$[\![\Gamma, x{:}_\kappa \kappa, \Delta \vdash \sigma{:}\mathsf{Kd}]\!]\,(\langle\langle \gamma, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma), \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \delta\rangle) = [\![\Gamma, \Delta\,[\tau/x] \vdash \sigma\,[\tau/x]\,{:}\mathsf{Kd}]\!]\,(\langle \gamma, \delta\rangle)\,.$$

Lemma C.7 (Semantic substitution: type into a type, refined). *If* $\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx *and* $\Gamma, x{:}_\kappa \kappa, \Delta \vdash \sigma{:}\mu$, *then for every* $\langle\langle \gamma, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma), \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \delta\rangle \in \mathrm{dom}([\![\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx$]\!])$,

$$[\![\Gamma, x{:}_\kappa \kappa, \Delta \vdash \sigma{:}\mu]\!]\,(\langle\langle \gamma, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma), \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \delta\rangle) = [\![\Gamma, \Delta\,[\tau/x] \vdash \sigma\,[\tau/x]\,{:}\mu\,[\tau/x]]\!]\,(\langle \gamma, \delta\rangle)\,.$$

In the unrefined semantics substitution into a kind, and substitution of a term, are degenerate: the unrefined interpretation of every kind is $\mathcal{CUPER}\,(\mathcal{D})$ and that of a type never inspects the term assignment.

Lemma C.8 (Semantic substitution: type into a type, unrefined). *If* $\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx *and* $\Gamma, x{:}_\kappa \kappa, \Delta \vdash \sigma{:}\mu$, *then for every* $\langle\langle \gamma, \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle, \delta\rangle \in \mathrm{dom}\,(\lfloor\!\lfloor \vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx$\rfloor\!\rfloor)$,

$$\lfloor\!\lfloor \Gamma, x{:}_\kappa \kappa, \Delta \vdash \sigma{:}\mu \rfloor\!\rfloor\,(\langle\langle \gamma, \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle, \delta\rangle) = \lfloor\!\lfloor \Gamma, \Delta\,[\tau/x] \vdash \sigma\,[\tau/x]\,{:}\mu\,[\tau/x] \rfloor\!\rfloor\,(\langle \gamma, \delta\rangle)\,.$$

Lemma C.9 (Semantic substitution: term into a proposition). *If* $\vdash \Gamma, x{:}_\tau \sigma, \Delta$ ctx *and* $\Gamma, x{:}_\tau \sigma, \Delta \vdash \phi$ prp, *then for every* $\langle\langle \gamma, (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle, \delta\rangle \in \mathrm{dom}\,([\![\vdash \Gamma, x{:}_\tau \sigma, \Delta$ ctx$]\!])$,

$$\{\!|\Gamma, x{:}_\tau \sigma, \Delta \vdash \phi \text{ prp}|\!\}\,(\langle\langle \gamma, (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle, \delta\rangle) = \{\!|\Gamma, \Delta\,[t/x] \vdash \phi\,[t/x] \text{ prp}|\!\}\,(\langle \gamma, \delta\rangle)\,,$$

Lemma C.10 (Semantic substitution: type into a proposition). *If* $\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx *and* $\Gamma, x{:}_\kappa \kappa, \Delta \vdash \phi$ prp, *then for every* $\langle\langle \gamma, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma), \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \delta\rangle \in \mathrm{dom}([\![\vdash \Gamma, x{:}_\kappa \kappa, \Delta$ ctx$]\!])$,

$$\{\!|\Gamma, x{:}_\kappa \kappa, \Delta \vdash \phi \text{ prp}|\!\}\,(\langle\langle \gamma, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma), \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \delta\rangle) = \{\!|\Gamma, \Delta\,[\tau/x] \vdash \phi\,[\tau/x] \text{ prp}|\!\}\,(\langle \gamma, \delta\rangle)\,,$$

Proof of Lemmas C.4 to C.10. The statements are proved by induction on the derivation of the substituted judgment, the cases that reach a term are discharged by Lemma C.3. The right-hand sides are well-formed by Lemma C.2. □

Lemma C.11 (Syntactic narrowing). *Let* $\Gamma \vdash \tau \preceq \sigma$ *and* $\Gamma \vdash \tau{:}\mathsf{Ty}$. *If* $\Gamma, x{:}_\tau \sigma, \Delta \vdash J$ *is derivable, then so is* $\Gamma, x{:}_\tau \tau, \Delta \vdash J$.

Proof. By induction on the derivation. Apply the Tm-Sub and Lemma C.1 in the Tm-Var case. □

Lemma C.12 (Semantic narrowing). *Let* $\Gamma \vdash \tau \preceq \sigma$, *and suppose both* $\Gamma, x{:}_\tau \tau, \Delta \vdash J$ *and* $\Gamma, x{:}_\tau \sigma, \Delta \vdash J$ *are derivable.*

(i) Refined: If J *is a kind or type judgment and* $\gamma \in \mathrm{dom}\,([\![\vdash \Gamma, x{:}_\tau \tau, \Delta$ ctx$]\!])$ *and* $\gamma \in \mathrm{dom}([\![\vdash \Gamma, x{:}_\tau \sigma, \Delta$ ctx$]\!])$, *then*

$$[\![\Gamma, x{:}_\tau \tau, \Delta \vdash J]\!]\,(\gamma) = [\![\Gamma, x{:}_\tau \sigma, \Delta \vdash J]\!]\,(\gamma)\,,$$

(ii) Unrefined: If J *is a kind or type judgment and* $\gamma \in \mathrm{dom}\,(\lfloor\!\lfloor \vdash \Gamma, x{:}_\tau \tau, \Delta$ ctx$\rfloor\!\rfloor)$ *and* $\gamma \in \mathrm{dom}(\lfloor\!\lfloor \vdash \Gamma, x{:}_\tau \sigma, \Delta$ ctx$\rfloor\!\rfloor)$, *then*

$$\lfloor\!\lfloor \Gamma, x{:}_\tau \tau, \Delta \vdash J \rfloor\!\rfloor\,(\gamma) = \lfloor\!\lfloor \Gamma, x{:}_\tau \sigma, \Delta \vdash J \rfloor\!\rfloor\,(\gamma)\,,$$

(iii) Propositions: If $\gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma,\ x{:}_\tau\tau,\ \Delta\ \mathsf{ctx} \rrbracket)$ *and* $\gamma \in \mathsf{dom}\,(\llbracket \vdash \Gamma,\ x{:}_\tau\sigma,\ \Delta\ \mathsf{ctx} \rrbracket)$*, then*

$$\{\!\{\Gamma,\ x{:}_\tau\tau,\ \Delta \vdash \phi\ \mathsf{prp}\}\!\}\,(\gamma) = \{\!\{\Gamma,\ x{:}_\tau\sigma,\ \Delta \vdash \phi\ \mathsf{prp}\}\!\}\,(\gamma)\,.$$

Proof. By structural induction on J. By Lemma C.11 the two derivations differ only by Tm-Sub steps inserted at occurrences of x; every kind, type, and proposition formation node is shared, and no such node inspects the type annotation of a context entry. The projections $\iota_-\,(-)$ and $\rho_-\,(-)$ either drop the entry for x or forward its value without reading its annotation, and terms enter the interpretations only through $(\!|-|\!)$, which is defined on raw syntax and depends only on $\rho_-\,(-)$; hence the inserted Tm-Sub steps are invisible to the semantics. □

D Domain theory

Definition 21 (Complete lattice). *A complete lattice is a partially ordered set* $(L, \leq)$ *such that every subset* $S \subseteq L$ *has a least upper bound* $\bigsqcup S$ *and a greatest lower bound* $\bigsqcap S$ *in* L*.*

Definition 22 (Complete partial order). *A complete partial order (CPO) is a partially ordered set* $(P, \leq)$ *such that every directed subset* $D \subseteq P$ *has a least upper bound* $\bigsqcup D$ *in* P*.*
A CPO is pointed *if it has a least element* $\bot \in P$*.*

Definition 23 (Monotone function). *Given two partially ordered sets* $(P, \leq_P)$ *and* $(Q, \leq_Q)$*, a function* $f : P \to Q$ *is monotone if for all* $p_1, p_2 \in P$*, if* $p_1 \leq_P p_2$ *then* $f(p_1) \leq_Q f(p_2)$*.*

Definition 24 (Continuous function). *Given two complete partial orders* $(P, \leq_P)$ *and* $(Q, \leq_Q)$*, a monotone* $f : P \to Q$ *is continuous if for every directed subset* $D \subseteq P$ $f(\bigsqcup D) = \bigsqcup f(D)$*.*

Lemma D.1 (Projection of tagged tuples). *For every* n *there exists* $n_j \leq n$ *such that*

$$\pi_{n+1}(\langle C, \overline{v_j}^{\,m}\rangle) = \langle \pi_n(C), \overline{\pi_{n_j}(v_j)}^{\,m}\rangle$$

E Supporting lemmas

The equivalent theorem to Lemma 7.2 for kinds.

Lemma E.1 ((Anti-)Monotonicity of kinds). *Let* $\Delta \triangleq \Gamma,\ K{:}_\kappa\mu,\ \overline{x{:}\sigma}^{\,i}$*, where* x_i *can bind both type and term variables; given two context interpretations different only on the assignment of* K $\langle\langle\gamma,\ \langle P,\ P_\bot\rangle\rangle,\ \overline{v}^{\,i}\rangle, \langle\langle\gamma,\ \langle Q,\ P_\bot\rangle\rangle,\ \overline{v}^{\,i}\rangle \in \mathsf{dom}\,(\llbracket \vdash \Delta\ \mathsf{ctx} \rrbracket)$*, where* $P \leq Q$*. Let* $\vdash \Delta\ \mathsf{ctx}$ *and* $\Delta \vdash \kappa{:}\mathsf{Kd}$ *be well-formed judgments, then*

(i) If $\mathsf{pol}_K\,(\kappa) \leq +$*, then* $\llbracket \Delta \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\langle\langle\gamma,\ \langle P,\ P_\bot\rangle\rangle,\ \overline{v}^{\,i}\rangle) \subseteq \llbracket \Delta \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\langle\langle\gamma,\ \langle Q,\ P_\bot\rangle\rangle,\ \overline{v}^{\,i}\rangle)$*,*
(ii) If $\mathsf{pol}_K\,(\kappa) \leq -$*, then* $\llbracket \Delta \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\langle\langle\gamma,\ \langle P,\ P_\bot\rangle\rangle,\ \overline{v}^{\,i}\rangle) \supseteq \llbracket \Delta \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\langle\langle\gamma,\ \langle Q,\ P_\bot\rangle\rangle,\ \overline{v}^{\,i}\rangle)$*.*

The following is the analogue of Lemma 7.3 for the unrefined semantics; as discussed in section 7.2, a less general statement suffices here, since no under-approximation of the inductive type is needed.

Lemma E.2 (Unrefined semantics is PER-respecting (Eliminator)).
Let $D \triangleq \mu K^{\langle\nu\rangle}\left\{\underline{C{:}(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i}\right\}, \Delta(i) \triangleq K{:}_\kappa(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty},\ \overline{y{:}\sigma}^{\,i}$ *and* $P_\bot(\gamma) \triangleq \llbracket \Gamma \vdash D{:}(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty} \rrbracket\,(\iota_\Gamma\,(\gamma))$*. If the following judgments are well-formed:* $\vdash \Gamma,\ \Delta(j)\ \mathsf{ctx}$ *and* $\Gamma,\ \overline{x{:}\tau}^{\,k},\ y{:}_\tau D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty}$*. Given two related context interpretations* $\langle\langle\gamma,\ P_\bot(\gamma)\rangle,\ \epsilon\rangle,\ \langle\langle\delta,\ P_\bot(\delta)\rangle,\ \theta\rangle\rangle \in \llbracket \vdash \Gamma,\ \Delta(j-1)\ \mathsf{ctx} \rrbracket$*, two related realizers of the eliminator* $\langle r,\ r'\rangle \in \llbracket \Gamma \vdash (\overline{x{:}_\tau\tau}^{\,k}) \to D \cdot \overline{x}^{\,k} \to \rho{:}\mathsf{Ty} \rrbracket\,(\gamma)$ *and some term* $\Gamma,\ \Delta(j-1) \vdash m{:}\sigma_j$*, then the eliminators realize induction hypothesis over argument* m*:*

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\,(\sigma_j,\ \gamma_\rho,\ (\!|m|\!)\,(\gamma_\rho))\,, \\ \mathsf{tmEl}_{K;r'}\,(\sigma_j,\ \delta_\rho,\ (\!|m|\!)\,(\delta_\rho)) \end{array} \right\rangle \in \left\llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\,(m,\ \sigma_j)\ {:}\mathsf{Ty} \right\rrbracket\,(\langle\langle\gamma,\ P_\bot(\gamma)\rangle,\ \epsilon\rangle)$$

Where $\langle\gamma_\rho,\ \delta_\rho\rangle \in \|\Gamma,\ \Delta(j-1)\|\,(\langle\langle\gamma,\ P_\bot(\gamma)\rangle,\ \epsilon\rangle)$

F Main Proofs

The unrefined statements (Theorem 5.1 (ii) , Theorem 5.2 (ii) , Theorem 5.3 (ii) , Theorem 5.6 (ii) , Theorem 6.2 (ii) , Theorem 7.1, and Lemma E.2) never mention the refined interpretations and are proved by mutual induction on typing derivations. Monotonicity (Lemmas E.1 and 7.2) and Theorem 5.4, together with Corollary 1 and Lemma 5.5, follow by induction on derivations, using only the statements above. These lemmas are statements *about* the refined interpretations but consume only their definitions, never the refined statements below; the definitions themselves are given by induction on derivations, simultaneously with Theorem 7.1 and Lemma 7.2, which justify the fixed points taken in the interpretation of inductive types at strictly smaller derivations. Finally, the refined statements (Theorem 5.1 (i) , Theorem 5.2 (i) , Theorem 5.3 (i) , Theorem 5.6 (i) , Theorem 6.2 (i) , Theorem 6.1, and Lemma 7.3) are proved by one further mutual induction on typing derivations; in particular, a refined statement may invoke its unrefined counterpart at the *same* judgment.

Proof of Lemma 3.2. If $\langle \pi_n(a),\, \pi_n(b)\rangle \in S_n$ with $\langle a,\, b\rangle \in S$, then $\langle \pi_n(a),\, \pi_n(b)\rangle \in S$ by uniformity, and $\pi_n(a), \pi_n(b) \in D_n$. Conversely, if $\langle x,\, y\rangle \in S$ with $x, y \in D_n$ then $x = \pi_n(x)$ and $y = \pi_n(y)$, so $\langle x,\, y\rangle \in S_n$. For the inclusion, $D_j \subseteq D_n$ when $j \le n$ because $\pi_n \circ \pi_j = \pi_j$, whence $S \cap (D_j \times D_j) \subseteq S \cap (D_n \times D_n)$. □

Proof of Lemma 3.3. Let $\langle a,\, b\rangle \in A$. By uniformity and lemma 3.2, $\langle \pi_n(a),\, \pi_n(b)\rangle \in A_n = B_n \subseteq B$ for every n. The family $\{\langle \pi_n(a),\, \pi_n(b)\rangle\}_n$ is directed with least upper bound $\langle a,\, b\rangle$, since $\bigsqcup_n \pi_n = id$; by completeness of B, $\langle a,\, b\rangle \in B$. Hence $A \subseteq B$, and symmetrically $B \subseteq A$. □

Proof of Lemma 3.4. *Uniqueness.* Suppose $\Phi(R) = R$ and $\Phi(S) = S$. We show $R_n = S_n$ by induction on n. Since π_0 is the constant map $\bot$, both approximations equal $\{\langle \bot,\, \bot\rangle\}$ at $n = 0$; and if $R_n = S_n$ then contractivity gives $R_{n+1} = \Phi(R)_{n+1} = \Phi(S)_{n+1} = S_{n+1}$. By lemma 3.3, $R = S$.

Existence. Put $R^0 \triangleq D \times D$ and $R^{k+1} \triangleq \Phi(R^k)$; each R^k is a CUPER. Using contractivity, a routine induction shows the iterates stabilise, $(R^m)_k = (R^k)_k$ whenever $m \ge k$, the base case holding because all CUPERs agree at level 0. Writing $R_n \triangleq (R^n)_n$ for the stabilised level-n data—not yet known to coincide with the approximation $(R)_n$ of the limit R defined below—the family is increasing and coherent, $(R_{n+1})_n = R_n$, by lemma 3.2 and stabilisation. Define

$$R \triangleq \left\{ \langle a,\, b\rangle \in D \times D \,\middle|\, \forall n.\ \langle \pi_n(a),\, \pi_n(b)\rangle \in R_n \right\}.$$

Then R is a CUPER: symmetry and transitivity hold because each $R_n = R^n \cap (D_n \times D_n)$ is a PER (lemma 3.2); pointedness is immediate, and uniformity follows from the monotonicity $R_j \subseteq R_n$ $(j \le n)$; and for completeness, if $\{\langle a_i,\, b_i\rangle\}_i \subseteq R$ is directed with least upper bound $\langle a,\, b\rangle$, then continuity of each π_n makes $\langle \pi_n(a),\, \pi_n(b)\rangle$ the least upper bound of the directed family $\{\langle \pi_n(a_i),\, \pi_n(b_i)\rangle\}_i \subseteq R_n$; since $R_n = R^n \cap (D_n \times D_n)$ is closed under directed least upper bounds—R^n being complete and $D_n \times D_n$ Scott-closed—this bound lies in R_n, so $\langle a,\, b\rangle \in R$. Moreover $(R)_n = R_n$: the inclusion $(R)_n \subseteq R_n$ is immediate, and conversely, given $\langle x,\, y\rangle \in R_n \subseteq D_n \times D_n$, coherence and monotonicity give $\langle \pi_m(x),\, \pi_m(y)\rangle \in R_m$ for every m, so $\langle x,\, y\rangle \in R$; as $x = \pi_n(x)$ and $y = \pi_n(y)$, this places $\langle x,\, y\rangle$ in $(R)_n$. Finally R is a fixed point: R and R^n share the n-th approximation R_n, so contractivity gives $\Phi(R)_{n+1} = \Phi(R^n)_{n+1} = (R^{n+1})_{n+1} = R_{n+1}$, while $\Phi(R)_0 = \{\langle \bot,\, \bot\rangle\} = R_0$; hence $\Phi(R) = R$ by lemma 3.3. □

Proof of Lemma D.1. In the bilimit, $\pi_{n+1}\,\langle a,\, b\rangle = \langle \pi_n(a),\, \pi_n(b)\rangle$ hence we can apply the identity recursively inside the nested pairs. □

Proof of Theorem 5.1 (i) . By induction on the structure of $\Gamma \vdash \kappa$:Kd.

- Kd-Type

$$[\![\Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma) = \mathcal{PER}\,(\mathcal{D}) = [\![\Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\delta)$$

- Kd-Fun

$$\begin{aligned}
&[\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\gamma) \\
&= \left\{ f \;\middle|\; \begin{array}{c} f \in \mathcal{D} \to (\mathcal{D} \to)^{\mathsf{arity}(\kappa)}\, \mathcal{PER}\,(\mathcal{D}) \\ \forall v \in \mathsf{dom}\,([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma))\;\; f(v) \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd}]\!]\,(\langle \gamma,\, v\rangle) \\ \forall\, \langle v,\, w\rangle \in \left([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma)) \cap \mathsf{dom}\,([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma))^2\right)\;\; f(v) = f(w) \end{array} \right\} \\
&\quad \text{By I. H. and Theorem 5.3} \\
&= \left\{ f \;\middle|\; \begin{array}{c} f \in \mathcal{D} \to (\mathcal{D} \to)^{\mathsf{arity}(\kappa)}\, \mathcal{PER}\,(\mathcal{D}) \\ \forall v \in \mathsf{dom}\,([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta))\;\; f(v) \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd}]\!]\,(\langle \delta,\, v\rangle) \\ \forall\, \langle v,\, w\rangle \in \left([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\delta)) \cap \mathsf{dom}\,([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta))^2\right)\;\; f(v) = f(w) \end{array} \right\} \\
&= [\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\delta)
\end{aligned}$$

□

Proof of Theorem 5.1 (ii). By induction on the structure of $\Gamma \vdash \kappa{:}\mathsf{Kd}$.

- Kd-Type

$$[\![\Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma) = \mathcal{CUPER}\,(\mathcal{D}) = [\![\Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\delta)$$

- Kd-Fun

$$\begin{aligned}
&[\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\gamma) \\
&= [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma) \\
&\quad \text{By I. H.} \\
&= [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta) \\
&= [\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\delta)
\end{aligned}$$

□

Proof of Theorem 5.2 (i). By induction on the structure of $\Gamma \vdash \tau{:}\kappa$.

- Ty-Nat and Ty-Bool

$$[\![\Gamma \vdash \mathcal{B}{:}\kappa]\!]\,(\gamma) = \left\{ \langle i_{\mathcal{B}}\,(v)\,,\, i_{\mathcal{B}}\,(v)\rangle \,\middle|\, v \in \mathcal{B} \right\} \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma) = \mathcal{PER}\,(\mathcal{D})$$

The diagonal relation over a subset of $\mathcal{D}$ is a PER.

- Ty-Fun

$$\begin{aligned}
&[\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \;\in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)\,. \qquad \langle f \cdot v,\, g \cdot w\rangle \;\in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, v\rangle) \\ \forall \langle v_\perp,\, w_\perp\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))\,.\; \langle f \cdot v_\perp,\, g \cdot w_\perp\rangle \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma)) \end{array} \right\} \\
&\in [\![\Gamma \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma) = \mathcal{PER}\,(\mathcal{D})
\end{aligned}$$

It is not enough to show that each of the two defining conditions determines a PER on its own, since neither is in general closed under symmetry by itself. Instead, we show directly that the whole set is a PER.

(1) **Symmetry**: Fix $\langle f,\, g\rangle$ in the set. We must show that $\langle g,\, f\rangle$ satisfies the defining condition for every $\langle v,\, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$ and $\langle v_\perp,\, w_\perp\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))$. By symmetry of the domain PERs, $\langle w,\, v\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$ and $\langle w_\perp,\, v_\perp\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))$. Instantiating the assumption for $\langle f,\, g\rangle$ at these swapped arguments gives

$$\langle f \cdot w,\, g \cdot v\rangle \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, w\rangle)$$

and

$$\langle f \cdot w_\perp,\, g \cdot v_\perp \rangle \in \llbracket \Gamma,\, x :_\tau \tau \vdash \sigma : \mathsf{Ty} \rrbracket\, (\iota_\Gamma\, (\gamma))\,.$$

By symmetry of these PERs we obtain the same pairs in the opposite order. Finally, $\langle \langle \gamma,\, w \rangle,\, \langle \gamma,\, v \rangle \rangle \in \llbracket \vdash \Gamma,\, x :_\tau \tau\ \mathsf{ctx} \rrbracket$, so Theorem 5.3 (i) transports the refined codomain relation from $[\langle \gamma,\, w \rangle]$ to $[\langle \gamma,\, v \rangle]$. Hence $\langle g,\, f \rangle$ is in the set.

(2) **Transitivity**: Fix $\langle f,\, g \rangle$ and $\langle g,\, h \rangle$ in the set. We again fix arbitrary $\langle v,\, w \rangle \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\, (\gamma)$ and $\langle v_\perp,\, w_\perp \rangle \in \lfloor \Gamma \vdash \tau : \mathsf{Ty} \rfloor\, (\iota_\Gamma\, (\gamma))$. By partial reflexivity, $\langle v,\, v \rangle \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\, (\gamma)$ and $\langle v_\perp,\, v_\perp \rangle \in \lfloor \Gamma \vdash \tau : \mathsf{Ty} \rfloor\, (\iota_\Gamma\, (\gamma))$. Instantiating the first assumption at these reflexive pairs gives

$$\langle f \cdot v,\, g \cdot v \rangle \in \llbracket \Gamma,\, x :_\tau \tau \vdash \sigma : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, v \rangle)$$

and

$$\langle f \cdot v_\perp,\, g \cdot v_\perp \rangle \in \lfloor \Gamma,\, x :_\tau \tau \vdash \sigma : \mathsf{Ty} \rfloor\, (\iota_\Gamma\, (\gamma))\,.$$

Instantiating the second assumption at $\langle v,\, w \rangle$ and $\langle v_\perp,\, w_\perp \rangle$ gives the corresponding pairs from g to h. Transitivity of the codomain PERs therefore yields

$$\langle f \cdot v,\, h \cdot w \rangle \in \llbracket \Gamma,\, x :_\tau \tau \vdash \sigma : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, v \rangle)$$

and

$$\langle f \cdot v_\perp,\, h \cdot w_\perp \rangle \in \lfloor \Gamma,\, x :_\tau \tau \vdash \sigma : \mathsf{Ty} \rfloor\, (\iota_\Gamma\, (\gamma))\,.$$

Hence $\langle f,\, h \rangle$ is in the set.

- Ty-All

$$\begin{aligned}
&\llbracket \Gamma \vdash \forall x{:}\kappa.\, \tau : \mathsf{Ty} \rrbracket\, (\gamma) \\
&= \left\{ \langle v,\, w \rangle \,\middle|\, \begin{array}{c} \forall P \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\, (\gamma)\,,\ \forall P_\perp \in \lfloor \Gamma \vdash \kappa : \mathsf{Kd} \rfloor\, (\iota_\Gamma\, (\gamma)) \\ \pi_{\Gamma;\,\kappa}\, (\gamma,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w \rangle \in \llbracket \Gamma,\, x :_\kappa \kappa \vdash \tau : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, \langle P,\, P_\perp \rangle \rangle) \end{array} \right\} \\
&\in \llbracket \Gamma \vdash \mathsf{Ty} : \mathsf{Kd} \rrbracket\, (\gamma) = \mathcal{PER}\, (\mathcal{D})
\end{aligned}$$

Since PERs are closed under intersection, it suffices to show that for each fixed $P \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\, (\gamma)$ and $P_\perp \in \lfloor \Gamma \vdash \kappa : \mathsf{Kd} \rfloor\, (\iota_\Gamma\, (\gamma))$ the set

$$\left\{ \langle v,\, w \rangle \,\middle|\, \pi_{\Gamma;\,\kappa}\, (\gamma,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w \rangle \in \llbracket \Gamma,\, x :_\kappa \kappa \vdash \tau : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, \langle P,\, P_\perp \rangle \rangle) \right\}$$

is a PER. If $\pi_{\Gamma;\,\kappa}\, (\gamma,\, P) \not\subseteq P_\perp$ the implication holds vacuously for every pair, so the set is all of $\mathcal{D} \times \mathcal{D}$, which is trivially a PER. Otherwise the set equals $\llbracket \Gamma,\, x :_\kappa \kappa \vdash \tau : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, \langle P,\, P_\perp \rangle \rangle)$, which we show is a PER.

(1) **Symmetry**: Fix $\langle v,\, w \rangle$ in the set, by I. H. and symmetry of PERs we have $\langle w,\, v \rangle \in \llbracket \Gamma,\, x :_\kappa \kappa \vdash \tau : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, \langle P,\, P_\perp \rangle \rangle)$ thus $\langle w,\, v \rangle$ is in the set.

(2) **Transitivity**: Fix $\langle v,\, w \rangle$ and $\langle w,\, q \rangle$ in the set, by I. H. and transitivity of PERs we have $\langle v,\, q \rangle \in \llbracket \Gamma,\, x :_\kappa \kappa \vdash \tau : \mathsf{Ty} \rrbracket\, (\langle \gamma,\, \langle P,\, P_\perp \rangle \rangle)$ thus $\langle v,\, q \rangle$ is in the set.

- Ty-Var

$$\llbracket \Gamma \vdash x : \kappa \rrbracket\, (\gamma) = \gamma\, (x)_1 \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\, (\gamma)$$

by assumption.

- Ty-Ref

$$\llbracket \Gamma \vdash \{x{:}\tau \mid \phi\} : \mathsf{Ty} \rrbracket\, (\gamma) = \left\{ \langle v,\, w \rangle \,\middle|\, \begin{array}{c} \langle v,\, w \rangle \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\, (\gamma) \\ \{\!\!\{ \Gamma,\, x :_\tau \tau \vdash \phi\ \mathsf{prp} \}\!\!\}\, (\langle \gamma,\, v \rangle) = tt \end{array} \right\}$$
$$\in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\, (\gamma) = \mathcal{PER}\, (\mathcal{D})$$

we show that it is a PER

(1) **Symmetry**: Fix $\langle v, w\rangle$ in the set, by I. H. and symmetry of PERs we have $\langle w, v\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$ and by Theorem 6.1 we have $\{\!|\Gamma, x{:_\mathrm{T}}\tau \vdash \phi\ \mathsf{prp}|\!\}\,(\langle\gamma, w\rangle) = tt$ thus $\langle w, v\rangle$ is in the set.

(2) **Transitivity**: Fix $\langle v, w\rangle$ and $\langle w, q\rangle$ in the set, by I. H. and transitivity of PERs we have $\langle v, q\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$ and thus $\langle v, q\rangle$ is in the set.

- Ty-App

$$[\![\Gamma \vdash \sigma\ t{:}\kappa\,[t/x]]\!]\,(\gamma) = [\![\Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa]\!]\,(\gamma)\,((\!|t|\!)\,(\rho_\Gamma\,(\gamma))) \in [\![\Gamma \vdash \kappa\,[t/x]\,{:}\mathsf{Kd}]\!]\,(\gamma)$$

by I. H. $f = [\![\Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa]\!]\,(\gamma) \in [\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\gamma)$ thus $f\,((\!|t|\!)\,(\rho_\Gamma\,(\gamma))) \in [\![\Gamma, x{:_\mathrm{T}}\tau \vdash \kappa{:}\mathsf{Kd}]\!]\,(\langle\gamma, (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\rangle)$ by Lemma C.4 $= [\![\Gamma \vdash \kappa\,[t/x]\,{:}\mathsf{Kd}]\!]\,(\gamma)$.

- Ty-IndPos and Ty-IndStrat

$$\left[\!\!\left[\Gamma \vdash \mu K^{\langle\nu\rangle}\left\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K\cdot\overline{t}^{\,k}}_{\,i}\right\}{:}\overline{\tau}^{\,k} \to \mathsf{Ty}\right]\!\!\right](\gamma)$$

$$= \mathsf{lfp}_X\Bigg(\underbrace{\overline{q}^{\,k} \mapsto \bigcup_i \left\{\langle\langle C_i, \overline{v}^{\,j}\rangle, \langle C_i, \overline{w}^{\,j}\rangle\rangle \,\middle|\, \langle v_j, w_j\rangle \in [\![\Gamma, K{:_\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty}, \overline{x{:_\mathrm{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle\gamma, \langle X, [\![\Gamma \vdash \mu K^{\langle\nu\rangle}\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K\cdot\overline{t}^{\,k}}_{\,i}\}{:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))\rangle\rangle, \overline{v}^{\,j-1}\rangle)\right\}}_{\alpha}\Bigg)$$

$$\in [\![\Gamma \vdash \overline{\tau}^{\,k} \to \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma)$$

First, for the shared part of the two rules, we show that if $X \in [\![\Gamma \vdash \overline{\tau}^{\,k} \to \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma)$ lies below the unrefined interpretation of the declaration, so that the environment in α is well formed, then $\alpha \in [\![\Gamma \vdash \overline{\tau}^{\,k} \to \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma)$; we have to show two things: Fixed $\langle\overline{q}^{\,k}, \overline{n}^{\,k}\rangle \in [\!|\Gamma \vdash \overline{\tau}^{\,k}{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma))$ we show that:

- Fixed $\overline{q}^{\,k} \in \mathrm{dom}\,([\![\Gamma \vdash \overline{\tau}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma))$, $\alpha(\overline{q}^{\,k})$ is a PER, since it is the union of disjoint C_i we just need to show that each C_i is a PER, by Theorem 5.3 (i) the semantics is invariant under related contexts and hence it is a PER.
- Fixed $\langle\overline{q}^{\,k}, \overline{n}^{\,k}\rangle \in [\!|\Gamma \vdash \overline{\tau}^{\,k}{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma)) \cap \mathrm{dom}\,([\![\Gamma \vdash \overline{\tau}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma))^2$ we show that $\alpha(\overline{q}^{\,k}) = \alpha(\overline{n}^{\,k})$; it follows that $\langle\overline{q}^{\,k}, \overline{n}^{\,k}\rangle \in [\!|\Gamma \vdash \overline{\tau}^{\,k}{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma))$ hence either $\forall k\ \langle q_k, (\!|t_k|\!)\,(\langle\rho_\Gamma\,(\gamma), \overline{v}^{\,j}\rangle)\rangle \in [\!|\Gamma, \overline{x{:_\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma))$ thus by transitivity of the PERs $\langle n_k, (\!|t_k|\!)\,(\langle\rho_\Gamma\,(\gamma), \overline{v}^{\,j}\rangle)\rangle \in [\!|\Gamma, \overline{x{:_\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma))$ and hence the corresponding C_i are equal, as the semantics of each σ_j does not depend on q_k; or if $\exists k\ \langle q_k, (\!|t_k|\!)\,(\langle\rho_\Gamma\,(\gamma), \overline{v}^{\,j}\rangle)\rangle \notin [\!|\Gamma, \overline{x{:_\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma))$ then by transitivity of the PERs $\langle n_k, (\!|t_k|\!)\,(\langle\rho_\Gamma\,(\gamma), \overline{v}^{\,j}\rangle)\rangle \notin [\!|\Gamma, \overline{x{:_\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}|\!]\,(\iota_\Gamma\,(\gamma))$ and hence the corresponding sets C_i are empty and hence equal.

Now given this we can show that the least fixed point of α exists, we consider the two cases separately:

- Ty-IndPos By assumption $\forall l, j.\ \mathrm{pol}_K\,(\tau_{l,j}) \leq +$ and by Lemma 7.2 each $[\![\Gamma, K{:_\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty}, \overline{x{:_\mathrm{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle\langle\gamma, \langle X, \ldots\rangle\rangle, \overline{v}^{\,j-1}\rangle)$ is monotone in X, thus α is monotone on such X, which form a complete lattice, and by Theorem 3.1 the least fixpoint of α exists.
- Ty-IndStrat By assumption each self reference in $\tau_{i,j}$ is on a strictly smaller index with respect to the stratification function m, hence we can give the alternative definition of the functional

$$\mathcal{F}_{rec}\,(\overline{v}^{\,k}) = (\alpha^{(1+n)}(\overline{q}^{\,k} \mapsto \emptyset))(\overline{v}^{\,k}) \text{ where } n \triangleq r_{\mathbb{N}}\,((\!|m|\!)\,(\rho_\Gamma\,(\gamma)) \cdot \overline{v}^{\,k})$$

where $\alpha^{(n)}$ is the n-fold composition of α.

Since all the recursive occurrences are on strictly smaller indices, this is well defined. This follows from the fact that fixed $v_k \in \mathrm{dom}([\![\Gamma, x{:_\mathrm{T}}\overline{\tau_k}^{\,k-1} \vdash \mathsf{Ty}{:}\mathsf{Ty}]\!]\,(\langle\gamma, \overline{v}^{\,k-1}\rangle))$

$\left[\!\left[\Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C\colon (\overline{y\colon\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} \colon \overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right] (\gamma) \cdot \overline{v}^{\,k}$ cannot depend on any recursive occurrences of $\overline{w}^{\,k}$ where $r_{\mathbb{N}}\,(\!(m)\!)\,(\rho_\Gamma\,(\gamma)) \cdot \overline{w}^{\,k}) \geq r_{\mathbb{N}}\,(\!(m)\!)\,(\rho_\Gamma\,(\gamma)) \cdot \overline{v}^{\,k})$ by the assumption of the stratification condition.

This definition is stable, meaning that forall $\overline{q}^{\,k} \in \mathsf{dom}(\llbracket \Gamma,\ x \colon_{\mathsf{T}} \overline{\tau}^{\,k-1} \vdash \mathsf{Ty}\colon\mathsf{Ty} \rrbracket(\langle \gamma,\ \overline{v}^{\,k-1} \rangle))$ where $n = r_{\mathbb{N}}\,(\!(m)\!)\,(\rho_\Gamma\,(\gamma)) \cdot \overline{q}^{\,k})\ (\alpha^{(1+n)}(\overline{q}^{\,k} \mapsto \emptyset))(\overline{q}^{\,k}) = (\alpha^{(1+m)}(\overline{q}^{\,k} \mapsto \emptyset))(\overline{q}^{\,k})$ for all $m \geq n$. By strong induction on n: $n = 0$ then there are no recursive occurrences and hence the definition is stable by definition as the argument to α does not matter. For $n > 0$ the semantics depends only on the recursive occurrences of strictly smaller indices, for which by the I. H. the definition is stable, hence the whole definition is stable.

Now we show that it is a fixed point, hence $\mathcal{F}_{rec} = \alpha(\mathcal{F}_{rec})$; fixed $v_k \in \mathsf{dom}(\llbracket \Gamma,\ x \colon_{\mathsf{T}} \overline{\tau}^{\,k-1} \vdash \mathsf{Ty}\colon\mathsf{Ty} \rrbracket(\langle \gamma,\ \overline{v}^{\,k-1} \rangle))$ and let $n \triangleq r_{\mathbb{N}}\,(\!(m)\!)\,(\rho_\Gamma\,(\gamma)) \cdot \overline{v}^{\,k})$ we have

$$
\begin{aligned}
& \mathcal{F}_{rec}(\overline{v}^{\,k}) \\
=\ & (\alpha^{(1+n)}(\overline{q}^{\,k} \mapsto \emptyset))(\overline{v}^{\,k}) \\
& \text{By stability} \\
=\ & (\alpha^{(2+n)}(\overline{q}^{\,k} \mapsto \emptyset))(\overline{v}^{\,k}) \\
=\ & \alpha(\mathcal{F}_{rec})(\overline{v}^{\,k})
\end{aligned}
$$

Finally we show that it is unique and hence trivially the least fixed point. Let G be another fixed point of α thus $G = \alpha(G)$, fixed $v_k \in \mathsf{dom}(\llbracket \Gamma,\ x \colon_{\mathsf{T}} \overline{\tau}^{\,k-1} \vdash \mathsf{Ty}\colon\mathsf{Ty} \rrbracket(\langle \gamma,\ \overline{v}^{\,k-1} \rangle))$ and let $n \triangleq r_{\mathbb{N}}\,(\!(m)\!)\,(\rho_\Gamma\,(\gamma)) \cdot \overline{v}^{\,k})$ by induction on n: if $n = 0$ then

$$
\begin{aligned}
& \mathcal{F}_{rec}(\overline{v}^{\,k}) \\
=\ & \alpha(\overline{q}^{\,k} \mapsto \emptyset)(\overline{v}^{\,k}) \\
& \text{since there are no recursive occurrences} \\
& \text{does not depend on the argument} \\
=\ & \alpha(G)(\overline{v}^{\,k}) \\
& \text{since } G \text{ is a fixed point of } \alpha \\
=\ & G(\overline{v}^{\,k})
\end{aligned}
$$

if $n > 0$ then

$$
\begin{aligned}
& \mathcal{F}_{rec}(\overline{v}^{\,k}) \\
=\ & (\alpha^{(2+n)}(\overline{q}^{\,k} \mapsto \emptyset))(\overline{v}^{\,k}) \\
& \text{By I. H.} \\
=\ & (\alpha(G))(\overline{v}^{\,k}) \\
& \text{since } G \text{ is a fixed point of } \alpha \\
=\ & G(\overline{v}^{\,k})
\end{aligned}
$$

Note that the proof does not rely on the fact that α is monotone but only on the uniqueness of the fixed point.

□

Proof of Theorem 5.2 (ii). By induction on the structure of $\Gamma \vdash \tau\colon\kappa$.

- Ty-Nat and Ty-Bool

$$\llbracket \Gamma \vdash \mathcal{B}{:}\kappa \rrbracket\,(\gamma) = \left\{ \langle i_{\mathcal{B}}\,(v)\,,\, i_{\mathcal{B}}\,(v)\rangle \,\middle|\, v \in \mathcal{B} \right\} \cup \{\langle \bot,\, \bot\rangle\} \in \llbracket \Gamma \vdash \mathsf{Ty{:}Kd} \rrbracket\,(\gamma) = \mathcal{CUPER}\,(\mathcal{D})$$

  The diagonal relation over a subset of $\mathcal{D}$ is a PER. Since $\langle \bot,\, \bot\rangle$ is in the set, it is clearly pointed, and since $\mathcal{B}$ is a flat domain, completeness and uniformity follow.
- Ty-Fun

$$\llbracket \Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) = \left\{ \langle f,\, g\rangle \,\middle|\, \forall\, \langle v,\, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)\,.\ \langle f \cdot v,\, g \cdot w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) \right\} \in \llbracket \Gamma \vdash \mathsf{Ty{:}Kd} \rrbracket\,(\gamma) = \mathcal{CUPER}\,(\mathcal{D})$$

  Since CUPERs are closed under intersection we just show that fixed $\langle v,\, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$ the set is a CUPER.
  (1) **Symmetry**: Fix $\langle f,\, g\rangle$ in the set, we have $\langle g \cdot v,\, f \cdot w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$ by I. H. and symmetry of CUPERs thus $\langle g,\, f\rangle$ is in the set.
  (2) **Transitivity**: Fix $\langle f,\, g\rangle$ and $\langle g,\, h\rangle$ in the set by I. H. and transitivity of CUPERs we have $\langle f \cdot v,\, h \cdot w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$ thus $\langle f,\, h\rangle$ is in the set.
  (3) **Pointedness**: By I. Hs. $\langle \bot,\, \bot\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$, and $\langle \bot,\, \bot\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$. Since $\bot \cdot v = \bot$ we have $\langle \bot,\, \bot\rangle$ is in the set.
  (4) **Completeness**: Let $\{\langle f_i,\, g_i\rangle\}_{i \in I}$ be a directed subset of the set. Fix $\langle v,\, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket(\gamma)$. By the definition of the set, for every $i \in I$ by I. H. $\langle f_i \cdot v,\, g_i \cdot w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket(\gamma)$, since application is continuous, $\langle (\bigsqcup_{i \in I} f_i) \cdot v,\, (\bigsqcup_{i \in I} g_i) \cdot w\rangle = \bigsqcup_{i \in I} \langle f_i \cdot v,\, g_i \cdot w\rangle$. By I. H. $\llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$ is complete, hence the lub pair is in the relation, so $\langle \bigsqcup_i f_i,\, \bigsqcup_i g_i\rangle$ is in the set.
  (5) **Uniformity**: Let $\langle f,\, g\rangle$ be in the set, and fix an arbitrary $\langle v,\, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$. By I. H. the domain PER is uniform, so $\langle \pi_n(v),\, \pi_n(w)\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$, hence $\langle f \cdot \pi_n(v),\, g \cdot \pi_n(w)\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$. By I. H. the codomain PER is uniform as well, so $\langle \pi_n(f \cdot \pi_n(v)),\, \pi_n(g \cdot \pi_n(w))\rangle$ is in it. The projections of the bilimit satisfy $\pi_{n+1}(f) \cdot v = \pi_n(f \cdot \pi_n(v))$, hence $\langle \pi_{n+1}(f) \cdot v,\, \pi_{n+1}(g) \cdot w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)$. Since $\langle v,\, w\rangle$ was arbitrary, $\langle \pi_{n+1}(f),\, \pi_{n+1}(g)\rangle$ is in the set; for $n = 0$ we have $\pi_0(f) = \bot = \pi_0(g)$ and the claim follows from pointedness.
- Ty-All

$$\llbracket \Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) = \left\{ \langle v,\, w\rangle \,\middle|\, \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)\,.\ \langle v,\, w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{K}}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\, P\rangle) \right\} \in \llbracket \Gamma \vdash \mathsf{Ty{:}Kd} \rrbracket\,(\gamma) = \mathcal{CUPER}\,(\mathcal{D})$$

  Since CUPERs are closed under intersection we just show that fixed $P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)$ the set is a CUPER, by I. H. each $\llbracket \Gamma,\, x{:}_{\mathsf{K}}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\, P\rangle)$ is a CUPER.
- Ty-Var

$$\llbracket \Gamma \vdash x{:}\kappa \rrbracket\,(\gamma) = \gamma\,(x) \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)$$

  by assumption.
- Ty-Ref

$$\llbracket \Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty} \rrbracket\,(\gamma) = \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \in \llbracket \Gamma \vdash \mathsf{Ty{:}Kd} \rrbracket\,(\gamma) = \mathcal{CUPER}\,(\mathcal{D})$$

  By I. H. $\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$ is a CUPER.
- Ty-App

$$\llbracket \Gamma \vdash \sigma\ t{:}\kappa\,[t/x] \rrbracket\,(\gamma) = \llbracket \Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa \rrbracket\,(\gamma) \in \llbracket \Gamma \vdash \kappa\,[t/x]\,{:}\mathsf{Kd} \rrbracket\,(\gamma)$$

By I. H. $[\![\Gamma \vdash \sigma : (x{:}\tau) \to \kappa]\!]\,(\gamma) \in [\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\gamma) = [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma) = [\![\Gamma \vdash \kappa\,[t/x]{:}\mathsf{Kd}]\!](\gamma) = \mathcal{CUPER}\,(\mathcal{D})$.

- Ty-IndPos and Ty-IndStrat

$$\left[\!\left[\Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C_i : (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} : \overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma)$$
$$= \mathsf{fix}_X \underbrace{\bigcup_i \left\{ \left\langle \langle C_i,\, \overline{v}^{\,j} \rangle,\, \langle C_i,\, \overline{w}^{\,j} \rangle \right\rangle \middle| \left\langle v_j,\, w_j \right\rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle) \right\} \cup \{\langle \bot,\, \bot \rangle\}}_{\alpha}$$
$$\in [\![\Gamma \vdash \overline{\tau}^{\,k} \to \mathsf{Ty}{:}\mathsf{Kd}]\!]\,(\gamma) = \mathcal{CUPER}\,(\mathcal{D})$$

First, we show that if $X \in \mathcal{CUPER}\,(\mathcal{D})$ then $\alpha \in \mathcal{CUPER}\,(\mathcal{D})$, since α is the union of disjoint C_i we just need to show that each C_i is a CUPER.

(1) **Symmetry**: Fix $\langle \langle C_i,\, \overline{v}^{\,j} \rangle,\, \langle C_i,\, \overline{w}^{\,j} \rangle \rangle$ in the set, then $\langle v_j,\, w_j \rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$ by definition. By I. H., symmetry of this PER gives $\langle w_j,\, v_j \rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$, hence $\langle \langle C_i,\, \overline{w}^{\,j} \rangle,\, \langle C_i,\, \overline{v}^{\,j} \rangle \rangle$ in the set.

(2) **Transitivity**: Fix $\langle \langle C_i,\, \overline{v}^{\,j} \rangle,\, \langle C_i,\, \overline{w}^{\,j} \rangle \rangle$ in the set and $\langle \langle C_i,\, \overline{w}^{\,j} \rangle,\, \langle C_i,\, \overline{q}^{\,j} \rangle \rangle$ in the set, then $\langle v_j,\, w_j \rangle$ and $\langle w_j,\, q_j \rangle$ are in $[\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$. By I. H., transitivity gives $\langle v_j,\, q_j \rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$, hence $\langle \langle C_i,\, \overline{v}^{\,j} \rangle,\, \langle C_i,\, \overline{q}^{\,j} \rangle \rangle$ is in the set.

(3) **Pointedness**: $\langle \bot,\, \bot \rangle$ in the set by definition.

(4) **Completeness**: Let $\{\langle a_k,\, b_k \rangle\}_{k \in I}$ be directed, then for each k, a_k, b_k are built from v_j^k, w_j^k with $\langle v_j^k,\, w_j^k \rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$ by I. H.. Since this PER is complete and pairing is continuous, we get closure under lubs, hence $\langle \bigsqcup_k a_k,\, \bigsqcup_k b_k \rangle$ is in the set.

(5) **Uniformity**: Let $\langle a,\, b \rangle$ in the set, then $\langle v_j,\, w_j \rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$ by definition. By I. H. and uniformity of this PER, $\langle \pi_n(v_j),\, \pi_n(w_j) \rangle \in [\![\Gamma,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, X \rangle)$. By compatibility of constructors with projection, this gives $\langle \pi_n(a),\, \pi_n(b) \rangle$ in the set.

By Theorem 7.1, each field interpretation is non-expansive in X, by Lemma D.1, agreement at level n of the fields yields agreement at level $n+1$ of the corresponding constructor summand. Hence α is contractive and by Lemma 3.4 the fixed point exists and is unique.

□

Proof of Theorem 5.3 (i). By induction on the structure of $\Gamma \vdash \tau{:}\kappa$.

- Ty-Nat and Ty-Bool

$$[\![\Gamma \vdash \mathcal{B}{:}\kappa]\!]\,(\gamma) = \left\{ \langle i_{\mathcal{B}}\,(v),\, i_{\mathcal{B}}\,(v) \rangle \middle| v \in \mathcal{B} \right\} = [\![\Gamma \vdash \mathcal{B}{:}\kappa]\!]\,(\delta)$$

- Ty-Fun

$$[\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!]\,(\gamma)$$
$$= \left\{ \langle f,\, g \rangle \middle| \begin{array}{l} \forall\, \langle v,\, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)\,. \quad \langle f \cdot v,\, g \cdot w \rangle \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, v \rangle) \\ \forall \langle v_\bot,\, w_\bot \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))\,.\, \langle f \cdot v_\bot,\, g \cdot w_\bot \rangle \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma)) \end{array} \right\}$$

By I. H. and Theorem 5.3 (ii)

$$= \left\{ \langle f,\, g \rangle \middle| \begin{array}{l} \forall\, \langle v,\, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta)\,. \quad \langle f \cdot v,\, g \cdot w \rangle \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta,\, v \rangle) \\ \forall \langle v_\bot,\, w_\bot \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\delta))\,.\, \langle f \cdot v_\bot,\, g \cdot w_\bot \rangle \in [\![\Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\delta)) \end{array} \right\}$$
$$= [\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!]\,(\delta)$$

- Ty-All

$$
\begin{aligned}
&[\![\Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)\,,\ \forall P_\perp \in \lfloor\!\lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma)) \\ \pi_{\Gamma;\,\kappa}\,(\gamma,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Gamma,\, x{:}_\kappa \kappa \vdash \tau{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, \langle P,\, P_\perp\rangle\rangle) \end{array} \right\} \\
&\quad \text{By I. H., Theorem 5.1 (i) and Theorem 5.1 (ii)} \\
&= \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta)\,,\ \forall P_\perp \in \lfloor\!\lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\!\rfloor\,(\iota_\Gamma\,(\delta)) \\ \pi_{\Gamma;\,\kappa}\,(\delta,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Gamma,\, x{:}_\kappa \kappa \vdash \tau{:}\mathsf{Ty}]\!]\,(\langle \delta,\, \langle P,\, P_\perp\rangle\rangle) \end{array} \right\} \\
&= [\![\Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty}]\!]\,(\delta)
\end{aligned}
$$

- Ty-Var

$$[\![\Gamma \vdash x{:}\kappa]\!]\,(\gamma) = \gamma\,(x)_1 = \delta\,(x)_1 = [\![\Gamma \vdash x{:}\kappa]\!]\,(\delta)$$

- Ty-Ref

$$
\begin{aligned}
&[\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle v,\, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \{\!\{\Gamma,\, x{:}_\tau \tau \vdash \phi\ \mathsf{prp}\}\!\}\,(\langle \gamma,\, v\rangle) = tt \right\} \\
&\quad \text{By I. H. and Theorem 6.1} \\
&= \left\{ \langle v,\, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta) \;\middle|\; \{\!\{\Gamma,\, x{:}_\tau \tau \vdash \phi\ \mathsf{prp}\}\!\}\,(\langle \delta,\, v\rangle) = tt \right\} \\
&= [\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\,(\delta)
\end{aligned}
$$

- Ty-App

$$
\begin{aligned}
&[\![\Gamma \vdash \sigma\ t{:}\kappa\ [t/x]]\!]\,(\gamma) \\
&= [\![\Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa]\!]\,(\gamma)\,(\!(t)\!)\,(\rho_\Gamma\,(\gamma))) \\
&\quad \text{By I. H.} \\
&= [\![\Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa]\!]\,(\delta)\,(\!(t)\!)\,(\rho_\Gamma\,(\gamma))) \\
&\quad \text{By Theorem 5.6 (i) } \langle (\!(t)\!)\,(\rho_\Gamma\,(\gamma))\,,\ (\!(t)\!)\,(\rho_\Gamma\,(\delta))\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\
&\quad \text{By Theorem 5.4 } \langle (\!(t)\!)\,(\rho_\Gamma\,(\gamma))\,,\ (\!(t)\!)\,(\rho_\Gamma\,(\delta))\rangle \in \lfloor\!\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma)) \\
&\quad \text{By } [\![\Gamma \vdash (x{:}\tau) \to \kappa{:}\mathsf{Kd}]\!]\,(\gamma) \text{ respecting} \\
&\quad \lfloor\!\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma)) \cap \mathsf{dom}\,([\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma))^2 \\
&= [\![\Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa]\!]\,(\delta)\,(\!(t)\!)\,(\rho_\Gamma\,(\delta))) \\
&= [\![\Gamma \vdash \sigma\ t{:}\kappa\ [t/x]]\!]\,(\delta)
\end{aligned}
$$

- Ty-IndPos and Ty-IndStrat

$$
\begin{aligned}
&\left[\!\!\left[\Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\!\right]\,(\gamma) \\
&= \mathsf{lfp}_X \left(\overline{q}^{\,k} \mapsto \bigcup_i \left\{ \langle \langle C_i,\, \overline{v}^{\,j}\rangle,\, \langle C_i,\, \overline{w}^{\,j}\rangle\rangle \;\middle|\; \begin{array}{c} \langle v_j,\, w_j\rangle \in [\![\Gamma,\, K{:}_\kappa \overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_\tau \sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle\langle \gamma,\, \langle X,\, [\![\Gamma \vdash \mu K^{\langle \nu \rangle} \{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \}\, {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))\rangle\rangle,\, \overline{v}^{\,j-1}\rangle\rangle) \\ \langle q_k,\, (\!(t_k)\!)\,(\langle \rho_\Gamma\,(\gamma),\, \overline{v}^{\,j}\rangle)\rangle \in \lfloor\!\lfloor \Gamma,\, \overline{x{:}_\tau \tau} \vdash \tau_k{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma)) \end{array} \right\} \right) \\
&\quad \text{By I. H., Theorem 5.1 (ii) , and Theorem 5.6 (i)} \\
&= \mathsf{lfp}_X \left(\overline{q}^{\,k} \mapsto \bigcup_i \left\{ \langle \langle C_i,\, \overline{v}^{\,j}\rangle,\, \langle C_i,\, \overline{w}^{\,j}\rangle\rangle \;\middle|\; \begin{array}{c} \langle v_j,\, w_j\rangle \in [\![\Gamma,\, K{:}_\kappa \overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_\tau \sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle\langle \delta,\, \langle X,\, [\![\Gamma \vdash \mu K^{\langle \nu \rangle} \{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \}\, {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Gamma\,(\delta))\rangle\rangle,\, \overline{v}^{\,j-1}\rangle\rangle) \\ \langle q_k,\, (\!(t_k)\!)\,(\langle \rho_\Gamma\,(\delta),\, \overline{v}^{\,j}\rangle)\rangle \in \lfloor\!\lfloor \Gamma,\, \overline{x{:}_\tau \tau} \vdash \tau_k{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\delta)) \end{array} \right\} \right) \\
&= \left[\!\!\left[\Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\!\right]\,(\delta)
\end{aligned}
$$

□

Proof of Theorem 5.3 (ii). By induction on the structure of $\Gamma \vdash \tau{:}\kappa$.

- Ty-Nat and Ty-Bool

$$\llbracket \Gamma \vdash \mathcal{B}{:}\kappa \rrbracket\,(\gamma) = \left\{ \langle i_{\mathcal{B}}\,(v)\,,\ i_{\mathcal{B}}\,(v)\rangle \,\middle|\, v \in \mathcal{B} \right\} \cup \{\langle \bot,\ \bot\rangle\} = \llbracket \Gamma \vdash \mathcal{B}{:}\kappa \rrbracket\,(\delta)$$

- Ty-Fun

$$\begin{aligned}
&\llbracket \Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
&= \left\{ \langle f,\ g\rangle \,\middle|\, \forall\, \langle v,\ w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)\,.\ \langle f\cdot v,\ g\cdot w\rangle \in \llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) \right\} \\
&\quad \text{By I. H.} \\
&= \left\{ \langle f,\ g\rangle \,\middle|\, \forall\, \langle v,\ w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\delta)\,.\ \langle f\cdot v,\ g\cdot w\rangle \in \llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\delta) \right\} \\
&= \llbracket \Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\delta)
\end{aligned}$$

- Ty-All

$$\begin{aligned}
&\llbracket \Gamma \vdash \forall x{:}\kappa.\ \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
&= \left\{ \langle v,\ w\rangle \,\middle|\, \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)\,.\ \langle v,\ w\rangle \in \llbracket \Gamma,\ x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\ P\rangle) \right\} \\
&\quad \text{By I. H. and Theorem 5.1 (ii)} \\
&= \left\{ \langle v,\ w\rangle \,\middle|\, \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\delta)\,.\ \langle v,\ w\rangle \in \llbracket \Gamma,\ x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \delta,\ P\rangle) \right\} \\
&= \llbracket \Gamma \vdash \forall x{:}\kappa.\ \tau{:}\mathsf{Ty} \rrbracket\,(\delta)
\end{aligned}$$

- Ty-Var

$$\llbracket \Gamma \vdash x{:}\kappa \rrbracket\,(\gamma) = \gamma\,(x) = \delta\,(x) = \llbracket \Gamma \vdash x{:}\kappa \rrbracket\,(\delta)$$

- Ty-Ref

$$\begin{aligned}
&\llbracket \Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
&= \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
&\quad \text{By I. H.} \\
&= \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\delta) \\
&= \llbracket \Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty} \rrbracket\,(\delta)
\end{aligned}$$

- Ty-App

$$\begin{aligned}
&\llbracket \Gamma \vdash \sigma\ t{:}\kappa\ [t/x] \rrbracket\,(\gamma) \\
&= \llbracket \Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa \rrbracket\,(\gamma) \\
&\quad \text{By Theorem 5.1 (ii)} \\
&= \llbracket \Gamma \vdash \sigma{:}\,(x{:}\tau) \to \kappa \rrbracket\,(\delta) \\
&= \llbracket \Gamma \vdash \sigma\ t{:}\kappa\ [t/x] \rrbracket\,(\delta)
\end{aligned}$$

- Ty-IndPos and Ty-IndStrat

$$\begin{aligned}
&\left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K\cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\,(\gamma) \\
&= \mathsf{fix}_X \left(\bigcup_i \left\{ \langle \langle C_i,\ \overline{v}^{\,j}\rangle,\ \langle C_i,\ \overline{w}^{\,j}\rangle\rangle \,\middle|\, \langle v_j,\ w_j\rangle \in \left\llbracket \Gamma,\ K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\ \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \right\rrbracket\,(\langle \gamma,\ X\rangle) \right\} \cup \{\langle \bot,\ \bot\rangle\} \right) \\
&\quad \text{By I. H.} \\
&= \mathsf{fix}_X \left(\bigcup_i \left\{ \langle \langle C_i,\ \overline{v}^{\,j}\rangle,\ \langle C_i,\ \overline{w}^{\,j}\rangle\rangle \,\middle|\, \langle v_j,\ w_j\rangle \in \left\llbracket \Gamma,\ K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\ \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \right\rrbracket\,(\langle \delta,\ X\rangle) \right\} \cup \{\langle \bot,\ \bot\rangle\} \right)
\end{aligned}$$

$$= \left[\!\!\left[\Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C \colon (\overline{y{:}\sigma}^{\,j}) \to K \cdot \bar{t}^{\,k}}_{\,i} \right\} \colon \bar{\tau}^{\,k} \to \mathsf{Ty} \right]\!\!\right] (\delta)$$

□

Proof of Theorem 5.4. By induction on the structure of $\Gamma \vdash \tau{:}\kappa$.

- Ty-Nat and Ty-Bool

$$\begin{aligned}
&[\![\Gamma \vdash \mathcal{B}{:}\kappa]\!] (\gamma) \\
&= \left\{ \langle i_{\mathcal{B}}(v),\, i_{\mathcal{B}}(v) \rangle \,\middle|\, v \in \mathcal{B} \right\} \\
&\subseteq \left\{ \langle i_{\mathcal{B}}(v),\, i_{\mathcal{B}}(v) \rangle \,\middle|\, v \in \mathcal{B} \right\} \cup \{ \langle \bot, \bot \rangle \} \\
&= \lfloor \Gamma \vdash \mathcal{B}{:}\kappa \rfloor (\iota_\Gamma(\gamma))
\end{aligned}$$

- Ty-Fun

$$\begin{aligned}
&[\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!] (\gamma) \\
&= \left\{ \langle f, g \rangle \,\middle|\, \begin{array}{l} \forall\, \langle v, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!] (\gamma) \,.\quad \langle f \cdot v, g \cdot w \rangle \in [\![\Gamma, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!] (\langle \gamma, v \rangle) \\ \forall \langle v_\bot, w_\bot \rangle \in \lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \,.\, \langle f \cdot v_\bot, g \cdot w_\bot \rangle \in \lfloor \Gamma, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \end{array} \right\} \\
&\subseteq \left\{ \langle f, g \rangle \,\middle|\, \forall \langle v_\bot, w_\bot \rangle \in \lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \,.\, \langle f \cdot v_\bot, g \cdot w_\bot \rangle \in \lfloor \Gamma, x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \right\}
\end{aligned}$$

  Dropping the first conjunct

$$= \lfloor \Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma))$$

- Ty-All

$$\begin{aligned}
&[\![\Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty}]\!] (\gamma) \\
&= \left\{ \langle v, w \rangle \,\middle|\, \begin{array}{c} \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!] (\gamma),\ \forall P_\bot \in \lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor (\iota_\Gamma(\gamma)) \\ \pi_{\Gamma;\,\kappa}(\gamma, P) \subseteq P_\bot \Rightarrow \langle v, w \rangle \in [\![\Gamma, x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty}]\!] (\langle \gamma, \langle P, P_\bot \rangle \rangle) \end{array} \right\}
\end{aligned}$$

  Instantiating at the constant empty-relation element $P_\emptyset \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!] (\gamma)$, for which $\pi_{\Gamma;\,\kappa}(\gamma, P_\emptyset) = \emptyset \subseteq P_\bot$ holds for every $P_\bot$, and applying I. H.

$$\begin{aligned}
&\subseteq \left\{ \langle v, w \rangle \,\middle|\, \forall P_\bot \in \lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor (\iota_\Gamma(\gamma)) \,.\, \langle v, w \rangle \in \lfloor \Gamma, x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty} \rfloor (\langle \iota_\Gamma(\gamma), P_\bot \rangle) \right\} \\
&= \lfloor \Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma))
\end{aligned}$$

- Ty-Var

$$\begin{aligned}
&\pi_{\Gamma;\,\kappa}(\gamma,\ [\![\Gamma \vdash x{:}\kappa]\!] (\gamma)) \\
&= \pi_{\Gamma;\,\kappa}(\gamma,\ \gamma(x)_1) \\
&\quad \text{By assumption} \\
&\subseteq \gamma(x)_2 \\
&= \iota_\Gamma(\gamma)(x) \\
&= \lfloor \Gamma \vdash x{:}\kappa \rfloor (\iota_\Gamma(\gamma))
\end{aligned}$$

- Ty-Ref

$$\begin{aligned}
&[\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!] (\gamma) \\
&= \left\{ \langle v, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!] (\gamma) \,\middle|\, \{\!\{\Gamma, x{:}_{\mathsf{T}}\tau \vdash \phi\ \mathsf{prp}\}\!\} (\langle \gamma, v \rangle) = tt \right\} \\
&\subseteq [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!] (\gamma) \\
&\quad \text{By I. H.} \\
&\subseteq \lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma))
\end{aligned}$$

$$= \llbracket \Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma))$$

- Ty-App

$$\begin{aligned}
&\pi_{\Gamma;\ \kappa[t/x]}\ (\gamma,\ \llbracket \Gamma \vdash \sigma\ t{:}\kappa\ [t/x] \rrbracket\ (\gamma)) \\
&= \pi_{\Gamma;\ \kappa[t/x]}\ (\gamma,\ \llbracket \Gamma \vdash \sigma{:}\ (x{:}\tau) \to \kappa \rrbracket\ (\gamma)\ (\llparenthesis t \rrparenthesis\ (\rho_\Gamma\ (\gamma)))) \\
&\subseteq \pi_{\Gamma;\ (x:\tau)\to\kappa}\ (\gamma,\ \llbracket \Gamma \vdash \sigma{:}\ (x{:}\tau) \to \kappa \rrbracket\ (\gamma)) \\
&\quad \text{By I. H.} \\
&\subseteq \llbracket \Gamma \vdash \sigma{:}\ (x{:}\tau) \to \kappa \rrbracket\ (\iota_\Gamma\ (\gamma)) \\
&= \llbracket \Gamma \vdash \sigma\ t{:}\kappa\ [t/x] \rrbracket\ (\iota_\Gamma\ (\gamma))
\end{aligned}$$

- Ty-IndPos and Ty-IndStrat We note the definitions

$$\left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\gamma)$$

$$= \mathsf{lfp}_X \left(\underbrace{\overline{q}^{\,k} \mapsto \bigcup_i \left\{ \langle \langle C_i,\ \overline{v}^{\,j} \rangle,\ \langle C_i,\ \overline{w}^{\,j} \rangle \rangle \,\middle|\, \substack{\langle v_j,\ w_j \rangle \in \llbracket \Gamma,\ K{:}_k \overline{\tau}^{\,k} \to \mathsf{Ty},\ \overline{x{:}_\tau \sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \rrbracket\ (\langle \langle \gamma,\ \langle X,\ \llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma)) \rangle \rangle,\ \overline{v}^{\,j-1} \rangle \rangle) \\ \langle q_k,\ \llparenthesis t_k \rrparenthesis\ (\langle \rho_\Gamma\ (\gamma),\ \overline{v}^{\,j} \rangle) \rangle \in \llbracket \Gamma,\ \overline{x{:}_\tau \tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma))}} \right\} }_{\alpha} \right)$$

$$\left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma))$$

$$= \mathsf{fix}_X \left(\bigcup_i \left\{ \langle \langle C_i,\ \overline{v}^{\,j} \rangle,\ \langle C_i,\ \overline{w}^{\,j} \rangle \rangle \,\middle|\, \langle v_j,\ w_j \rangle \in \left\llbracket \Gamma,\ K{:}_k \overline{\tau}^{\,k} \to \mathsf{Ty},\ \overline{x{:}_\tau \sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \right\rrbracket\ (\langle \gamma,\ X \rangle) \right\} \cup \{\langle \bot,\ \bot \rangle\} \right)$$

First we observe that if

$$\pi_{\Gamma;\ \overline{\tau}^{\,k} \to \mathsf{Ty}}\ (\gamma,\ X) \subseteq \left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma))$$

then by I. Hs..

$$\pi_{\Gamma;\ \overline{\tau}^{\,k} \to \mathsf{Ty}}\ (\gamma,\ \alpha(X)) \subseteq \llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma))$$

Now we proceed by cases on the formation rule:

  - Ty-IndPos: Since α is a monotone operator we can apply the pre-fixed point induction principle, by definition:

$$\begin{aligned}
\pi_{\Gamma;\ \overline{\tau}^{\,k} \to \mathsf{Ty}} \left(\gamma,\ \overline{q}^{\,k} \mapsto \left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma)) \right) \\
\subseteq \left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma))
\end{aligned}$$

    From the previous observation,

$$\begin{aligned}
\alpha(\overline{q}^{\,k} \mapsto \left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma))) \\
\subseteq \left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma))
\end{aligned}$$

    Since the least fixed point is bounded above by any pre-fixed point, it follows that

$$\pi_{\Gamma;\ \overline{\tau}^{\,k} \to \mathsf{Ty}}\ (\gamma,\ \mathsf{lfp}_X(\alpha)) \subseteq \left\llbracket \Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\ (\iota_\Gamma\ (\gamma))$$

  - Ty-IndStrat: Instead here since we lack monotonicity, we will show it by subdividing the construction by the stratification index

$$P(n) \triangleq \overline{q}^{\,k} \mapsto \begin{cases} \llbracket \Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \rrbracket\ (\gamma) \cdot \overline{q}^{\,k} & \text{if } n > r_{\mathbb{N}}\ (\llparenthesis m \rrparenthesis\ (\rho_\Gamma\ (\gamma)) \cdot \overline{q}^{\,k}) \\ \emptyset & \text{otherwise} \end{cases}$$

Clearly $\forall n \in \mathbb{N}.\ P(n) \leq [\![\Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma)$.
By induction on n that $\forall n \in \mathbb{N}.\ \pi_{\Gamma;\,\overline{\tau}^{\,k} \to \mathsf{Ty}}\,(\gamma,\, P(n)) \subseteq [\![\Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!](\iota_\Gamma\,(\gamma))$. Since each $P(n) = \alpha(P(n-1))$ for $n > 0$, the inductive case follows from the previous observation. The base case is trivial since $P(0) = \emptyset$.

□

Proof of Corollary 1. From Theorem 5.4 $\langle v, v' \rangle, \langle w, w' \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))$

(i) ($\Rightarrow$) By transitivity and symmetry $\langle v', w' \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))$.
(ii) ($\Leftarrow$) By transitivity and symmetry $\langle v, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\gamma))$.

□

Proof of Theorem 5.6 (i) . By induction on the structure of $\Gamma \vdash t{:}\tau$.

- Tm-Nat and Tm-Bool

$$\begin{aligned}
&\langle (\!|b|\!)\,(\rho_\Gamma\,(\gamma)),\ (\!|b|\!)\,(\rho_\Gamma\,(\delta)) \rangle \\
&= \langle i_{\mathcal{B}}\,(b),\ i_{\mathcal{B}}\,(b) \rangle \\
&\in \left\{ \langle i_{\mathcal{B}}\,(b),\ i_{\mathcal{B}}\,(b) \rangle \,\middle|\, b \in \mathcal{B} \right\} \\
&= [\![\Gamma \vdash \mathcal{B}{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}$$

- Tm-Var

$$\begin{aligned}
&\langle (\!|x|\!)\,(\rho_\Gamma\,(\gamma)),\ (\!|x|\!)\,(\rho_\Gamma\,(\delta)) \rangle \\
&= \langle \gamma(x),\ \delta(x) \rangle \\
&\quad \text{By assumption} \\
&\in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}$$

- Tm-Lam

$$\begin{aligned}
&\langle (\!|\lambda x.\, t|\!)\,(\rho_\Gamma\,(\gamma)),\ (\!|\lambda x.\, t|\!)\,(\rho_\Gamma\,(\delta)) \rangle \\
&= \langle i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \rho_\Gamma\,(\gamma),\, v \rangle)),\ i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \rho_\Gamma\,(\delta),\, v \rangle)) \rangle \\
&\quad \text{We need to show} \\
&\in [\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle f,\, g \rangle \,\middle|\, \begin{array}{l} \forall\, \langle v,\, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)\,. \quad \langle f \cdot v,\, g \cdot w \rangle \in [\![\Gamma,\, x{:}_{\mathsf{r}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, v \rangle)\ \text{(i)} \\ \forall \langle v_\perp,\, w_\perp \rangle \in \lfloor\!\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\,.\ \langle f \cdot v_\perp,\, g \cdot w_\perp \rangle \in \lfloor\!\lfloor \Gamma,\, x{:}_{\mathsf{r}}\tau \vdash \sigma{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\ \text{(ii)} \end{array} \right\}
\end{aligned}$$

It is enough to show that fixed arbitrary $\langle v, w \rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$ and $\langle v_\perp, w_\perp \rangle \in \lfloor\!\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))$ the conditions (i) and (ii) hold. We show them separately.

(i)

$$\begin{aligned}
&\langle i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \rho_\Gamma\,(\gamma),\, v \rangle)) \cdot v,\ i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \rho_\Gamma\,(\delta),\, v \rangle)) \cdot w \rangle \\
&= \langle (\!|t|\!)\,(\langle \rho_\Gamma\,(\gamma),\, v \rangle),\ (\!|t|\!)\,(\langle \rho_\Gamma\,(\delta),\, w \rangle) \rangle \\
&\quad \text{By I. H.} \\
&\in [\![\Gamma,\, x{:}_{\mathsf{r}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, v \rangle)
\end{aligned}$$

(ii)

$$\begin{aligned}
&\langle i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \rho_\Gamma\,(\gamma),\, v \rangle)) \cdot v_\perp,\ i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \rho_\Gamma\,(\delta),\, v \rangle)) \cdot w_\perp \rangle \\
&= \langle (\!|t|\!)\,(\langle \rho_\Gamma\,(\gamma),\, v_\perp \rangle),\ (\!|t|\!)\,(\langle \rho_\Gamma\,(\delta),\, w_\perp \rangle) \rangle \\
&\quad \text{By Theorem 5.6 (ii) and Lemma 5.5}
\end{aligned}$$

$$\in \llbracket \Gamma,\ x :_{\mathrm{T}} \tau \vdash \sigma : \mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma))$$

- Tm-App

$$\begin{aligned}
&\langle (\!| t \cdot u |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t \cdot u |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
=&\langle (\!| t |\!)\ (\rho_\Gamma\ (\gamma)) \cdot (\!| u |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t |\!)\ (\rho_\Gamma\ (\delta)) \cdot (\!| u |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
&\text{We need to show} \\
\in &\llbracket \Gamma \vdash \sigma\ [u/x] : \mathsf{Ty} \rrbracket\ (\gamma)
\end{aligned}$$

By I. H. applied to u and t we have, respectively,

$$\langle (\!| u |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| u |\!)\ (\rho_\Gamma\ (\delta)) \rangle \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\ (\gamma),\ \text{and}$$

$$\begin{aligned}
&\langle (\!| t |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
\in &\llbracket \Gamma \vdash (x : \tau) \to \sigma : \mathsf{Ty} \rrbracket\ (\gamma) \\
= &\left\{ \langle f,\ g \rangle \;\middle|\; \begin{array}{l} \forall\ \langle v,\ w \rangle\ \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\ (\gamma)\,.\qquad \langle f \cdot v,\ g \cdot w \rangle\ \in \llbracket \Gamma,\ x :_{\mathrm{T}} \tau \vdash \sigma : \mathsf{Ty} \rrbracket\ (\langle \gamma,\ v \rangle)\ \text{(i)} \\ \forall \langle v_\perp,\ w_\perp \rangle \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma))\,.\ \langle f \cdot v_\perp,\ g \cdot w_\perp \rangle \in \llbracket \Gamma,\ x :_{\mathrm{T}} \tau \vdash \sigma : \mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma))\text{(ii)} \end{array} \right\}.
\end{aligned}$$

Finally, we can show

$$\begin{aligned}
&\langle (\!| t |\!)\ (\rho_\Gamma\ (\gamma)) \cdot (\!| u |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t |\!)\ (\rho_\Gamma\ (\delta)) \cdot (\!| u |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
&\text{By (i)} \\
\in &\llbracket \Gamma,\ x :_{\mathrm{T}} \tau \vdash \sigma : \mathsf{Ty} \rrbracket\ (\langle \gamma,\ (\!| u |\!)\ (\rho_\Gamma\ (\gamma)) \rangle) \\
&\text{By Lemma C.5} \\
= &\llbracket \Gamma \vdash \sigma\ [u/x] : \mathsf{Ty} \rrbracket\ (\gamma)
\end{aligned}$$

- Tm-TyLam

$$\begin{aligned}
&\langle (\!| \Lambda x.\ t |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| \Lambda x.\ t |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
=&\langle (\!| t |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
&\text{We need to show} \\
\in &\llbracket \Gamma \vdash \forall x{:}\kappa.\ \tau : \mathsf{Ty} \rrbracket\ (\gamma) \\
= &\left\{ \langle v,\ w \rangle \;\middle|\; \begin{array}{c} \forall P \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\ (\gamma),\ \forall P_\perp \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\ (\iota_\Gamma\ (\gamma)) \\ \pi_{\Gamma;\,\kappa}\ (\gamma,\ P) \subseteq P_\perp \Rightarrow \langle v,\ w \rangle \in \llbracket \Gamma,\ x :_{\kappa} \kappa \vdash \tau : \mathsf{Ty} \rrbracket\ (\langle \gamma,\ \langle P,\ P_\perp \rangle \rangle) \end{array} \right\}
\end{aligned}$$

It is enough to show that for arbitrary fixed $P \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket\ (\gamma)$ and $P_\perp \in \llbracket \Gamma \vdash \kappa : \mathsf{Kd} \rrbracket (\iota_\Gamma\ (\gamma))$ with $\pi_{\Gamma;\,\kappa}\ (\gamma,\ P) \subseteq P_\perp$ the pair belongs to the body interpretation, which is exactly the I. H. for t in the extended context.

$$\begin{aligned}
&\langle (\!| t |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
&\text{By I. H.} \\
\in &\llbracket \Gamma,\ x :_{\kappa} \kappa \vdash \tau : \mathsf{Ty} \rrbracket\ (\langle \gamma,\ \langle P,\ P_\perp \rangle \rangle)
\end{aligned}$$

- Tm-TyApp

$$\begin{aligned}
&\langle (\!| t\ \tau |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t\ \tau |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
=&\langle (\!| t |\!)\ (\rho_\Gamma\ (\gamma)),\ (\!| t |\!)\ (\rho_\Gamma\ (\delta)) \rangle \\
&\text{We need to show} \\
\in &\llbracket \Gamma \vdash \sigma\ [\tau/x] : \mathsf{Ty} \rrbracket\ (\gamma)
\end{aligned}$$

By I. H. we have

$$\begin{aligned}
&\langle (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\,,\ (\!|t|\!)\,(\rho_\Gamma\,(\delta))\rangle \\
&\in [\![\Gamma \vdash \forall x{:}\kappa.\,\sigma{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)\,,\ \forall P_\perp \in \lfloor\!\lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma)) \\ \pi_{\Gamma;\,\kappa}\,(\gamma,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Gamma,\, x{:}_\kappa\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, \langle P,\, P_\perp\rangle\rangle) \end{array} \right\}.
\end{aligned}$$

The interpretations of the type argument τ form a valid index: by Theorem 5.2 (i) $[\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma) \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)$, by Theorem 5.2 (ii) $\lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma)) \in \lfloor\!\lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))$, and by Theorem 5.4 $\pi_{\Gamma;\,\kappa}\,(\gamma,\, [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma)) \subseteq \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))$. Instantiating the universal quantifier at $[\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma)$ and $\lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))$, whose implication premise holds, gives

$$\begin{aligned}
&\langle (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\,,\ (\!|t|\!)\,(\rho_\Gamma\,(\delta))\rangle \\
&\in [\![\Gamma,\, x{:}_\kappa\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, \langle [\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma)\,,\ \lfloor\!\lfloor \Gamma \vdash \tau{:}\kappa \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))\rangle\rangle) \\
&\quad \text{By Lemma C.7} \\
&= [\![\Gamma \vdash \sigma\,[\tau/x]\,{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}$$

- Tm-Refl By I. H. the pair lies in $[\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$. Write $v_t \triangleq (\!|t|\!)\,(\rho_\Gamma\,(\gamma))$ for its first component. Partial reflexivity gives $\langle v_t,\, v_t\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$, and by Theorem 5.4 $\langle v_t,\, v_t\rangle \in \lfloor\!\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\!\rfloor\,(\iota_\Gamma\,(\gamma))$; since t does not mention x, this is exactly $\{\!|\Gamma,\, x{:}_\mathsf{T}\tau \vdash x \equiv_\tau t\ \mathsf{prp}|\!\}\,(\langle \gamma,\, v_t\rangle) = \mathit{tt}$. Hence the pair, whose first component is v_t, satisfies the refinement predicate, so

$$\begin{aligned}
&\langle (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\,,\ (\!|t|\!)\,(\rho_\Gamma\,(\delta))\rangle \\
&\in \left\{ \langle v,\, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \{\!|\Gamma,\, x{:}_\mathsf{T}\tau \vdash x \equiv_\tau t\ \mathsf{prp}|\!\}\,(\langle \gamma,\, v\rangle) = \mathit{tt} \right\} \\
&= [\![\Gamma \vdash \{x{:}\tau \mid x \equiv_\tau t\}{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}$$

- Tm-Sub

$$\begin{aligned}
&\langle (\!|t|\!)\,(\rho_\Gamma\,(\gamma))\,,\ (\!|t|\!)\,(\rho_\Gamma\,(\delta))\rangle \\
&\quad \text{By I. H.} \\
&\in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\
&\quad \text{By Theorem 6.2 (i)} \\
&\subseteq [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}$$

- Tm-Intro Let $D \triangleq \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\}$. The constructor is interpreted by the constant function

$$(\!|C(i)_D|\!)\,(\rho_\Gamma\,(\gamma)) = i_{\to}\,(\overline{v}^{\,j} \mapsto \langle C_i,\, \overline{v}^{\,j}\rangle)\,,$$

so the two components of the required pair coincide. Since the type is a function type, it suffices to fix related arguments

$$\left\langle v_j,\, w_j \right\rangle \in \left[\!\!\left[\Gamma,\, \overline{x{:}_\mathsf{T}\sigma\,[D/K]}^{\,j-1} \vdash \sigma_j\,[D/K]\,{:}\mathsf{Ty}\right]\!\!\right]\,(\langle \gamma,\, \overline{v}^{\,j-1}\rangle)$$

and show that the application lands in the codomain,

$$\langle \langle C_i,\, \overline{v}^{\,j}\rangle,\, \langle C_i,\, \overline{w}^{\,j}\rangle\rangle \in [\![\Gamma,\, \overline{x{:}_\mathsf{T}\sigma\,[D/K]}^{\,j} \vdash D \cdot \overline{t}^{\,k}{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, \overline{v}^{\,j}\rangle)\,.$$

The interpretation of the function type also imposes the analogous condition on unrefined-related arguments; this is precisely Theorem 5.6 (ii) applied to $C(i)_D$, so we are left with the refined condition above. Let $\mathcal{F}$ be the operator defining the refined interpretation of D. As that interpretation is a fixed point of $\mathcal{F}$, the codomain above equals

$$\mathcal{F}([\![\Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma)) \cdot \overline{(\!|t|\!)\,(\langle \rho_\Gamma\,(\gamma)\,,\, \overline{v}^{\,j}\rangle)}^{\,k},$$

whose i-th disjunct is

$$\left\{ \langle \langle C_i, \overline{v}^{\,j} \rangle, \langle C_i, \overline{w}^{\,j} \rangle \rangle \;\middle|\; \begin{array}{c} \langle v_j, w_j \rangle \in \left[\!\left[\Gamma,\, K{:_{\mathsf{k}}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:_{\mathsf{T}}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \right]\!\right] \left(\langle \langle \gamma,\, \langle \left[\!\left[\Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma),\, \lfloor \Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \rangle \rangle, \overline{v}^{\,j-1} \rangle \right) \quad \text{(i)} \\ \langle q_k,\, (\!| t_k |\!) (\langle \rho_\Gamma(\gamma), \overline{v}^{\,j} \rangle) \rangle \in \lfloor \Gamma,\, \overline{x{:_{\mathsf{T}}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \quad \text{(ii)} \end{array} \right\}$$

The required pair lies in this disjunct. Condition (ii) holds: its two sides coincide, and the common index denotation lies in the interpretation of the corresponding index type, so partial reflexivity relates it to itself. For condition (i), each argument pair satisfies, by assumption,

$$\langle v_j,\, w_j \rangle$$

by assumption

$$\in \left[\!\left[\Gamma,\, \overline{x{:_{\mathsf{T}}}\sigma\,[D/K]}^{\,j-1} \vdash \sigma_j\,[D/K]{:}\mathsf{Ty} \right]\!\right] (\langle \gamma,\, \overline{v}^{\,j-1} \rangle)$$

By Lemma C.7

$$= \left[\!\left[\Gamma,\, K{:_{\mathsf{k}}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:_{\mathsf{T}}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \right]\!\right] (\langle \langle \gamma,\, \langle \left[\!\left[\Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma),\, \lfloor \Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \rangle \rangle, \overline{v}^{\,j-1} \rangle)$$

which is condition (i). Hence both conditions hold, the pair lies in the disjunct, and therefore in the codomain.

- Tm-Elim We use the following abbreviations:

$$D \triangleq \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\}$$

$$e(v,r) \triangleq \begin{cases} (\!| c_i |\!)\,(\rho_\Gamma(\gamma)) \cdot \overline{q}^{\,j} \cdot \overline{\mathsf{tmEl}_{K;r}\left(\sigma,\, \langle \rho_\Gamma(\gamma),\, \overline{q}^{\,j-1} \rangle,\, q_j\right)}^{\,j} & \text{if } v = \langle C_i,\, \overline{q}^{\,j} \rangle \\ \bot & \text{otherwise} \end{cases}$$

$$\begin{aligned} & \langle (\!| elim_D\,(\overline{c}^{\,i}) |\!)\,(\rho_\Gamma(\gamma)),\, (\!| elim_D\,(\overline{c}^{\,i}) |\!)\,(\rho_\Gamma(\delta)) \rangle \\ = & \left\langle i_{\to}\left(\mathsf{lfp}_r \overline{v}^{\,k},\, w \mapsto e(w,r)\right),\, i_{\to}\left(\mathsf{lfp}_{r'} \overline{v}^{\,k},\, w \mapsto e(w,r')\right) \right\rangle \\ \triangleq & \langle r,\, r' \rangle \end{aligned}$$

This pair exists and is an element of $\mathcal{D}$ as it is the least fixed point of continuous operators in $\mathcal{D}$. We name the two components r and r'.
What we need to show is that this pair lies in

$$\left[\!\left[\Gamma \vdash (\overline{x{:_{\mathsf{T}}}\tau}^{\,k}) \to (y{:}D \cdot \overline{x}^{\,k}) \to \rho{:}\mathsf{Ty} \right]\!\right] (\gamma)$$

We distinguish two cases based on whether the type is stratified or positive:

  - **Positive case:**
    We define the collection of all those elements of the domain for which the property holds.

$$P \triangleq \overline{q}^{\,k} \mapsto \left\{ \langle v,\, w \rangle \;\middle|\; \begin{array}{c} \langle v,\, w \rangle \in \left[\!\left[\Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma) \cdot \overline{q}^{\,k} \\ \langle e(v,r),\, e(w,r') \rangle \in \left[\!\left[\Gamma,\, \overline{x{:_{\mathsf{T}}}\sigma}^{\,k},\, z{:_{\mathsf{T}}}D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty} \right]\!\right] (\langle \langle \gamma,\, \overline{q}^{\,k} \rangle,\, v \rangle) \end{array} \right\}$$

    The goal is to show that $P = \left[\!\left[\Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma)$.
    By definition we already have $P \leq \left[\!\left[\Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma)$, we need to show the other inclusion; since $\left[\!\left[\Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \right]\!\right](\gamma)$ is defined as the least fixed point of $\mathcal{F}$, it is enough to show that $P \geq \mathcal{F}(P)$, i.e. that P is a pre-fixed point of $\mathcal{F}$.

$$\mathcal{F}(P)$$

$$= \overline{q}^{\,k} \mapsto \bigcup_i \left\{ \langle \langle C_i, \overline{v}^{\,j} \rangle, \langle C_i, \overline{w}^{\,j} \rangle \rangle \;\middle|\; \begin{array}{c} \langle v_j, w_j \rangle \in \left[\!\left[\Gamma,\, K{:_{\mathsf{k}}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:_{\mathsf{T}}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \right]\!\right] \left(\left\langle \left\langle \gamma,\, \left\langle P,\, \left[\!\left[\Gamma \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:} \overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right] (\iota_\Gamma(\gamma)) \right\rangle \right\rangle, \overline{v}^{\,j-1} \right\rangle \right) \\ \langle q_k,\, (\!| t_k |\!) (\langle \rho_\Gamma(\gamma), \overline{v}^{\,j} \rangle) \rangle \in \lfloor \Gamma,\, \overline{x{:_{\mathsf{T}}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty} \rfloor (\iota_\Gamma(\gamma)) \end{array} \right\}$$

Before continuing, note that by Definition 19 the order on type operators compares them pointwise, but only at indices in the refined domain $\mathrm{dom}(\llbracket \Gamma,\ \overline{x{:}_{\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty} \rrbracket\ (\langle \gamma,\ \overline{q}^{\,k-1}\rangle))$. Fixed arbitrary $q_k \in \llbracket \Gamma,\ \overline{x{:}_{\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty} \rrbracket\ (\langle \gamma,\ \overline{q}^{\,k-1}\rangle)$, we need to show that if $\langle\langle C_i,\ \overline{v}^{\,j}\rangle,\ \langle C_i,\ \overline{w}^{\,j}\rangle\rangle \in \mathcal{F}(P)\cdot\overline{q}^{\,k}$, then $\left\langle \left\langle C_i,\ \overline{v}^{\,j}\right\rangle,\ \left\langle C_i,\ \overline{w}^{\,j}\right\rangle \right\rangle \in P\cdot\overline{q}^{\,k}$.
To do this it is enough to show that

$$\begin{aligned}
&\langle e(\langle C_i,\ \overline{v}^{\,j}\rangle, r),\ e(\langle C_i,\ \overline{w}^{\,j}\rangle, r')\rangle \\
=&\left\langle \begin{array}{l} (\!|c_i|\!)\ (\rho_\Gamma\ (\gamma))\cdot\overline{q}^{\,j}\cdot \overline{\mathsf{tmEl}_{K;r}\left(\sigma,\ \left\langle \rho_\Gamma\ (\gamma),\ \overline{q}^{\,j-1}\right\rangle,\ q_j\right)}^{\,j}, \\ (\!|c_i|\!)\ (\rho_\Gamma\ (\delta))\cdot\overline{q}^{\,j}\cdot \overline{\mathsf{tmEl}_{K;r'}\left(\sigma,\ \left\langle \rho_\Gamma\ (\delta),\ \overline{q}^{\,j-1}\right\rangle,\ q_j\right)}^{\,j} \end{array} \right\rangle \\
\in&\ \llbracket \Gamma,\ \overline{x{:}_{\mathrm{T}}\sigma}^{\,k},\ z{:}_{\mathrm{T}} D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty} \rrbracket\ (\langle\langle \gamma,\ \overline{q}^{\,k}\rangle,\ \langle C_i,\ \overline{v}^{\,j}\rangle\rangle)
\end{aligned}$$

We apply Lemma 7.3 to each mention of tmEl with the approximant P, which is below $\llbracket \Gamma \vdash D{:}(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty} \rrbracket\ (\gamma)$, by $\mathcal{F}$ being monotonic it follows that $\mathcal{F}(P) \le \mathcal{F}(\llbracket \Gamma \vdash D{:}(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty} \rrbracket\ ($ $\llbracket \Gamma \vdash D{:}(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty} \rrbracket\ (\gamma)$, thus by I. H. on c_i we have that the pair is in the interpretation of the type, as required.
Since P is therefore a pre-fixed point of $\mathcal{F}$, the least fixed point $\llbracket \Gamma \vdash D{:}(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty} \rrbracket(\gamma)$ is below P, giving the equality.

– **Stratified case:** We prove it by induction on the metric. We define the n-th approximation of the type as:

$$P(n) \triangleq \overline{q}^{\,k} \mapsto \begin{cases} \llbracket \Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \rrbracket\ (\gamma)\cdot\overline{q}^{\,k} & \text{if } n > r_{\mathbb{N}}\ ((\!|m|\!)\ (\rho_\Gamma\ (\gamma))\cdot\overline{q}^{\,k}) \\ \emptyset & \text{otherwise} \end{cases}$$

Clearly $\forall n \in \mathbb{N}.\ P(n) \le \llbracket \Gamma \vdash D{:}\overline{\sigma}^{\,k} \to \mathsf{Ty} \rrbracket\ (\gamma)$. As in the positive case, by Definition 19 it suffices to range over refined-domain indices, where the motive interpretation is defined. We show that $\forall n \in \mathbb{N}$

$$\begin{aligned}
&\forall q_k \in \llbracket \Gamma,\ \overline{x{:}_{\mathrm{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty} \rrbracket\ (\langle \gamma,\ \overline{q}^{\,k-1}\rangle) \\
&\forall\ \langle v,\ w\rangle \in P(n)\cdot\overline{q}^{\,k}. \\
&\langle e(v, r),\ e(w, r')\rangle \in \llbracket \Gamma,\ \overline{x{:}_{\mathrm{T}}\sigma}^{\,k},\ z{:}_{\mathrm{T}} D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty} \rrbracket\ (\langle\langle \gamma,\ \overline{q}^{\,k}\rangle,\ v\rangle)
\end{aligned}$$

By induction over n:

* $n = 0$ is trivial, since $P(0)$ is empty.
* $n = m + 1$: Let $\langle v,\ w\rangle = \langle\langle C_i,\ \overline{v}^{\,j}\rangle,\ \langle C_i,\ \overline{w}^{\,j}\rangle\rangle$ We can show

$$\begin{aligned}
&\langle e(\langle C_i,\ \overline{v}^{\,j}\rangle, r),\ e(\langle C_i,\ \overline{w}^{\,j}\rangle, r')\rangle \\
=&\left\langle \begin{array}{l} (\!|c_i|\!)\ (\rho_\Gamma\ (\gamma))\cdot\overline{q}^{\,j}\cdot \overline{\mathsf{tmEl}_{K;r}\left(\sigma,\ \left\langle \rho_\Gamma\ (\gamma),\ \overline{q}^{\,j-1}\right\rangle,\ q_j\right)}^{\,j}, \\ (\!|c_i|\!)\ (\rho_\Gamma\ (\delta))\cdot\overline{q}^{\,j}\cdot \overline{\mathsf{tmEl}_{K;r'}\left(\sigma,\ \left\langle \rho_\Gamma\ (\delta),\ \overline{q}^{\,j-1}\right\rangle,\ q_j\right)}^{\,j} \end{array} \right\rangle \\
\in&\ \llbracket \Gamma,\ \overline{x{:}_{\mathrm{T}}\sigma}^{\,k},\ z{:}_{\mathrm{T}} D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty} \rrbracket\ (\langle\langle \gamma,\ \overline{q}^{\,k}\rangle,\ \langle C_i,\ \overline{v}^{\,j}\rangle\rangle)
\end{aligned}$$

We apply Lemma 7.3 to each mention of tmEl with the approximant $P(m)$, which is below $\llbracket \Gamma \vdash D{:}(\overline{x{:}\tau}^{\,k}) \to \mathsf{Ty} \rrbracket\ (\gamma)$ and by the I. H. they satisfy the required property, thus by I. H. on c_i we have that the pair is in the interpretation of the type, as required.

□

Proof of Lemma 5.5. By induction on the structure of $\vdash \Gamma\ \mathsf{ctx}$.

- Ctx-Empty

$$\begin{aligned}
&\langle \rho.\,(\star)\,,\ \rho.\,(\star)\rangle \\
=\ &\langle \star,\ \star\rangle \\
\in\ &[\![\vdash \cdot\ \mathsf{ctx}]\!]\ (\iota.\,(\star))
\end{aligned}$$

- Ctx-Ty

$$\begin{aligned}
&\left\langle \rho_{\Gamma, x:_{\mathrm{T}}\tau}\ (\gamma,\ v)\,,\ \rho_{\Gamma, x:_{\mathrm{T}}\tau}\ (\delta,\ w)\right\rangle \\
=\ &\langle \rho_\Gamma\ (\gamma)\,,\ v,\ \rho_\Gamma\ (\delta)\,,\ w\rangle \\
&\text{By I. H. and Theorem 5.4} \\
\in\ &[\![\vdash \Gamma\ \mathsf{ctx}]\!]\ (\iota_\Gamma\ (\gamma)) \times [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\iota_\Gamma\ (\gamma)) \\
=\ &[\![\vdash \Gamma,\ x:_{\mathrm{T}}\tau\ \mathsf{ctx}]\!]\ \left(\iota_{\Gamma, x:_{\mathrm{T}}\tau}\ (\gamma,\ v)\right)
\end{aligned}$$

- Ctx-Kd

$$\begin{aligned}
&\left\langle \rho_{\Gamma, x:_{\mathrm{K}}\kappa}\ (\gamma,\ P)\,,\ \rho_{\Gamma, x:_{\mathrm{K}}\kappa}\ (\delta,\ Q)\right\rangle \\
=\ &\langle \rho_\Gamma\ (\gamma)\,,\ \rho_\Gamma\ (\delta)\rangle \\
&\text{By I. H.} \\
\in\ &[\![\vdash \Gamma\ \mathsf{ctx}]\!]\ (\iota_\Gamma\ (\gamma)) \\
=\ &[\![\vdash \Gamma,\ x:_{\mathrm{K}}\kappa\ \mathsf{ctx}]\!]\ \left(\iota_{\Gamma, x:_{\mathrm{K}}\kappa}\ (\gamma,\ P)\right)
\end{aligned}$$

□

Proof of Theorem 5.6 (ii). By induction on the structure of $\Gamma \vdash t{:}\tau$.

- Tm-Nat and Tm-Bool

$$\begin{aligned}
&\langle (\!|b|\!)\ (\delta)\,,\ (\!|b|\!)\ (\epsilon)\rangle \\
=\ &\langle i_{\mathcal{B}}\ (b)\,,\ i_{\mathcal{B}}\ (b)\rangle \\
\in\ &\left\{ \langle i_{\mathcal{B}}\ (b)\,,\ i_{\mathcal{B}}\ (b)\rangle \,\middle|\, b \in \mathcal{B} \right\} \cup \{\langle \bot,\ \bot\rangle\} \\
=\ &[\![\Gamma \vdash \mathcal{B}{:}\mathsf{Ty}]\!]\ (\gamma)
\end{aligned}$$

- Tm-Var

$$\begin{aligned}
&\langle (\!|x|\!)\ (\delta)\,,\ (\!|x|\!)\ (\epsilon)\rangle \\
=\ &\langle \delta(x),\ \epsilon(x)\rangle \\
&\text{By assumption} \\
\in\ &[\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\gamma)
\end{aligned}$$

- Tm-Lam

$$\begin{aligned}
&\langle (\!|\lambda x.\,t|\!)\ (\delta)\,,\ (\!|\lambda x.\,t|\!)\ (\epsilon)\rangle \\
=\ &\langle i_{\to}\ (v \mapsto (\!|t|\!)\ (\langle \delta,\ v\rangle))\,,\ i_{\to}\ (v \mapsto (\!|t|\!)\ (\langle \epsilon,\ v\rangle))\rangle \\
&\text{We need to show} \\
\in\ &[\![\Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty}]\!]\ (\gamma) \\
=\ &\left\{ \langle f,\ g\rangle \,\middle|\, \forall\, \langle v,\ w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\ (\gamma)\,.\ \langle f \cdot v,\ g \cdot w\rangle \in [\![\Gamma,\ x:_{\mathrm{T}}\tau \vdash \sigma{:}\mathsf{Ty}]\!]\ (\gamma) \right\}
\end{aligned}$$

Fix arbitrary $\langle v, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$. Hence

$$\begin{aligned}
&\langle i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \delta, v\rangle)) \cdot v,\ i_{\to}\,(v \mapsto (\!|t|\!)\,(\langle \epsilon, v\rangle)) \cdot w\rangle \\
=\ &\langle (\!|t|\!)\,(\langle \delta, v\rangle)\,,\ (\!|t|\!)\,(\langle \epsilon, w\rangle)\rangle \\
&\text{By I. H.} \\
\in\ &\llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)
\end{aligned}$$

- Tm-App

$$\begin{aligned}
&\langle (\!|t \cdot u|\!)\,(\delta)\,,\ (\!|t \cdot u|\!)\,(\epsilon)\rangle \\
=\ &\langle (\!|t|\!)\,(\delta) \cdot (\!|u|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon) \cdot (\!|u|\!)\,(\epsilon)\rangle \\
&\text{We need to show} \\
\in\ &\llbracket \Gamma \vdash \sigma\,[u/x]\,{:}\mathsf{Ty} \rrbracket\,(\gamma)
\end{aligned}$$

By the I. H. applied to u and t we have, respectively,

$$\langle (\!|u|\!)\,(\delta)\,,\ (\!|u|\!)\,(\epsilon)\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)$$

and

$$\begin{aligned}
&\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle \\
\in\ &\llbracket \Gamma \vdash (x{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
=\ &\left\{ \langle f, g\rangle \,\middle|\, \forall\, \langle v, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)\,.\ \langle f \cdot v, g \cdot w\rangle \in \llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) \right\}.
\end{aligned}$$

Instantiating the universal quantifier with the argument pair gives

$$\begin{aligned}
&\langle (\!|t|\!)\,(\delta) \cdot (\!|u|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon) \cdot (\!|u|\!)\,(\epsilon)\rangle \\
\in\ &\llbracket \Gamma,\ x{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
&\text{By noticing that the unrefined semantics of types is independent of terms} \\
=\ &\llbracket \Gamma \vdash \sigma\,[u/x]\,{:}\mathsf{Ty} \rrbracket\,(\gamma)
\end{aligned}$$

- Tm-TyLam

$$\begin{aligned}
&\langle (\!|\Lambda x.\, t|\!)\,(\delta)\,,\ (\!|\Lambda x.\, t|\!)\,(\epsilon)\rangle \\
=\ &\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle \\
&\text{We need to show} \\
\in\ &\llbracket \Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \\
=\ &\left\{ \langle v, w\rangle \,\middle|\, \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)\,.\ \langle v, w\rangle \in \llbracket \Gamma,\ x{:}_{\mathsf{K}}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma, P\rangle) \right\}
\end{aligned}$$

For arbitrary fixed $P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma)$.

$$\begin{aligned}
&\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle \\
&\text{By I. H.} \\
\in\ &\llbracket \Gamma,\ x{:}_{\mathsf{K}}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma, P\rangle)
\end{aligned}$$

- Tm-TyApp

$$\begin{aligned}
&\langle (\!|t\ \tau|\!)\,(\delta)\,,\ (\!|t\ \tau|\!)\,(\epsilon)\rangle \\
=\ &\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle \\
&\text{We need to show} \\
\in\ &\llbracket \Gamma \vdash \sigma\,[\tau/x]\,{:}\mathsf{Ty} \rrbracket\,(\gamma)
\end{aligned}$$

By the I. H. we have

$$\begin{aligned}&\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle\\ \in\ &[\![\Gamma \vdash \forall x{:}\kappa.\,\sigma{:}\mathsf{Ty}]\!]\,(\gamma)\\ =\ &\left\{\,\langle v,\,w\rangle \,\middle|\, \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)\,.\ \langle v,\,w\rangle \in [\![\Gamma,\,x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle\gamma,\,P\rangle)\,.\,\right\}\end{aligned}$$

By Theorem 5.2 (ii) , the unrefined interpretation of the type argument is a valid index. Instantiating the universal quantifier at $[\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma)$ gives

$$\begin{aligned}&\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle\\ \in\ &[\![\Gamma,\,x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle\gamma,\,[\![\Gamma \vdash \tau{:}\kappa]\!]\,(\gamma)\rangle)\\ &\text{By Lemma C.8}\\ =\ &[\![\Gamma \vdash \sigma\,[\tau/x]\,{:}\mathsf{Ty}]\!]\,(\gamma)\end{aligned}$$

- Tm-Refl

$$\begin{aligned}&\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle\\ &\text{By the I. H.}\\ \in\ &[\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)\\ =\ &[\![\Gamma \vdash \{x{:}\tau \mid x \equiv_\tau t\}{:}\mathsf{Ty}]\!]\,(\gamma)\end{aligned}$$

- Tm-Sub

$$\begin{aligned}&\langle (\!|t|\!)\,(\delta)\,,\ (\!|t|\!)\,(\epsilon)\rangle\\ &\text{By the I. H.}\\ \in\ &[\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)\\ &\text{By Theorem 6.2 (ii)}\\ =\ &[\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma)\end{aligned}$$

- Tm-Intro Let $D \triangleq \mu K^{\langle\nu\rangle}\left\{\underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K\cdot\overline{t}^{\,k}}_{\,i}\right\}$. The constructor is interpreted by the constant function

$$(\!|C(i)_D|\!)\,(\delta) = i_{\to}\,(\overline{v}^{\,j} \mapsto \langle C_i,\,\overline{v}^{\,j}\rangle)\,,$$

so the two components of the required pair coincide. Since the type is a function type, it suffices to fix related arguments

$$\left\langle v_j,\,w_j\right\rangle \in \left[\!\!\left[\Gamma,\,\overline{x{:}_{\mathsf{T}}\sigma\,[D/K]}^{\,j-1} \vdash \sigma_j\,[D/K]\,{:}\mathsf{Ty}\right]\!\!\right]\,(\gamma)$$

and show that the application lands in the codomain,

$$\langle\langle C_i,\,\overline{v}^{\,j}\rangle,\,\langle C_i,\,\overline{w}^{\,j}\rangle\rangle \in [\![\Gamma,\,\overline{x{:}_{\mathsf{T}}\sigma\,[D/K]}^{\,j} \vdash D\cdot\overline{t}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma)\,.$$

Let $\mathcal{F}$ be the operator defining the unrefined interpretation of D. As that interpretation is a fixed point of $\mathcal{F}$, the codomain above equals

$$\mathcal{F}([\![\Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma))$$

whose i-th disjunct is

$$\left\{\,\left\langle\left\langle C_i,\,\overline{v}^{\,j}\right\rangle,\,\left\langle C_i,\,\overline{w}^{\,j}\right\rangle\right\rangle \,\middle|\, \left\langle v_j,\,w_j\right\rangle \in \left[\!\!\left[\Gamma,\,K{:}_{\mathsf{K}}\overline{\tau}^{\,k} \to \mathsf{Ty},\,\overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}\right]\!\!\right]\,(\langle\gamma,\,[\![\Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma)\rangle)\,\right\}$$

Each argument pair satisfies, by assumption,

$$\begin{aligned}&\left\langle v_j,\,w_j\right\rangle\\ &\text{by assumption}\end{aligned}$$

$$\in \left\llbracket \Gamma,\ \overline{x{:}_{\mathsf{T}}\sigma\ [D/K]}^{\,j-1} \vdash \sigma_j\ [D/K] : \mathsf{Ty} \right\rrbracket\ (\gamma)$$

By Lemma C.8

$$= \left\llbracket \Gamma,\ K{:}_{\mathsf{K}}\overline{\tau}^{\,k} \to \mathsf{Ty},\ \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j : \mathsf{Ty} \right\rrbracket\ (\langle \gamma,\ \llbracket \Gamma \vdash D{:}\overline{\tau}^{\,k} \to \mathsf{Ty} \rrbracket\ (\gamma) \rangle)$$

Hence the pair lies in the disjunct, and therefore in the codomain.

- Tm-Elim We use the following abbreviations:

$$D \triangleq \mu K^{\langle \nu \rangle} \left\{ \underbrace{C{:}\ (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{i} \right\}$$

$$e(v, r) \triangleq \begin{cases} \overline{\left(\!\left(c_i\right)\!\right)\ (\delta) \cdot \overline{q}^{\,j} \cdot \mathsf{tmEl}_{K;r}\left(\sigma,\ \left\langle \delta,\ \overline{q}^{\,j-1} \right\rangle,\ q_j\right)}^{\,j} & \text{if } v = \langle C_i,\ \overline{q}^{\,j} \rangle \\ \bot & \text{otherwise} \end{cases}$$

$$\begin{aligned} &\langle \left(\!\left(elim_D\ (\overline{c}^{\,i})\right)\!\right)\ (\delta),\ \left(\!\left(elim_D\ (\overline{c}^{\,i})\right)\!\right)\ (\epsilon) \rangle \\ &= \left\langle i_{\to}\left(\mathsf{lfp}_r \overline{v}^{\,k},\ w \mapsto e(w, r)\right),\ i_{\to}\left(\mathsf{lfp}_{r'} \overline{v}^{\,k},\ w \mapsto e(w, r')\right) \right\rangle \\ &\triangleq \langle r,\ r' \rangle \end{aligned}$$

This pair exists and is an element of $\mathcal{D}$ as it is the least fixed point of continuous operators in $\mathcal{D}$. We name the two components r and r'.
What we need to show is that this pair lies in

$$\llbracket \Gamma \vdash (\overline{x{:}_{\mathsf{T}}\tau}^{\,k}) \to (y{:}D \cdot \overline{x}^{\,k}) \to \rho{:}\mathsf{Ty} \rrbracket\ (\gamma)$$

By Lemma E.2, and the fact that all unrefined types are complete the least fixed point of the operator defining the eliminator semantics is contained in the unrefined interpretation of the type of the eliminator.

□

Proof of Corollary 2. Now suppose by contradiction that there exists a closed term t such that $\cdot \vdash t{:}\forall x{:}\mathsf{Ty}.\ x$, then:

$$\begin{aligned} &\langle \left(\!\left(t\right)\!\right)\ (\star),\ \left(\!\left(t\right)\!\right)\ (\star) \rangle \\ &\text{By Theorem 5.6 (i)} \\ &\in \llbracket \cdot \vdash \forall x{:}\mathsf{Ty}.\ x{:}\mathsf{Ty} \rrbracket\ (\star) \\ &= \left\{ \langle v,\ w \rangle \,\middle|\, \begin{array}{c} \forall P \in \llbracket \cdot \vdash \mathsf{Ty}{:}\mathsf{Kd} \rrbracket\ (\star),\ \forall P_\bot \in \lfloor \cdot \vdash \mathsf{Ty}{:}\mathsf{Kd} \rfloor\ (\star) \\ \pi_{\cdot;\,\mathsf{Ty}}\ (\star,\ P) \subseteq P_\bot \Rightarrow \langle v,\ w \rangle \in \llbracket \cdot,\ x{:}_{\mathsf{K}}\mathsf{Ty} \vdash x{:}\mathsf{Ty} \rrbracket\ (\langle \star,\ \langle P,\ P_\bot \rangle \rangle) \end{array} \right\} \\ &= \left\{ \langle v,\ w \rangle \,\middle|\, \begin{array}{c} \forall P \in \mathcal{PER}\ (\mathcal{D}),\ \forall P_\bot \in \mathcal{CUPER}\ (\mathcal{D}) \\ P \subseteq P_\bot \Rightarrow \langle v,\ w \rangle \in P \end{array} \right\} \end{aligned}$$

Instantiating at $P = \emptyset \in \mathcal{PER}\ (\mathcal{D})$ and $P_\bot = \{\langle \bot,\ \bot \rangle\} \in \mathcal{CUPER}\ (\mathcal{D})$, for which $\emptyset \subseteq \{\langle \bot,\ \bot \rangle\}$, any $\langle v,\ w \rangle$ in the set above would have to satisfy $\langle v,\ w \rangle \in \emptyset$, which is impossible. Hence $\llbracket \cdot \vdash \forall x{:}\mathsf{Ty}.\ x{:}\mathsf{Ty} \rrbracket(\star) = \emptyset$, contradicting $\langle \left(\!\left(t\right)\!\right)\ (\star),\ \left(\!\left(t\right)\!\right)\ (\star) \rangle$ being an element of it. □

Proof of Theorem 6.1. We proceed by induction on the structure of $\Gamma \vdash \phi\ \mathsf{prp}$.

- Prop-Eq

$$\begin{aligned} &\{\!\{\Gamma \vdash t \equiv_\tau u\ \mathsf{prp}\}\!\}\ (\gamma) \\ &= \begin{cases} tt & \text{if } \langle \left(\!\left(t\right)\!\right)\ (\rho_\Gamma\ (\gamma)),\ \left(\!\left(u\right)\!\right)\ (\rho_\Gamma\ (\gamma)) \rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\ (\iota_\Gamma\ (\gamma)) \\ ff & \text{otherwise} \end{cases} \\ &\text{By Theorem 5.6 (i) and Corollary 1} \end{aligned}$$

$$= \begin{cases} \mathit{tt} & \text{if } \langle (\!|t|\!)\,(\rho_\Gamma\,(\delta))\,,\ (\!|u|\!)\,(\rho_\Gamma\,(\delta))\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\iota_\Gamma\,(\delta)) \\ \mathit{ff} & \text{otherwise} \end{cases}$$
$$= \{\!|\Gamma \vdash t \equiv_\tau u\ \mathsf{prp}|\!\}\,(\delta)$$

- Prop-And

$$\{\!|\Gamma \vdash \phi \wedge \psi\ \mathsf{prp}|\!\}\,(\gamma)$$
$$= \begin{cases} \{\!|\Gamma \vdash \psi\ \mathsf{prp}|\!\}\,(\gamma) & \text{if } \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\,(\gamma) = \mathit{tt} \\ \mathit{ff} & \text{otherwise} \end{cases}$$
By I. H.
$$= \begin{cases} \{\!|\Gamma \vdash \psi\ \mathsf{prp}|\!\}\,(\delta) & \text{if } \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\,(\delta) = \mathit{tt} \\ \mathit{ff} & \text{otherwise} \end{cases}$$
$$= \{\!|\Gamma \vdash \phi \wedge \psi\ \mathsf{prp}|\!\}\,(\delta)$$

- Prop-Not

$$\{\!|\Gamma \vdash \neg\phi\ \mathsf{prp}|\!\}\,(\gamma)$$
$$= \begin{cases} \mathit{ff} & \text{if } \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\,(\gamma) = \mathit{tt} \\ \mathit{tt} & \text{otherwise} \end{cases}$$
By I. H.
$$= \begin{cases} \mathit{ff} & \text{if } \{\!|\Gamma \vdash \phi\ \mathsf{prp}|\!\}\,(\delta) = \mathit{tt} \\ \mathit{tt} & \text{otherwise} \end{cases}$$
$$= \{\!|\Gamma \vdash \neg\phi\ \mathsf{prp}|\!\}\,(\delta)$$

□

Proof of Theorem 6.2 (1). By structural induction on the derivation of $\Gamma \vdash \tau \preceq \sigma$:

- Sub-Refl

$$[\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$$
$$= [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$$

- Sub-Top

$$[\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)$$
$$= \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \,\middle|\, \{\!|\Gamma,\ x{:}_\mathsf{T}\tau \vdash \mathit{tt} \equiv_{\mathsf{Bool}} \mathit{tt}\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\}$$
$$= [\![\Gamma \vdash \{x{:}\tau \mid \mathit{tt} \equiv_{\mathsf{Bool}} \mathit{tt}\}{:}\mathsf{Ty}]\!]\,(\gamma)$$

- Sub-Impl

$$[\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma)$$
$$= \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \,\middle|\, \{\!|\Gamma,\ x{:}_\mathsf{T}\tau \vdash \phi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\}$$
by Assumption 1
$$\subseteq \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \,\middle|\, \{\!|\Gamma,\ x{:}_\mathsf{T}\tau \vdash \psi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\}$$
$$= [\![\Gamma \vdash \{x{:}\tau \mid \psi\}{:}\mathsf{Ty}]\!]\,(\gamma)$$

- Sub-Fun

$$
\begin{aligned}
&[\![\Gamma \vdash (x{:}\tau) \to \rho{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle f, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma)\,. \quad \langle f\cdot v, g\cdot w\rangle \in [\![\Gamma, x{:}_{\mathsf{T}}\tau \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle\gamma, v\rangle) \\ \forall \langle v_\perp, w_\perp\rangle \in \lfloor\Gamma \vdash \tau{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma))\,.\ \langle f\cdot v_\perp, g\cdot w_\perp\rangle \in \lfloor\Gamma, x{:}_{\mathsf{T}}\tau \vdash \rho{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma)) \end{array} \right\} \\
&\quad \text{By I. H. and Theorem 6.2 (ii)} \\
&\subseteq \left\{ \langle f, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v, w\rangle \in [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma)\,. \quad \langle f\cdot v, g\cdot w\rangle \in [\![\Gamma, x{:}_{\mathsf{T}}\tau \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle\gamma, v\rangle) \\ \forall \langle v_\perp, w_\perp\rangle \in \lfloor\Gamma \vdash \sigma{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma))\,.\ \langle f\cdot v_\perp, g\cdot w_\perp\rangle \in \lfloor\Gamma, x{:}_{\mathsf{T}}\tau \vdash \rho{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma)) \end{array} \right\} \\
&\quad \text{By I. H. and Lemma C.12} \\
&\subseteq \left\{ \langle f, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v, w\rangle \in [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma)\,. \quad \langle f\cdot v, g\cdot w\rangle \in [\![\Gamma, x{:}_{\mathsf{T}}\sigma \vdash \pi{:}\mathsf{Ty}]\!]\,(\langle\gamma, v\rangle) \\ \forall \langle v_\perp, w_\perp\rangle \in \lfloor\Gamma \vdash \sigma{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma))\,.\ \langle f\cdot v_\perp, g\cdot w_\perp\rangle \in \lfloor\Gamma, x{:}_{\mathsf{T}}\sigma \vdash \pi{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma)) \end{array} \right\} \\
&= [\![\Gamma \vdash (x{:}\sigma) \to \pi{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}
$$

- Sub-Flat

$$
\begin{aligned}
&[\![\Gamma \vdash \{x{:}\{y{:}\tau \mid \psi\} \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \{y{:}\tau \mid \psi\}{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \{\!|\Gamma, x{:}_{\mathsf{T}}\{y{:}\tau \mid \psi\} \vdash \phi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\} \\
&= \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \begin{array}{ll} \{\!|\Gamma, x{:}_{\mathsf{T}}\{y{:}\tau \mid \psi\} \vdash \phi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) & \text{(i)} \\ \quad \{\!|\Gamma, y{:}_{\mathsf{T}}\tau \vdash \psi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) & \text{(ii)} \end{array} \right\} \\
&\quad \text{By Lemma C.9, the well-formedness hypothesis} \\
&\quad \text{on the right-hand type, and the definition of } \wedge \\
&\quad \text{if (ii) holds then the value is in dom } ([\![\Gamma \vdash \{y{:}\tau \mid \psi\}{:}\mathsf{Ty}]\!]\,(\gamma)) \\
&= \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \{\!|\Gamma, y{:}_{\mathsf{T}}\tau \vdash \psi \wedge \phi\,[y/x]\ \ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\} \\
&= [\![\Gamma \vdash \{y{:}\tau \mid \psi \wedge \phi\,[y/x]\}{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}
$$

- Sub-Ix:

$$
\begin{aligned}
&[\![\Gamma \vdash \tau \cdot \overline{t}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= [\![\Gamma \vdash \tau{:}\overline{\sigma}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma) \cdot \overline{(\!|t|\!)\,(\rho_\Gamma\,(\gamma))}^{\,k} \\
&\quad \text{By Theorem 5.6 (i) } (\!|t_k|\!)\,(\rho_\Gamma\,(\gamma)), (\!|u_k|\!)\,(\rho_\Gamma\,(\gamma)) \in \mathrm{dom}\,([\![\Gamma \vdash \tau_k{:}\mathsf{Ty}]\!]\,(\gamma)) \\
&\quad \text{By Assumption 1 } \langle (\!|t_k|\!)\,(\rho_\Gamma\,(\gamma)), (\!|u_k|\!)\,(\rho_\Gamma\,(\gamma))\rangle \in \lfloor\Gamma \vdash \tau_k{:}\mathsf{Ty}\rfloor\,(\iota_\Gamma\,(\gamma)), \\
&\quad \text{Theorem 5.4, and kind being invariant under related arguments} \\
&= [\![\Gamma \vdash \tau{:}\overline{\sigma}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma) \cdot \overline{(\!|u|\!)\,(\rho_\Gamma\,(\gamma))}^{\,k} \\
&= [\![\Gamma \vdash \tau \cdot \overline{u}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}
$$

- Sub-Refine

$$
\begin{aligned}
&[\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\
&= \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \{\!|\Gamma, x{:}_{\mathsf{T}}\tau \vdash \phi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\} \\
&\quad \text{By I. H. and Lemma C.12 (iii)} \\
&\subseteq \left\{ \langle v, w\rangle \in [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma) \;\middle|\; \{\!|\Gamma, x{:}_{\mathsf{T}}\sigma \vdash \phi\ \mathsf{prp}|\!\}\,(\langle\gamma, v\rangle) \right\} \\
&= [\![\Gamma \vdash \{x{:}\sigma \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma)
\end{aligned}
$$

- Sub-Forget

$$\begin{aligned}&\llbracket \Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty} \rrbracket\,(\gamma)\\ =\ &\left\{ \langle v,\, w\rangle \in \llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma) \,\middle|\, \{\!\{ \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \phi\ \mathsf{prp} \}\!\}\,(\langle \gamma,\, v\rangle) \right\}\\ \subseteq\ &\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)\end{aligned}$$

- Sub-Trans

$$\begin{aligned}&\llbracket \Gamma \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)\\ &\quad \text{By I. H. on the first premise, which is well-formed by the second}\\ \subseteq\ &\llbracket \Gamma \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)\\ &\quad \text{By I. H. on the third premise}\\ \subseteq\ &\llbracket \Gamma \vdash \rho{:}\mathsf{Ty} \rrbracket\,(\gamma)\end{aligned}$$

- Sub-All

$$\begin{aligned}&\llbracket \Gamma \vdash \forall x{:}\kappa.\, \tau{:}\mathsf{Ty} \rrbracket\,(\gamma)\\ =\ &\left\{ \langle v,\, w\rangle \,\middle|\, \begin{array}{c} \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma),\ \forall P_\perp \in \lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\,(\iota_\Gamma\,(\gamma))\\ \pi_{\Gamma;\,\kappa}\,(\gamma,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{K}}\kappa \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\, \langle P,\, P_\perp\rangle\rangle) \end{array} \right\}\end{aligned}$$

The conditions on P and $P_\perp$ are those of $\langle \gamma,\, \langle P,\, P_\perp\rangle\rangle \in \mathrm{dom}\,(\llbracket \vdash \Gamma,\, x{:}_{\mathsf{K}}\kappa\ \mathsf{ctx} \rrbracket)$, so by I. H.

$$\begin{aligned}&\subseteq \left\{ \langle v,\, w\rangle \,\middle|\, \begin{array}{c} \forall P \in \llbracket \Gamma \vdash \kappa{:}\mathsf{Kd} \rrbracket\,(\gamma),\ \forall P_\perp \in \lfloor \Gamma \vdash \kappa{:}\mathsf{Kd} \rfloor\,(\iota_\Gamma\,(\gamma))\\ \pi_{\Gamma;\,\kappa}\,(\gamma,\, P) \subseteq P_\perp \Rightarrow \langle v,\, w\rangle \in \llbracket \Gamma,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\langle \gamma,\, \langle P,\, P_\perp\rangle\rangle) \end{array} \right\}\\ &= \llbracket \Gamma \vdash \forall x{:}\kappa.\, \sigma{:}\mathsf{Ty} \rrbracket\,(\gamma)\end{aligned}$$

□

Proof of Theorem 6.2 (ii). By structural induction on the derivation of $\Gamma \vdash \tau \preceq \sigma$:

- Sub-Refl

$$\begin{aligned}&\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\gamma)\\ =\ &\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\gamma)\end{aligned}$$

- Sub-Top

$$\begin{aligned}&\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\gamma)\\ =\ &\lfloor \Gamma \vdash \{x{:}\tau \mid tt \equiv_{\mathsf{Bool}} tt\}{:}\mathsf{Ty} \rfloor\,(\gamma)\end{aligned}$$

- Sub-Impl

$$\begin{aligned}&\lfloor \Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty} \rfloor\,(\gamma)\\ =\ &\lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\gamma)\\ =\ &\lfloor \Gamma \vdash \{x{:}\tau \mid \psi\}{:}\mathsf{Ty} \rfloor\,(\gamma)\end{aligned}$$

- Sub-Fun

$$\begin{aligned}&\lfloor \Gamma \vdash (x{:}\tau) \to \rho{:}\mathsf{Ty} \rfloor\,(\gamma)\\ =\ &\left\{ \langle f,\, g\rangle \,\middle|\, \forall\, \langle v,\, w\rangle \in \lfloor \Gamma \vdash \tau{:}\mathsf{Ty} \rfloor\,(\gamma)\,.\ \langle f \cdot v,\, g \cdot w\rangle \in \lfloor \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \rho{:}\mathsf{Ty} \rfloor\,(\gamma) \right\}\\ &\quad \text{By I. H. on the first premise and Lemma C.12 (ii)}\\ &\quad \lfloor \Gamma,\, x{:}_{\mathsf{T}}\tau \vdash \rho{:}\mathsf{Ty} \rfloor\,(\gamma) = \lfloor \Gamma,\, x{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty} \rfloor\,(\gamma)\\ =\ &\left\{ \langle f,\, g\rangle \,\middle|\, \forall\, \langle v,\, w\rangle \in \lfloor \Gamma \vdash \sigma{:}\mathsf{Ty} \rfloor\,(\gamma)\,.\ \langle f \cdot v,\, g \cdot w\rangle \in \lfloor \Gamma,\, x{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty} \rfloor\,(\gamma) \right\}\\ &\quad \text{By I. H. on the second premise}\end{aligned}$$

$$= \left\{ \langle f,\, g\rangle \,\middle|\, \forall \langle v,\, w\rangle \in [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma)\,.\ \langle f \cdot v,\, g \cdot w\rangle \in [\![\Gamma,\, x{:}_{\mathrm{T}}\sigma \vdash \pi{:}\mathsf{Ty}]\!]\,(\gamma) \right\}$$
$$= [\![\Gamma \vdash (x{:}\sigma) \to \pi{:}\mathsf{Ty}]\!]\,(\gamma)$$

- Sub-Flat

$$\begin{aligned} & [\![\Gamma \vdash \{x{:}\{y{:}\tau \mid \psi\} \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \{y{:}\tau \mid \psi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \{y{:}\tau \mid \psi \wedge \phi\,[y/x]\}{:}\mathsf{Ty}]\!]\,(\gamma) \end{aligned}$$

- Sub-Ix:

$$\begin{aligned} & [\![\Gamma \vdash \tau \cdot \overline{t}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \tau{:}\overline{\sigma}^{\,k} \to \mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \tau \cdot \overline{u}^{\,k}{:}\mathsf{Ty}]\!]\,(\gamma) \end{aligned}$$

- Sub-Refine:

$$\begin{aligned} & [\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\ & \text{By I. H.} \\ = {} & [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \{x{:}\sigma \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \end{aligned}$$

- Sub-Forget

$$\begin{aligned} & [\![\Gamma \vdash \{x{:}\tau \mid \phi\}{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \end{aligned}$$

- Sub-Trans

$$\begin{aligned} & [\![\Gamma \vdash \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\ & \text{By I. H. on the first premise, which is well-formed by the second} \\ = {} & [\![\Gamma \vdash \sigma{:}\mathsf{Ty}]\!]\,(\gamma) \\ & \text{By I. H. on the third premise} \\ = {} & [\![\Gamma \vdash \rho{:}\mathsf{Ty}]\!]\,(\gamma) \end{aligned}$$

- Sub-All

$$\begin{aligned} & [\![\Gamma \vdash \forall x{:}\kappa.\ \tau{:}\mathsf{Ty}]\!]\,(\gamma) \\ = {} & \left\{ \langle v,\, w\rangle \,\middle|\, \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)\,.\ \langle v,\, w\rangle \in [\![\Gamma,\, x{:}_{\kappa}\kappa \vdash \tau{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, P\rangle) \right\} \\ & \text{By I. H. at each } \langle \gamma,\, P\rangle \in \mathrm{dom}\,([\![\vdash \Gamma,\, x{:}_{\kappa}\kappa\ \mathsf{ctx}]\!]) \\ = {} & \left\{ \langle v,\, w\rangle \,\middle|\, \forall P \in [\![\Gamma \vdash \kappa{:}\mathsf{Kd}]\!]\,(\gamma)\,.\ \langle v,\, w\rangle \in [\![\Gamma,\, x{:}_{\kappa}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \gamma,\, P\rangle) \right\} \\ = {} & [\![\Gamma \vdash \forall x{:}\kappa.\ \sigma{:}\mathsf{Ty}]\!]\,(\gamma) \end{aligned}$$

□

Proof of Theorem 7.1. We abbreviate $\Theta \triangleq \Gamma,\, x{:}_{\kappa}\kappa,\, \Delta$ and $\epsilon(X) \triangleq \langle \langle \gamma,\, X\rangle,\, \delta\rangle$. By induction on the derivation of $\Theta \vdash \tau{:}\mu$.

- Ty-Nat and Ty-Bool:

$$\llbracket \Theta \vdash \mathcal{B}{:}\mathsf{Ty} \rrbracket\,(\epsilon(A)) = \left\{ \langle i_{\mathcal{B}}\,(b)\,,\ i_{\mathcal{B}}\,(b)\rangle \,\middle|\, b \in \mathcal{B} \right\} \cup \{\langle \bot,\ \bot\rangle\} = \llbracket \Theta \vdash \mathcal{B}{:}\mathsf{Ty} \rrbracket\,(\epsilon(B))$$

  The operator is contractive as it is constant.
- Ty-Var: $\llbracket \Theta \vdash y{:}\mathsf{Ty} \rrbracket\,(\epsilon(A)) = \epsilon(A)(y)$ and $\llbracket \Theta \vdash y{:}\mathsf{Ty} \rrbracket\,(\epsilon(B)) = \epsilon(B)(y)$, if $x = y$ then the operator is the identity, otherwise it is constant and hence contractive.
- Ty-Fun:

$$\llbracket \Theta \vdash (y{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(C))$$
$$= \left\{ \langle f,\ g\rangle \,\middle|\, \forall\, \langle v,\ w\rangle \in \llbracket \Theta \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\epsilon(C))\,.\ \langle f \cdot v,\ g \cdot w\rangle \in \llbracket \Theta,\ y{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(C)) \right\}$$

  Let $\langle f,\ g\rangle \in \llbracket \Theta \vdash (y{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(A))$; we show that $\langle \pi_{n+1}(f),\ \pi_{n+1}(g)\rangle \in \llbracket \Theta \vdash (y{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(B))$.

$$\begin{aligned}
&\langle v,\ w\rangle \in \llbracket \Theta \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\epsilon(B)) \\
&\quad \text{By the I. H. they agree at least up to level } n \\
\implies &\langle \pi_n(v),\ \pi_n(w)\rangle \in \llbracket \Theta \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\epsilon(A)) \\
\implies &\langle f \cdot \pi_n(v),\ g \cdot \pi_n(w)\rangle \in \llbracket \Theta,\ y{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(A)) \\
&\quad \text{By the I. H. they agree at least up to level } n \\
\implies &\langle \pi_n(f \cdot \pi_n(v)),\ \pi_n(g \cdot \pi_n(w))\rangle \in \llbracket \Theta,\ y{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(B)) \\
&\quad \text{By the fact that } \pi_{n+1}(f) \cdot v = \pi_n(f \cdot \pi_n(v)) \text{ in the bilimit} \\
\implies &\langle \pi_n(f \cdot \pi_n(v)),\ \pi_n(g \cdot \pi_n(w))\rangle \in \llbracket \Theta,\ y{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(B)) \\
\implies &\langle \pi_{n+1}(f) \cdot v,\ \pi_{n+1}(g) \cdot w\rangle \in \llbracket \Theta,\ y{:}_{\mathsf{T}}\tau \vdash \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(B)) \\
\implies &\langle \pi_{n+1}(f),\ \pi_{n+1}(g)\rangle \in \llbracket \Theta \vdash (y{:}\tau) \to \sigma{:}\mathsf{Ty} \rrbracket\,(\epsilon(B))
\end{aligned}$$

  The other direction is symmetric. Hence the operator is contractive.
- Ty-All:

$$\llbracket \Theta \vdash \forall y{:}\nu.\ \tau{:}\mathsf{Ty} \rrbracket\,(\epsilon(C))$$
$$= \left\{ \langle v,\ w\rangle \,\middle|\, \forall P \in \llbracket \Theta \vdash \nu{:}\mathsf{Kd} \rrbracket\,(\epsilon)\,.\ \langle v,\ w\rangle \in \llbracket \Theta,\ y{:}_{\mathsf{K}}\nu \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\langle \epsilon(C),\ P\rangle) \right\}$$

  We can apply the I. H. on each instance of the body operator; the alternative that holds depends only on the derivation of τ, so it is the same for every P.
  If every instance is the identity, then the whole operator is the identity. Otherwise every instance is contractive. Hence for every P the corresponding body interpretations agree at level $n + 1$.
- Ty-Ref:

$$\llbracket \Theta \vdash \{y{:}\tau \mid \phi\}{:}\mathsf{Ty} \rrbracket\,(\epsilon(C)) = \llbracket \Theta \vdash \tau{:}\mathsf{Ty} \rrbracket\,(\epsilon(C))$$

  By the I. H. the operator is either the identity or contractive.
- Ty-App:

$$\llbracket \Theta \vdash \tau \cdot t{:}\kappa\,[t/y] \rrbracket\,(\epsilon(C)) = \llbracket \Theta \vdash \sigma{:}\,(x{:}\tau) \to \kappa \rrbracket\,(\epsilon(C))$$

  By the I. H. the operator is either the identity or contractive.
- Ty-IndPos and Ty-IndStrat:

$$\left\llbracket \Theta \vdash \mu K^{\langle \nu \rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\,\overline{\tau}^{\,k} \to \mathsf{Ty} \right\rrbracket\,(\epsilon(C))$$
$$= \mathrm{fix}_X \bigcup_i \left\{ \langle \langle C_i,\ \overline{v}^{\,j}\rangle,\ \langle C_i,\ \overline{w}^{\,j}\rangle\rangle \,\middle|\, \langle v_j,\ w_j\rangle \in \llbracket \Theta,\ K{:}_{\mathsf{K}}\overline{\tau}^{\,k} \to \mathsf{Ty},\ \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty} \rrbracket\,(\langle \epsilon(C),\ X\rangle) \right\} \cup \{\langle \bot,\ \bot\rangle\}$$

Let $\Phi_C(X)$ denote the operator under the fixed point computed in $\epsilon(C)$. We first show that if $A_n = B_n$ and $X_n = Y_n$ then $\Phi_A(X)_{n+1} = \Phi_B(Y)_{n+1}$: let $\langle\langle C_i,\, \overline{v_j}^{\,j}\rangle,\, \langle C_i,\, \overline{w_j}^{\,j}\rangle\rangle \in \Phi_A(X)$, we show that its $\langle \pi_{n+1}(\langle C_i,\, \overline{v_j}^{\,j}\rangle),\, \pi_{n+1}(\langle C_i,\, \overline{w_j}^{\,j}\rangle)\rangle$ is in $\Phi_B(Y)$.

$$\left\langle v_j,\, w_j \right\rangle \in [\![\Theta,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\tau}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!](\langle \epsilon(A),\, X\rangle)$$

By the I. H., applied once for x and once for K, they agree at least up to level n

$$\implies \langle \pi_n(v_j),\, \pi_n(w_j)\rangle \in [\![\Theta,\, K{:}_{\kappa}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\tau}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!](\langle \epsilon(B),\, Y\rangle)$$

By Lemma D.1 and the fact that $\pi_n(a) = \pi_n(b)$ implies $\pi_m(a) = \pi_m(b)$ for $m \le n$

$$\implies \langle \pi_{n+1}(\langle C_i,\, \overline{v_j}^{\,j}\rangle),\, \pi_{n+1}(\langle C_i,\, \overline{w_j}^{\,j}\rangle)\rangle \in \Phi_B(Y)$$

The other direction is symmetric. Hence if $A_n = B_n$ and $X_n = Y_n$ then $\Phi_A(X)_{n+1} = \Phi_B(Y)_{n+1}$; in particular, taking $A = B$, each Φ_C is contractive.
We show by induction on m that $A_n = B_n$ implies $\Phi_A^m(\{\langle \bot,\, \bot\rangle\})_{n+1} = \Phi_B^m(\{\langle \bot,\, \bot\rangle\})_{n+1}$: the base case is trivial, and the inductive step follows from the previous argument, as agreement at level $n+1$ implies agreement at level n.
Since all CUPERs agree at level 0, contractivity of Φ_C gives $(\Phi_C^m(S))_m = (\Phi_C^m(\{\langle \bot,\, \bot\rangle\}))_m$ for every S by induction on m; instantiating S with the fixed point, $(\mathrm{fix}_X\, \Phi_C(X))_m = (\Phi_C^m(\{\langle \bot,\, \bot\rangle\}))_m$. Taking $m = n+1$, the two fixed points agree at level $n+1$. Hence the operator is contractive.

□

Proof of Lemma 7.2. We use the following abbreviations

- $\Delta \triangleq \Gamma,\, K{:}_{\kappa}\kappa,\, \overline{x{:}\sigma}^{\,i}$
- $\delta(x) = \left\langle \langle \gamma,\, \langle x,\, P_{\bot}\rangle\rangle,\, \overline{v}^{\,i} \right\rangle$

by induction on the structure of $\Delta \vdash \rho{:}\mu$,

- Ty-Nat and Ty-Bool when $p \in \{+, -\}$

$$[\![\Delta \vdash \mathcal{B}{:}\mathsf{Ty}]\!]\,(\delta(P)) = [\![\Delta \vdash \mathcal{B}{:}\mathsf{Ty}]\!]\,(\delta(Q))$$

- Ty-Fun when $p = +$

$$[\![\Delta \vdash (y{:}\sigma) \to \rho{:}\mathsf{Ty}]\!]\,(\delta(P))$$

$$= \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, v\rangle) \\ \forall\, \langle v_{\bot},\, w_{\bot}\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(P)))\,.\ \langle f\cdot v_{\bot},\, g\cdot w_{\bot}\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(P))) \end{array} \right\}$$

By I. H. $[\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q)) \subseteq [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P))$

$$\subseteq \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, v\rangle) \\ \forall\, \langle v_{\bot},\, w_{\bot}\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(P)))\,.\ \langle f\cdot v_{\bot},\, g\cdot w_{\bot}\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(P))) \end{array} \right\}$$

Notice $\iota_{\Delta}\,(\delta(P)) = \iota_{\Delta}\,(\delta(Q))$

$$= \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, v\rangle) \\ \forall\, \langle v_{\bot},\, w_{\bot}\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(Q)))\,.\ \langle f\cdot v_{\bot},\, g\cdot w_{\bot}\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(Q))) \end{array} \right\}$$

By I. H. $[\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, v\rangle) \supseteq [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, v\rangle)$

$$\subseteq \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, v\rangle) \\ \forall\, \langle v_{\bot},\, w_{\bot}\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(Q)))\,.\ \langle f\cdot v_{\bot},\, g\cdot w_{\bot}\rangle \in [\![\Delta,\, y{:}_{\tau}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\iota_{\Delta}\,(\delta(Q))) \end{array} \right\}$$

$$= [\![\Delta \vdash (y{:}\sigma) \to \rho{:}\mathsf{Ty}]\!]\,(\delta(Q))$$

- Ty-Fun when $p = -$

$$
\begin{aligned}
&[\![\Delta \vdash (y{:}\sigma) \to \rho{:}\mathsf{Ty}]\!]\,(\delta(Q)) \\
&= \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, v\rangle) \\ \forall\, \langle v_\perp,\, w_\perp\rangle \in \lfloor\Delta \vdash \sigma{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(Q)))\,.\ \langle f\cdot v_\perp,\, g\cdot w_\perp\rangle \in \lfloor\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(Q))) \end{array} \right\} \\
&\quad \text{By I. H. } [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P)) \subseteq [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q)) \\
&\subseteq \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, v\rangle) \\ \forall\, \langle v_\perp,\, w_\perp\rangle \in \lfloor\Delta \vdash \sigma{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(Q)))\,.\ \langle f\cdot v_\perp,\, g\cdot w_\perp\rangle \in \lfloor\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(Q))) \end{array} \right\} \\
&\quad \text{Notice } \iota_\Delta\,(\delta(P)) = \iota_\Delta\,(\delta(Q)) \\
&= \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, v\rangle) \\ \forall\, \langle v_\perp,\, w_\perp\rangle \in \lfloor\Delta \vdash \sigma{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(P)))\,.\ \langle f\cdot v_\perp,\, g\cdot w_\perp\rangle \in \lfloor\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(P))) \end{array} \right\} \\
&\quad \text{By I. H. } [\![\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, v\rangle) \supseteq [\![\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, v\rangle) \\
&\subseteq \left\{ \langle f,\, g\rangle \;\middle|\; \begin{array}{l} \forall\, \langle v,\, w\rangle \in [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P))\,.\ \langle f\cdot v,\, g\cdot w\rangle \in [\![\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, v\rangle) \\ \forall\, \langle v_\perp,\, w_\perp\rangle \in \lfloor\Delta \vdash \sigma{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(P)))\,.\ \langle f\cdot v_\perp,\, g\cdot w_\perp\rangle \in \lfloor\Delta,\, y{:}_{\mathsf{T}}\sigma \vdash \rho{:}\mathsf{Ty}\rfloor\,(\iota_\Delta\,(\delta(P))) \end{array} \right\} \\
&= [\![\Delta \vdash (y{:}\sigma) \to \rho{:}\mathsf{Ty}]\!]\,(\delta(P))
\end{aligned}
$$

- Ty-All when $p = +$

$$
\begin{aligned}
&[\![\Delta \vdash \forall x{:}\kappa.\ \sigma{:}\mathsf{Ty}]\!]\,(\delta(P)) \\
&= \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall R \in [\![\Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta(P))\,,\ \forall R_\perp \in \lfloor\Delta \vdash \kappa{:}\mathsf{Kd}\rfloor\,(\iota_\Delta\,(\delta(P))) \\ \pi_{\Delta;\,\kappa}\,(\delta(P),\, R) \subseteq R_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Delta,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, \langle R,\, R_\perp\rangle\rangle) \end{array} \right\} \\
&\quad \text{By Lemma E.1 and } \iota_\Delta\,(\delta(P)) = \iota_\Delta\,(\delta(Q)) \\
&\subseteq \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall R \in [\![\Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta(Q))\,,\ \forall R_\perp \in \lfloor\Delta \vdash \kappa{:}\mathsf{Kd}\rfloor\,(\iota_\Delta\,(\delta(Q))) \\ \pi_{\Delta;\,\kappa}\,(\delta(Q),\, R) \subseteq R_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Delta,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, \langle R,\, R_\perp\rangle\rangle) \end{array} \right\} \\
&\quad \text{By I. H.} \\
&\subseteq \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall R \in [\![\Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta(Q))\,,\ \forall R_\perp \in \lfloor\Delta \vdash \kappa{:}\mathsf{Kd}\rfloor\,(\iota_\Delta\,(\delta(Q))) \\ \pi_{\Delta;\,\kappa}\,(\delta(Q),\, R) \subseteq R_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Delta,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, \langle R,\, R_\perp\rangle\rangle) \end{array} \right\} \\
&= [\![\Delta \vdash \forall x{:}\kappa.\ \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q))
\end{aligned}
$$

- Ty-All when $p = -$

$$
\begin{aligned}
&[\![\Delta \vdash \forall x{:}\kappa.\ \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q)) \\
&= \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall R \in [\![\Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta(Q))\,,\ \forall R_\perp \in \lfloor\Delta \vdash \kappa{:}\mathsf{Kd}\rfloor\,(\iota_\Delta\,(\delta(Q))) \\ \pi_{\Delta;\,\kappa}\,(\delta(Q),\, R) \subseteq R_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Delta,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, \langle R,\, R_\perp\rangle\rangle) \end{array} \right\} \\
&\quad \text{By Lemma E.1 and } \iota_\Delta\,(\delta(Q)) = \iota_\Delta\,(\delta(P)) \\
&\subseteq \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall R \in [\![\Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta(P))\,,\ \forall R_\perp \in \lfloor\Delta \vdash \kappa{:}\mathsf{Kd}\rfloor\,(\iota_\Delta\,(\delta(P))) \\ \pi_{\Delta;\,\kappa}\,(\delta(P),\, R) \subseteq R_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Delta,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta(Q),\, \langle R,\, R_\perp\rangle\rangle) \end{array} \right\} \\
&\quad \text{By I. H.} \\
&\subseteq \left\{ \langle v,\, w\rangle \;\middle|\; \begin{array}{c} \forall R \in [\![\Delta \vdash \kappa{:}\mathsf{Kd}]\!]\,(\delta(P))\,,\ \forall R_\perp \in \lfloor\Delta \vdash \kappa{:}\mathsf{Kd}\rfloor\,(\iota_\Delta\,(\delta(P))) \\ \pi_{\Delta;\,\kappa}\,(\delta(P),\, R) \subseteq R_\perp \Rightarrow \langle v,\, w\rangle \in [\![\Delta,\, x{:}_{\mathsf{K}}\kappa \vdash \sigma{:}\mathsf{Ty}]\!]\,(\langle \delta(P),\, \langle R,\, R_\perp\rangle\rangle) \end{array} \right\} \\
&= [\![\Delta \vdash \forall x{:}\kappa.\ \sigma{:}\mathsf{Ty}]\!]\,(\delta(P))
\end{aligned}
$$

- Ty-Var when $p \in \{+, -\}$

$$\begin{aligned} & [\![\Delta \vdash x{:}\mathsf{Ty}]\!]\,(\delta(P)) \\ = {} & \delta(P)(x) \\ & \text{By assumption} \\ \subseteq {} & /\supseteq\ \delta(Q)(x) \\ = {} & [\![\Delta \vdash x{:}\mathsf{Ty}]\!]\,(\delta(Q)) \end{aligned}$$

- Ty-Ref when $p \in \{+, -\}$

$$\begin{aligned} & [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(P)) \\ = {} & \left\{ \langle v, w\rangle \in [\![\Delta \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta(P)) \;\middle|\; \{\!\{\Delta,\, x{:}_{\mathsf{T}}\tau \vdash \phi\ \mathsf{prp}\}\!\}\,(\langle \delta(P), v\rangle) \right\} \\ & \text{Notice that } \{\!\{-\}\!\} \text{ is invariant under kind assignments} \\ = {} & \left\{ \langle v, w\rangle \in [\![\Delta \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta(P)) \;\middle|\; \{\!\{\Delta,\, x{:}_{\mathsf{T}}\tau \vdash \phi\ \mathsf{prp}\}\!\}\,(\langle \delta(Q), v\rangle) \right\} \\ & \text{By I. H.} \\ \subseteq {} & /\supseteq \left\{ \langle v, w\rangle \in [\![\Delta \vdash \tau{:}\mathsf{Ty}]\!]\,(\delta(Q)) \;\middle|\; \{\!\{\Delta,\, x{:}_{\mathsf{T}}\tau \vdash \phi\ \mathsf{prp}\}\!\}\,(\langle \delta(Q), v\rangle) \right\} = [\![\Delta \vdash \sigma{:}\mathsf{Ty}]\!]\,(\delta(Q)) \end{aligned}$$

- Ty-App when $p \in \{+, -\}$

$$\begin{aligned} & [\![\Delta \vdash \sigma \cdot t{:}\kappa]\!]\,(\delta(P)) \\ = {} & [\![\Delta \vdash \sigma{:}(x{:}\tau) \to \kappa]\!]\,(\delta(P)) \cdot (\!|t|\!)\,(\rho_\Delta\,(\delta(P))) \\ & \text{By noticing that } \rho_\Delta\,(\delta(P)) = \rho_\Delta\,(\delta(Q)) \\ = {} & [\![\Delta \vdash \sigma{:}(x{:}\tau) \to \kappa]\!]\,(\delta(P)) \cdot (\!|t|\!)\,(\rho_\Delta\,(\delta(Q))) \\ & \text{By I. H.} \\ \leq {} & /\geq\ [\![\Delta \vdash \sigma{:}(x{:}\tau) \to \kappa]\!]\,(\delta(Q)) \cdot (\!|t|\!)\,(\rho_\Delta\,(\delta(Q))) \\ = {} & [\![\Delta \vdash \sigma \cdot t{:}\mathsf{Ty}]\!]\,(\delta(Q)) \end{aligned}$$

- Ty-IndPos when $p \in \{+, -\}$ Since

$$[\![\Delta \vdash \mu K^{\langle \nu\rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\delta(P))$$

$$= \mathsf{lfp}_X \left(\bar{q}^{\,k} \mapsto \bigcup_i \left\{ \langle \langle C_i, \bar{v}^{\,j}\rangle, \langle C_i, \overline{w}^{\,j}\rangle\rangle \;\middle|\; \begin{array}{l} \langle v_j, w_j\rangle \in [\![\Delta,\, K{:}_{\mathsf{k}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \delta(P), \langle X, [\![\Delta \vdash \mu K^{\langle \nu\rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Delta\,(\delta(P)))\rangle\rangle, \bar{v}^{\,j-1}\rangle) \\ \langle q_k, (\!|t_k|\!)\,(\langle \rho_\Delta\,(\delta(P)), \bar{v}^{\,j}\rangle)\rangle \in [\![\Delta,\, \overline{x{:}_{\mathsf{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}]\!]\,(\iota_\Delta\,(\delta(P))) \end{array} \right\} \right)$$

$$[\![\Delta \vdash \mu K^{\langle \nu\rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\delta(Q))$$

$$= \mathsf{lfp}_X \left(\bar{q}^{\,k} \mapsto \bigcup_i \left\{ \langle \langle C_i, \bar{v}^{\,j}\rangle, \langle C_i, \overline{w}^{\,j}\rangle\rangle \;\middle|\; \begin{array}{l} \langle v_j, w_j\rangle \in [\![\Delta,\, K{:}_{\mathsf{k}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \delta(Q), \langle X, [\![\Delta \vdash \mu K^{\langle \nu\rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Delta\,(\delta(Q)))\rangle\rangle, \bar{v}^{\,j-1}\rangle) \\ \langle q_k, (\!|t_k|\!)\,(\langle \rho_\Delta\,(\delta(Q)), \bar{v}^{\,j}\rangle)\rangle \in [\![\Delta,\, \overline{x{:}_{\mathsf{T}}\tau}^{\,k-1} \vdash \tau_k{:}\mathsf{Ty}]\!]\,(\iota_\Delta\,(\delta(Q))) \end{array} \right\} \right)$$

Since they are both defined as the least fixed point of a monotone functional, it suffices to show that for all X the functionals respect $\leq /\geq$ pointwise; for this it is enough to show that each component of the functional respects the order $\leq /\geq$:

$$\begin{array}{l} [\![\Delta,\, K{:}_{\mathsf{k}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \delta(P), \langle X, [\![\Delta \vdash \mu K^{\langle \nu\rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Delta\,(\delta(P)))\rangle\rangle, \bar{v}^{\,j-1}\rangle) \\ \qquad\qquad \subseteq /\supseteq \\ [\![\Delta,\, K{:}_{\mathsf{k}}\overline{\tau}^{\,k} \to \mathsf{Ty},\, \overline{x{:}_{\mathsf{T}}\sigma}^{\,j-1} \vdash \sigma_j{:}\mathsf{Ty}]\!]\,(\langle \delta(Q), \langle X, [\![\Delta \vdash \mu K^{\langle \nu\rangle} \left\{ \underline{C{:}\,(\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} {:}\overline{\tau}^{\,k} \to \mathsf{Ty}]\!]\,(\iota_\Delta\,(\delta(Q)))\rangle\rangle, \bar{v}^{\,j-1}\rangle) \end{array}$$

which follows from the I. H. and the fact that $\rho_\Delta\,(\delta(P)) = \rho_\Delta\,(\delta(Q))$ and $\iota_\Delta\,(\delta(P)) = \iota_\Delta\,(\delta(Q))$

- Ty-IndStrat when $p \in \{+,-\}$ the only possible case is that $\mathrm{pol}_K(-) = \circ$ and hence

$$\left[\!\left[\Delta \vdash \mu K^{\langle m\rangle} \left\{ \underline{C\colon (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} \colon \overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right](\delta(P))$$
$$= \left[\!\left[\Delta \vdash \mu K^{\langle m\rangle} \left\{ \underline{C\colon (\overline{y{:}\sigma}^{\,j}) \to K \cdot \overline{t}^{\,k}}_{\,i} \right\} \colon \overline{\tau}^{\,k} \to \mathsf{Ty} \right]\!\right](\delta(Q))$$

as the semantics of the kind is independent of the refined kind assignment of K.

□

Proof of Lemma E.1. We use the following abbreviations

- $\Delta \triangleq \Gamma,\ K :_{\kappa} \mu,\ \overline{x{:}\sigma}^{\,i}$
- $\delta(x) = \left\langle \langle \gamma,\ \langle x,\ P_\perp \rangle \rangle,\ \overline{v}^{\,i} \right\rangle$

We prove the case for $p = -$, the case for $p = +$ is identical but with the inequalities reversed.

First, we notice that all kinds have the shape $\overline{\tau}^{\,k} \to \mathsf{Ty}$, thus if $\mathrm{pol}_K(\kappa) \le -$ then $\forall k\ \mathrm{pol}_K(\tau_k) \le +$. Now, we can proceed by induction on k:

- $k = 0$, then $\kappa = \mathsf{Ty}$, and $[\![\Delta \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!](\delta(P)) = \mathcal{PER}(\mathcal{D}) = [\![\Delta \vdash \mathsf{Ty}{:}\mathsf{Kd}]\!](\delta(Q))$.
- $k > 0$, then $\kappa = \tau \to \kappa$, and

$$[\![\Delta \vdash \tau \to \kappa{:}\mathsf{Kd}]\!](\delta(P))$$
$$= \left\{ f \;\middle|\; \begin{array}{c} f \in \mathcal{D} \to (\mathcal{D} \to)^{\mathrm{arity}(\kappa)}\, \mathcal{PER}(\mathcal{D}) \\ \forall v \in \mathrm{dom}\left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\delta(P))\right)\ f(v) \in [\![\Delta,\ x :_{\mathsf{T}} \tau \vdash \kappa{:}\mathsf{Kd}]\!](\langle \delta(P),\ v\rangle) \\ \forall \langle v,\ w\rangle \in \left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\iota_\Delta(\delta(P))) \cap \mathrm{dom}\left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\delta(P))\right)^2\right)\ f(v) = f(w) \end{array} \right\}$$

By Lemma 7.2 and $\iota_\Delta(\delta(P)) = \iota_\Delta(\delta(Q))$

$$\supseteq \left\{ f \;\middle|\; \begin{array}{c} f \in \mathcal{D} \to (\mathcal{D} \to)^{\mathrm{arity}(\kappa)}\, \mathcal{PER}(\mathcal{D}) \\ \forall v \in \mathrm{dom}\left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\delta(Q))\right)\ f(v) \in [\![\Delta,\ x :_{\mathsf{T}} \tau \vdash \kappa{:}\mathsf{Kd}]\!](\langle \delta(P),\ v\rangle) \\ \forall \langle v,\ w\rangle \in \left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\iota_\Delta(\delta(Q))) \cap \mathrm{dom}\left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\delta(Q))\right)^2\right)\ f(v) = f(w) \end{array} \right\}$$

By I. H.

$$\supseteq \left\{ f \;\middle|\; \begin{array}{c} f \in \mathcal{D} \to (\mathcal{D} \to)^{\mathrm{arity}(\kappa)}\, \mathcal{PER}(\mathcal{D}) \\ \forall v \in \mathrm{dom}\left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\delta(Q))\right)\ f(v) \in [\![\Delta,\ x :_{\mathsf{T}} \tau \vdash \kappa{:}\mathsf{Kd}]\!](\langle \delta(Q),\ v\rangle) \\ \forall \langle v,\ w\rangle \in \left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\iota_\Delta(\delta(Q))) \cap \mathrm{dom}\left([\![\Delta \vdash \tau{:}\mathsf{Ty}]\!](\delta(Q))\right)^2\right)\ f(v) = f(w) \end{array} \right\}$$
$$= [\![\Delta \vdash \tau \to \kappa{:}\mathsf{Kd}]\!](\delta(Q))$$

□

Proof of Lemma 7.3. By induction on the structure of τ:

- If $\tau = K \cdot \overline{t}^{\,k}$,

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(K \cdot \overline{t}^{\,k},\ \gamma_\rho,\ (\!|m|\!)\left(\gamma_\rho\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(K \cdot \overline{t}^{\,k},\ \delta_\rho,\ (\!|m|\!)\left(\delta_\rho\right)\right) \end{array} \right\rangle$$
$$= \left\langle \begin{array}{l} r \cdot \overline{(\!|t|\!)\left(\gamma_\rho\right)}^{\,k} \cdot (\!|m|\!)\left(\gamma_\rho\right), \\ r' \cdot \overline{(\!|t|\!)\left(\delta_\rho\right)}^{\,k} \cdot (\!|m|\!)\left(\delta_\rho\right) \end{array} \right\rangle$$

As $m : K \cdot \overline{t}^{\,k}$ gives $(\!|m|\!) \in P \cdot \overline{(\!|t|\!)}^{\,k}$, the realizer hypothesis yields

$$\in [\![\Gamma,\ \overline{x :_{\mathsf{T}} \tau}^{\,k},\ y :_{\mathsf{T}} D\ \overline{x}^{\,k} \vdash \rho{:}\mathsf{Ty}]\!]\left(\left\langle \langle \gamma,\ \overline{(\!|t|\!)\left(\gamma_\rho\right)}^{\,k} \rangle,\ (\!|m|\!)\left(\gamma_\rho\right) \right\rangle\right)$$

By Lemma C.5

$$= [\![\Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\left(m,\ K \cdot \overline{t}^{\,k}\right){:}\mathsf{Ty}]\!]\left(\left\langle \langle \gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1} \right\rangle\right)$$

- If $\tau \in \{\mathcal{B}, \tau \cdot \overline{t}^{\,n}\}$

$$
\begin{aligned}
&\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\tau,\, \gamma_\rho,\, (\!|m|\!)\left(\gamma_\rho\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(\tau,\, \delta_\rho,\, (\!|m|\!)\left(\delta_\rho\right)\right) \end{array} \right\rangle \\
&= \langle \star,\, \star \rangle \\
&\in [\![\Gamma,\, \Delta(j-1) \vdash \mathsf{Unit{:}Ty}]\!] \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right) \\
&= [\![\Gamma,\, \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\left(m,\, \tau\right) {:}\mathsf{Ty}]\!] \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right)
\end{aligned}
$$

- If $\tau = \{x{:}\tau \mid \phi\}$

$$
\begin{aligned}
&\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\{x{:}\tau \mid \phi\},\, \gamma_\rho,\, (\!|m|\!)\left(\gamma_\rho\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(\{x{:}\tau \mid \phi\},\, \delta_\rho,\, (\!|m|\!)\left(\delta_\rho\right)\right) \end{array} \right\rangle \\
&= \left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\tau,\, \gamma_\rho,\, (\!|m|\!)\left(\gamma_\rho\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(\tau,\, \delta_\rho,\, (\!|m|\!)\left(\delta_\rho\right)\right) \end{array} \right\rangle \\
&\quad \text{By induction hypothesis} \\
&\in [\![\Gamma,\, \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\left(m,\, \tau\right) {:}\mathsf{Ty}]\!] \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right) \\
&= [\![\Gamma,\, \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\left(m,\, \{x{:}\tau \mid \phi\}\right) {:}\mathsf{Ty}]\!] \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right)
\end{aligned}
$$

- If $\tau = x{:}\tau \to \sigma$ $\mathsf{Fixed}\ \langle q,\, n \rangle \in [\![\Gamma,\, \Delta(j-1) \vdash \tau{:}\mathsf{Ty}]\!] \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right)$ by I. H.

$$
\begin{aligned}
&\text{(i)} \left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\sigma,\, \left\langle \gamma_\rho,\, q \right\rangle,\, (\!|m \cdot x|\!)\left(\left\langle \gamma_\rho,\, q \right\rangle\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(\sigma,\, \left\langle \delta_\rho,\, n \right\rangle,\, (\!|m \cdot x|\!)\left(\left\langle \delta_\rho,\, n \right\rangle\right)\right) \end{array} \right\rangle \\
&\in [\![\Gamma,\, \Delta(j-1),\, x{:}_{\mathsf{T}}\tau \vdash \mathsf{El}_{K;\rho}\left(m \cdot x,\, \sigma\right) {:}\mathsf{Ty}]\!] \left(\left\langle \left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle,\, q \right\rangle \right)
\end{aligned}
$$

And fixed $\langle q_\perp,\, n_\perp \rangle \in [\!|\Gamma,\, \Delta(j-1) \vdash \tau{:}\mathsf{Ty}|\!] \left(\iota_{\Gamma,\, \Delta(j-1)} \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right) \right)$ by Lemma E.2

$$
\begin{aligned}
&\text{(ii)} \left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\sigma,\, \left\langle \gamma_\rho,\, q_\perp \right\rangle,\, (\!|m \cdot x|\!)\left(\left\langle \gamma_\rho,\, q_\perp \right\rangle\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(\sigma,\, \left\langle \delta_\rho,\, n_\perp \right\rangle,\, (\!|m \cdot x|\!)\left(\left\langle \delta_\rho,\, n_\perp \right\rangle\right)\right) \end{array} \right\rangle \\
&\in [\!|\Gamma,\, \Delta(j-1),\, x{:}_{\mathsf{T}}\tau \vdash \mathsf{El}_{K;\rho}\left(m \cdot x,\, \sigma\right) {:}\mathsf{Ty}|\!] \left(\left\langle \iota_{\Gamma,\, \Delta(j-1)} \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right),\, q_\perp \right\rangle \right)
\end{aligned}
$$

Hence,

$$
\begin{aligned}
&\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(x{:}\tau \to \sigma,\, \gamma_\rho,\, (\!|m|\!)\left(\gamma_\rho\right)\right), \\ \mathsf{tmEl}_{K;r'}\left(x{:}\tau \to \sigma,\, \delta_\rho,\, (\!|m|\!)\left(\delta_\rho\right)\right) \end{array} \right\rangle \\
&= \left\langle \begin{array}{l} i_{\to}\left(q \mapsto \mathsf{tmEl}_{K;r}\left(\sigma,\, \left\langle \gamma_\rho,\, q \right\rangle,\, (\!|m|\!)\left(\gamma_\rho\right) \cdot q\right)\right), \\ i_{\to}\left(q \mapsto \mathsf{tmEl}_{K;r'}\left(\sigma,\, \left\langle \delta_\rho,\, q \right\rangle,\, (\!|m|\!)\left(\delta_\rho\right) \cdot q\right)\right) \end{array} \right\rangle \\
&\quad \text{We notice that } (\!|t \cdot x|\!)\left(\langle \gamma,\, v \rangle\right) = (\!|t|\!)\left(\gamma\right) \cdot v \text{ if } x \text{ is the last bound variable} \\
&= \left\langle \begin{array}{l} i_{\to}\left(q \mapsto \mathsf{tmEl}_{K;r}\left(\sigma,\, \left\langle \gamma_\rho,\, q \right\rangle,\, (\!|m \cdot x|\!)\left(\left\langle \gamma_\rho,\, q \right\rangle\right)\right)\right), \\ i_{\to}\left(q \mapsto \mathsf{tmEl}_{K;r'}\left(\sigma,\, \left\langle \delta_\rho,\, q \right\rangle,\, (\!|m \cdot x|\!)\left(\left\langle \delta_\rho,\, q \right\rangle\right)\right)\right) \end{array} \right\rangle \\
&\quad \text{By (i) and (ii)} \\
&\in [\![\Gamma,\, \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\left(m,\, x{:}\tau \to \sigma\right) {:}\mathsf{Ty}]\!] \left(\left\langle \langle \gamma,\, \langle P,\, P_\perp(\gamma) \rangle \rangle,\, \overline{v}^{\,j-1} \right\rangle \right)
\end{aligned}
$$

- $\tau = \forall x{:}\kappa.\, \tau$

First, we notice that for any $Q \in [\![\Gamma,\ \Delta(j-1) \vdash \kappa\text{:Kd}]\!]\left(\left\langle\langle\gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1}\right\rangle\right)$, $Q_\perp \in [\![\Gamma,\ \Delta(j-1) \vdash \kappa\text{:Kd}]\!]\left(\left\langle\langle\gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1}\right\rangle\right)$, by IH

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\tau,\ \rho_{\Gamma,\Delta(j-1),x:_\kappa\kappa}\left(\left\langle\langle\gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1}\right\rangle,\ \langle Q,\ Q_\perp\rangle\right\rangle\right),\ (\!|m \cdot x|\!)\ (\gamma_\rho)\right), \\ \mathsf{tmEl}_{K;r'}\left(\tau,\ \rho_{\Gamma,\Delta(j-1),x:_\kappa\kappa}\left(\left\langle\langle\delta,\ \langle P,\ P_\perp(\delta)\rangle\rangle,\ \overline{w}^{\,j-1}\right\rangle,\ \langle Q,\ Q_\perp\rangle\right\rangle\right),\ (\!|m \cdot x|\!)\ (\delta_\rho)\right) \end{array} \right\rangle$$
$$\in [\![\Gamma,\ \Delta(j-1),\ x:_\kappa\kappa \vdash \mathsf{El}_{K;\rho}\ (m \cdot x,\ \tau)\text{:Ty}]\!]\left(\left\langle\langle\gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1}\right\rangle,\ \langle Q,\ Q_\perp\rangle\right\rangle\right)$$

$$\begin{aligned} &\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\forall x{:}\kappa.\ \tau,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)\right), \\ \mathsf{tmEl}_{K;r'}\left(\forall x{:}\kappa.\ \tau,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)\right) \end{array} \right\rangle \\ =&\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\tau,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)\right), \\ \mathsf{tmEl}_{K;r'}\left(\tau,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)\right) \end{array} \right\rangle \\ &\rho_{\Gamma,x:_\kappa\kappa}\left(\langle\gamma,\ x\rangle\right) = \rho_\Gamma\left(\gamma\right) \\ =&\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\tau,\ \gamma_\rho,\ (\!|m \cdot x|\!)\ (\gamma_\rho)\right), \\ \mathsf{tmEl}_{K;r'}\left(\tau,\ \delta_\rho,\ (\!|m \cdot x|\!)\ (\delta_\rho)\right) \end{array} \right\rangle \\ &\text{By the first observation} \\ \in\ &[\![\Gamma,\ \Delta(j-1) \vdash \forall x{:}\kappa.\ \mathsf{El}_{K;\rho}\ (m \cdot x,\ \tau)\text{:Ty}]\!]\left(\left\langle\langle\gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1}\right\rangle\right) \\ =\ &[\![\Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ \forall x{:}\kappa.\ \tau)\text{:Ty}]\!]\left(\left\langle\langle\gamma,\ \langle P,\ P_\perp(\gamma)\rangle\rangle,\ \overline{v}^{\,j-1}\right\rangle\right) \end{aligned}$$

□

Proof of Lemma E.2. By induction on the structure of τ:

- If $\tau = K \cdot \overline{t}^{\,k}$,

$$\begin{aligned} &\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(K \cdot \overline{t}^{\,k},\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)\right), \\ \mathsf{tmEl}_{K;r'}\left(K \cdot \overline{t}^{\,k},\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)\right) \end{array} \right\rangle \\ =&\left\langle \begin{array}{l} r \cdot \overline{(\!|t|\!)\ (\gamma_\rho)}^{\,k} \cdot (\!|m|\!)\ (\gamma_\rho), \\ r' \cdot \overline{(\!|t|\!)\ (\delta_\rho)}^{\,k} \cdot (\!|m|\!)\ (\delta_\rho) \end{array} \right\rangle \end{aligned}$$

  By Theorem 5.6 (ii) , since γ_ρ, δ_ρ are related, the index terms give a related pair $\left\langle \overline{(\!|t|\!)\ (\gamma_\rho)}^{\,k},\ \overline{(\!|t|\!)\ (\delta_\rho)}^{\,k} \right\rangle$, and, as $m : K \cdot \overline{t}^{\,k}$,

  $\left\langle (\!|m|\!)\ (\gamma_\rho),\ (\!|m|\!)\ (\delta_\rho) \right\rangle \in P_\perp(\gamma) \cdot \overline{(\!|t|\!)\ (\gamma_\rho)}^{\,k}$; the realizer hypothesis yields

  $\in [\![\Gamma,\ \overline{x:_\tau\tau}^{\,k},\ y:_\tau K \cdot \overline{x}^{\,k} \vdash \rho\text{:Ty}]\!]\ (\langle\gamma,\ P_\perp(\gamma)\rangle)$

  Since $[\![-]\!]$ does not depend on the term assignment

  $= [\![\Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ K \cdot \overline{t}^{\,k})\text{:Ty}]\!]\ (\langle\langle\gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$

- If $\tau \in \{\mathcal{B}, \tau \cdot \overline{t}^{\,n}\}$,

$$\begin{aligned} &\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\left(\tau,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)\right), \\ \mathsf{tmEl}_{K;r'}\left(\tau,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)\right) \end{array} \right\rangle \\ =&\left\langle \star,\ \star \right\rangle \end{aligned}$$

$$\in \llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{Unit{:}Ty} \rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$$
$$= \left\llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ \tau)\ {:}\mathsf{Ty} \right\rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$$

- If $\tau = \{x{:}\tau \mid \phi\}$,

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\{x{:}\tau \mid \phi\},\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (\{x{:}\tau \mid \phi\},\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)) \end{array} \right\rangle$$
$$= \left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\tau,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (\tau,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)) \end{array} \right\rangle$$

By I. H.

$$\in \left\llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ \tau)\ {:}\mathsf{Ty} \right\rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$$
$$= \left\llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ \{x{:}\tau \mid \phi\})\ {:}\mathsf{Ty} \right\rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$$

- If $\tau = x{:}\tau \to \sigma$, fixed $\langle q_\perp,\ n_\perp\rangle \in \llbracket \Gamma,\ \Delta(j-1) \vdash \tau{:}\mathsf{Ty} \rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$ by I. H.

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\sigma,\ \langle \gamma_\rho,\ q_\perp\rangle,\ (\!|m \cdot x|\!)\ (\langle \gamma_\rho,\ q_\perp\rangle)), \\ \mathsf{tmEl}_{K;r'}\ (\sigma,\ \langle \delta_\rho,\ n_\perp\rangle,\ (\!|m \cdot x|\!)\ (\langle \delta_\rho,\ n_\perp\rangle)) \end{array} \right\rangle$$
$$\in \left\llbracket \Gamma,\ \Delta(j-1),\ x{:}_{\mathsf{T}}\tau \vdash \mathsf{El}_{K;\rho}\ (m \cdot x,\ \sigma)\ {:}\mathsf{Ty} \right\rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$$

Hence,

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (x{:}\tau \to \sigma,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (x{:}\tau \to \sigma,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)) \end{array} \right\rangle$$
$$= \left\langle \begin{array}{l} i_\to\ (q \mapsto \mathsf{tmEl}_{K;r}\ (\sigma,\ \langle \gamma_\rho,\ q\rangle,\ (\!|m|\!)\ (\gamma_\rho) \cdot q)), \\ i_\to\ (q \mapsto \mathsf{tmEl}_{K;r'}\ (\sigma,\ \langle \delta_\rho,\ q\rangle,\ (\!|m|\!)\ (\delta_\rho) \cdot q)) \end{array} \right\rangle$$

We notice that $(\!|t \cdot x|\!)\ (\langle \gamma,\ v\rangle) = (\!|t|\!)\ (\gamma) \cdot v$ if x is the last bound variable

$$= \left\langle \begin{array}{l} i_\to\ (q \mapsto \mathsf{tmEl}_{K;r}\ (\sigma,\ \langle \gamma_\rho,\ q\rangle,\ (\!|m \cdot x|\!)\ (\langle \gamma_\rho,\ q\rangle))), \\ i_\to\ (q \mapsto \mathsf{tmEl}_{K;r'}\ (\sigma,\ \langle \delta_\rho,\ q\rangle,\ (\!|m \cdot x|\!)\ (\langle \delta_\rho,\ q\rangle))) \end{array} \right\rangle$$

By the above

$$\in \left\llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ x{:}\tau \to \sigma)\ {:}\mathsf{Ty} \right\rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$$

- $\tau = \forall x{:}\kappa.\ \tau$

First, we notice that for any $Q \in \llbracket \Gamma,\ \Delta(j-1) \vdash \kappa{:}\mathsf{Kd} \rrbracket\ (\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle)$, since the runtime projection does not depend on the newly bound kind variable, by IH

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\tau,\ \gamma_\rho,\ (\!|m \cdot x|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (\tau,\ \delta_\rho,\ (\!|m \cdot x|\!)\ (\delta_\rho)) \end{array} \right\rangle$$
$$\in \left\llbracket \Gamma,\ \Delta(j-1),\ x{:}_\kappa\kappa \vdash \mathsf{El}_{K;\rho}\ (m \cdot x,\ \tau)\ {:}\mathsf{Ty} \right\rrbracket\ (\langle\langle\langle \gamma,\ P_\perp(\gamma)\rangle,\ \epsilon\rangle,\ Q\rangle)$$

$$\left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\forall x{:}\kappa.\ \tau,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (\forall x{:}\kappa.\ \tau,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)) \end{array} \right\rangle$$
$$= \left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\tau,\ \gamma_\rho,\ (\!|m|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (\tau,\ \delta_\rho,\ (\!|m|\!)\ (\delta_\rho)) \end{array} \right\rangle$$
$$\rho_{\Gamma,\,x:_\kappa\kappa}\ (\langle \gamma,\ x\rangle) = \rho_\Gamma\ (\gamma)$$
$$= \left\langle \begin{array}{l} \mathsf{tmEl}_{K;r}\ (\tau,\ \gamma_\rho,\ (\!|m \cdot x|\!)\ (\gamma_\rho)), \\ \mathsf{tmEl}_{K;r'}\ (\tau,\ \delta_\rho,\ (\!|m \cdot x|\!)\ (\delta_\rho)) \end{array} \right\rangle$$

By the first observation

$$\in \llbracket \Gamma,\ \Delta(j-1) \vdash \forall x{:}\kappa.\ \mathsf{El}_{K;\rho}\ (m \cdot x,\ \tau) : \mathsf{Ty} \rrbracket\ (\langle \langle \gamma,\ P_\perp(\gamma) \rangle,\ \epsilon \rangle)$$
$$= \llbracket \Gamma,\ \Delta(j-1) \vdash \mathsf{El}_{K;\rho}\ (m,\ \forall x{:}\kappa.\ \tau) : \mathsf{Ty} \rrbracket\ (\langle \langle \gamma,\ P_\perp(\gamma) \rangle,\ \epsilon \rangle)$$

□