A Framework for Internalizing Relations into Type Theory

Peng Fu, Aaron Stump, Jeffrey Vaughan

A Framework for Internalizing Relations into Type Theory

Peng Fu[1], Aaron Stump[1], and Jeff Vaughan[2]

[1] Computer Science, The University of Iowa
[2] Computer Science, University of California, Los Angeles

Abstract. This paper introduces the concept of *internalization structure*, which can be used to incorporate certain relations into $\mathbf{F}^{\Pi}$, a variant of system $\mathbf{F}$, while maintaining termination of the new system. We will call this process of incorporation *internalization*, $\mathbf{F}^{\Pi}$ the *base system* and the new system after the incorporation the *internalized system*. We first specify the syntax, and then the semantics of $\mathbf{F}^{\Pi}$ via the Tait-Girard reducibility method. We then define internalization structure. We show that we can obtain a terminating internalized system from an internalization structure. Finally, as motivating examples, we demonstrate how our framework can be applied to internalize subtyping, full-beta term equality and term-type inhabitation relations.

1 Introduction

Type systems, including the Calculus of Inductive Constructions [7] and $\mathbf{F}_{<:}$ [13], incorporate auxiliary judgments in their typing relations. For example, the subtype judgment is a premise of $\mathbf{F}_{<:}$'s subsumption rule:

$$\frac{\Gamma \vdash t : T \quad T <: T'}{\Gamma \vdash t : T'} \; sub$$

Likewise, the type-equivalence judgment appears in CIC's conversion rule:

$$\frac{\Gamma \vdash t : T \quad T \equiv T'}{\Gamma \vdash t : T'} \; conv$$

Conventionally, adding a rule like *sub* or *conv* to a lambda calculus requires ad-hoc reasoning about the auxiliary judgment system and its relation to the underlying type theory. This paper proposes a new framework for defining language extensions in the style of *sub* or *conv*, and shows how to systematically verify that such extensions yield terminationing systems under call-by-name reduction.

We call deduction systems producing auxiliary judgments *metasystems*, and refer to typing rules that modify types based such metasystem judgments *automatic conversion rules*. We will also consider cut-down type systems without automatic conversion rules; these are called base systems.

For instance the subtyping ($<:$) and type-equivalence ($\equiv$) derivation systems are metasystems, and rules *sub* and *conv* are automatic conversion rules. System **F** is $\mathbf{F}_{<:}$'s base system. Automatic conversion rules may be viewed as a bridge between a base system and a metasystem; metasystem derivations serve as evidence that type conversion or subsumption should be allowed.

Type structure may be used to reflect metasystem judgments. Indeed this has been done in several languages: equality sets in Martin-Löf type theory enable reasoning about equality relations [11], Sjöberg and Stump's $T^{\mathtt{vec}}$ uses types to reflect call-by-value term equality in the presence of divergence [16], and the AuraConf language uses proofs of type e **isa** t to indicate expression e may be cast to type t [18]. As we will see, the language extension framework proposed in this paper is implemented using a generalization of these ideas.

The Curry-Howard isomorphism enable us to view the base system as a kind of constructive logic. Once we internalized metasystem judgments as types, we can reason about these judgments using the logic provided by the base system. Moreover, using features of the internalized system we can derive new, admissible judgments that may not derivable in either the base system or the metasystem alone. Later we will see how the internalized system makes this possible.

This paper introduces a new framework, internalization, that enables the systemic definition and investigation of λ-calculi with automatic conversion rules. We use a dependent variant of polymorphic lambda calculus $\mathbf{F}^{\Pi}$ as a base system (Section 2). We then show how to create generalizations of $\mathbf{F}^{\Pi}$—internalized systems—given a compact internalization structure (Section 3). A *sound* internalization structure yields a terminating internalized system (Section 4). This framework is powerful enough to augment $\mathbf{F}^{\Pi}$ with interesting features, including subtyping, conversion based on full-beta term joinability, and a reflected typing relation (Section 5). Finally this paper compares internalization with several related ideas from type theory (Section 6).

An internalization structure is a triple $\langle D, E, \mathcal{I} \rangle$. *Reflective relational sentences* D define the syntax of metasystem propositions and identify valid metasystem judgments. *Elimination relation* E defines automatic conversion rules based on judgments from D. Finally, *interpretation* $\mathcal{I}$ defines semantics for reflective relational sentences as relations over sets of terms. All internalization structures require that D and E are *sound*. As a central result of our work, we show that any sound internalized system constructed from an internalization structure is guaranteed to be terminating.

The contributions of this work are as follows. We define internalization, a framework that enables the systematic study of automatic conversion rules and metasystems relative to a base system. We instantiate this framework with respect to system $\mathbf{F}^{\Pi}$, and prove that *every* internalization structure yields a new, terminating lambda calculus. Finally we demonstrate the utility of this methodology with examples.

2 Base system $\mathbf{F}^{\Pi}$

Internalization builds off of base system $\mathbf{F}^{\Pi}$, a variant of system $\mathbf{F}$. $\mathbf{F}^{\Pi}$'s syntax and operational semantics are given in Figure 1. We use call-by-name operational semantics. Key differences between $\mathbf{F}$ and $\mathbf{F}^{\Pi}$ are as follows:

1. $\mathbf{F}^{\Pi}$ is parametrized by a finite set B of constant types and it contains a constant term **axiom**.
2. **Values** is extended by including constant terms.
3. $\mathbf{F}^{\Pi}$ uses dependent product Π instead of arrow $\rightarrow$ as the function type constructor.

A word about the use of call-by-name reduction is warranted. The main result of this paper is normalization for systems derived by internalization from the base system $\mathbf{F}^{\Pi}$. Strong normalization does not hold for all such systems, as we show by example in Section 5.1. So (weak) normalization is all that we can obtain. An interesting result of our investigation into internalization is that normalization with respect to call-by-name reduction imposes fewer requirements on internalization structures than with call-by-value reduction. Specifically, the λ-abstraction case of the proof of Theorem 1 goes through more directly using call-by-name reduction; with call-by-value reduction, dependent typing imposes additional requirements on the internalization structure.

For $\mathbf{F}^{\Pi}$, we use dependent product instead of arrow anticipating the use of internalization structures whose types mention terms. $\mathbf{F}^{\Pi}$ contains constant **axiom** but does not give it a type; later, internalized systems will use **axiom** to inhabit special types. Moreover, defining **axiom** now allows us to fix a single universe of reducibility candidates and to use $\mathbf{F}^{\Pi}$ type interpretations directly when interpreting these special internalized-system types.

2.1 Syntax of $\mathbf{F}^{\Pi}$

Figure 2 gives the type assignment rules for $\mathbf{F}^{\Pi}$. The definition of typing context is standard. We define the judgment $\Gamma \vdash \mathbf{OK}$ for well-formedness of context Γ as follows:

$$\frac{}{\cdot \vdash \mathbf{OK}} \qquad \frac{\Gamma \vdash \mathbf{OK}}{\Gamma, X \vdash \mathbf{OK}} \qquad \frac{\Gamma \vdash \mathbf{OK} \qquad \mathit{FVar}(T) \subseteq \mathit{dom}(\Gamma)}{\Gamma, x : T \vdash \mathbf{OK}}$$

$\mathit{FVar}(T)$ means the set of free type variables and free term variables in type T. $\mathit{dom}(\Gamma)$ means the domain of the context, i.e., $e \in \mathit{dom}(\Gamma)$ iff e is either a type variable such that $\Gamma \equiv \Gamma_1, e, \Gamma_2$, or a term variable such that $\Gamma \equiv \Gamma_1, e : T, \Gamma_2$.

2.2 Interpretation of Types in $\mathbf{F}^{\Pi}$

Reducibility is a well-known technique for proving the normalization of type systems such as $\mathbf{F}$. In this paper, we use it to interpret $\mathbf{F}^{\Pi}$'s types. Reducibility will both provide intuition for $\mathbf{F}^{\Pi}$'s semantics and yield a normalization result. To begin, we define *reducibility candidates* following Girard's *Proofs and Types* [9].

$$
\begin{array}{lll}
\textbf{Types} & T & ::= \ B \mid X \mid \Pi x : T.T \mid \forall X.T \\
\textbf{Terms} & t, u & ::= \textbf{axiom} \mid x \mid (t\ t) \mid \lambda x.t \\
\textbf{Contexts} & \mathscr{C} & ::= [] \mid \mathscr{C}\ t \\
\textbf{Values} & v & ::= \lambda x.t \mid \textbf{axiom}
\end{array}
$$

$$\mathscr{C}[(\lambda x.t)\ t'] \rightsquigarrow \mathscr{C}[[t'/x]t]$$

Fig. 1. Syntax and Operational Semantics of $\mathbf{F}^{\Pi}$

$$\frac{(x:T) \in \Gamma \quad \Gamma \vdash \textbf{OK}}{\Gamma \vdash x : T}\ \mathit{Var} \qquad \frac{\Gamma, x : T_1 \ \vdash t : T_2}{\Gamma \vdash \lambda x.t : \Pi x : T_1.T_2}\ \Pi\text{-}\mathit{intro}$$

$$\frac{\Gamma \vdash t_1 : \Pi x : T_1.T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : [t_2/x]T_2}\ \Pi\text{-}\mathit{elim} \qquad \frac{\Gamma, X \vdash t : T}{\Gamma \vdash t : \forall X.T}\ \forall\text{-}\mathit{intro}$$

$$\frac{\Gamma \vdash t : \forall X.T \quad FVar(T') \subseteq dom(\Gamma)}{\Gamma \vdash t : [T'/X]T}\ \forall\text{-}\mathit{elim}$$

Fig. 2. Type Assignment System of $\mathbf{F}^{\Pi}$

Definition 1. *A reducibility candidate $\mathcal{R}$ is a set of terms that satisfies the following conditions:*
CR 1 *If $t \in \mathcal{R}$,then $t \in \mathcal{V}$, where $\mathcal{V}$ is the set of closed terms that that reduces to a value in* **Values**.
CR 2 *If $t \in \mathcal{R}$ and $t \rightsquigarrow t'$, then $t' \in \mathcal{R}$.*
CR 3 *If t is a closed term, $t \rightsquigarrow t'$ and $t' \in \mathcal{R}$, then $t \in \mathcal{R}$.*

Definition 2. *Let $\Re$ be the set of all reducibility candidates. Let $TVar$ be the set of all type variables. Let ϕ be a finite function with $dom(\phi) \subseteq TVar$ and $range(\phi) \subseteq \Re$. If $dom(\phi) = \{X_1, X_2, ...X_n\}$, then we usually write ϕ as $[\mathcal{R}_1/X_1, ...\mathcal{R}_n/X_n]$.*

Figure 3 defines the interpretation $[\![T]\!]_\phi$ of a type T as a set of terms. Note that constant types B and their interpretations $\mathcal{R}_B$ are left unspecified; these may be filled in later. For any $[\![T]\!]_\phi$, let $FV(T)$ be the set of free type variable in T.we assume $FV(T) \subseteq dom(\phi)$.

$$
\begin{array}{ll}
t \in [\![B]\!]_\phi & \text{iff } t \in \mathcal{R}_B, \text{ where } \mathcal{R}_B \in \Re \\
t \in [\![X]\!]_\phi & \text{iff } t \in \phi(X) \\
t \in [\![\Pi x : T_1.T_2]\!]_\phi & \text{iff } t \in \mathcal{V} \text{ and } (\forall u \in [\![T_1]\!]_\phi \Rightarrow (t\ u) \in [\![[u/x]T_2]\!]_\phi) \\
t \in [\![\forall X.T]\!]_\phi & \text{iff } \forall \mathcal{R} \in \Re, t \in [\![T]\!]_{\phi[\mathcal{R}/X]}
\end{array}
$$

Fig. 3. Interpretation of Types in $\mathbf{F}^{\Pi}$

2.3 Type Soundness

The theorem below shows that any typable closed term is normalizing, and can be shown in a standard way using Tait-Girard reducibility (cf. [9]). Several properties of the interpretation of types are required, which can all be proved by induction on the structure of types in $\mathbf{F}^{\Pi}$. The proofs of the lemmas and theorems below are relegated to the appendix.

Lemma 1. $[\![T]\!]_\phi \in \Re$, *in the other words, the interpretation of a type is indeed a reducibility candidate.*

Lemma 2. *Let Sub be the set of all capture avoiding term-level substitutions with a domain of term variables and a range of terms that are in* $\mathcal{V}$. $\forall \sigma \in Sub, [\![\sigma T]\!]_\phi = [\![T]\!]_\phi$.

Since $\mathbf{F}^{\Pi}$ essentially is system $\mathbf{F}$, it does not contain terms in the types, we have $\sigma T \equiv T$ in $\mathbf{F}^{\Pi}$, thus this lemma is true.

Lemma 3 (Substitution lemma). $[\![[T'/X]T]\!]_\phi = [\![T]\!]_{\phi[[\![T']\!]_\phi/X]}$.

Definition 3. *We define the set* $[\Gamma]$ *of well-typed substitutions* (σ, δ) *w.r.t.* Γ *as follows:*

$$\frac{}{(\emptyset,\emptyset) \in [.]} \quad \frac{(\sigma,\delta) \in [\Gamma] \quad \mathcal{R} \in \Re}{(\sigma, \delta \cup \{(X,\mathcal{R})\}) \in [\Gamma, X]} \quad \frac{(\sigma,\delta) \in [\Gamma] \quad t \in [\![\sigma T]\!]_\delta}{(\sigma \cup \{(x,t)\}, \delta) \in [\Gamma, x:T]}$$

Theorem 1 (Type Soundness). *If* $\Gamma \vdash t : T$, *then* $\forall (\sigma,\delta) \in [\Gamma], (\sigma\ t) \in [\![\sigma T]\!]_\delta$.

3 Internalized Structure

Internalization is based on internalization structure. The internalization structure contains the information of how to construct reflective relational sentences and how these reflective relational sentences interact with the base system. It also gives the meaning of the reflective relational sentences through the interpretation of types in the base system. Once we define an internalization structure, we can then begin the process of internalization by first internalizing the reflective relational sentences as types, then add two new typing rules to deal with these reflective relational sentences.

Internalization structure consists of three parts, they are reflective relational sentences, elimination relation and interpretation. Besides these three parts, an internalization structure has two soundness properties, which we will identify later. In this section, we will first describe these three parts of the internalization structure and the two soundness properties of internalization structure. Then we will illustrate how to construct a new system, which is called internalized system, from an internalization structure. The internalized system is shown to be terminating.

3.1 Reflective Relational Sentence-D

We define the kind of judgments or relations that could be integrated into the base system. Essentially these are the relations on the terms and types from the base system.

Definition 4. *Let signature* $\Sigma \subseteq$ **Symbols** $\times \mathbb{N} \times \mathbb{N}$*, where* **Symbols** *means a set of relation symbols, and* $\mathbb{N}$ *is the set of natural numbers.* $R^{n\times m} \in \Sigma$ *means* $R \in$ **Symbols** *and the arity of* R *is* $n + m$*.*

Definition 5. *A relational sentence on the basic system is a syntactic object of form* $R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m)$*, where* t, T *are defined in* $\boldsymbol{F}^{\Pi}$ *and* $R^{(n\times m)} \in \Sigma$*.*

Note that in this paper we do not deal with nesting relational sentences.

Definition 6. *Let* $\mathfrak{A}$ *be the set of all relational sentences. A set of reflective relational sentences* D *is a subset of all relational sentences, i.e.* $D \subseteq \mathfrak{A}$*.*

Reflective relational sentences are used to formalize a metasystem's derivable judgments. When we define specifically how to recognize the reflective relational sentences from relational sentences, we obtain a kind of metasystem. This metasystem need not be recursive; it can be defined axiomatically.

3.2 Elimination Relation-E

An elimination relation is a syntactic constraint used to specify how the metasystem influences the base system. We will appeal to an elimination relation when we add the elimination rule to the base system for the reflective relational sentences. Since the elimination relation is used after internalizing reflective relational sentences as types, we need to extend the definition of types and the context accordingly.

Definition 7. *We define extended types and extended contexts as follows:*

$$\begin{array}{ll} \textbf{RTypes} & A ::= R_1^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m) \mid ... \mid R_l^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m) \\ \textbf{ETypes} & S ::= B \mid X \mid \Pi x : S.S \mid \forall X.S \mid A \\ \textbf{EContext} & \Delta ::= \cdot \mid \Delta, x : S \mid \Delta, X \end{array}$$

We can see that the extended types and the extended context defined above are really extended in the sense of adding the *relational sentences* as new types.

Definition 8. *We specify an elimination relation* E *by:*
$E \subseteq$ **EContext** $\times$ **Terms** $\times$ **Terms** $\times \mathfrak{A} \times$ **ETypes** $\times$ **ETypes**.

For example, when we consider the specific internalization structure for subtyping below, we will define an elimination relation where $(\Delta, t, t', T < T', T, T') \in E$ holds iff in extended context Δ, t has the type T, t' has the type $T < T'$, and we can change the type of t to T'.

3.3 Interpretation-$\mathcal{I}$

We defined the interpretation of types of $\mathbf{F}^{\Pi}$ before. Since interpretation of types is a set of terms and the reflective relational sentences are relations about between terms and types in $\mathbf{F}^{\Pi}$, it is natural to understand the meaning of these reflective relational sentences as set-theoretic relations between interpretation of types. Take subtyping as an example; we interpret subtype judgment $<:$ as the mathematical subset relation $\subseteq$ on interpretation of types [14].

Interpretation-$\mathcal{I}$ is defined to capture this intuition. Later we will relate interpretation-$\mathcal{I}$ to reflective relational sentences and elimination relation through two soundness properties.

Definition 9. *Let $\Re$ be the set of all reducibility candidates as defined in $\boldsymbol{F}^{\Pi}$. We define an interpretation of $R^{(n\times m)}$–$\mathcal{I}_{R^{(n\times m)}}$ to be $\mathcal{I}_{R^{(n\times m)}} \subseteq \mathbf{Terms}^n \times \Re^m$.*

3.4 Soundness Properties

Now that we have defined all parts of an internalization structure, we can formulate two soundness properties for an internalization structure. Since one of the soundness properties is related to the extended types, we first define the interpretation for extended types. Then we identify the soundness properties.

Definition 10. *Let ϕ be an environment function w.r.t. type S, which is defined in the same way as definition 2 except we extend it to type S. Let $\mathcal{A}$ be the set of closed terms that normalize at* **axiom**. *The interpretation of types $[\![S]\!]_\phi$ is defined inductively as follows:*

- $t \in [\![B]\!]_\phi$ *iff* $t \in \mathcal{R}_B$.
- $t \in [\![X]\!]_\phi$ *iff* $t \in \phi(X)$.
- $t \in [\![\Pi x : S_1.S_2]\!]_\phi$ *iff* $t \in \mathcal{V}$ *and* $(\forall u \in [\![S_1]\!]_\phi \Rightarrow (t\ u) \in [\![[u/x]S_2]\!]_\phi)$.
- $t \in [\![\forall X.S]\!]_\phi$ *iff* $\forall \mathcal{R} \in \Re, t \in [\![S]\!]_{\phi[\mathcal{R}/X]}$.
- $t \in [\![R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m)]\!]_\phi$ *iff* $t \in \mathcal{A}$ *and* $(t_1, ..., t_n, [\![T_1]\!]_\phi, ..., [\![T_m]\!]_\phi) \in \mathcal{I}_{R^{(n\times m)}}$.

Define $(\sigma, \delta) \in [\Delta]$ in the same way as $(\sigma, \delta) \in [\Gamma]$, except with extended contexts and extended types.

Definition 11. *We say a tuple $\langle D, E, \mathcal{I}\rangle$ is an internalization structure iff it satisfies the following soundness properties:*

- *Soundness of reflective relational sentences*
  If $R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m) \in D$, then
  $\forall \phi, \forall \sigma \in Sub, (\sigma t_1, ..., \sigma t_n, [\![\sigma T_1]\!]_\phi, ..., [\![\sigma T_m]\!]_\phi) \in \mathcal{I}_{R^{(n\times m)}}$.
- *Soundness of the elimination relation*
  Suppose $(\Delta, t, t', R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m), S, S') \in E$, and $(\sigma, \delta) \in [\Delta]$. Also, suppose $\sigma(t) \in [\![\sigma S]\!]_\delta$ and $R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m) \in D$. Then $\sigma(t) \in [\![\sigma S']\!]_\delta$.

Soundness of reflective relational sentences means that the reflective relational sentences are a conservative approximation of interpretation-$\mathcal{I}$. *Soundness of the elimination relation* will imply that the elimination rule for internalized systems respects the Girard-Tait type interpretation and is semantically compatible with substitutions that arise duing CBN evaluation. We will see that in next section. We can extend the lemmas from Section 2.3 above to our extended interpretation of types:

Lemma 4. $\forall \phi, [\![S]\!]_\phi \in \Re$.

Lemma 5 (Substitution lemma). *If* $[S'/X]S \in \mathbf{ETypes}$*, then* $[\![[S'/X]S]\!]_\phi = [\![S]\!]_{\phi[[\![S']\!]_\phi/X]}$.

4 Internalized System

So far, we defined the internalization structure–$\langle D, E, \mathcal{I}\rangle$. Now using an internalization structure, we can construct a new system–we call it the internalized system–from the internalization structure and $\mathbf{F}^{\Pi}$. The term syntax and operational semantics of internalized system are the same as $\mathbf{F}^{\Pi}$, while the syntax of types and contexts are the $\mathbf{RTypes}, \mathbf{ETypes}, \mathbf{EContexts}$ in definition 7. The well-formed extended context $\Delta \vdash \mathbf{OK}$ is defined just as before except using $\mathbf{EContexts}$. Figure 4 shows the new type assignment rules for the internalized system. We can see that the new type assignment system contains two new rules:A-intro and A-elim. The A-intro rule is used to introduce reflective relational sentences as types in the internalized system, while the A-elim rule is for using the reflective relational sentences to change the type of a term accordingly.

$$\frac{A \in D \quad FVar(A) \subseteq dom(\Delta) \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash \mathbf{axiom} : A}\ A\text{-}intro \qquad \frac{\Delta(x) = S \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash x : S}\ Var$$

$$\frac{\Delta \vdash t : T \quad \Delta \vdash t' : A \quad E(\Delta, t, t', A, T, T')}{\Delta \vdash t : T'}\ A\text{-}elim \qquad \frac{\Delta, x : S_1 \ \vdash t : S_2}{\Delta \vdash \lambda x.t : \Pi x : S_1.S_2}\ \Pi\_intro$$

$$\frac{\Delta \vdash t_1 : \Pi x : S_1.S_2 \quad \Delta \vdash t_2 : S_1}{\Delta \vdash t_1\ t_2 : [t_2/x]S_2}\ \Pi\_elim \qquad \frac{\Delta, X \vdash t : S}{\Delta \vdash t : \forall X.S}\ \forall\_intro$$

$$\frac{\Delta \vdash t : \forall X.S \quad [S'/X]S \in \mathbf{ETypes} \quad FVar(S') \subseteq dom(\Delta)}{\Delta \vdash t : [S'/X]S}\ \forall\_elim$$

Fig. 4. Type Assignment System of Internalized System

The theorem below guarantees that the internalized system generated from $\mathbf{F}^{\Pi}$ and internalization structure is *terminating*, which is the central result of internalization.

Theorem 2 (Type Soundness). *If $\langle D, E, \mathcal{I}\rangle$ is an internalization structure and $\Delta \vdash t : S$, then $\forall(\sigma, \delta) \in [\Delta], (\sigma\ t) \in [\![\sigma S]\!]_\delta$.*

The proof of this theorem is in appendix.

Corollary 1. *If $\cdot \vdash t : S$, then $t \in \mathcal{V}$.*

Because typing contexts may introduce spurious assumptions, some open contexts may assign a type to a diverging term. Section 5.1 shows gives an example. This is an expected outcome of reasoning from invalid premises. Indeed Corollary 1 may be strengthened to allow contexts where all variables are classified by inhabited types.

5 Examples

In previous section, we capsule our development of internalized system as constructing a sound internalization structure. Now let us see how we can apply our formalization of internalization to internalize subtyping, full-beta term equality and term-type inhabitation relations as types. First, we specify an instance of $\mathbf{F}^{\Pi}$. Essentially, we instantiate constant types as $B ::= \top \mid \bot$. Additionally, we define $[\![\bot]\!]_\phi := \emptyset, [\![\top]\!]_\phi := \mathcal{V}$.

Recall that internalization works as follows. We first define the set of reflective relational sentences that contains all the derivable judgments from subtyping, full-beta term equality and term-type inhabitation. Then we define the elimination relation and interpretation. We show our definition interpretation structure is sound. Finally we present the internalized system as the result of internalization. We will follow this recipe in the sequel.

5.1 Subtyping

We need to instantiate the three parts of internalization structure-$\langle D, E, I\rangle$. First, we specify $\Sigma := \{<^{0+2}\}$. Then we know all the reflective relational sentences should be in the form $T_1 < T_2$. We identify reflective relational sentences D as follows:

$$\frac{}{T < \top \in D} \qquad \frac{}{\bot < T \in D}$$

$$\frac{}{X < X \in D} \qquad \frac{T_1 < T_2 \in D}{\forall X.T_1 < \forall X.T_2 \in D}$$

$$\frac{T_1' < T_1 \in D \quad T_2 < T_2' \in D}{\Pi x : T_1.T_2 < \Pi x : T_1'.T_2' \in D}$$

We can see that the way we identify D is similar to the way we write subtyping rules. Now we define E as follows:

$$(\Delta, t, t', T < T', T, T') \in E$$

The meaning of this elimination relation is that if t has type T in context Δ and t' has type $T < T'$, then t can also has the type T'.

We define interpretation $\mathcal{I}_<$:

$$\mathcal{I}_< := \{(\mathcal{R}_1, \mathcal{R}_2) \mid \mathcal{R}_1 \subseteq \mathcal{R}_2\}$$

We can see that $\mathcal{I}_<$ capture all the subset relations on reducibility candidates. The following two lemmas make sure we obtain a sound internalization structure from $\langle D, E, \mathcal{I}_< \rangle$ we defined above.

Lemma 6 (Soundness of the Reflective Relational Sentence). *If* $(T < T') \in D$*, then* $\forall \sigma \in Sub, \forall \phi, (\llbracket \sigma T \rrbracket_\phi, \llbracket \sigma T' \rrbracket_\phi) \in \mathcal{I}_<$.

Proof. Since $\llbracket \sigma T \rrbracket_\phi = \llbracket T \rrbracket_\phi$ by lemma 2. We just need to show: If $(T < T') \in D$, then $\forall \phi, (\llbracket T \rrbracket_\phi, \llbracket T' \rrbracket_\phi) \in \mathcal{I}_<$. We will prove this by induction on the structure of T.
Case: $T = \top$ or $T = \bot$
By inversion, it holds.
Case: $T = X$
By inversion, we know $T' = X$ *or* $\top$, again, it is the case.
Case: $T = \Pi x : T_1.T_2$
By inversion, $T' = \top$ or $T' = \Pi x : T_1'.T_2'$. Let us consider $T' = \Pi x : T_1'.T_2'$. In this case, by inversion, $T_1' < T_1 \in D, T_2 < T_2' \in D$. By IH, we have $\llbracket T_1' \rrbracket_\phi \subseteq \llbracket T_1 \rrbracket_\phi$. Again, by IH, we have $\llbracket T_2 \rrbracket_\phi \subseteq \llbracket T_2' \rrbracket_\phi$. For any $u \in \llbracket T_1' \rrbracket_\phi \subseteq \llbracket T_1 \rrbracket_\phi$, if $t \in \llbracket \Pi x : T_1.T_2 \rrbracket_\phi$, we have $tu \in \llbracket [u/x]T_2 \rrbracket_\phi = \llbracket T_2 \rrbracket_\phi \subseteq \llbracket T_2' \rrbracket_\phi$. So $t \in \llbracket \Pi x : T_1'.T_2' \rrbracket_\phi$.
Case: $T = \forall X.T$
By inversion, $T' = \top$ or $\forall X.T'$. So let's consider $T' = \forall X.T'$. By inversion, we know $T < T' \in D$. We know for $t \in \llbracket \forall X.T \rrbracket_\phi$, $\forall \mathcal{R} \in \Re, t \in \llbracket T \rrbracket_{\phi[\mathcal{R}/X]}$. By IH, $\llbracket T \rrbracket_{\phi[\mathcal{R}/X]} \subseteq \llbracket T' \rrbracket_{\phi[\mathcal{R}/X]}$. So $t \in \llbracket \forall X.T' \rrbracket_\phi$.

Lemma 7 (Soundness of the Elimination Relation).
If $(\Delta, t, t', T_1 < T_2, T_1, T_2) \in E$, $(\sigma, \delta) \in [\Delta]$ *and* $\sigma(t) \in \llbracket \sigma T_1 \rrbracket_\delta = \llbracket T_1 \rrbracket_\delta$ *and* $T_1 < T_2 \in D$*, then* $\sigma(t) \in \llbracket \sigma T_2 \rrbracket_\delta = \llbracket T_2 \rrbracket_\delta$.

It is the case by *soundness of the reflective relational sentences*.

For this internalization structure–$\langle D, E, \mathcal{I}_< \rangle$, the *A-intro* and *A-elim* rules are equivalent to:

$$\frac{T_1 < T_2 \in D \quad FVar(T_1 < T_2) \subseteq dom(\Delta) \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash \mathbf{axiom} : T_1 < T_2} \; A\text{-}intro$$

$$\frac{\Delta \vdash t : T_1 \quad \Delta \vdash t' : T_1 < T_2}{\Delta \vdash t : T_2} \; A\text{-}elim$$

The subtyping setting also provides a clear illustration that it is possible to have diverging term under open terms and full-beta reduction in internalized system. Here it is possible to derive

$$y : (\top < (\top \to \top)) \vdash (\lambda x.xx)(\lambda x.xx) : \top$$

using the underivable fact $\top < (\top \to \top)$ and derivable $(\top \to \top) < \top$ to establish an isomorphism between types $\top$ and $\top \to \top$. Sticking to closed terms means we need not worry about this derivation directly. And call-by-name evaluation ensures that

$$\cdot \vdash \lambda y.(\lambda x.xx)(\lambda x.xx) : (\top < (\top \to \top)) \to \top$$

does not reduce. In contrast, full reduction would loop.

5.2 Full-beta Term Equality, Term-Type Inhabitation

We can go even further to explore the internalization structure. We add two more relation symbols to signature so that $\Sigma = \{\downarrow^{(2+0)}, <^{(0+2)}, \lhd^{(1+1)}\}$. For simplicity, we usually do not specify the arity. Thus the relational sentences have form: $t_1 \downarrow t_2, T_1 < T_2$, and $t \lhd T$ for base-system t and T.

Now we are ready to specify more reflective relational sentences. We define $\lhd$ reflective relational sentences by the following condition:

$$t \lhd T \in D \text{ iff } \forall \phi, t \in [\![T]\!]_\phi$$

Notice that this definition is not algorithmic, which is fine since our framework does not require decidability for the set D for reflective relational sentences.

The $\lhd$ symbol allows us to give "morally correct" types to terms which cannot otherwise be checked. In practice, such terms are created when extracting computational content from mechanically checked proofs. As a concrete example, the Coq proof assistant uses an expressive language to define functional programs and exports that code to OCaml for efficient compilation. Resulting OCaml programs do not go wrong, but must use `Obj.magic`$:\alpha \to \beta$ to pass ML's weaker type system. Likewise, AuraConf [18] uses a type constructor resembling $\lhd$ to inform the type checker about the concealed types of opaque ciphertexts. Note that weaker variants of $\lhd$ may be possible when, as in the case of extracted proofs, there is a conservative procedure for checking semantic type inclusion, $t \lhd_{alt} T \in D$ iff $Oracle(t, T)$. (We do not consider such variants further.)

We define $t_1 \downarrow t_2 \in D$ by the following rules:

$$\frac{}{t \downarrow t \in D} \qquad \frac{}{(\lambda x.t)t' \downarrow [t'/x]t \in D} \qquad \frac{t_1 \downarrow t_2 \in D}{t_1\ t \downarrow t_2\ t \in D}$$

$$\frac{t_1 \downarrow t_2 \in D}{\lambda x.t_1 \downarrow \lambda x.t_2 \in D} \qquad \frac{t_1 \downarrow t_2 \in D}{t\ t_1 \downarrow t\ t_2 \in D} \qquad \frac{t_1 \downarrow t_2 \in D \quad t_2 \downarrow t_3 \in D}{t_1 \downarrow t_3 \in D}$$

$$\frac{t_1 \downarrow t_2 \in D}{t_2 \downarrow t_1 \in D}$$

The rules above are the same as how we define the conversion in lambda calculus. In this case, the syntax of extended types (as defined by the internalization framework) is:

$$\textbf{EType } S ::= \top \mid \bot \mid X \mid \Pi x : S.S \mid \forall X.S \mid t_1 \downarrow t_2 \mid T_1 < T_2 \mid t \lhd T$$

The additional elimination relations are:

$$(\Delta, t, t', t_1 \downarrow t_2, [t_1/x](t_3 \downarrow t_4), [t_2/x](t_3 \downarrow t_4)) \in E.$$

$$(\Delta, t, t', t \lhd T', T, T') \in E$$

The additional interpretations $\mathcal{I}_\downarrow, \mathcal{I}_\lhd$ are:

- $\mathcal{I}_\downarrow \subseteq \mathbf{Terms} \times \mathbf{Terms}$ defined by $\mathcal{I}_\downarrow := \{(t_1, t_2) \mid t_1 \downarrow t_2 \in D\}$.
- $\mathcal{I}_\lhd \subseteq \mathbf{Terms} \times \Re$ defined by $\mathcal{I}_\lhd := \{(t, \mathcal{R}) \mid t \in \mathcal{R}\}$.

We have now defined the three parts of the internalization structure. We need to show that this structure is sound. For that purpose, we have following lemmas.

Lemma 8 (Soundness of the Reflective Relational Sentence).

- *If* $(t_1 \downarrow t_2) \in D$, *then* $\forall \sigma \in Sub, (\sigma t_1, \sigma t_2) \in \mathcal{I}_\downarrow$.
- *If* $(t \lhd T) \in D$, *then* $\forall \sigma \in Sub, \forall \phi, (\sigma t, [\![\sigma T]\!]_\phi) \in \mathcal{I}_\lhd$.

Proof. If $(t_1 \downarrow t_2) \in D$, we have $\forall \sigma \in Sub, (\sigma t_1, \sigma t_2) \in D$. This is because we define the $t \downarrow t'$ relation same as the conversion in lambda calculus and this is one of its properties. (see [4]) Thus $(\sigma t_1, \sigma t_2) \in \mathcal{I}_\downarrow$ by definition of $\mathcal{I}_\downarrow$.

If $(t \lhd T) \in D$, by definition, we have $\forall \phi, t \in [\![T]\!]_\phi$. Since t is closed, $\forall \sigma \in Sub, \sigma t \equiv t$. And we have $[\![\sigma T]\!]_\phi = [\![T]\!]_\phi$. So $\forall \phi, \forall \sigma \in Sub, \sigma t \in [\![\sigma T]\!]_\phi$. Thus $\forall \sigma \in Sub, \forall \phi, (\sigma t, [\![\sigma T]\!]_\phi) \in \mathcal{I}_\lhd$.

Lemma 9 (Soundness of the Elimination Relation).

- *If* $(\Delta, t, t', t_1 \downarrow t_2, [t_1/x](t_3 \downarrow t_4), [t_2/x](t_3 \downarrow t_4)) \in E$, $(\sigma, \delta) \in [\Delta]$ *and* $\sigma(t) \in [\![\sigma[t_1/x](t_3 \downarrow t_4)]\!]_\delta$ *and* $t_1 \downarrow t_2 \in D$, *then* $\sigma(t) \in [\![\sigma[t_2/x](t_3 \downarrow t_4)]\!]_\delta$.
- *If* $(\Delta, t, t', t \lhd T', T, T') \in E$, $(\sigma, \delta) \in [\Delta]$ *and* $\sigma(t) \in [\![\sigma T]\!]_\delta$ *and* $t \lhd T' \in D$, *then* $\sigma(t) \in [\![\sigma T']\!]_\delta$.

Proof. We have $\sigma(t) \in [\![\sigma[t_1/x](t_3 \downarrow t_4)]\!]_\delta$, thus $\sigma(t) \in \mathcal{A}$ and $(\sigma[t_1/x]t_3) \downarrow (\sigma[t_1/x]t_4) \in D$. Since $t_1 \downarrow t_2 \in D$, then we have $(\sigma[t_2/x]t_3) \downarrow (\sigma[t_2/x]t_4) \in D$. This is also followed by the property of $t \downarrow t'$ (see [4]). So $\sigma(t) \in [\![\sigma[t_2/x](t_3 \downarrow t_4)]\!]_\delta$.

By *soundness of reflective relational sentences*, $t \lhd T' \in D$ implies $\forall \phi, \sigma t = t \in [\![T']\!]_\phi$. So it is the case.

So the structure $\langle D, E, \mathcal{I}_<, \mathcal{I}_\downarrow, \mathcal{I}_\lhd \rangle$ we have defined is a sound internalization structure. Let us see some instances of *A-elim* rule and *A-intro* rule for the internalized system based on this internalization structure:

$$\frac{t_1 \downarrow t_2 \in D \quad FVar(t_1 \downarrow t_2) \subseteq dom(\Delta) \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash \mathbf{axiom} : t_1 \downarrow t_2} \; A\text{-}intro$$

$$\frac{\Delta \vdash t : [t_1/x](t_3 \downarrow t_4) \quad \Delta \vdash t' : t_1 \downarrow t_2}{\Delta \vdash t : [t_2/x](t_3 \downarrow t_4)} \; A\text{-}elim$$

$$\frac{t \lhd T' \in D \quad FVar(t \lhd T') \subseteq dom(\Delta) \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash \mathbf{axiom} : t \lhd T'} \; A\text{-}intro$$

$$\frac{\Delta \vdash t : T \quad \Delta \vdash t' : t \lhd T'}{\Delta \vdash t : T'} \; A\text{-}elim$$

We can see that our elimination rule for $\downarrow$ realizes a more general form of transitivity. For example, if we have a term with a type $[t_2/y](t_1 \downarrow y)$ and $t_2 \downarrow t_3 \in D$, then we can assign this term a new type $[t_3/y](t_1 \downarrow y)$ by the elimination rule.

6 Related Work

Reflection allows users of logical frameworks such as Coq or NuPRL to use object-level definitions and computations as proxies or replacements for meta-level processes such as tactic application. NuPRL includes a built-in reflection rule, showing (roughly) that $H \vdash G$ follows from $\vdash \exists p.p \ proves \ rep(H \vdash G)$. A key difficulty in making this precise is reflecting uses of the reflection rule itself [6]. Solutions to this problem are given by [1] and [5]. Like the internalization discussed in the present paper, reflection involves a subtle interaction between the object- and meta-level. However, while reflection uses the object language to replace meta-level reasoning, internalization uses systematic meta-level operations to extend and enhance the object language. Scharp seems to address similar issues of internalization, from the perspective of natural language with Tarski semantics [15].

Many type theories make direct use of auxiliary judgments. For example, Pure Type Systems includes a conversion rule based on a convertibility judgment that is is not internalized as a type [3]. In CIC (and similar systems), one can define an equality type inductively. In a sense this internalizes the convertibility judgment [12], but such an equality type cannot be used to change the type of a given term; rather, an explicit cast is required. In contrast, the current paper's internalized equality can be used to change the type of a term without an explicit cast. Cast terms raise a number of well-known problems, addressed using axioms like Streicher's axiom K [17]. Systems like the Implicit Calculus of Constructions have sought to alleviate similar problems by leaving term annotations implicit, although not for equality casts [10]. More recently, Sjöberg and Stump have proposed a system where all such equality casts are implicit, eliminating the practical need for methods like axiom K [16]. Indeed, providing a framework to support systems like their T^{vec} system is one motivation for

the present work. The problem of equality in type theory is, of course, one of the most longstanding and central ones, and has been addressed in too many works to summarize further (but see also [2] for a recent approach combining intensional and extensional features).

Also related is language extension using meta-programming techniques. Felleisen builds a formal model describing the power of and limitations of macros for extending programming languages with new features [8]. As with internalization, macros can define "local" language extensions. In Felleisen's setting locality is induced by a requirement that macro expansion be a homomorphism on base-language syntax. In contrast to this syntactic approach, our internalization framework specifies semantic constraints which, if satisfied, enable preservation of deep semantic properties of the system being extended.

7 Conclusion and Future Work

We have formalized the notion of internalization structure and demonstrated that the internalized system is terminating. We also have shown how our formalization can be applied to full-beta term equality, subtyping and term-type inhabitation relation. Our approach makes it easier to establish normalization for type theories with these features, since the framework provides the analysis for all but the internalization-specific parts of the language. In future work, we hope to change our base system to the Calculus of Construction and see if we can obtain a similar internalization structure from it. Also, we plan to prove other standard meta-theoretic results, such as type preservation (i.e. subject reduction). Another direction would be trying to find out more relations that can be incorporated into base system through internalization.

References

1. Stuart F. Allen, Robert L. Constable, Douglas J. Howe, and William Aitken. The semantics of reflected proof. In *Proceedings of Fifth IEEE Symposium on Logic in Computer Science*, pages 95–197, 1990.
2. T. Altenkirch, C. McBride, and W. Swierstra. Observational Equality, Now! In A. Stump and H. Xi, editors, *PLPV '07: Proceedings of the 2007 Workshop on Programming Languages meets Program Verification*, pages 57–68, 2007.
3. H. Barendregt. Lambda Calculi with Types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.
4. H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Sole Distributors for the U.S.A. And Canada, Elsevier Science Pub. Co., 1984.
5. Eli Barzilay. *Implementing Reflection in Nuprl*. PhD thesis, Cornell, 2006.
6. Robert L. Constable. Using reflection to explain and enhance type theory. In *Proof and Computation, volume 139 of NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 20-August 1, NATO Series F*, pages 65–100. Springer, 1994.

7. Thierry Coquand and Christine Paulin. Inductively defined types. In Per Martin-Lf and Grigori Mints, editors, *COLOG-88*, volume 417 of *Lecture Notes in Computer Science*, pages 50–66. Springer Berlin / Heidelberg, 1990.
8. Matthias Felleisen. On the expressive power of programming languages. In *Science of Computer Programming*, pages 134–151. Springer-Verlag, 1990.
9. Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.
10. A. Miquel. The Implicit Calculus of Constructions. In *Typed Lambda Calculi and Applications*, volume 2044 of *Lecture Notes in Computer Science*, pages 344–359. Springer, 2001.
11. Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, USA, July 1990.
12. F. Pfenning and C. Paulin-Mohring. Inductively Defined Types in the Calculus of Constructions. In *Proceedings of the 5th International Conference on Mathematical Foundations of Programming Semantics*, pages 209–228, London, UK, 1990. Springer-Verlag.
13. Benjamin C. Pierce. *Types and programming languages*. MIT Press, Cambridge, MA, USA, 2002.
14. Jakob Rehof. Strong normalization for non-structural subtyping via saturated sets. *Inf. Process. Lett.*, 58:157–162, May 1996.
15. Kevin Scharp. Truth and internalizability. Draft available from `http://people.cohums.ohio-state.edu/scharp1`, 2010.
16. V. Sjöberg and A. Stump. Equality, Quasi-Implicit Products, and Large Eliminations. In B. Venneri, editor, *Workshop on Intersection Types and Related Systems (ITRS)*, 2010.
17. T. Streicher. *Investigations into Intensional Type Theory*. PhD thesis, Ludwig Maximilians University, Munich, 1993. Habilitation thesis.
18. Jeffrey A. Vaughan. *Aura: Programming with Authorization and Audit*. PhD thesis, University of Pennsylvania, Philadelphia, 2009.

A Proofs

A.1 Proof of Lemma 1

Proof. By induction on the structure of T.

Case: $T = X, B$

CR 1–CR 3 Obvious from the definition.

Case: $T = \Pi x : T_1.T_2$

CR 1 Obvious from the definition of $[\![\Pi x : T_1.T_2]\!]_\phi$.

CR 2 Assume $t \in [\![\Pi x : T_1.T_2]\!]_\phi$ and $t \rightsquigarrow t'$. Take arbitrary $u \in [\![T_1]\!]_\phi$. By definition, we know $(t\ u) \in [\![[u/x]T_2]\!]_\phi$. With our reduction strategy, $(t\ u) \rightsquigarrow (t'\ u)$. By IH(CR 2), $(t'\ u) \in [\![[u/x]T_2]\!]_\phi$. So by definition of $[\![\Pi x : T_1.T_2]\!]_\phi$, $t' \in [\![\Pi x : T_1.T_2]\!]_\phi$.

CR 3 Assume t is closed, $t \rightsquigarrow t'$ and $t' \in [\![\Pi x : T_1.T_2]\!]_\phi$. Take arbitrary $u \in [\![T_1]\!]_\phi$. By definition, we know $(t'\ u) \in [\![[u/x]T_2]\!]_\phi$. With our reduction strategy, $(t\ u) \rightsquigarrow (t'\ u)$. By IH(CR 1), u is closed, thus we know $(t\ u)$ is closed. By IH(CR 3), $(t\ u) \in [\![[u/x]T_2]\!]_\phi$. So by definition of $[\![\Pi x : T_1.T_2]\!]_\phi$, $t \in [\![\Pi x : T_1.T_2]\!]_\phi$.

Case: $T = \forall X.T$

CR 1 Assume $t \in [\![\forall X.T]\!]_\phi$. We know that $\Re$ is non-empty (for example, $\mathcal{V} \in \Re$, this follow by lemma 10). Take an arbitrary reducibility candidate $\mathcal{R}$. By definition of $[\![\forall X.T]\!]_\phi$, $t \in [\![T]\!]_{\phi[\mathcal{R}/X]}$. By IH(CR 1), $t \in \mathcal{V}$.

CR 2 Assume $t \in [\![\forall X.T]\!]_\phi$ and $t \rightsquigarrow t'$. Consider arbitrary reducibility candidate $\mathcal{R}$. By definition, $t \in [\![T]\!]_{\phi[\mathcal{R}/X]}$. By IH(CR 2), $t' \in [\![T]\!]_{\phi[\mathcal{R}/X]}$. So by definition of $[\![\forall X.T]\!]_\phi$, $t' \in [\![\forall X.T]\!]_\phi$.

CR 3 Assume t is closed, $t \rightsquigarrow t'$ and $t' \in [\![\forall X.T]\!]_\phi$. Take arbitrary reducibility candidate $\mathcal{R}$. By definition, $t' \in [\![T]\!]_{\phi[\mathcal{R}/X]}$. We know that $t \rightsquigarrow t'$ and t is closed. So by IH(CR 3), $t \in [\![T]\!]_{\phi[\mathcal{R}/X]}$. So by definition of $[\![\forall X.T]\!]_\phi$, $t \in [\![\forall X.T]\!]_\phi$.

Lemma 10. *Let $\mathcal{B} \subseteq \mathcal{V}$. We have $\hat{\mathcal{B}} = \{t \mid \exists v \in \mathcal{B}, t \overset{*}{\rightsquigarrow} v$ and t closed $\} \in \Re$.*

Proof. CR1: $t \in \hat{\mathcal{B}}$ implies $t \in \mathcal{V}$.

CR2: If $t \in \hat{\mathcal{B}}$ and $t \rightsquigarrow t'$, again by definition of $\hat{\mathcal{B}}$, we have $t' \in \hat{\mathcal{B}}$.

CR3: Similar argument as CR2.

A.2 Proof of Lemma 3

Proof. By induction on the structure of T.

Case: $T = X, B$

If $T = B$, trivial. If $T = X$, we need to show $[\![T']\!]_\phi = [\![X]\!]_{\phi[[\![T']\!]_\phi/X]}$. By definition, $[\![X]\!]_{\phi[[\![T']\!]_\phi/X]} = \phi[[\![T']\!]_\phi/X](X) = [\![T']\!]_\phi$. If $T = Y \neq X$, then $[\![[T'/X]Y]\!]_\phi = [\![Y]\!]_\phi = [\![Y]\!]_{\phi[[\![T']\!]_\phi/X]}$. So it is the case.

Case: $T = \forall X_1.T_1$

Then we need to show $[\![(\forall X_1.[T'/X]T_1)]\!]_\phi = [\![\forall X_1.T_1]\!]_{\phi[[\![T']\!]_\phi/X]}$. Take arbitrary $\mathcal{R} \in \Re$ and arbitrary $t \in [\![(\forall X_1.[T'/X]T_1)]\!]_\phi$. By definition, $t \in [\![[T'/X]T_1]\!]_{\phi[\mathcal{R}/X_1]}$. By IH, $[\![[T'/X]T_1]\!]_{\phi[\mathcal{R}/X_1]} = [\![T_1]\!]_{\phi[\mathcal{R}/X_1,[\![T']\!]_{\phi[\mathcal{R}/X_1]}/X]} = [\![T_1]\!]_{\phi[\mathcal{R}/X_1,[\![T']\!]_\phi/X]}$, since we may assume $X_1 \notin FV(T')$. So $t \in [\![T_1]\!]_{\phi[\mathcal{R}/X_1,[\![T']\!]_\phi/X]}$. By definition, $t \in [\![\forall X_1.T_1]\!]_{\phi[[\![T']\!]_\phi/X]}$.

Now let's prove the other direction. Take arbitrary $t \in [\![\forall X_1.T_1]\!]_{\phi[[\![T']\!]_\phi/X]}$ and arbitrary $\mathcal{R} \in \Re$. By definition, $t \in [\![T_1]\!]_{\phi[[\![T']\!]_\phi/X,\mathcal{R}/X_1]}$. By IH, $[\![T_1]\!]_{\phi[[\![T']\!]_\phi/X,\mathcal{R}/X_1]} = [\![[T'/X]T_1]\!]_{\phi[\mathcal{R}/X_1]}$. So $t \in [\![[T'/X]T_1]\!]_{\phi[\mathcal{R}/X_1]}$. By definition, $t \in [\![\forall X_1.[T'/X]T_1]\!]_\phi$. So it is the case.

Case: $T = \Pi x : T_1.T_2$

Then we need to show $[\![(\Pi x : [T'/X]T_1.[T'/X]T_2)]\!]_\phi = [\![(\Pi x : T_1.T_2)]\!]_{\phi[[\![T']\!]_\phi/X]}$. Take arbitrary $u \in [\![([T'/X]T_1)]\!]_\phi$ and $t \in [\![\Pi x : [T'/X]T_1.[T'/X]T_2)]\!]_\phi$. By definition, $(t\ u) \in [\![([u/x][T'/X]T_2)]\!]_\phi = [\![[T'/X]([u/x]T_2)]\!]_\phi$. By IH, $[\![([T'/X]T_1)]\!]_\phi = [\![T_1]\!]_{\phi[[\![T']\!]_\phi/X]}$ and $[\![[T'/X]([u/x]T_2)]\!]_\phi = [\![[u/x]T_2]\!]_{\phi[[\![T']\!]_\phi/X]}$. So $t \in [\![\Pi x : T_1.T_2]\!]_{\phi[[\![T']\!]_\phi/X]}$. The other direction is similar.

A.3 Proof of Theorem 1

Proof. By induction on the typing derivation of $\Gamma \vdash t : T$

Case:

$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T}$$

By definition of $(\sigma, \delta) \in [\Gamma]$, there exists t such that $\{(x, t)\} \subseteq \sigma$ and $t \in [\![\sigma(T)]\!]_\delta$, so $\sigma\ x = t \in [\![\sigma(T)]\!]_\delta$.

Case:

$$\frac{\Gamma \vdash t_1 : \Pi x : T_1.T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : [t_2/x]T_2}$$

We need to prove that $\sigma(t_1\ t_2) \in [\![\sigma([t_2/x]T_2)]\!]_\delta$. By IH, for any $(\sigma, \delta) \in [\Gamma], \sigma\ t_1 \in [\![\Pi x : \sigma T_1.\sigma T_2]\!]_\delta$ and $\sigma\ t_2 \in [\![\sigma T_1]\!]_\delta$. Then from definition of $[\![\Pi x : \sigma T_1.\sigma T_2]\!]_\delta$, we have $(\sigma t_1)(\sigma t_2) = \sigma(t_1\ t_2) \in [\![[\sigma(t_2)/x](\sigma T_2)]\!]_\delta$. So we need to show $[\![\sigma([t_2/x]T_2)]\!]_\delta = [\![[\sigma(t_2)/x](\sigma T_2)]\!]_\delta$, which is true.

Case:

$$\frac{\Gamma, x : T_1\ \vdash t : T_2}{\Gamma \vdash \lambda x.t : \Pi x : T_1.T_2}$$

We need to show that for all $(\sigma, \delta) \in [\Gamma]$, we have $\sigma(\lambda x.t) = \lambda x.(\sigma\ t) \in [\![\Pi x : \sigma T_1.\sigma T_2]\!]_\delta$. First, we know that $\lambda x.(\sigma\ t) \in \mathcal{V}$. Then by definition of $[\![\Pi x : \sigma T_1.\sigma T_2]\!]_\delta$,we need to show for arbitrary $u \in [\![\sigma T_1]\!]_\delta, (\lambda x.(\sigma\ t))\ u \in [\![[u/x](\sigma T_2)]\!]_\delta$. Since we use LTR-CBN, we have $(\lambda x.(\sigma\ t))\ u \rightsquigarrow \sigma[u/x]t$. Since $u \in [\![\sigma T_1]\!]_\delta, (\sigma \cup \{(x, u)\}, \delta) \in [\Gamma, x : T_1]$. By IH, $\sigma[u/x](t) \in [\![(\sigma[u/x]T_2)]\!]_\delta$. Since $(\lambda x.(\sigma\ t))\ u$ is closed ,by CR 3, $(\lambda x.(\sigma\ t))\ u \in [\![(\sigma[u/x]T_2)]\!]_\delta$. So it is the case.

Case:

$$\frac{\Gamma, X\ \vdash t : T}{\Gamma \vdash t : \forall X.T}$$

We need to show for any $(\sigma, \delta) \in [\Gamma]$, $\sigma(t) \in [\![\sigma \forall X.T]\!]_\delta$. By definition of $[\![\sigma \forall X.T]\!]_\delta$, we just need to show for arbitrary $\mathcal{R} \in \Re$, $\sigma(t) \in [\![\sigma T]\!]_{\delta[\mathcal{R}/X]}$. By IH, for any $(\sigma, \delta \cup \{(X, \mathcal{R})\}) \in [\Gamma, X]$, so $\sigma(t) \in [\![(\sigma T)]\!]_{\delta[\mathcal{R}/X]}$. So it is the case.

Case:

$$\frac{\Gamma \vdash t : \forall X.T}{\Gamma \vdash t : ([T'/X]T)}$$

We need to show for any $(\sigma, \delta) \in [\Gamma]$, $\sigma(t) \in [\![([\sigma T'/X]\sigma T)]\!]_\delta$. By IH, we know that $\sigma(t) \in [\![(\forall X.\sigma T)]\!]_\delta$. By definition, for any $\mathcal{R} \in \Re, \sigma(t) \in [\![(\sigma T)]\!]_{\delta[\mathcal{R}/X]}$. Since $[\![\sigma T']\!]_\delta$ is a reducibility candidate, we have $\sigma(t) \in [\![(\sigma T)]\!]_{\delta[[\![\sigma T']\!]_\delta/X]}$. By lemma 3 (Substitution Lemma), we know $[\![([\sigma T'/X]\sigma T)]\!]_\delta = [\![(\sigma T)]\!]_{\delta[[\![\sigma T']\!]_\delta/X]}$. So it is the case.

A.4 Proof of Lemma 4

Proof. By induction on the structure of S.

Case: $S = X, B$

CR 1–CR 3 Obvious from the definition.

Case: $S = \Pi x : S_1.S_2$

CR 1 Obvious from the definition of $[\![\Pi x : S_1.S_2]\!]_\phi$.

CR 2 Assume $t \in [\![\Pi x : S_1.S_2]\!]_\phi$ and $t \rightsquigarrow t'$. Take arbitrary $u \in [\![S_1]\!]_\phi$. By definition, we know $(t\ u) \in [\![[u/x]S_2]\!]_\phi$. With our reduction strategy, $(t\ u) \rightsquigarrow (t'\ u)$. By IH(CR 2), $(t'\ u) \in [\![[u/x]S_2]\!]_\phi$. So by definition of $[\![\Pi x : S_1.S_2]\!]_\phi$, $t' \in [\![\Pi x : S_1.S_2]\!]_\phi$.

CR 3 Assume t is closed, $t \rightsquigarrow t'$ and $t' \in [\![\Pi x : S_1.S_2]\!]_\phi$. Take arbitrary $u \in [\![S_1]\!]_\phi$. By definition, we know $(t'\ u) \in [\![[u/x]S_2]\!]_\phi$. With our reduction strategy, $(t\ u) \rightsquigarrow (t'\ u)$. By IH(CR 1), u is closed, thus we know $(t\ u)$ is closed. By IH(CR 3), $(t\ u) \in [\![[u/x]S_2]\!]_\phi$. So by definition of $[\![\Pi x : S_1.S_2]\!]_\phi$, $t \in [\![\Pi x : S_1.S_2]\!]_\phi$.

Case: $S = \forall X.S$

CR 1 Assume $t \in [\![\forall X.S]\!]_\phi$. We know that $\Re$ is non-empty (for example, $\mathcal{V} \in \Re$). Take an arbitrary reducibility candidate $\mathcal{R}$. By definition of $[\![\forall X.S]\!]_\phi$, $t \in [\![S]\!]_{\phi[\mathcal{R}/X]}$. By IH(CR 1), $t \in \mathcal{V}$.

CR 2 Assume $t \in [\![\forall X.S]\!]_\phi$ and $t \rightsquigarrow t'$. Consider arbitrary reducibility candidate $\mathcal{R}$. By definition, $t \in [\![S]\!]_{\phi[\mathcal{R}/X]}$. By IH(CR 2), $t' \in [\![S]\!]_{\phi[\mathcal{R}/X]}$. So by definition of $[\![\forall X.S]\!]_\phi$, $t' \in [\![\forall X.S]\!]_\phi$.

CR 3 Assume t is closed, $t \rightsquigarrow t'$ and $t' \in [\![\forall X.S]\!]_\phi$. Take arbitrary reducibility candidate $\mathcal{R}$. By definition, $t' \in [\![S]\!]_{\phi[\mathcal{R}/X]}$. We know that $t \rightsquigarrow t'$ and t is closed. So by IH(CR 3), $t \in [\![S]\!]_{\phi[\mathcal{R}/X]}$. So by definition of $[\![\forall X.S]\!]_\phi$, $t \in [\![\forall X.S]\!]_\phi$.

Case: $S = A$

By definition we know that $[\![S]\!]_\phi$ is either an empty set or $\mathcal{A}$. Since $\{\mathbf{axiom}\} \subseteq \mathcal{V}$, by lemma 10, we have $\mathcal{A} \in \Re$.

A.5 Proof of Lemma 5

Proof. Prove by induction on the structure of S.

Case: $S = X, B$

If $S = B$, trivial. If $S = X$, we need to show $[\![S']\!]_\phi = [\![X]\!]_{\phi[[\![S']\!]_\phi/X]}$. By definition, $[\![X]\!]_{\phi[[\![S']\!]_\phi/X]} = \phi[[\![S']\!]_\phi/X](X) = [\![S']\!]_\phi$.

Case: $S = \forall X_1.S_1$

We have $\forall X_1.[S'/X]S_1 \in \mathbf{ETypes}$. Thus we also have $[S'/X]S_1 \in \mathbf{ETypes}$. We need to show $[\![(\forall X_1.[S'/X]S_1)]\!]_\phi = [\![\forall X_1.S_1]\!]_{\phi[[\![S']\!]_\phi/X]}$. Take arbitrary $\mathcal{R} \in \Re$

and arbitrary $t \in [\![(\forall X_1.[S'/X]S_1)]\!]_\phi$. By definition, $t \in [\![[S'/X]S_1]\!]_{\phi[\mathcal{R}/X_1]}$. By IH, $[\![[S'/X]S_1]\!]_{\phi[\mathcal{R}/X_1]} = [\![S_1]\!]_{\phi[\mathcal{R}/X_1,[\![S']\!]_{\phi[\mathcal{R}/X_1]}/X]} = [\![S_1]\!]_{\phi[\mathcal{R}/X_1,[\![S']\!]_\phi/X]}$. So $t \in [\![S_1]\!]_{\phi[\mathcal{R}/X_1,[\![S']\!]_\phi/X]}$. By definition, $t \in [\![\forall X_1.S_1]\!]_{\phi[[\![S']\!]_\phi/X]}$.

Now let's prove the other direction. Take arbitrary $t \in [\![\forall X_1.S_1]\!]_{\phi[[\![S']\!]_\phi/X]}$ and arbitrary $\mathcal{R} \in \Re$. By definition, $t \in [\![S_1]\!]_{\phi[[\![S']\!]_\phi/X,\mathcal{R}/X_1]}$. By IH, $[\![S_1]\!]_{\phi[[\![S']\!]_\phi/X,\mathcal{R}/X_1]} = [\![[S'/X]S_1]\!]_{\phi[\mathcal{R}/X_1]}$. So $t \in [\![[S'/X]S_1]\!]_{\phi[\mathcal{R}/X_1]}$. By definition, $t \in [\![\forall X_1.[S'/X]S_1]\!]_\phi$. So it is the case.

Case: $S = \Pi x : S_1.S_2$

We have $\Pi x : [S'/X]S_1.[S'/X]S_2 \in \mathbf{ETypes}$ and $[S'/X]S_1, [S'/X]S_2 \in \mathbf{ETypes}$. Then we need to show $[\![(\Pi x : [S'/X]S_1.[S'/X]S_2)]\!]_\phi = [\![(\Pi x : S_1.S_2)]\!]_{\phi[[\![S']\!]_\phi/X]}$. Take arbitrary $u \in [\![([S'/X]S_1)]\!]_\phi$ and $t \in [\![\Pi x : [S'/X]S_1.[S'/X]S_2)]\!]_\phi$. By definition, $(t\ u) \in [\![([u/x][S'/X]S_2)]\!]_\phi = [\![[S'/X]([u/x]S_2)]\!]_\phi$. By IH, $[\![([S'/X]S_1)]\!]_\phi = [\![S_1]\!]_{\phi[[\![S']\!]_\phi/X]}$ and $[\![[S'/X]([u/x]S_2)]\!]_\phi = [\![[u/x]S_2]\!]_{\phi[[\![S']\!]_\phi/X]}$. So $t \in [\![\Pi x : S_1.S_2]\!]_{\phi[[\![S']\!]_\phi/X]}$. The other direction is similar.

Case: $S = A$

We have $R^{(n\times m)}(t_1, ..., t_n, [S'/X]T_1, ..., [S'/X]T_m) \in \mathbf{ETypes}$. If $\forall i \in \{1, 2, ..., m\}, X \notin FV(T_i)$, then it is trivially true. Otherwise we have $S' \in \mathbf{Types}$. We need to show
$[\![R^{(n\times m)}(t_1, ..., t_n, [S'/X]T_1, ..., [S'/X]T_m)]\!]_\phi =$
$[\![R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m)]\!]_{\phi[[\![S']\!]_\phi/X]}$.
By definition, we need to show
$(t_1, ..., t_n, [\![[S'/X]T_1]\!]_\phi, ..., [\![[S'/X]T_m]\!]_\phi) \in \mathcal{I}_{R^{(n\times m)}} \Leftrightarrow$
$(t_1, ..., t_n, [\![T_1]\!]_{\phi[[\![S']\!]_\phi/X]}, ..., [\![T_m]\!]_{\phi[[\![S']\!]_\phi/X]}) \in \mathcal{I}_{R^{(n\times m)}}$
which is true because lemma 3.

A.6 Proof of Theorem 2

proof By induction on the typing derivation of $\Delta \vdash t : S$

Case:

$$\frac{(x : S) \in \Delta \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash x : S}$$

By definition of $(\sigma, \delta) \in [\Delta]$, there exists t such that $t \in [\![\sigma(S)]\!]_\delta$. Thus $(x, t) \in \sigma$. So $\sigma\ x = t \in [\![\sigma(S)]\!]_\delta$.

Case:

$$\frac{\Delta \vdash t_1 : \Pi x.S_1.S_2 \quad \Delta \vdash t_2 : S_1}{\Delta \vdash t_1\ t_2 : [t_2/x]S_2}$$

We need to prove that $\sigma(t_1\ t_2) \in [\![\sigma([t_2/x]S_2)]\!]_\delta$. By IH, for any $(\sigma, \delta) \in [\Delta], \sigma\ t_1 \in [\![\Pi x : \sigma S_1.\sigma S_2]\!]_\delta$ *and* $\sigma\ t_2 \in [\![\sigma S_1]\!]_\delta$. Then from definition of $[\![\Pi x : \sigma S_1.\sigma S_2]\!]_\delta$, we have $(\sigma t_1)(\sigma t_2) = \sigma(t_1\ t_2) \in [\![[\sigma(t_2)/x](\sigma S_2)]\!]_\delta$. So we need to show $[\![\sigma([t_2/x]S_2)]\!]_\delta = [\![[\sigma(t_2)/x](\sigma S_2)]\!]_\delta$, which is true.

Case:

$$\frac{\Delta, x : S_1\ \vdash t : S_2}{\Delta \vdash \lambda x.t : \Pi x : S_1.S_2}$$

We need to show that for any $(\sigma, \delta) \in [\Delta]$, we have $\sigma(\lambda x.t) = \lambda x.(\sigma\ t) \in [\![\Pi x : \sigma S_1.\sigma S_2]\!]_\delta$. First, we know that $\lambda x.(\sigma\ t) \in \mathcal{V}$. Then by definition of $[\![\Pi x : \sigma S_1.\sigma S_2]\!]_\delta$,we need to show for arbitrary $u \in [\![\sigma S_1]\!]_\delta, (\lambda x.(\sigma\ t))\ u \in [\![[u/x](\sigma S_2)]\!]_\delta$. u is closed by CR 1. By LTR-CBN, we have $(\lambda x.(\sigma\ t))\ u \rightsquigarrow \sigma[u/x]t$. Since $u \in [\![\sigma S_1]\!]_\delta$,$(\sigma \cup \{(x, u)\}, \delta) \in [\Delta, x : S_1]$. By IH, $\sigma[u/x](t) \in [\![(\sigma[u/x]S_2)]\!]_\delta$. Since $(\lambda x.(\sigma\ t))\ u$ is closed, by CR 3, $(\lambda x.(\sigma\ t))\ u \in [\![(\sigma[u/x]S_2)]\!]_\delta$. So it is the case.

Notice that if we use LTR-CBV as underlying operational semantics, to show $\lambda x.(\sigma\ t) \in [\![\Pi x : \sigma S_1.\sigma S_2]\!]_\delta$, we need to show for any $u \in [\![\sigma S_1]\!]_\delta, (\lambda x.(\sigma\ t))\ u \in [\![[u/x](\sigma S_2)]\!]_\delta$. Since we use CBV, we have $(\lambda x.(\sigma\ t))\ u \overset{*}{\rightsquigarrow} (\lambda x.(\sigma\ t))\ v \rightsquigarrow [v/x]\sigma t$. Since $v \in [\![\sigma S_1]\!]_\delta$, by IH and CR3, we have $(\lambda x.(\sigma\ t))\ u \in [\![(\sigma[v/x]S_2)]\!]_\delta$. Thus we need another lemma stating: if $t \rightsquigarrow t'$, then $[\![[t/x]S]\!]_\delta = [\![[t'/x]S_2]\!]_\delta$. Thus it is more complicated in this situation.

Case:

$$\frac{\Delta, X \vdash t : S}{\Delta \vdash t : \forall X.S}$$

We need to show for any $(\sigma, \delta) \in [\Delta]$, $\sigma(t) \in [\![\sigma\forall X.S]\!]_\delta$. By definition of $[\![\sigma\forall X.S]\!]_\delta$, we just need to show for arbitrary $\mathcal{R} \in \Re$, $\sigma(t) \in [\![\sigma S]\!]_{\delta[\mathcal{R}/X]}$. By IH, for $(\sigma, \delta \cup \{(X, \mathcal{R})\}) \in [\Delta]$, we have $\sigma(t) \in [\![(\sigma S)]\!]_{\delta[\mathcal{R}/X]}$. So it is the case.

Case:

$$\frac{\Delta \vdash t : \forall X.S \quad [S'/X]S \in \mathbf{ETypes} \quad FVar(S') \subseteq dom(\Delta)}{\Delta \vdash t : ([S'/X]S)}$$

We need to show for any $(\sigma, \delta) \in [\Delta]$, $\sigma(t) \in [\![([\sigma S'/X]\sigma S)]\!]_\delta$. By IH, we know that $\sigma(t) \in [\![(\forall X.\sigma S)]\!]_\delta$. By definition, for any $\mathcal{R} \in \Re, \sigma(t) \in [\![(\sigma S)]\!]_{\delta[\mathcal{R}/X]}$. Since $[\![\sigma S']\!]_\delta$ is a reducibility candidate, we have $\sigma(t) \in [\![(\sigma S)]\!]_{\delta[[\![\sigma S']\!]_\delta/X]}$. By lemma 5 (Substitution Lemma), we know $[\![([\sigma S'/X]\sigma S)]\!]_\delta = [\![(\sigma S)]\!]_{\delta[[\![\sigma S']\!]_\delta/X]}$. So it is the case.

Case:

$$\frac{A \in D \quad FVar(A) \subseteq dom(\Delta) \quad \Delta \vdash \mathbf{OK}}{\Delta \vdash \mathbf{axiom} : A}$$

We need to show if $(\sigma, \delta) \in [\Delta], dom(\Delta) = FVar(A)$ and $A \in D$, then $\mathbf{axiom} \in [\![\sigma A]\!]_\delta$. Let $A = R^{(n\times m)}(t_1, ..., t_n, T_1, ..., T_m)$. Thus we need to show $(\sigma t_1, ..., \sigma t_n, [\![\sigma T_1]\!]_\delta, ..., [\![\sigma T_m]\!]_\delta) \in \mathcal{I}_{R^{(n\times m)}}$. By the **soundness of reflective relational sentences**, we know it is the case.

Case:

$$\frac{\Delta \vdash t : T \quad \Delta \vdash t' : A \quad E(\Delta, t, t', A, T, T')}{\Delta \vdash t : T'} \; A\text{-}elim$$

We have $(\sigma, \delta) \in [\Delta]$, $\sigma(t) \in [\![\sigma T]\!]_\delta, \sigma(t') \in [\![\sigma A]\!]_\delta$ and $E(\Delta, t, t', A, T, T')$. We know that it is the case by **Soundness of elimination relation**.