Typed Applicative Structures and Normalization by Evaluation for System F$^\omega$

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

Abstract. We present a normalization-by-evaluation (NbE) algorithm for System F$^\omega$ with $\beta\eta$-equality, the simplest impredicative type theory with computation on the type level. Values are kept abstract and requirements on values are kept to a minimum, allowing many different implementations of the algorithm. The algorithm is verified through a general model construction using typed applicative structures, called type and object structures. Both soundness and completeness of NbE are conceived as an instance of a single fundamental theorem.

1 Introduction and Related Work

The Curry-Howard isomorphism, which identifies proofs with programs and propositions with types, is the basis of several type-theoretic theorem provers, such as Coq [INR07], Agda [Nor07], Epigram [CAM07] and LEGO [Pol94a]. In these systems, checking the validity of proofs or typings relies on deciding equality of types. Types are recognized as equal if they have the same normal form, this is why normalization plays a key role in the study of the theories underlying these theorem provers, such as the Calculus of Constructions (CC) which underlies Coq and LEGO, and Martin-Löf Type Theory which is the basis of Agda and Epigram. The hardwired type equality of these systems, often referred to as definitional equality, is necessarily intensional, since fully extensional equality is undecidable. The minimal equality is induced by just the computational laws (β-laws), yet, the stronger the hardwired equality, the less manual equality proofs, and the more succinct proofs can get. Thus, it is desirable to add bits of extensionality as long as decidability is preserved. For the CC, different directions have been explored, such as rewriting [Bla05,CWC07] and decision procedures [BJS07]. Our goal is to integrate η-laws. which provide some extensionality for functions, into definitional equality of CC.

Normalization by evaluation (NbE) [BS91,Dan99] is a systematic method to perform $\beta\eta$-normalization. In a first step, the object t of type T is evaluated. The resulting value d is then *reified* to an η-long β-normal form v. The reification process is directed by the shape of type T. NbE has proven a valid tool to structure extensional normalization, especially in the notoriously difficult case of sum types [ADHS01,BCF04,Bar08]. We are convinced it is the perfect tool to account for the meta-theory of η, thus, we are researching the type systems of the lambda cube with $\beta\eta$-equality given by judgements.

The lambda cube organizes type systems in three dimensions: *dependency*, *impredicativity*, and *higher-order*. In previous work [ACD07], we have adapted NbE to a dependent type theory with one predicative universe and judgmental $\beta\eta$-equality. What is the challenge when stepping up to impredicativity? Predicative type theories allow to define the semantics of types from below via induction-recursion [Dyb00], and the reification function can be defined by induction on types. This fails in the presence of impredicativity, where one first has to lay out a lattice of semantic type candidates and then define impredicative quantification using an intersection over all candidates [GLT89]. Hence, the semantic type structure is not inductive, and reification cannot be defined by induction on types.[1] There are at least two ways out of this dilemma: Altenkirch, Hofmann, and Streicher [AHS96] construct a total normalization function type-wise while building a model for System F. In previous work [Abe08], I have conceived reification as a deterministic relation between value d and normal form v and their type T, and showed through a model construction that it corresponds to a total function.

In this work, we are moving one step closer to NbE for the CC: we are considering the simplest type system which features impredicativity and computation on the type level: the *higher-order* polymorphic lambda-calculus F^ω. It adds to the problem of impredicativity the difficulty that types are no longer fixed syntactic expressions as in System F, but they need to be normalized as well. One solution would be to keep types always in long normal form, e. g., by the use of hereditary substitutions [WCPW03,AR08]. This is possible since the types are simply-kinded, so normalization can be defined by induction on kinds. However, this approach would not scale, e.g., to CC, where "serious" computation is happening on the type level. Furthermore, we would like to use the same normalization procedure both on object and type level.

In our solution, reification of objects is directed by *type values* A. Syntactic types T are interpreted by a pair $(A, \mathcal{A})$ of a type value A and a semantic type $\mathcal{A}$ which is a set of objects that are reifiable at type A. Furthermore, type value A reifies to a normal form V which is $\beta\eta$-equal to T. These considerations lead us to the concept of a *type structure* which captures the similarities between syntactic types, type values, and semantic types. Consequently, syntactic *objects* and their values both form an *object structure* over a type structure, the syntactical type structure in case of syntactic objects and the structure of type values in case of (object) values.

We reorganize a typical normalization proof for System F^ω by model construction [Gir72] into our framework of type and object structures. Central to such a normalization proof is the fundamental theorem of typing which states that $(\!|t|\!) \in [\![T]\!]$ for any object t of type T. Herein, $(\!|t|\!)$ interprets object t in some applicative structure, for instance in the structure of syntactical objects. The semantic type $[\![T]\!]$ is a collection of values, typically a set of normalizing objects, where the function type $[\![T_1 \to T_2]\!]$ is defined à la Tait [Tai67], i. e., as all values f such that f applied to d inhabits $[\![T_2]\!]$ for all d in $[\![T_1]\!]$. The essence of the fundamental theorem is that the (hereditarily) normalizing terms model the typing rules where these terms are a (non-proper) subset of the typable

[1] The reflection function $\uparrow$, which is defined mutually inductive with reification, yields at universal quantification $(\uparrow^{\forall X A} t)(B) = \uparrow^{A[B/X]}(t\, B)$. Clearly $A[B/X]$ can be bigger as $\forall X A$. For similar reasons, the set of η-long normal forms cannot be defined by induction on the type in System F; it lacks the subformula property for normal forms.

terms.[2] In our abstract setting, the fundamental theorem proves that a part of an object structure with the above function type definition is an object *substructure*, casting the fundamental theorem into an algebraic setting.

Or notions of type and object structures are very general, in essence typed versions of Barendregt's syntactical applicative structures [Bar84, Def. 5.3.1]. The fundamental theorems we prove are also very general since we do not fix an interpretation of types; we only require that semantic types inhabit a *candidate space*. By choosing different candidate spaces we can harvest different results from the same fundamental theorem, e. g., soundness of NbE, completeness of NbE, or weak normalization of β- or $\beta\eta$-reduction [Abe08].

	syntax	value	semantics	structures
kind	$\kappa \in \mathsf{Ki}$		Kripke family $K_\Xi \subseteq \mathcal{T}_\Xi^\kappa$ of sets of types	$\mathsf{Ki}, \widehat{\mathcal{T}}$
type	$T \in \mathsf{Ty}_\Xi^\star$	$A \in \mathcal{T}_\Xi^\star$	Kripke family $\mathcal{A}_\Delta \subseteq D_\Delta^{\Xi \vdash A}$ of sets of objects	$\mathsf{Ty}, \mathcal{T}, \widehat{D}$
	$T \in \mathsf{Ty}_\Xi^{\kappa \to \kappa'}$	$F \in \mathcal{T}_\Xi^{\kappa \to \kappa'}$	operator $\mathcal{F} \in \widehat{D}^{\Xi \vdash F : \kappa \to \kappa'}$	
object	$t \in \mathsf{Obj}_\Gamma^{\Xi \vdash T}$	$d \in D_\Delta^{\Xi \vdash A}$	—	Obj, D

Table 1. Systematics of kind, type, and object structures.

1.1 Overview

Table 1 systematizes the main structures introduced in this article. *Kind structures* are inhabited by kinds, the types of types of System F^ω. The free kind structure Ki is inhabited by the syntactic kinds κ which coincide with their values, since no computation takes place on the kind level. A kind κ can be interpreted as a family K of types which are drawn from a type structure $\mathcal{T}$; with Tait's interpretation of the function arrow between kinds, the set $\widehat{\mathcal{T}}$ of subsets of $\mathcal{T}$ forms another kind structure. Instances of *type structures* are Ty, which contains the syntactical types T, or $\mathcal{T}$, which contains type values A of some sort, or $\widehat{D}$ which contains semantics types $\mathcal{A}$, sets of objects, hence, subsets of an object structure D. At higher kinds $\kappa \to \kappa'$, Ty is inhabited by proper type constructors, $\mathcal{T}$ by their values F, and $\widehat{D}$ by operators $\mathcal{F}$ on sets of objects. On the level of objects, we have syntactical objects t in structure Obj and values d in structure D.

All structures are *sorted*, so types always have a kind, and objects always have a type, syntactical objects a syntactical type, and values a type value. The reason is that we prove the fundamental theorems just once, and instantiate them, amongst other uses, to show soundness of NbE, which is formulated in terms of sorted, i. e., judgmental, equality.

1.2 Preliminaries

Contexts $\Xi, \Theta, \Gamma, \Delta, \Phi, \Psi$ are functions from variables to some codomain. We write $\diamond$ for the totally undefined function and $\Phi, x\!:\!a$ for the function Φ' with domain $\mathsf{dom}(\Phi) \uplus$

[2] As a consequence, the hereditarily normalizing typable terms coincide with the typable terms.

$\{x\}$ such that $\Phi'(x) = a$ and $\Phi'(y) = \Phi(y)$ for $y \neq x$. We say Ψ' *extends* Ψ, written $\Psi' \leq \Psi$, if $\Psi'(x) = \Psi(x)$ for all $x \in \mathsf{dom}(\Psi)$.

Families $\mathcal{T}_\Xi$ indexed by a context Ξ are always understood to be *Kripke*, i. e., $\Xi' \leq \Xi$ implies $\mathcal{T}_\Xi \subseteq \mathcal{T}_{\Xi'}$. The notion *Kripke family* is also used for maps M_Ξ. There it implies that M does not depend on the context parameter, i. e., $M_\Xi(a) = M_{\Xi'}(a)$ for $a \in \mathsf{dom}(M_\Xi)$ and $\Xi' \leq \Xi$. (Note that $\mathsf{dom}(M_\Xi) \subseteq \mathsf{dom}(M_{\Xi'})$ since M is Kripke.)

We identify a pair of functions (f, g) with the function $h(x) = (f(x), g(x))$, especially in the case of contexts (Δ, Γ) or environments (σ, ρ). We will sometimes drop the parentheses and write just the comma. We write $(a \in \mathcal{A}) \to \mathcal{B}(a)$ for the *dependent function space* $\{f \in \mathcal{A} \to \bigcup_{a \in \mathcal{A}} \mathcal{B}(a) \mid f(a) \in \mathcal{B}(a) \text{ for all } a \in \mathcal{A}\}$.

2 Syntax

In this section, we present the syntax and inference rules for System F^ω. The system consists of three levels: On the lowest level there live the *objects*, meaning polymorphic, purely functional programs. On the middle level live the *types* of objects, and the *type constructors*, which are classified by *kinds* that themselves inhabit the highest level.

Kinds $\kappa \in \mathsf{Ki}$ are given by the grammar $\kappa ::= \star \mid \kappa \to \kappa'$. Kind $\star$ classifies type constructors which are actually types, and kind $\kappa \to \kappa'$ classifies the type constructors which map type constructors of kind κ to type constructors of kind κ'. In the following, we will refer to all type constructors as *types*.

Assume a countably infinite set of *type variables* TyVar whose members are denoted by X, Y, Z. *Kinding contexts* $\Xi, \Theta \in \mathsf{KiCxt}$ are partial maps from the type variables into Ki. The set $\mathsf{TyCst} = \{\to, \forall^\kappa \mid \kappa \in \mathsf{Ki}\}$ contains the *type constants* C. Their kinds are given by the signature $\Sigma \in \mathsf{TyCst} \to \mathsf{Ki}$ defined by

$$\begin{aligned} \Sigma(\to) &= \star \to \star \to \star \\ \Sigma(\forall^\kappa) &= (\kappa \to \star) \to \star \qquad \text{for all } \kappa \in \mathsf{Ki}. \end{aligned}$$

Types are given by the grammar $T, U, V ::= C \mid X \mid \lambda X\!:\!\kappa.\, T \mid T\, U$, where $X \in \mathsf{TyVar}$, and form a “simply-kinded” lambda calculus. As usual, we write $T \to U$ for $\to T\, U$. *Objects* are given by the grammar $t, u, v ::= x \mid \lambda x\!:\!T.\, t \mid t\, u \mid \Lambda X\!:\!\kappa.\, t \mid t\, U$ and form a polymorphic lambda-calculus with type abstraction and type application. Herein, object variables x are drawn from a countably infinite set ObjVar which is disjoint from TyVar. We write $b[a/x]$ for capture-avoiding substitution of a for variable x in syntactic expression b, and FV for the function returning the set of all free type and object variables of a syntactic expression.

Kinding, typing, and equality for System F^ω is given by five judgements whose inference rules are displayed in Figure 1.

$$\begin{aligned} &\Xi \vdash T : \kappa \\ &\Xi \vdash T = T' : \kappa \\ &\Xi; \Gamma \vdash t : T \\ &\Xi; \Gamma \vdash t = t' : T \end{aligned}$$

Herein, the auxiliary judgement $\Xi \vdash \Gamma$, read “*Γ is a well-formed typing context in Ξ*”, is defined as $\Xi \vdash \Gamma(x) : \star$ for all $x \in \mathsf{dom}(\Gamma)$.

Kinding $\Xi \vdash T : \kappa$. "In context Ξ, type T has kind κ."

$$\frac{}{\Xi \vdash C : \Sigma(C)} \qquad \frac{}{\Xi \vdash X : \Xi(X)}$$

$$\frac{\Xi, X{:}\kappa \vdash T : \kappa'}{\Xi \vdash \lambda X{:}\kappa.T : \kappa \to \kappa'} \qquad \frac{\Xi \vdash T : \kappa \to \kappa' \qquad \Xi \vdash U : \kappa}{\Xi \vdash T\,U : \kappa'}$$

Type equality $\Xi \vdash T = T' : \kappa$. "In context Ξ, types T and T' are $\beta\eta$-equal of kind κ."

$$\frac{\Xi, X{:}\kappa \vdash T : \kappa' \qquad \Xi \vdash U : \kappa}{\Xi \vdash (\lambda X{:}\kappa.T)\,U = T[U/X] : \kappa'} \qquad \frac{\Xi \vdash T : \kappa \to \kappa'}{\Xi \vdash \lambda X{:}\kappa.T\,X = T : \kappa \to \kappa'}\ X \notin \mathsf{dom}(\Xi)$$

symmetry transitivity $$\frac{}{\Xi \vdash C = C : \Sigma(C)} \qquad \frac{}{\Xi \vdash X = X : \Xi(X)}$$

$$\frac{\Xi, X{:}\kappa \vdash T = T' : \kappa'}{\Xi \vdash \lambda X{:}\kappa.T = \lambda X{:}\kappa.T' : \kappa \to \kappa'} \qquad \frac{\Xi \vdash T = T' : \kappa \to \kappa' \qquad \Xi \vdash U = U' : \kappa}{\Xi \vdash T\,U = T'\,U' : \kappa'}$$

Typing $\Xi;\Gamma \vdash t : T$. "In contexts Ξ, Γ, object t has type T."

$$\frac{\Xi \vdash \Gamma}{\Xi;\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Xi;\Gamma, x{:}U \vdash t : T}{\Xi;\Gamma \vdash \lambda x{:}U.t : U \to T} \qquad \frac{\Xi;\Gamma \vdash t : U \to T \qquad \Xi;\Gamma \vdash u : U}{\Xi;\Gamma \vdash t\,u : T}$$

$$\frac{\Xi \vdash T : \kappa \to \star \qquad \Xi, X{:}\kappa;\Gamma \vdash t : T\,X}{\Xi;\Gamma \vdash \Lambda X{:}\kappa.t : \forall^{\kappa} T}\ X \notin \mathsf{dom}(\Xi) \qquad \frac{\Xi;\Gamma \vdash t : \forall^{\kappa} T \qquad \Xi \vdash U : \kappa}{\Xi;\Gamma \vdash t\,U : T\,U}$$

$$\frac{\Xi;\Gamma \vdash t : T \qquad \Xi \vdash T = T' : \star}{\Xi;\Gamma \vdash t : T'}$$

Object equality $\Xi;\Gamma \vdash t = t' : T$. "In contexts Ξ, Γ, objects t and t' are $\beta\eta$-equal of type T."

$$\frac{\Xi;\Gamma, x{:}U \vdash t : T \qquad \Xi;\Gamma \vdash u : U}{\Xi;\Gamma \vdash (\lambda x{:}U.t)\,u = t[u/x] : T} \qquad \frac{\Xi;\Gamma \vdash t : U \to T}{\Xi;\Gamma \vdash \lambda x{:}U.t\,x = t : U \to T}\ x \notin \mathsf{dom}(\Gamma)$$

$$\frac{\Xi, X{:}\kappa;\Gamma \vdash t : T \qquad \Xi \vdash U : \kappa}{\Xi;\Gamma \vdash (\Lambda X{:}\kappa.t)\,U = t[U/X] : T[U/X]} \qquad \frac{\Xi;\Gamma \vdash t : \forall^{\kappa} T}{\Xi;\Gamma \vdash \Lambda X{:}\kappa.t\,X = t : \forall^{\kappa} T}\ X \notin \mathsf{dom}(\Xi)$$

symmetry transitivity

$$\frac{\Xi \vdash \Gamma}{\Xi;\Gamma \vdash x = x : \Gamma(x)} \qquad \frac{\Xi \vdash U = U' : \star \qquad \Xi;\Gamma, x{:}U \vdash t = t' : T}{\Xi;\Gamma \vdash \lambda x{:}U.t = \lambda x{:}U'.t' : U \to T}$$

$$\frac{\Xi;\Gamma \vdash t = t' : U \to T \qquad \Xi;\Gamma \vdash u = u' : U}{\Xi;\Gamma \vdash t\,u = t'\,u' : T}$$

$$\frac{\Xi \vdash T : \kappa \to \star \qquad \Xi, X{:}\kappa;\Gamma \vdash t = t' : T\,X}{\Xi;\Gamma \vdash \Lambda X{:}\kappa.t = \Lambda X{:}\kappa.t' : \forall^{\kappa} T}\ X \notin \mathsf{dom}(\Xi) \qquad \frac{\Xi;\Gamma \vdash t = t' : \forall^{\kappa} T \qquad \Xi \vdash U = U' : \kappa}{\Xi;\Gamma \vdash t\,U = t'\,U' : T\,U}$$

$$\frac{\Xi;\Gamma \vdash t = t' : T \qquad \Xi \vdash T = T' : \star}{\Xi;\Gamma \vdash t = t' : T'}$$

Fig. 1. F^{ω}: kinding, type equality, typing, object equality.

3 Abstract Normalization by Evaluation

In the following, we present normalization by evaluation (NbE) for System F^ω for an abstract domain D of *values* and type values. This leaves the freedom to implement values in different ways, e. g., β-normal forms, weak head normal forms (as in Pollack's constructive engine [Pol94b]), closures (as in Coquand's type checker [Coq96]), tagged functions (Epigram 2 [CAM07]) or virtual machine instructions (compiled reduction in Coq [GL02]). All implementations of values that satisfy the interface given in the following can be used with our NbE algorithm, and in this article we provide a framework to prove all these implementations correct.

In this section, we will understand functions in terms of a programming language, i. e., partial and possibly non-terminating. We unify the syntax of kinds, types, and objects into a grammar of expressions Exp. Let $\mathsf{Var} = \mathsf{TyVar} \cup \mathsf{ObjVar}$.

$$\begin{array}{lll} \text{Expressions} & \mathsf{Exp} \ni M, N ::= \star \mid C \mid X \mid x \mid \lambda x\!:\!M.\,N \mid \varLambda X\!:\!M.\,N \mid M\,N \\ \text{Values} & \mathsf{D} \ \ \ni d, e, f, A, B, F, G \quad \text{(abstract)} \end{array}$$

Environments Env are finite maps from variables to values. Look-up of variable x in environment ρ is written $\rho(x)$, update of environment ρ with new value v for variable x is written $\rho[x \mapsto v]$, and the empty environment is written $\diamond$. The call $\mathsf{fresh}(\rho)$ returns a variable x which is not in $\mathsf{dom}(\rho)$.

Application and evaluation (see Fig. 2) make values into a *syntactical applicative structure* [Bar84, 5.3.1], provided the equations below are satisfied. Such structures will appear later, in a sorted setting, as type and object structures (defs. 3 and 19). Note that establishing the laws of evaluation can be arbitrarily hard, e. g., if $(\!|\_|\!)_\_$ involves an optimizing compiler.

Values are converted back to expressions through reification. However, this process can only be implemented for *term-like* value domains, in particular, we require an embedding of variables into D, and an analysis neView of values that arise as iterated application of a variable (a so-called *neutral* value) or as iterated application of a constant (a *constructed* value). Some constructed values are types or kinds, they are analyzed by tyView, which can actually be defined from neView.

Values d of type V in context $\varDelta$, which assigns type values to variables, are reified by a call to $\searrow^{\Uparrow}(\varDelta, d, V)$. It is mutually defined with $\searrow^{\Uparrow}(\varDelta, n)$ which returns the normal form M and type V of neutral value n. Later in this article, reification will be presented as two relations $\varDelta \vdash d \searrow M \Downarrow\!\Uparrow V$ such that $\varDelta \vdash d \searrow M \Uparrow V$ iff $\searrow^{\Uparrow}(\varDelta, d, V) = M$ and $\varDelta \vdash d \searrow M \Downarrow V$ iff $\searrow^{\Downarrow}(\varDelta, d) = (M, V)$.

NbE is now obtained as reification after evaluation. For closed expressions M of type or kind N we define

$$\mathsf{nbe}(M, N) = \searrow^{\Uparrow}(\diamond, (\!|M|\!)_\diamond, \mathsf{tyView}(\!|N|\!)_\diamond).$$

A concrete instance of NbE is obtained by defining a recursive data type D with the constructors:

$$\begin{array}{ll} \mathsf{Constr} & : \mathsf{TyCst} \to \mathsf{D}^* \to \mathsf{D} \\ \mathsf{Ne} & : \mathsf{Var} \to \mathsf{D}^* \to \mathsf{D} \\ \mathsf{Abs} & : (\mathsf{D} \to \mathsf{D}) \to \mathsf{D} \end{array}$$

Applicative structure D of values.

$$\begin{array}{lll}
\text{Application} & \_\cdot\_ & : \mathsf{D} \to \mathsf{D} \to \mathsf{D} \\
\text{Evaluation} & \llparenthesis \_ \rrparenthesis_{\_} & : \mathsf{Exp} \to \mathsf{Env} \to \mathsf{D} \\
 & \llparenthesis x \rrparenthesis_\rho & = \rho(x) \\
 & \llparenthesis \lambda x : M.\, N \rrparenthesis_\rho \cdot d & = \llparenthesis N \rrparenthesis_{\rho[x \mapsto d]} \\
 & \llparenthesis X \rrparenthesis_\rho & = \rho(X) \\
 & \llparenthesis \Lambda X : M.\, N \rrparenthesis_\rho \cdot G & = \llparenthesis N \rrparenthesis_{\rho[X \mapsto G]} \\
 & \llparenthesis M\, N \rrparenthesis_\rho & = \llparenthesis M \rrparenthesis_\rho \cdot \llparenthesis N \rrparenthesis_\rho
\end{array}$$

D is term-like.

$$\begin{array}{llll}
\text{Embedding} & \mathsf{var} & : & \mathsf{Var} \to \mathsf{D} \\
\text{View as neutral} & \mathsf{NeView} \ni n & ::= & C \mid X \mid x \mid e\, d \\
 & \mathsf{neView} & : & \mathsf{D} \to \mathsf{NeView} \\
 & \mathsf{neView} \llparenthesis C \rrparenthesis_\rho & = & C \\
 & \mathsf{neView}(\mathsf{var}\, X) & = & X \\
 & \mathsf{neView}(\mathsf{var}\, x) & = & x \\
 & \mathsf{neView}(e \cdot d) & = & e\, d \quad \text{if } \mathsf{neView}\, e \text{ is defined} \\
\text{View as type} & \mathsf{TyView} \ni V & ::= & \star \mid A \to B \mid \forall^\kappa F \\
 & \mathsf{tyView} & : & \mathsf{D} \to \mathsf{TyView} \\
 & \mathsf{tyView}\, \llparenthesis \star \rrparenthesis_\rho & = & \star \\
 & \mathsf{tyView}\, \llparenthesis M \to N \rrparenthesis_\rho & = & \mathsf{tyView}\, \llparenthesis M \rrparenthesis_\rho \to \mathsf{tyView}\, \llparenthesis N \rrparenthesis_\rho \\
 & \mathsf{tyView}\, \llparenthesis \forall^\kappa M \rrparenthesis_\rho & = & \forall^\kappa\, \mathsf{tyView}\, \llparenthesis M \rrparenthesis_\rho
\end{array}$$

Reification.

$$\begin{array}{ll}
\searrow^{\Uparrow} & : \mathsf{Env} \to \mathsf{D} \to \mathsf{TyView} \to \mathsf{Exp} \\
\searrow^{\Uparrow}(\Delta, f, A \to B) & = \mathsf{let}\ \ x \qquad\ = \mathsf{fresh}(\Delta) \\
 & \qquad\quad (U, \_) = \searrow^{\Downarrow}(\Delta, \mathsf{neView}\, A) \\
 & \quad \mathsf{in}\ \lambda x : U.\ \searrow^{\Uparrow}(\Delta[x \mapsto A], f \cdot \mathsf{var}\, x, \mathsf{tyView}\, B) \\
\searrow^{\Uparrow}(\Delta, d, \forall^\kappa F) & = \mathsf{let}\ X = \mathsf{fresh}(\Delta)\ \mathsf{in}\ \Lambda X : \kappa.\ \searrow^{\Uparrow}(\Delta[X \mapsto \kappa], d \cdot \mathsf{var}\, X, \mathsf{tyView}(F \cdot \mathsf{var}\, X)) \\
\searrow^{\Uparrow}(\Delta, e, \star) & = \mathsf{let}\ (M, \_) = \searrow^{\Downarrow}(\Delta, \mathsf{neView}\, e)\ \mathsf{in}\ M
\end{array}$$

$$\begin{array}{ll}
\searrow^{\Downarrow} & : \mathsf{Env} \to \mathsf{NeView} \to \mathsf{Exp} \times \mathsf{TyView} \\
\searrow^{\Downarrow}(\Delta, C) & = (C, \Sigma(C)) \\
\searrow^{\Downarrow}(\Delta, X) & = (X, \mathsf{tyView}(\Delta(X))) \\
\searrow^{\Downarrow}(\Delta, x) & = (x, \mathsf{tyView}(\Delta(x))) \\
\searrow^{\Downarrow}(\Delta, e\, d) & = \mathsf{let}\ (M, V) = \searrow^{\Downarrow}(\Delta, e)\ \mathsf{in\ case}\ V\ \mathsf{of} \\
 & \qquad A \to B \mapsto (M\ (\searrow^{\Uparrow}(\Delta, d, \mathsf{tyView}\, A)), \mathsf{tyView}\, B) \\
 & \qquad \forall^\kappa F \quad\ \ \mapsto (M\ (\searrow^{\Uparrow}(\Delta, d, \kappa)), \mathsf{tyView}(F \cdot d))
\end{array}$$

Normalization by evaluation.

$$\mathsf{nbe}(M, N) = \searrow^{\Uparrow}(\diamond, \llparenthesis M \rrparenthesis_\diamond, \mathsf{tyView} \llparenthesis N \rrparenthesis_\diamond)$$

Fig. 2. Specification of an NbE algorithm.

Application, evaluation, and variable embedding are given by the following equations.

$$
\begin{array}{lll}
(\mathsf{Constr}\, C\ Gs) \cdot G & = \mathsf{Constr}\, C\ (Gs, G) & \\
(\mathsf{Ne}\, x\ ds) \quad\ \cdot d & = \mathsf{Ne}\, x\ (ds, d) & \\
(\mathsf{Abs}\, f) \qquad\ \cdot d & = f\ d & \\
(\!|\lambda x\!:\!M.\,N|\!)_\rho & = \mathsf{Abs}\, f & \text{where } f\ d = (\!|N|\!)_{\rho[x\mapsto d]} \\
(\!|\Lambda X\!:\!M.\,N|\!)_\rho & = \mathsf{Abs}\, f & \text{where } f\ G = (\!|N|\!)_{\rho[X\mapsto G]} \\
(\!|C|\!)_\rho & = \mathsf{Constr}\, C\ () & \\
\mathsf{var}\, X & = \mathsf{Ne}\, X\ () & \\
\mathsf{var}\, x & = \mathsf{Ne}\, x\ () &
\end{array}
$$

This instance of NbE is now easily completed using the equations of the specification, and can be implemented directly in Haskell.

In this article we show that *any instance* of the NbE-specification terminates with the correct result for well-formed expressions of F^ω, i. e., we show the following two properties:

1. Soundness: if $\vdash M : N$, then $\vdash \mathsf{nbe}(M, N) = M : N$.
2. Completeness: if $\vdash M : N$ and $\vdash M' : N$, then $\mathsf{nbe}(M, N) = \mathsf{nbe}(M', N)$ (same expression up to α).

In contrast to the untyped presentation in this section, which saves us from some repetition, we will distinguish the three levels of F^ω consequently in the remainder of the article.

4 Kind Structures

Definition 1 (Kind structure). *A* kind structure *is a set $\mathbb{K}$ with a distinguished element $\star \in \mathbb{K}$ and a binary operation $\to \in \mathbb{K} \to \mathbb{K} \to \mathbb{K}$.*

The free kind structure is Ki. A kind structure can also consist of semantic kinds which are *sets of types* drawn from a type structure $\mathcal{T}$. In the following, we present this abstractly.

Definition 2 (Kind candidate space). *Let $(\mathbb{K}^\kappa, \sqsubseteq^\kappa)$ be a Ki-indexed family of posets with a (polymorphic) binary operation $\to \in \mathbb{K}^\kappa \to \mathbb{K}^{\kappa'} \to \mathbb{K}^{\kappa\to\kappa'}$. A kind candidate space $\mathbb{C}$ consists of two families $\underline{\mathbb{C}}^\kappa, \overline{\mathbb{C}}^\kappa \in \mathbb{K}^\kappa$ indexed by $\kappa \in \mathsf{Ki}$, written $\underline{\kappa}, \overline{\kappa}$ if no ambiguity arises, such that*

$$
\begin{array}{lll}
\text{K-BASE} & \underline{\star} & \sqsubseteq \overline{\star}, \\
\text{K-FUN-I} & \underline{\kappa} \to \overline{\kappa'} & \sqsubseteq \overline{\kappa \to \kappa'}, \\
\text{K-FUN-E} & \underline{\kappa \to \kappa'} & \sqsubseteq \overline{\kappa} \to \underline{\kappa'}.
\end{array}
$$

For $K \in \mathbb{K}^\kappa$ we write $\kappa \Vdash_{\mathbb{C}} K$ and say κ realizes K w. r. t. $\mathbb{C}$ *iff $\underline{\kappa} \sqsubseteq K \sqsubseteq \overline{\kappa}$. For a family $K^\kappa \in \mathbb{K}^\kappa$ we write $\Vdash_{\mathbb{C}} K$ if $\kappa \Vdash_{\mathbb{C}} K^\kappa$ for all κ.*

Lemma 1 (Realizability kind structure). *In the context of the above definition: If $\to$ is antitone in its first argument and monotone in its second argument, then $\mathbb{C} := \{(\kappa, K) \mid \kappa \Vdash_{\mathbb{C}} K\}$ constitutes a kind structure with distinguished element $(\star, \underline{\star})$ and operation $(\kappa, K) \to (\kappa', K') = (\kappa \to \kappa', K \to K')$.*

Proof. We have $\star \Vdash \underline{\star}$, and $\kappa \Vdash K$ and $\kappa' \Vdash K'$ imply $\kappa \to \kappa' \Vdash K \to K'$. □

5 Type Structures

In this section, we define type structures as an abstraction over syntactic types, type values, and semantic types. Type structures form a category which has finite products. Let $\mathsf{Ty}^{\kappa}_{\Xi} = \{T \mid \Xi \vdash T : \kappa\}$.

Definition 3 (Type structure). *An* (F^{ω}) type structure *is a tuple* $(\mathcal{T}, \mathsf{Cst}, \mathsf{App}, [\![\_]\!]_{\_})$ *where $\mathcal{T}$ is a Kripke family $\mathcal{T}^{\kappa}_{\Xi}$ of sets with the following Kripke families of maps:*

$$\begin{array}{ll} \mathsf{Cst}_{\Xi} & \in (C \in \mathsf{TyCst}) \to \mathcal{T}^{\Sigma(C)}_{\Xi} \\ \mathsf{App}^{\kappa\to\kappa'}_{\Xi} & \in \mathcal{T}^{\kappa\to\kappa'}_{\Xi} \to \mathcal{T}^{\kappa}_{\Xi} \to \mathcal{T}^{\kappa'}_{\Xi} \end{array}$$

Usually, we will just write $F \cdot G$ for $\mathsf{App}^{\kappa\to\kappa'}_{\Xi}(F, G)$. Let $\rho \in \mathcal{T}^{\Xi}_{\Theta}$ iff $\rho(X) \in \mathcal{T}^{\Xi(X)}_{\Theta}$ for all $X \in \mathsf{dom}(\Xi)$. The interpretation function has the following properties:

$$\begin{array}{ll} [\![\_]\!]_{\_} & \in \mathsf{Ty}^{\kappa}_{\Xi} \to \mathcal{T}^{\Xi}_{\Theta} \to \mathcal{T}^{\kappa}_{\Theta} \\ [\![C]\!]_{\rho} & = \mathsf{Cst}_{\Theta}(C) \\ [\![X]\!]_{\rho} & = \rho(X) \\ [\![\lambda X\!:\!\kappa.\, T]\!]_{\rho} \cdot G & = [\![T]\!]_{\rho[X\mapsto G]} \\ [\![T\, U]\!]_{\rho} & = [\![T]\!]_{\rho} \cdot [\![U]\!]_{\rho} \\ [\![T[U/X]]\!]_{\rho} & = [\![T]\!]_{\rho[X\mapsto [\![U]\!]_{\rho}]} \quad (*) \end{array}$$

If the condition $(*)$ is fulfilled, we speak of a *combinatory* type structure, since $(*)$ is a characterizing property of combinatory algebras. The condition $(*)$ is only necessary since we chose to use eager substitution in the inference rules of F^{ω}, it can be dropped when switching to explicit substitutions [ACD08].

We use “interpretation” and “evaluation” synonymously. Note that while the equations determine the interpretation of constants, variables, and application, there is some freedom in the interpretation of functions $[\![\lambda X\!:\!\kappa.\, T]\!]_{\rho}$. It could be lambda-terms (taking $\mathcal{T} = \mathsf{Ty}$), set-theoretical functions (see Def. 28), functional values in an interpreter, machine code etc.

Since Cst_{Ξ} is independent of Ξ, we have $\mathsf{Cst}_{\Xi} = \mathsf{Cxt}_{\diamond}$, we usually suppress the index Ξ in Cst_{Ξ}. We may even drop Cst altogether, i. e., we just write $\to\, \in \mathcal{T}^{\star\to\star\to\star}_{\Xi}$ instead of $\mathsf{Cst}(\to) \in \mathcal{T}^{\star\to\star\to\star}_{\Xi}$.

To avoid ambiguities when different type structures are in scope, we may write $\to_{\mathcal{T}}$, $\forall^{\kappa}_{\mathcal{T}}$, $\_ \cdot_{\mathcal{T}} \_$ and $\mathcal{T}[\![\_]\!]_{\_}$ to emphasize that we mean the type structure operations of $\mathcal{T}$.

Simple examples of type structures are Ty and Ty modulo β, $\beta\eta$, or judgmental equality. In these instances, the interpretation function is parallel substitution.

Our notion of type structure essentially coincides with Barendregt's *syntactical applicative structure* [Bar84, 5.3.1], except that we are working in a kinded setting and Barendregt in untyped lambda-calculus.

Definition 4 (Type structure morphism). *Given two type structures $\mathcal{S}$ and $\mathcal{T}$, a* type structure morphism $M : \mathcal{S} \to \mathcal{T}$ *is a Kripke family of maps $M^{\kappa}_{\Xi} \in \mathcal{S}^{\kappa}_{\Xi} \to \mathcal{T}^{\kappa}_{\Xi}$ that commute with the operations of $\mathcal{S}$, i. e.,*

$$\begin{array}{lll} M^{\kappa}_{\Xi}(C_{\mathcal{S}}) & = C_{\mathcal{T}} & (C:\kappa) \in \Sigma \\ M^{\kappa'}_{\Xi}(F \cdot_{\mathcal{S}} G) & = M^{\kappa \to \kappa'}_{\Xi}(F) \cdot_{\mathcal{T}} M^{\kappa}_{\Xi}(G) & F \in \mathcal{S}^{\kappa \to \kappa'}_{\Xi}, G \in \mathcal{S}^{\kappa}_{\Xi} \\ M^{\kappa}_{\Theta}(\mathcal{S}[\![T]\!]_{\rho}) & = \mathcal{T}[\![T]\!]_{M^{\Xi}_{\Theta} \circ \rho} & T \in \mathsf{Ty}^{\kappa}_{\Xi}, \rho \in \mathcal{S}^{\Xi}_{\Theta} \end{array}$$

where $(M^{\Xi}_{\Theta} \circ \rho)(X) := M^{\Xi(X)}_{\Theta}(\rho(X))$.

Definition 5 (Environment pairing). *Given two type structures $\mathcal{S}, \mathcal{T}$ and environments $\rho \in \mathcal{S}^{\Xi}_{\Theta}$ and $\rho' \in \mathcal{T}^{\Xi}_{\Theta}$, we define*

$$(\rho, \rho') \in (X \in \mathsf{dom}(\Xi)) \to \mathcal{S}^{\Xi(X)}_{\Theta} \times \mathcal{T}^{\Xi(X)}_{\Theta}$$

pointwise by $(\rho, \rho')(X) = (\rho(X), \rho'(X))$.

For the environment update operation then holds $(\rho, \rho')[X \mapsto (G, G')] = (\rho[X \mapsto G], \rho'[X \mapsto G'])$.

Def. and Lem. 1 (Product type structure) *Given two type structures $\mathcal{S}, \mathcal{T}$, the pointwise product*

$$(\mathcal{S} \times \mathcal{T})^{\kappa}_{\Xi} = \mathcal{S}^{\kappa}_{\Xi} \times \mathcal{T}^{\kappa}_{\Xi}$$

forms a new type structure with

$$\begin{array}{lll} C_{\mathcal{S} \times \mathcal{T}} & := (C_{\mathcal{S}}, C_{\mathcal{T}}) & \text{for all } C \in \mathsf{TyCst} \\ (F, F') \cdot (G, G') & := (F \cdot G, F' \cdot G') & \\ [\![F]\!]_{\rho, \rho'} & := ([\![F]\!]_{\rho}, [\![F]\!]_{\rho'}). & \end{array}$$

Proof. One easily validates the laws

$$\begin{array}{ll} [\![X]\!]_{\rho,\rho'} & = (\rho, \rho')(X) \\ [\![\lambda X\!:\!\kappa.\, T]\!]_{\rho,\rho'} \cdot (G, G') & = [\![T]\!]_{(\rho,\rho')[X \mapsto (G, G')]} \\ [\![T\, U]\!]_{\rho,\rho'} & = [\![T]\!]_{\rho,\rho'} \cdot [\![U]\!]_{\rho,\rho'} \end{array}$$

using the definition of application and evaluation. □

The two projections $\pi_1 : \mathcal{S} \times \mathcal{T} \to \mathcal{S}$ and $\pi_2 : \mathcal{S} \times \mathcal{T} \to \mathcal{T}$ are trivially type structure morphisms, and $\times$ is a product in the category of type structures and their morphisms.

5.1 Type Substructures and the Fundamental Theorem for Kinding

Definition 6 (Type substructure). *Given a type structure $\mathcal{T}$, the Kripke family $\mathcal{S}^{\kappa}_{\Xi} \subseteq \mathcal{T}^{\kappa}_{\Xi}$ is a* type substructure of $\mathcal{T}$ *if*

$$\begin{array}{lll} C_{\mathcal{T}} & \in \mathcal{S}^{\kappa}_{\Xi} & \textit{for all } (C\!:\!\kappa) \in \Sigma \\ \_ \cdot_{\mathcal{T}} \_ & \in \mathcal{S}^{\kappa\to\kappa'}_{\Xi} \to \mathcal{S}^{\kappa}_{\Xi} \to \mathcal{S}^{\kappa'}_{\Xi} & \\ \mathcal{T}[\![\_]\!]_{\_} & \in \mathsf{Ty}^{\kappa}_{\Xi} \to \mathcal{S}^{\Xi}_{\Theta} \to \mathcal{S}^{\kappa}_{\Theta} & \end{array}$$

Basically, a type substructure is a subfamily which is still a type structure with the original operations. One could also say that $\mathcal{S}$ is a type substructure of $\mathcal{T}$ iff the identity on $\mathcal{S}$ is a type structure morphism from $\mathcal{S}$ to $\mathcal{T}$. However, then the notion of type structure morphism must be set up more liberally, not requiring $\mathcal{S}$ to be a type structure *a priori*. In the following we simply write $\mathcal{S} \subseteq \mathcal{T}$ to mean $\mathcal{S}^{\kappa}_{\Xi} \subseteq \mathcal{T}^{\kappa}_{\Xi}$ for all κ, Ξ.

Lemma 2 (Projection type substructure). *If $\mathcal{S} \subseteq \mathcal{T}_1 \times \mathcal{T}_2$ is a type substructure, so are $\pi_1(\mathcal{S}) \subseteq \mathcal{T}_1$ and $\pi_2(\mathcal{S}) \subseteq \mathcal{T}_2$.*

Definition 7 (Function space). *We write $K \in \widehat{\mathcal{T}}^{\kappa}$ if K is a Kripke family of subsets $K_{\Xi} \subseteq \mathcal{T}^{\kappa}_{\Xi}$. Given $K \in \widehat{\mathcal{T}}^{\kappa}$ and $K' \in \widehat{\mathcal{T}}^{\kappa'}$ we define the* Kripke function space

$$(K \to_{\widehat{\mathcal{T}}} K')_{\Xi} = \{F \in \mathcal{T}^{\kappa\to\kappa'}_{\Xi} \mid F \cdot G \in K'_{\Xi'} \textit{ for all } \Xi' \leq \Xi \textit{ and } G \in K_{\Xi'}\}$$

The set $\{(\kappa, K) \mid K \subseteq \mathcal{T}^{\kappa}$ Kripke $\}$ forms the powerset kind structure *with operation $(\kappa, K) \to (\kappa', K') = (\kappa \to \kappa', K \to_{\widehat{\mathcal{T}}} K')$ and an arbitrarily chosen distinguished element $(\star, K_0)$ with $K_0 \subseteq \mathcal{T}^{\star}$.*

If no ambiguities arise, we write $\to$ for $\to_{\widehat{\mathcal{T}}}$.

Definition 8 (Induced type structure). *Let $\mathcal{T}$ be a type structure and $\mathcal{S} \subseteq \mathcal{T}$ be Kripke. If*

$$\begin{array}{lll} C_{\mathcal{T}} & \in \mathcal{S}^{\kappa}_{\Xi} & \textit{for all } (C\!:\!\kappa) \in \Sigma \\ \mathcal{S}^{\kappa\to\kappa'}_{\Xi} & = (\mathcal{S}^{\kappa} \to_{\widehat{\mathcal{T}}} \mathcal{S}^{\kappa'})_{\Xi} & \end{array}$$

then $\mathcal{S}$ is called induced *or an* induced type substructure *of $\mathcal{T}$ (see Thm. 2).*

Such an $\mathcal{S}$ is called induced since it is already determined by the choice of the denotation of the base kind $\mathcal{S}^{\star}$.

Theorem 2 (Fundamental theorem of kinding). *Let $\mathcal{T}$ be a type structure. If $\mathcal{S} \subseteq \mathcal{T}$ is induced, then $\mathcal{S}$ is a type substructure of $\mathcal{T}$.*

Proof. We mainly need to show that evaluation is well-defined. This is shown by induction on the kinding derivation, as usual.

Let $\rho \in \mathcal{S}^{\Xi}_{\Theta}$. We show $(\!|T|\!)_{\rho} \in \mathcal{S}^{\kappa}_{\Theta}$ by induction on $\Xi \vdash T : \kappa$. The cases for variables and constants are trivial.

Case

$$\frac{\Xi, X\!:\!\kappa \vdash T : \kappa'}{\Xi \vdash \lambda X\!:\!\kappa.\, T : \kappa \to \kappa'}$$

Since $\mathcal{S}$ is induced, it is sufficient to show $(\!|\lambda X : \kappa.\, T|\!)_\rho \cdot G \in \mathcal{S}^{\kappa'}_{\Theta'}$ for arbitrary $\Theta' \leq \Theta$ and $G \in \mathcal{S}^{\kappa}_{\Theta'}$. This follows from the induction hypothesis $(\!|T|\!)_{\rho'} \in \mathcal{S}^{\kappa'}_{\Theta'}$ with $\rho' = \rho[X \mapsto G] \in \mathcal{S}^{\Xi, X:\kappa}_{\Theta'}$.

Case

$$\frac{\Xi \vdash T : \kappa \to \kappa' \qquad \Xi \vdash U : \kappa}{\Xi \vdash T\, U : \kappa'}$$

Since $\mathcal{S}^{\kappa \to \kappa'} \subseteq \mathcal{S}^{\kappa} \to_{\widehat{\mathcal{T}}} \mathcal{S}^{\kappa'}$, the goal follows by the induction hypotheses and $(\!|T\, U|\!)_\rho = (\!|T|\!)_\rho \cdot (\!|U|\!)_\rho$.

□

5.2 NbE for Types and Its Soundness

We are ready to define the reification relation for type values and show that NbE, i. e., the composition of evaluation of a syntactic type T and reification to a normal form V, is sound, i. e., T and V are judgmentally equal. As a byproduct, we show totality of NbE on well-kinded types. The structure $\mathcal{T}$ of type values is left abstract, a concrete definition will be given in Sect. **??**. However, not every $\mathcal{T}$ permits reification of its inhabitants. It needs to include the variables which need to be distinguishable from each other and other type values. *Neutral* types, i. e., of the shape $X \cdot \boldsymbol{G}$, need to be analyzable into head X and spine $\boldsymbol{G}$. We call a suitable $\mathcal{T}$ *term-like*; on such a $\mathcal{T}$ we can define contextual reification [ACD08,Abe08].

Definition 9 (Type structure with variables). *A type structure $\mathcal{T}$* has variables *if there exists a Kripke family of maps* $\mathsf{Var}_\Xi \in (X \in \mathsf{dom}(\Xi)) \to \mathcal{T}^{\Xi(X)}_{\Xi}$ *such that* $\mathsf{Var}_\Xi(X) \in \mathcal{T}^{\kappa'}_{\Xi'}$ *iff $\Xi'(X) = \kappa'$. Usually, we simply write X for* $\mathsf{Var}_\Xi(X)$.

1. *Types $X \cdot \boldsymbol{G}$ are called* neutral.
2. *Types $C \cdot \boldsymbol{G}$ are called* constructed.

If clear from the context of discourse, we drop the index Ξ to Var. To disambiguate, we sometimes write $\mathsf{Var}_{\mathcal{T}}$ to refer to the variable embedding of type structure $\mathcal{T}$. Usually, we write $\mathcal{T}[\![T]\!]$ instead of $\mathcal{T}[\![T]\!]_{\mathsf{Var}_{\mathcal{T}}}$ for the interpretation in the identity environment.

Definition 10 (Term-like type structure). *A type structure $\mathcal{T}$ is* term-like *if it has variables and there exists a Kripke family of partial maps*

$$\begin{array}{rl}\mathsf{View}^{\kappa}_{\Xi} \in \mathcal{T}^{\kappa}_{\Xi} \rightharpoonup & \{(C, \boldsymbol{G}) \in \mathsf{TyCst} \times \mathcal{T}^{\boldsymbol{\kappa}}_{\Xi} \mid \Sigma(C) = \boldsymbol{\kappa} \to \kappa\} \\ + & \{(X, \boldsymbol{G}) \in \mathsf{TyVar} \times \mathcal{T}^{\boldsymbol{\kappa}}_{\Xi} \mid \Xi(X) = \boldsymbol{\kappa} \to \kappa\}\end{array}$$

such that:

- $\mathsf{View}(F) = (C, \boldsymbol{G})$ *iff* $F = \mathsf{Cst}(C) \cdot \boldsymbol{G}$.
- $\mathsf{View}(F) = (X, \boldsymbol{G})$ *iff* $F = \mathsf{Var}(X) \cdot \boldsymbol{G}$.

Lemma 3 (Injectivity in term-like type structures). *In a term-like type structure $\mathcal{T}$,* Cst *and* Var *are families of* injective *maps and neutral and constructed application is injective, i. e., $H \cdot G = H' \cdot G'$ implies $H = H'$ and $G = G'$ for neutral or constructed H, H'.*

Proof. Assume, for instance $H = X \cdot \boldsymbol{G}, H' = X' \cdot \boldsymbol{G}'$ neutral. We have $(X; \boldsymbol{G}, G) = \mathsf{View}(H \cdot G) = \mathsf{View}(H' \cdot G') = (X'; \boldsymbol{G}', G')$, thus $X = X'$, $\boldsymbol{G} = \boldsymbol{G}'$, and $G = G'$. Immediately, $H = H'$ follows. □

Trivially, Ty is a term-like type structure with $\mathsf{Var}_{\mathsf{Ty}}(X) = X$. The product of term-like type structures $\mathcal{S}$ and $\mathcal{T}$ is again term-like with $\mathsf{Var}_{\mathcal{S}\times\mathcal{T}} = (\mathsf{Var}_{\mathcal{S}}, \mathsf{Var}_{\mathcal{T}})$. $\mathsf{Ty}/{=_\beta}$ is term-like due to confluence of β-reduction. Ty modulo judgmental equality is not trivially term-like, since first injectivity of the type constructors needs to be proven.

Definition 11 (Reifiable type structure). *A type structure $\mathcal{T}$ with variables is* reifiable*, if there are relations*

$$\begin{array}{ll} \Xi \vdash F \searrow V \Uparrow \kappa & \text{in } \Xi,\ F \text{ reifies to } V \text{ at kind } \kappa, \\ \Xi \vdash H \searrow U \Downarrow \kappa & \text{in } \Xi,\ H \text{ reifies to } U, \text{ inferring kind } \kappa, \end{array}$$

(where $F, H \in \mathcal{T}^\kappa_\Xi$ with H neutral or constructed, and $V, U \in \mathsf{Ty}^\kappa_\Xi$) which obey the following rules:

$$\frac{}{\Xi \vdash C \searrow C \Downarrow \Sigma(C)} \qquad \frac{}{\Xi \vdash X \searrow X \Downarrow \Xi(X)}$$

$$\frac{\Xi \vdash H \searrow U \Downarrow \kappa \to \kappa' \qquad \Xi \vdash G \searrow V \Uparrow \kappa}{\Xi \vdash H \cdot G \searrow U\,V \Downarrow \kappa'}$$

$$\frac{\Xi \vdash H \searrow U \Downarrow \star}{\Xi \vdash H \searrow U \Uparrow \star} \qquad \frac{\Xi, X{:}\kappa \vdash F \cdot X \searrow V \Uparrow \kappa'}{\Xi \vdash F \searrow \lambda X{:}\kappa.\, V \Uparrow \kappa \to \kappa'}$$

Further, these reification relations *must be* deterministic *in the following sense: For all Ξ, κ, F (inputs) and neutral or constructed H (input) there is at most one V (output) such that $\Xi \vdash F \searrow V \Uparrow \kappa$ and at most one U and κ' (outputs) such that $\Xi \vdash H \searrow U \Downarrow \kappa'$.*

Seen as logic programs with inputs and outputs as indicated above, these relations denote partial functions, where $\searrow^{\Uparrow}$ is defined by cases on the kind κ and and $\searrow^{\Downarrow}$ by cases on the neutral value H.

Lemma 4 (Reification returns long normal form). *If $\Xi \vdash F \searrow V \Uparrow \kappa$, then V is η-long β-normal.*

Lemma 5 (Term-like type structure is reifiable). *Any term-like type structure $\mathcal{T}$ is reifiable.*

Proof. In the presence of View, the two relations can simply be defined inductively by the above rules. □

Why did we then bother to introduce the concept of a reifiable type structure, instead of just speaking of term-like type structures and define reification inductively? It is because we will show in Sec. 6.3 that the quotient of a reifiable type structure modulo some suitable equality remains reifiable.

We continue by constructing a model of the kinding rules which proves soundness of NbE for types. Kinds κ are interpreted as sets $\mathsf{G}^{\kappa}_{\varXi}$ of pairs (F,T) *glued together* [CD97] by reification, i. e., the type value F reifies to syntactic type T up to $\beta\eta$-equality. Function kinds are interpreted via Tait's function space (see Def. 7), thus, the fundamental theorem of kinding yields that G is indeed a type structure.

Definition 12 (Glueing candidate). *Fix a reifiable type structure $\mathcal{T}$. We define the families $\underline{\mathsf{Gl}}, \overline{\mathsf{Gl}} \subseteq \mathcal{T} \times \mathsf{Ty}$ by*

$$\begin{aligned}\overline{\mathsf{Gl}}^{\kappa}_{\varXi} &= \{(F,T) \in \mathcal{T}^{\kappa}_{\varXi} \times \mathsf{Ty}^{\kappa}_{\varXi} \mid \varXi \vdash F \searrow V \Uparrow \kappa \textit{ and } \varXi \vdash T = V : \kappa\},\\ \underline{\mathsf{Gl}}^{\kappa}_{\varXi} &= \{(H,T) \in \mathcal{T}^{\kappa}_{\varXi} \times \mathsf{Ty}^{\kappa}_{\varXi} \mid \varXi \vdash H \searrow U \Downarrow \kappa \textit{ and } \varXi \vdash T = U : \kappa\}.\end{aligned}$$

A family $\mathcal{S}$ with $\underline{\mathsf{Gl}}^{\kappa} \subseteq \mathcal{S}^{\kappa} \subseteq \overline{\mathsf{Gl}}^{\kappa}$ is called a glueing candidate.

Lemma 6. *Any glueing candidate $\mathcal{S}$ contains the constants.*

Proof. Since $\varXi \vdash C \searrow C \Downarrow \Sigma(C)$ we have $\mathsf{Cst}_{\mathcal{S}}(C) = (C,C) \in \underline{\mathsf{Gl}}^{\Sigma(C)}_{\varXi} \subseteq \mathcal{S}^{\Sigma(C)}_{\varXi}$. □

Lemma 7 (Glueing candidate space). $\underline{\mathsf{Gl}}^{\kappa}$, $\overline{\mathsf{Gl}}^{\kappa}$ *form a kind candidate space according to Def. 2.*

Proof. Let us write $\underline{\kappa}, \overline{\kappa}$ for $\underline{\mathsf{Gl}}^{\kappa}, \overline{\mathsf{Gl}}^{\kappa}$. We show K-FUN-I. Assume $(F,T) \in (\underline{\kappa} \to \overline{\kappa'})_{\varXi}$ and show $(F,T) \in \overline{\kappa \to \kappa'}_{\varXi}$. We have $(X,X) \in \underline{\kappa}_{\varXi,X:\kappa}$, hence, $(F \cdot X, T\,X) \in \overline{\kappa'}_{\varXi,X:\kappa}$. That is, $\varXi, X : \kappa \vdash F \cdot X \searrow V \Uparrow \kappa'$ and $\varXi, X : \kappa \vdash T\,X = V : \kappa'$. It follows $\varXi \vdash F \searrow \lambda X{:}\kappa.\, V \Uparrow \kappa \to \kappa'$ and $\varXi \vdash \lambda X{:}\kappa.\, T\,X = \lambda X{:}\kappa.\, V : \kappa'$. Since $X \notin \mathsf{dom}(\varXi)$ we have $\varXi \vdash T = \lambda X{:}\kappa.\, V : \kappa'$. Thus, $(F,T) \in \overline{\kappa \to \kappa'}_{\varXi}$. □

The last proof makes apparent why we work with *kinded* type structures; if elements of the Kripke function space had not been well-kinded, we would not have the information $T \in \mathsf{Ty}^{\kappa\to\kappa'}_{\varXi}$ which is necessary to conclude $\varXi \vdash \lambda X{:}\kappa.\, T\,X = T : \kappa'$.

Def. and Lem. 3 (Glueing type structure) *Given a type structure $\mathcal{T}$, we define $\mathsf{G} \subseteq \mathcal{T} \times \mathsf{Ty}$ by*

$$\begin{aligned}\mathsf{G}^{\star}_{\varXi} &:= \underline{\star}_{\varXi},\\ \mathsf{G}^{\kappa\to\kappa'}_{\varXi} &:= (\mathsf{G}^{\kappa} \to_{\widehat{\mathcal{T}\times\mathsf{Ty}}} \mathsf{G}^{\kappa'})_{\varXi}.\end{aligned}$$

G *is a glueing candidate, i. e., $\underline{\mathsf{Gl}}^{\kappa} \subseteq \mathsf{G}^{\kappa} \subseteq \overline{\mathsf{Gl}}^{\kappa}$ for all κ.*

Proof. By induction on κ. By definition, $\star \Vdash_{\mathsf{Gl}} \mathsf{G}^{\star}$, and $\kappa \Vdash_{\mathsf{Gl}} \mathsf{G}^{\kappa}$ and $\kappa' \Vdash_{\mathsf{Gl}} \mathsf{G}^{\kappa'}$ imply

$$\underline{\kappa \to \kappa'} \subseteq \overline{\kappa} \to \underline{\kappa'} \subseteq \mathsf{G}^{\kappa} \to \mathsf{G}^{\kappa'} \subseteq \overline{\kappa} \to \underline{\kappa'} \subseteq \overline{\kappa \to \kappa'}.$$

□

Corollary 1. G *is induced.*

Proof. $C_{\mathsf{G}} \in \mathsf{G}_{\Xi}^{\Sigma(C)}$ since any glueing candidate contains the constants, and $\mathsf{G}^{\kappa \to \kappa'} = \mathsf{G}^{\kappa} \to \mathsf{G}^{\kappa'}$ by definition. □

Since G is induced, by the fundamental theorem of kinding it is a type substructure of $\mathcal{T} \times \mathsf{Ty}$.

Theorem 4 (Soundness of NbE for types). *Let $\mathcal{T}$ be a reifiable type structure. If $\Xi \vdash T : \kappa$ then there is a $V \in \mathsf{Ty}_{\Xi}^{\kappa}$ such that $\Xi \vdash \mathcal{T}[\![T]\!]_{\mathsf{Var}_{\mathcal{T}}} \searrow V \Uparrow \kappa$ and $\Xi \vdash T = V : \kappa$.*

Proof. Since $(X, X) \in \underline{\mathsf{Gl}}_{\Xi}^{\Xi(X)} \subseteq \mathsf{G}_{\Xi}^{\Xi(X)}$ for all $X \in \mathsf{dom}(\Xi)$, for the identity valuation $\mathsf{Var}_{\mathcal{T}\times\mathsf{Ty}}(X) = (X, X)$ we have $\mathsf{Var}_{\mathsf{G}} = \mathsf{Var}_{\mathcal{T}\times\mathsf{Ty}} \in \mathsf{G}_{\Xi}^{\Xi}$.

By the fundamental theorem, $\mathsf{G}[\![T]\!]_{\mathsf{Var}_{\mathsf{G}}} = (\mathcal{T}[\![T]\!]_{\mathsf{Var}_{\mathcal{T}}}, T) \in \mathsf{G}_{\Xi}^{\kappa}$. Since $\mathsf{G}_{\Xi}^{\kappa} \subseteq \overline{\mathsf{Gl}}_{\Xi}^{\kappa}$, we have that $\Xi \vdash \mathcal{T}[\![T]\!]_{\mathsf{Var}_{\mathcal{T}}} \searrow V \Uparrow \kappa$ with $\Xi \vdash T = V : \kappa$. □

The V returned by reification is the long normal form of T.

6 Type Groupoids

Completeness of NbE means that it models judgmental type equality, i. e., if $\Xi \vdash T = T' : \kappa$ then $\Xi \vdash [\![T]\!] \searrow V \Uparrow \kappa$ and $\Xi \vdash [\![T']\!] \searrow V \Uparrow \kappa$. Completeness will be shown by a fundamental theorem of type equality. Judgmental equality is usually modelled by partial equivalence relations (PERs), which can be seen as groupoids. Hence, we introduce the notion of a *groupoidal type structure*, or *type groupoid*. The advantage over PERs is that we can directly reuse the fundamental theorem of kinding, instantiated to a groupoidal type structure ${}^{2}\mathcal{T}$ of pairs of types, instead of having to prove this theorem again for kinds modelled as PERs.

Definition 13 (From Wikipedia, 2008-10-30:). *A* groupoid *is a set $\mathcal{G}$ with a function $\_^{-1} : \mathcal{G} \to \mathcal{G}$ and a partial function $\_ * \_ : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ that have the following properties:*

1. *Associativity: For all $a, b, c \in \mathcal{G}$, if $a * b$ and $b * c$ are defined, then $(a * b) * c$ and $a * (b * c)$ are defined and equal. Conversely, if either of these last two expressions is defined, then so is the other (and again they are equal).*
2. *Inverse: For all $a \in \mathcal{G}$, $a^{-1} * a$ and $a * a^{-1}$ are defined.*
3. *Identity: For all $a, b \in \mathcal{G}$, if $a * b$ is defined, then $a * b * b^{-1} = a$, and $a^{-1} * a * b = b$. (The previous axioms already show that these expressions are defined and unambiguous.)*

Example 1 (PER as groupoid). A partial equivalence relation R over set S is a groupoid with $(s, t)^{-1} = (t, s)$ and $(r, s) * (s, t) = (r, t)$.

Example 2 (Discrete groupoid). Any set S gives rise to a trivial groupoid with $s^{-1} = s$ and $*$ is defined exactly on the diagonal of $S \times S$, and there $s * s = s$.

Lemma 8. $(b^{-1})^{-1} = b$.

Proof. (Uli Schoepp) First, note that $b^{-1} = b^{-1} * b * b^{-1}$. Hence $(b^{-1})^{-1} = (b^{-1})^{-1} * b^{-1} * (b^{-1})^{-1} = (b^{-1})^{-1} * b^{-1} * b * b^{-1} * (b^{-1})^{-1} = b * b^{-1} * (b^{-1})^{-1} = b$. □

Lemma 9. *If $a * b$ is defined then $(a * b)^{-1} = b^{-1} * a^{-1}$.*

Proof. First $a = a * b * b^{-1}$. Since $(a * b)^{-1} * (a * b)$ is defined, $(a * b)^{-1} = (a * b)^{-1} * a * a^{-1} = (a * b)^{-1} * a * b * b^{-1} * a^{-1} = b^{-1} * a^{-1}$. □

Definition 14 (Subgroupoid). *Given a groupoid $\mathcal{G}$, a set $\mathcal{H} \subseteq \mathcal{G}$ is a subgroupoid if it is closed under inversion and composition, i.e., if*

1. *$a \in \mathcal{H}$ implies $a^{-1} \in \mathcal{H}$, and*
2. *$a, b \in \mathcal{H}$ implies $a * b \in \mathcal{H}$ if $a * b$ is defined.*

6.1 Type Groupoids and the Fundamental Theorem of Type Equality

Definition 15 (Type groupoid). *A type structure is* groupoidal *if each $\mathcal{T}^{\kappa}_{\Xi}$ is a groupoid, constants are preserved under inversion, and inversion and composition distribute over application, i. e.,*

$$\begin{array}{ll} C^{-1} & = C \\ & \quad \textit{for all } C \in \mathsf{TyCst}, \\ (F \cdot G)^{-1} & = F^{-1} \cdot G^{-1}, \\ (F \cdot G) * (F' \cdot G') & = (F * F') \cdot (G * G'). \end{array}$$

Example 3. Each type structure is groupoidal for the trivial choice of inversion (identity) and composition (idempotent, defined on the diagonal).

Def. and Lem. 5 (Square type groupoid) *Given a type structure $\mathcal{T}$ we define the square type groupoid ${}^2\mathcal{T}$ as the product type structure $\mathcal{T} \times \mathcal{T}$ equipped with*

$$\begin{array}{ll} (F, G)^{-1} & = (G, F), \\ (F, G) * (G, H) & = (F, H). \end{array}$$

Proof. The laws of a type groupoid are satisfied, e.g., for the last law we have

$$\begin{array}{l} \quad ((F_1, F_2) * (F_2, F_3)) \cdot ((G_1, G_2) * (G_2, G_3)) \\ = (F_1, F_3) \cdot (G_1, G_3) \\ = (F_1 \cdot G_1, F_3 \cdot G_3) \\ = (F_1 \cdot G_1, F_2 \cdot G_2) * (F_2 \cdot G_2, F_3 \cdot G_3) \\ = ((F_1, F_2) \cdot (G_1, G_2)) * ((F_2, F_3) \cdot (G_2, G_3)) \end{array}$$

□

Lemma 10 (Function space is groupoid). *If $K \in \widehat{\mathcal{T}}^{\kappa}$ and $K' \in \widehat{\mathcal{T}}^{\kappa'}$ are groupoids, so is $K \to_{\widehat{\mathcal{T}}} K' \in \widehat{\mathcal{T}}^{\kappa \to \kappa'}$.*

Proof. Analogous to the proof of Lemma 21. □

Definition 16 (Induced type groupoid). *Let $\mathcal{T}$ be a type structure and $\mathcal{E} \subseteq {}^2\mathcal{T}$. We say $\mathcal{E}$ is* induced *if $\mathcal{E}$ is an induced type structure and $\mathcal{E}^{\star}_{\Xi}$ is groupoidal for all Ξ.*

Since type equality refers to kinding, we will have to refer to the fundamental theorem of kinding in the proof of the fundamental theorem of type equality.

Lemma 11 (Fundamental theorem of kinding for type groupoids). *Let $\mathcal{T}$ be a type structure and $\mathcal{E} \subseteq {}^2\mathcal{T}$ be induced. Then,*

1. *$\mathcal{E}$ is a type subgroupoid of ${}^2\mathcal{T}$, and*
2. *if $\Xi \vdash T : \kappa$ and $(\rho, \rho') \in \mathcal{E}^{\Xi}_{\Theta}$ then $(\mathcal{T}[\![T]\!]_{\rho}, \mathcal{T}[\![T]\!]_{\rho'}) \in \mathcal{E}^{\kappa}_{\Theta}$.*

Proof. 1. By induction on κ and Lemma 10 we show that $\mathcal{E}^{\kappa}$ is groupoidal.
2. This is just an instance of Theorem 2, since $\mathcal{E}[\![T]\!]_{(\rho,\rho')} = (\mathcal{T}[\![T]\!]_{\rho}, \mathcal{T}[\![T]\!]_{\rho'})$. □

Definition 17 (Model/respect type equality). *Let $\mathcal{T}$ be a type structure. We say that $\mathcal{E} \subseteq {}^2\mathcal{T}$* models type equality *if $\Xi \vdash T = T' : \kappa$ and $(\rho, \rho') \in \mathcal{E}^{\Xi}_{\Theta}$ imply $(\mathcal{T}[\![T]\!]_{\rho}, \mathcal{T}[\![T']\!]_{\rho'}) \in \mathcal{E}^{\kappa}_{\Theta}$. A type structure $\mathcal{T}'$* respects type equality *if $\Xi \vdash T = T' : \kappa$ implies $\mathcal{T}'[\![T]\!]_{\rho} = \mathcal{T}'[\![T']\!]_{\rho}$ for all $\rho \in \mathcal{T}'^{\Xi}_{\Theta}$.*

Lemma 12 (Type structure modulo). *Let $\mathcal{T}$ and $\mathcal{E} \subseteq {}^2\mathcal{T}$ be type structures such that $\mathcal{E}$ models type equality. Then $\mathcal{T}/\mathcal{E}$ is a type structure respecting type equality.*

Proof. Application and evaluation are well-defined in $\mathcal{T}/\mathcal{E}$ since $\mathcal{E}$ is a type structure. $\mathcal{T}/\mathcal{E}$ respects type equality since $\mathcal{E}$ models type equality. □

Theorem 6 (Fundamental theorem of type equality). *Let $\mathcal{T}$ be a combinatory type structure and $\mathcal{E} \subseteq {}^2\mathcal{T}$ an induced type structure. Then $\mathcal{E}$ models type equality.*

Proof. By induction on $\Xi \vdash T = T' : \kappa$. We give a few representative cases. Assume $(\rho, \rho') \in \mathcal{E}^{\Xi}_{\Theta}$.

Case
$$\frac{\Xi \vdash T : \kappa \to \kappa'}{\Xi \vdash \lambda X\!:\!\kappa.\, T\, X = T : \kappa \to \kappa'}\ X \notin \mathsf{dom}(\Xi)$$
Since $\mathcal{E}$ is induced, it is sufficient to assume arbitrary $\Theta' \leq \Theta$ and $(G, G') \in \mathcal{E}^{\kappa}_{\Theta'}$ and show $([\![\lambda X\!:\!\kappa.\, T\, X]\!]_{\rho}, [\![T]\!]_{\rho'}) \cdot (G, G') \in \mathcal{E}^{\kappa'}_{\Theta'}$. Because $X \notin \mathsf{dom}(\Xi)$ we have $(\rho_G, \rho') \in \mathcal{E}^{\Xi}_{\Theta'}$ for $\rho_G := \rho[X \mapsto G]$. By induction hypothesis and Lemma 11, $([\![T]\!]_{\rho_G}, [\![T]\!]_{\rho'}) \in \mathcal{E}^{\kappa \to \kappa'}_{\Theta}$, hence, $([\![T]\!]_{\rho_G} \cdot G, [\![T]\!]_{\rho'} \cdot G') \in \mathcal{E}^{\kappa'}_{\Theta'}$. We are done, since $[\![\lambda X\!:\!\kappa.\, T\, X]\!]_{\rho} \cdot G = [\![T\, X]\!]_{\rho_G} = [\![T]\!]_{\rho_G} \cdot G$.

Case
$$\frac{\Xi, X\!:\!\kappa \vdash T : \kappa' \qquad \Xi \vdash U : \kappa}{\Xi \vdash (\lambda X\!:\!\kappa.\, T)\, U = T[U/X] : \kappa'}$$
Here we use the law $[\![T[U/X]]\!]_{\rho} = [\![T]\!]_{\rho[X \mapsto [\![U]\!]_{\rho}]}$ of combinatory type structures.

Case
$$\frac{\Xi \vdash T_1 = T_2 : \kappa \qquad \Xi \vdash T_2 = T_3 : \kappa}{\Xi \vdash T_1 = T_3 : \kappa}$$
Let $F_i = [\![T_i]\!]_{\rho}$ for $i = 1, 2, 3$. By induction hypothesis, $(F_1, F_2), (F_2, F_3) \in \mathcal{E}^{\kappa}_{\Theta}$. Since $\mathcal{E}^{\kappa}_{\Theta}$ is a subgroupoid of ${}^2\mathcal{T}^{\kappa}_{\Theta}$ and $(F_1, F_2) * (F_2, F_3)$ is defined, we have $(F_1, F_3) \in \mathcal{E}^{\kappa}_{\Theta}$. □

6.2 Completeness of NbE for Types

In the following we show that the relation "reify to the same η-long form" gives rise to an equivalence relation on types which models type equality. This implies that NbE is complete.

Def. and Lem. 7 (Kind candidate space for completeness) *Let $\mathcal{T}$ be reifiable.*

$$\overline{\mathsf{Per}}^{\kappa}_{\Xi} = \{(F,F') \in {}^{2}\mathcal{T}^{\kappa}_{\Xi} \mid \Xi \vdash F \searrow V \Uparrow \kappa \textit{ and } \Xi \vdash F' \searrow V \Uparrow \kappa \textit{ for some } V \in \mathsf{Ty}^{\kappa}_{\Xi}\}$$
$$\underline{\mathsf{Per}}^{\kappa}_{\Xi} = \{(F,F') \in {}^{2}\mathcal{T}^{\kappa}_{\Xi} \mid \Xi \vdash F \searrow V \Downarrow \kappa \textit{ and } \Xi \vdash F' \searrow V \Downarrow \kappa \textit{ for some } V \in \mathsf{Ty}^{\kappa}_{\Xi}\}$$

$\underline{\mathsf{Per}}^{\kappa}$ *and* $\overline{\mathsf{Per}}^{\kappa}$ *are Kripke families of subgroupoids, and form a kind candidate space.*

Proof. Composition $(F_1,F_2)*(F_2,F_3)$ is well-defined since reification is deterministic. □

Def. and Lem. 8 (Type groupoid for completeness) *Let $\mathcal{T}$ be a type structure. We define $\mathsf{P}^{\kappa} \subseteq {}^{2}\mathcal{T}^{\kappa}$ by recursion on κ:*

$$\begin{aligned} \mathsf{P}^{\star} &:= \underline{\mathsf{Per}}^{\star}, \\ \mathsf{P}^{\kappa\to\kappa'} &:= \mathsf{P}^{\kappa} \to_{\widehat{2\mathcal{T}}} \mathsf{P}^{\kappa'}. \end{aligned}$$

P *is an induced type groupoid.*

Theorem 9 (Completeness of NbE for types). *Let $\mathcal{T}$ be a reifiable type structure. If $\Xi \vdash T = T' : \kappa$ then $\Xi \vdash [\![T]\!]_{\mathsf{Var}} \searrow V \Uparrow \kappa$ and $\Xi \vdash [\![T']\!]_{\mathsf{Var}} \searrow V \Uparrow \kappa$ for some V.*

Proof. Since $(\mathsf{Var},\mathsf{Var}) \in \mathsf{P}^{\Xi}_{\Xi}$, by the fundamental theorem of type equality we have $([\![T]\!]_{\mathsf{Var}}, [\![T']\!]_{\mathsf{Var}}) \in \mathsf{P}^{\kappa}_{\Xi} \subseteq \overline{\mathsf{Per}}^{\kappa}_{\Xi}$ which entails the goal. □

6.3 Type structure modulo reification

In the following we show that a type structure $\mathcal{T}$ modulo the equality P induced by Per is still reifiable, i. e., reification is compatible with P-equality. Since P is a kind candidate of space Per, it is clear that for $(F,F') \in \mathsf{P}^{\kappa}_{\Xi}$, the types F and F' reify to the same expression under the $\Uparrow$-relation. What remains to show that his holds also for the $\Downarrow$-relation, provided F and F' are neutral or constructed. Thus, we need to show that $(F,F') \in \overline{\mathsf{Per}}^{\kappa}_{\Xi}$ implies $(F,F') \in \underline{\mathsf{Per}}^{\kappa}_{\Xi}$ for such F,F'. The proof rests on the following strengthening lemma.

Lemma 13 (Strengthening of reification). *Let $\mathcal{T}$ be a reifiable structure, $F \in \mathcal{T}^{\kappa}_{\Xi}$, and $\Downarrow\!\Uparrow \in \{\Downarrow,\Uparrow\}$. If $\Xi' \vdash F \searrow T \Downarrow\!\Uparrow \kappa$ for some $\Xi' \leq \Xi$, then $\Xi \vdash F \searrow T \Downarrow\!\Uparrow \kappa$.*

Proof. By induction on $T \in \mathsf{Ty}^{\kappa}_{\Xi}$, first proving the statement for $\Downarrow$, then for $\Uparrow$. In the case of a variable $X \in \mathcal{T}^{\kappa}_{\Xi}$, we know $\Xi(X) = \kappa$, hence, $\Xi \vdash X \searrow X \Downarrow \kappa$. In case $\Xi' \vdash F \searrow \lambda X : \kappa_1 .\, V \Uparrow \kappa_1 \to \kappa_2$ we may assume w. l. o. g. that $X \notin \mathsf{dom}(\Xi')$. Hence, $\Xi', X : \kappa_1 \leq \Xi, X : \kappa_1$ and by induction hypothesis on $V \in \mathsf{Ty}^{\kappa_2}_{\Xi, X:\kappa_1}$ we have $\Xi, X : \kappa_1 \vdash F \cdot X \searrow V \Uparrow \kappa_2$, thus, $\Xi \vdash F \searrow \lambda X : \kappa_1 .\, V \Uparrow \kappa_1 \to \kappa_2$. □

Using strengthening, we can show that $\Uparrow$ for neutral and constructed types, which returns the η-long form, embeds into $\Downarrow$, modulo η-equality.

Lemma 14 (Reification of neutral and constructed types). *Let $\mathcal{T}$ be a reifiable type structure. Let $H \in \mathcal{T}^{\kappa}_{\Xi}$ be neutral or constructed. If $\Xi \vdash H \searrow V \Uparrow \kappa$ then $\Xi \vdash H \searrow U \Downarrow \kappa$ and $\Xi \vdash U = V : \kappa$.*

Proof. By induction on κ. Trivial for $\kappa = \star$. In case $\kappa = \kappa_1 \to \kappa_2$, we have

$$\frac{\Xi, X\!:\!\kappa_1 \vdash H \cdot X \searrow V \Uparrow \kappa_2}{\Xi \vdash H \searrow \lambda X\!:\!\kappa_1.\, V \Uparrow \kappa_1 \to \kappa_2}$$

Analyzing the induction hypothesis $\Xi, X : \kappa_1 \vdash H \cdot X \searrow U' \Downarrow \kappa_2$, we get $\Xi, X : \kappa_1 \vdash H \searrow U \Downarrow \kappa_1 \to \kappa_2$ and $\Xi, X : \kappa_1 \vdash U\, X = V : \kappa_2$. Since $H \in \mathcal{T}^{\kappa}_{\Xi}$, we get by Lemma 13, $\Xi \vdash H \searrow U \Downarrow \kappa_1 \to \kappa_2$, hence, $\Xi \vdash U : \kappa_1 \to \kappa_2$. This entails $\Xi \vdash U = \lambda X\!:\!\kappa_1.\, V : \kappa_1 \to \kappa_2$. $\square$

Corollary 2. *If $(H, T) \in \overline{\mathsf{Per}}^{\kappa}_{\Xi}$ and H is neutral or constructed, then $(H, T) \in \underline{\mathsf{Per}}^{\kappa}_{\Xi}$.*

Theorem 10 (Type structure modulo reification). *Let $\mathcal{T}$ be a reifiable type structure and P defined as above. Then $\mathcal{T}/\mathsf{P}$ is reifiable and respects type equality.*

Proof. Since P models type equality, we only need to show that both reification relations are well-defined on $\mathcal{T}/\mathsf{P}$. First, if $\Xi \vdash F \searrow V \Uparrow \kappa$ and $\Xi \vdash F' \searrow V' \Uparrow \kappa$ and $(F, F') \in \mathsf{P}^{\kappa}_{\Xi} \subseteq \overline{\mathsf{Per}}^{\kappa}_{\Xi}$ then $V = V'$ since reification is deterministic. Secondly, if $\Xi \vdash H \searrow U \Downarrow \kappa$ and $\Xi \vdash H' \searrow U' \Downarrow \kappa$ for neutral or constructed $(H, H') \in \mathsf{P}^{\kappa}_{\Xi}$, then by Cor. 2 $(H, H') \in \underline{\mathsf{Per}}^{\kappa}_{\Xi}$, hence, $U = U'$. $\square$

Observe however, that $\mathcal{T}$ is term-like whereas $\mathcal{T}/\mathsf{P}$ is not.

7 Object Structures

In this section, we introduce object structures which model both the syntactic object structure Obj indexed by syntactic types in Ty and structures of values D indexed by type values from a structure $\mathcal{T}$. The following development, leading up the fundamental theorem of typing and the soundness of NbE for objects, parallels the preceding one on the type level. However, while we could define the glueing type structure G^{κ} by induction on kind κ, we cannot define a similar glueing objects structure gl by induction on types, due to impredicativity. Hence, we will define gl as a structure of *candidates* for semantic types.

Definition 18 (Typing context). *Given a type structure $\mathcal{T}$, a $\mathcal{T}_{\Xi}$-context $\Delta \in \mathcal{T}^{\mathsf{cxt}}_{\Xi}$ is a partial map from the term variables into $\mathcal{T}^{\star}_{\Xi}$. If $\Gamma \in \mathsf{Ty}^{\mathsf{cxt}}_{\Xi}$ and $\rho \in \mathcal{T}^{\Xi}_{\Theta}$ then $[\![\Gamma]\!]_{\rho} \in \mathcal{T}^{\mathsf{cxt}}_{\Theta}$ is defined by $[\![\Gamma]\!]_{\rho}(x) = [\![\Gamma(x)]\!]_{\rho}$.*

Let $\mathsf{Obj}^{\Xi \vdash T}_{\Gamma} = \{t \mid \Xi; \Gamma \vdash t : T\}$.

Definition 19 (Object structure). *Let $\mathcal{T}$ be a type structure. An* object structure over *$\mathcal{T}$ is a family $D^{\Xi \vdash A}$ ($A \in \mathcal{T}^{\star}_{\Xi}$) of Kripke sets indexed by $\mathcal{T}_{\Xi}$-contexts Δ such that $\Xi' \leq \Xi$ implies $D^{\Xi \vdash A}_{\Delta} = D^{\Xi' \vdash A}_{\Delta}$. It respects type equality, i. e., $\Xi \vdash T = T' : \star$ implies $D^{\Theta \vdash [\![T]\!]_\rho} = D^{\Theta \vdash [\![T']\!]_\rho}$ for any $\rho \in \mathcal{T}^{\Xi}_{\Theta}$, and there are operations:*

$$\begin{array}{ll} \mathsf{app}^{\Xi \vdash A \to B}_{\Delta} & \in D^{\Xi \vdash A \to B}_{\Delta} \to D^{\Xi \vdash A}_{\Delta} \to D^{\Xi \vdash B}_{\Delta}, \\ \mathsf{TyApp}^{\Xi \vdash \forall^\kappa F}_{\Delta} & \in D^{\Xi \vdash \forall^\kappa F}_{\Delta} \to (G \in \mathcal{T}^{\kappa}_{\Xi}) \to D^{\Xi \vdash F \cdot G}_{\Delta}. \end{array}$$

We write $\_ \cdot \_$ for both of these operations. For $\Delta, \Psi \in \mathcal{T}^{\mathsf{cxt}}_{\Theta}$, let $\eta \in D^{\Theta \vdash \Psi}_{\Delta}$ iff $\eta(x) \in D^{\Theta \vdash \Psi(x)}_{\Delta}$ for all $x \in \mathsf{dom}(\Psi)$. We stipulate a family of evaluation functions

$$(\!|\_|\!)^{\rho}_{\_} \in \mathsf{Obj}^{\Xi \vdash T}_{\Gamma} \to D^{\Theta \vdash [\![\Gamma]\!]_\rho}_{\Delta} \to D^{\Theta \vdash [\![T]\!]_\rho}_{\Delta}$$

indexed by $\rho \in \mathcal{T}^{\Xi}_{\Theta}$ which satisfy the following equations:

$$\begin{array}{rlrll} (\!|x|\!)^{\rho}_{\eta} &= \eta(x) & (\!|\lambda x\!:\!U.\, t|\!)^{\rho}_{\eta} \cdot d &= (\!|t|\!)^{\rho}_{\eta[x \mapsto d]} & \textit{if } d \in D^{\Theta \vdash [\![U]\!]_\rho}_{\Delta} \\ (\!|r\, s|\!)^{\rho}_{\eta} &= (\!|r|\!)^{\rho}_{\eta} \cdot (\!|s|\!)^{\rho}_{\eta} & (\!|\Lambda X\!:\!\kappa.\, t|\!)^{\rho}_{\eta} \cdot G &= (\!|t|\!)^{\rho[X \mapsto G]}_{\eta} & \textit{if } G \in \mathcal{T}^{\kappa}_{\Theta} \\ (\!|t\, U|\!)^{\rho}_{\eta} &= (\!|t|\!)^{\rho}_{\eta} \cdot [\![U]\!]_\rho & (\!|t[U/X]|\!)^{\sigma}_{\eta} &= (\!|t|\!)^{\sigma[X \mapsto [\![U]\!]_\sigma]}_{\eta} \; (*) & \\ (\!|t[u/x]|\!)^{\sigma}_{\eta} &= (\!|t|\!)^{\sigma}_{\eta[x \mapsto (\!|u|\!)^{\sigma}_{\eta}]} \; (*) & & & \end{array}$$

Again, $(*)$ have to hold only in *combinatory* object structures.

With parallel substitution, Obj (modulo β, $\beta\eta$, or judgmental equality) forms an object structure over Ty (modulo the same equality).

Definition 20 (Object substructure). *Let $\mathcal{S}, \mathcal{T}$ be type structures with $\mathcal{S} \subseteq \mathcal{T}$ and let D be an object structure over $\mathcal{T}$. Let $E^{\Xi \vdash A} \subseteq D^{\Xi \vdash A}$ be a Kripke family of subsets indexed by $\Delta \in \mathcal{S}^{\mathsf{cxt}}_{\Xi}$ for all $A \in \mathcal{S}^{\star}_{\Xi}$. Then E is an* object substructure of D over $\mathcal{S}$ *if application and evaluation are well defined on E.*

$$\begin{array}{ll} \mathsf{app}^{\Xi \vdash A \to B}_{\Delta} & \in E^{\Xi \vdash A \to B}_{\Delta} \to E^{\Xi \vdash A}_{\Delta} \to E^{\Xi \vdash B}_{\Delta} \\ \mathsf{TyApp}^{\Xi \vdash \forall^\kappa F}_{\Delta} & \in E^{\Xi \vdash \forall^\kappa F}_{\Delta} \to (G \in \mathcal{S}^{\kappa}_{\Xi}) \to E^{\Xi \vdash F \cdot G}_{\Delta} \\ (\!|\_|\!)^{\rho}_{\_} & \in \mathsf{Obj}^{\Xi \vdash T}_{\Gamma} \to E^{\Theta \vdash \mathcal{S}[\![\Gamma]\!]_\rho}_{\Delta} \to E^{\Theta \vdash \mathcal{S}[\![T]\!]_\rho}_{\Delta} \\ & \quad \textit{for all } \Delta \in \mathcal{S}^{\mathsf{cxt}}_{\Theta} \textit{ and } \rho \in \mathcal{S}^{\Xi}_{\Theta}. \end{array}$$

Definition 21 (Reindexed object structure). *Let $M : \mathcal{S} \to \mathcal{T}$ be a type structure morphism and D an object structure over $\mathcal{T}$. The type structure $E^{\Xi \vdash A} := D^{\Xi \vdash M(A)}$ over $\mathcal{S}$ with*

$$\begin{array}{l} f \cdot_E d := f \cdot_D d \\ d \cdot_E G := d \cdot_D (M(G)) \\ E(\!|\_|\!)^{\rho}_{\_} := D(\!|\_|\!)^{M \circ \rho}_{\_} \end{array}$$

is called D reindexed by M.

Definition 22 (Object structure morphism). *Let $M : \mathcal{S} \to \mathcal{T}$ be a type structure morphism and D an object structure over $\mathcal{S}$ and E one over $\mathcal{T}$. An* object structure morphism $m : D \to E$ *is a Kripke family of maps*

$$m^{\Xi \vdash A}_{\Delta} \in D^{\Xi \vdash A}_{\Delta} \to E^{\Xi \vdash M^{\star}_{\Xi}(A)}_{M^{\star}_{\Xi} \circ \Delta}$$

which commute with application and evaluation, i. e.,

$$\begin{aligned}
m(f \cdot_D d) &= m(f) \cdot_E m(d) \\
&\quad \textit{for all } f \in D_\Delta^{\Xi \vdash A \to B}, d \in D_\Delta^{\Xi \vdash A} \\
m(d \cdot_D G) &= m(d) \cdot_E M(G) \\
&\quad \textit{for all } d \in D_\Delta^{\Xi \vdash \forall^\kappa F}, G \in \mathcal{T}_\Xi^\kappa \\
m(D(\!|t|\!)_\eta^\rho) &= E(\!|t|\!)_{m \circ \eta}^{M \circ \rho} \\
&\quad \textit{for all } T \in \mathsf{Ty}_\Xi^\star, t \in \mathsf{Obj}_\Gamma^{\Xi \vdash T}, \rho \in \mathcal{S}_\Theta^\Xi, \eta \in D_\Delta^{\Theta \vdash \mathcal{S}[\![\Gamma]\!]_\rho}.
\end{aligned}$$

Definition 23 (Product object structure). *Given object structures D_1 over $\mathcal{T}_1$ and D_2 over T_1 we define the* product object structure $D_1 \times D_2$ *over* $\mathcal{T}_1 \times \mathcal{T}_2$ *by* $(D_1 \times D_2)_{(\Delta_1, \Delta_2)}^{\Xi \vdash (A_1, A_2)} := D_1{}_{\Delta_1}^{\Xi \vdash A_1} \times D_2{}_{\Delta_2}^{\Xi \vdash A_2}$ *with*

$$\begin{aligned}
(f_1, f_2) \cdot_{D_1 \times D_2} (d_1, d_2) &:= (f_1 \cdot_{D_1} d_1, f_2 \cdot_{D_2} d_2) \\
(d_1, d_2) \cdot_{D_1 \times D_2} (G_1, G_2) &:= (d_1 \cdot_{D_1} G_1, d_2 \cdot_{D_2} G_2) \\
(D_1 \times D_2)(\!|t|\!)_{\eta_1, \eta_2}^{(\rho_1, \rho_2)} &:= (D_1(\!|t|\!)_{\eta_1}^{\rho_1}, D_2(\!|t|\!)_{\eta_2}^{\rho_2}).
\end{aligned}$$

7.1 Realizability Type Structure and the Fundamental Theorem of Typing

Fix some term-like type structure $\mathcal{T}$ and an object structure D over $\mathcal{T}$. Let $\mathcal{A} \in \widehat{D}_\Xi^A$ if $\mathcal{A}_\Delta \subseteq D_\Delta^{\Xi \vdash A}$ and $\mathcal{A}$ is Kripke. $\widehat{D}_\Xi^A$ forms a complete lattice for all Ξ, A.

Lemma 15. *If $\Xi' \leq \Xi$ then $\widehat{D}_\Xi^A = \widehat{D}_{\Xi'}^A$.*

Proof. $D_\Delta^{\Xi \vdash A} = D_\Delta^{\Xi' \vdash A}$. □

Definition 24 (Function space and type abstraction on $\widehat{D}$).

$$\begin{aligned}
\_ \to_{\widehat{D}} \_ &\in \widehat{D}_\Xi^A \to \widehat{D}_\Xi^B \to \widehat{D}_\Xi^{A \to B} \\
(\mathcal{A} \to \mathcal{B})_\Delta &:= \{ f \in D_\Delta^{\Xi \vdash A \to B} \mid \textit{for all } d, \Delta' \leq \Delta, d \in \mathcal{A}_{\Delta'} \textit{ implies } f \cdot d \in \mathcal{B}_{\Delta'} \}
\end{aligned}$$

$$\begin{aligned}
(\_.\_)^{\forall^\kappa F} &\in (G \in \mathcal{T}_\Xi^\kappa) \to \widehat{D}_\Xi^{F \cdot G} \to \widehat{D}_\Xi^{\forall^\kappa F} \\
(G.\mathcal{A})_\Delta^{\forall^\kappa F} &:= \{ d \in D_\Delta^{\Xi \vdash \forall^\kappa F} \mid d \cdot G \in \mathcal{A}_\Delta \}
\end{aligned}$$

Constructors of higher kind are interpreted as operators on Kripke sets.

Definition 25 (Kripke operators of higher kind). *We define $\widehat{D}_\Xi^{F:\kappa}$ by*

$$\begin{aligned}
\widehat{D}_\Xi^{A:\star} &:= \widehat{D}_\Xi^A, \\
\widehat{D}_\Xi^{F:\kappa \to \kappa'} &:= (G \in \mathcal{T}_\Xi^\kappa) \to \widehat{D}_\Xi^{G:\kappa} \to \widehat{D}_\Xi^{F \cdot G:\kappa'}.
\end{aligned}$$

Definition 26 (Type candidate space). *A* type candidate space $\mathcal{C}$ for D over $\mathcal{T}$ *consists of two Kripke sets $\underline{\mathcal{C}}^{\Xi \vdash A}, \overline{\mathcal{C}}^{\Xi \vdash A} \in \widehat{D}_\Xi^A$, (written $\underline{A}, \overline{A}$ if no ambiguities arise) for each type $A \in \mathcal{T}_\Xi^\star$ such that the following conditions hold.*

K-NE	$\underline{H}$	$\subseteq$	$\overline{H}$	$\in$	$\widehat{D}_\Xi^H$	*if H neutral*
K-FUN-E	$\underline{A \to B}$	$\subseteq$	$\overline{A} \to_{\widehat{D}} \underline{B}$	$\in$	$\widehat{D}_\Xi^{A \to B}$	
K-FUN-I	$\underline{A} \to_{\widehat{D}} \overline{B}$	$\subseteq$	$\overline{A \to B}$	$\in$	$\widehat{D}_\Xi^{A \to B}$	
K-ALL-E	$\underline{\forall^\kappa F}$	$\subseteq$	$G.\underline{F \cdot G}$	$\in$	$\widehat{D}_\Xi^{\forall^\kappa F}$	*if $G \in \mathcal{T}_\Xi^\kappa$*
K-ALL-I	$X.\overline{F \cdot X}$	$\subseteq$	$\overline{\forall^\kappa F}$	$\in$	$\widehat{D}_\Xi^{\forall^\kappa F}$	*if $X \notin \mathsf{dom}(\Xi)$*

Definition 27 (Realizable semantic types). *If $F \in \mathcal{T}^{\kappa}_{\Xi}$ and $\mathcal{F} \in \widehat{D}^{F:\kappa}_{\Xi}$ then $F \Vdash^{\kappa}_{\mathcal{C}} \mathcal{F}$ (pronounced F* realizes *$\mathcal{F}$) is defined by induction on κ as follows:*

$$\begin{array}{ll} A \Vdash^{\star}_{\mathcal{C}} \mathcal{A} & :\Longleftrightarrow \underline{A} \subseteq \mathcal{A} \subseteq \overline{A} \\ F \Vdash^{\kappa \to \kappa'}_{\mathcal{C}} \mathcal{F} & :\Longleftrightarrow F \cdot G \Vdash^{\kappa'}_{\mathcal{C}} \mathcal{F}(G, \mathcal{G}) \text{ for all } G \Vdash^{\kappa}_{\mathcal{C}} \mathcal{G} \end{array}$$

We define the Kripke families $\mathcal{T}\widehat{D}$ and $\mathcal{C} \subseteq \mathcal{T}\widehat{D}$ by

$$\begin{array}{ll} \mathcal{T}\widehat{D}^{\kappa}_{\Xi} & = \{(F, \mathcal{F}) \in \mathcal{T}^{\kappa}_{\Xi} \times \widehat{D}^{F:\kappa}_{\Xi}\} \\ \mathcal{C}^{\kappa}_{\Xi} & = \{(F, \mathcal{F}) \in \mathcal{T}^{\kappa}_{\Xi} \times \widehat{D}^{F:\kappa}_{\Xi} \mid F \Vdash^{\kappa}_{\mathcal{C}} \mathcal{F}\}. \end{array}$$

For the remainder of this section, we fix a type candidate space $\mathcal{C}$. Type variables are embedded into the semantic type structure as $\underline{X} \in \widehat{D}^{X:\Xi(X)}_{\Xi}$. To define $\underline{X}$, we have to actually define $\underline{H}$ for all neutral H.

Def. and Lem. 11 (Neutral realizable semantic types) *Given a type candidate space, for a neutral $H \in T^{\kappa}_{\Xi}$ we define $\underline{H}, \overline{H} \in \widehat{D}^{H:\kappa}_{\Xi}$ by induction on κ. For base kind $\star$, it is already defined, for higher kinds we set*

$$\begin{array}{l} \overline{H}(G, \mathcal{G}) := \overline{H\,G}, \\ \underline{H}(G, \mathcal{G}) := \underline{H\,G}. \end{array}$$

Clearly, $H \Vdash^{\kappa} \underline{H}$ and $H \Vdash^{\kappa} \overline{H}$.

Definition 28 (Interpretation into $\widehat{D}$). *For $T \in \mathsf{Ty}^{\kappa}_{\Xi}$ and $(\sigma, \rho) \in \mathcal{T}\widehat{D}^{\Xi}_{\Theta}$ we define $\widehat{D}[\![T]\!]_{\sigma,\rho} \in \widehat{D}^{\mathcal{T}[\![T]\!]_{\sigma}:\kappa}_{\Theta}$ as follows:*

$$\begin{array}{lll} & \widehat{D}[\![X]\!]_{\sigma,\rho} & := \rho(X) \\ & \widehat{D}[\![\lambda X\!:\!\kappa.\,T]\!]_{\sigma,\rho} & := ((G, \mathcal{G}) \in \mathcal{T}\widehat{D}^{\kappa}_{\Theta}) \mapsto \widehat{D}[\![T]\!]_{(\sigma,\rho)[X \mapsto (G, \mathcal{G})]} \\ & \widehat{D}[\![T\,U]\!]_{\sigma,\rho} & := \widehat{D}[\![T]\!]_{\sigma,\rho}(\mathcal{T}[\![U]\!]_{\sigma}, \widehat{D}[\![U]\!]_{\sigma,\rho}) \\ & \widehat{D}[\![C]\!]_{\sigma,\rho} & := C_{\widehat{D}} \\ \textit{where} & \to_{\widehat{D}} & \in \widehat{D}^{\to:\star\to\star\to\star}_{\Xi} \\ & \to_{\widehat{D}}(A, \mathcal{A})(B, \mathcal{B}) & := \mathcal{A} \to \mathcal{B} \\ & \forall^{\kappa}_{\widehat{D}} & \in \widehat{D}^{\forall^{\kappa}:(\kappa\to\star)\to\star}_{\Xi} \\ & \forall^{\kappa}_{\widehat{D}}(F, \mathcal{F}) & := \bigcap_{G \Vdash^{\kappa} \mathcal{G}} G.\mathcal{F}(G, \mathcal{G}) \end{array}$$

Lemma 16 (Well-definedness of interpretation into $\widehat{D}$). *If $T \in \mathsf{Ty}^{\kappa}_{\Xi}$ and $(\sigma, \rho) \in \mathcal{T}\widehat{D}^{\Xi}_{\Theta}$ then $\widehat{D}[\![T]\!]_{\sigma,\rho} \in \widehat{D}^{\mathcal{T}[\![T]\!]_{\sigma}:\kappa}_{\Theta}$.*

Proof.

$$\begin{array}{ll}
\widehat{D}\llbracket C\rrbracket_{\sigma,\rho} & \in \widehat{D}_{\Theta}^{\mathcal{T}\llbracket C\rrbracket_{\sigma}:\Sigma(C)} \\
& = \widehat{D}_{\Theta}^{C_{\mathcal{T}}:\Sigma(C)} \\
\widehat{D}\llbracket C\rrbracket_{\sigma,\rho} & = C_{\widehat{D}} \\
\widehat{D}\llbracket X\rrbracket_{\sigma,\rho} & \in \widehat{D}_{\Theta}^{\mathcal{T}\llbracket X\rrbracket_{\sigma}:\kappa} \\
& = \widehat{D}_{\Theta}^{\sigma(X):\kappa} \\
\widehat{D}\llbracket X\rrbracket_{\sigma,\rho} & = \rho(X) \\
\widehat{D}\llbracket \lambda X:\kappa.\, T\rrbracket_{\sigma,\rho} & \in \widehat{D}_{\Theta}^{\mathcal{T}\llbracket \lambda X:\kappa.\, T\rrbracket_{\sigma}} \\
& = ((G,\mathcal{G}) \in \mathcal{T}\widehat{D}_{\Theta}^{\kappa}) \to \widehat{D}_{\Theta}^{\mathcal{T}\llbracket \lambda X:\kappa.\, T\rrbracket_{\sigma}\cdot G} \\
& = ((G,\mathcal{G}) \in \mathcal{T}\widehat{D}_{\Theta}^{\kappa}) \to \widehat{D}_{\Theta}^{\mathcal{T}\llbracket T\rrbracket_{\sigma[X\mapsto G]}} \\
\widehat{D}\llbracket \lambda X:\kappa.\, T\rrbracket_{\sigma,\rho} & = ((G,\mathcal{G}) \in \mathcal{T}\widehat{D}_{\Theta}^{\kappa}) \mapsto \widehat{D}\llbracket T\rrbracket_{(\sigma,\rho)[X\mapsto(G,\mathcal{G})]} \\
\widehat{D}\llbracket T\,U\rrbracket_{\sigma,\rho} & \in \widehat{D}_{\Theta}^{\mathcal{T}\llbracket T\,U\rrbracket_{\sigma}} \\
& = \widehat{D}_{\Theta}^{\mathcal{T}\llbracket T\rrbracket_{\sigma}\cdot\mathcal{T}\llbracket U\rrbracket_{\sigma}} \\
\widehat{D}\llbracket T\,U\rrbracket_{\sigma,\rho} & = \widehat{D}\llbracket T\rrbracket_{\sigma,\rho}(\mathcal{T}\llbracket U\rrbracket_{\sigma}, \widehat{D}\llbracket U\rrbracket_{\sigma,\rho})
\end{array}$$

where, in the last line,

$$\begin{array}{ll}
\widehat{D}\llbracket T\rrbracket_{\sigma,\rho} & \in \widehat{D}_{\Theta}^{\mathcal{T}\llbracket T\rrbracket_{\sigma}:\kappa\to\kappa'} \\
& = ((G,\mathcal{G}) \in \mathcal{T}\widehat{D}_{\Xi}^{\kappa}) \to \widehat{D}_{\Theta}^{\mathcal{T}\llbracket T\rrbracket_{\sigma}\cdot G:\kappa'}.
\end{array}$$

□

Since the kind function space is the full set-theoretic one, $\widehat{D}$ is combinatory and respects type equality.

Lemma 17 **($\widehat{D}$ is combinatory).** *Let* $U \in \mathsf{Ty}_{\Xi}^{\kappa}$, $T \in \mathsf{Ty}_{\Xi,X:\kappa}^{\kappa'}$ *and* $(\sigma,\rho),(\sigma',\rho') \in \mathcal{T}\widehat{D}_{\Theta}^{\Xi}$.

1. $\widehat{D}\llbracket U\rrbracket_{\sigma,\rho} = \widehat{D}\llbracket U\rrbracket_{\sigma',\rho'}$ *if* $\rho(X) = \rho'(X)$ *for all* $X \in \mathsf{FV}(U)$.
2. $\widehat{D}\llbracket T[U/X]\rrbracket_{\sigma,\rho} = \widehat{D}\llbracket T\rrbracket_{(\sigma,\rho)[X\mapsto\mathcal{T}\widehat{D}\llbracket U\rrbracket_{\sigma,\rho}]}$.

Proof. The first proposition is needed to establish the second.

1. By induction on U.
2. By induction on T, using 1.
   Case $\kappa' = \kappa_1 \to \kappa_2$ and $T = \lambda Y:\kappa_1.\, T'$. Using the induction hypothesis and 1, we have for all $(G,\mathcal{G}) \in \mathcal{T}\widehat{D}_{\Theta}^{\kappa_1}$, since $Y \notin \mathsf{FV}(U)$,

$$\begin{array}{l}
\quad \widehat{D}\llbracket T'[U/X]\rrbracket_{(\sigma,\rho)[Y\mapsto(G,\mathcal{G})]} \\
= \widehat{D}\llbracket T'\rrbracket_{(\sigma,\rho)[Y\mapsto(G,\mathcal{G})][X\mapsto\mathcal{T}\widehat{D}\llbracket U\rrbracket_{(\sigma,\rho)[Y\mapsto(G,\mathcal{G})]}]} \\
= \widehat{D}\llbracket T'\rrbracket_{(\sigma,\rho)[Y\mapsto(G,\mathcal{G})][X\mapsto\mathcal{T}\widehat{D}\llbracket U\rrbracket_{\sigma,\rho}]} \\
= \widehat{D}\llbracket T'\rrbracket_{(\sigma,\rho)[X\mapsto\mathcal{T}\widehat{D}\llbracket U\rrbracket_{\sigma,\rho}][Y\mapsto(G,\mathcal{G})]}
\end{array}$$

   hence, $\widehat{D}\llbracket \lambda Y:\kappa_1.\, T'[U/X]\rrbracket_{\sigma,\rho} = \widehat{D}\llbracket \lambda Y:\kappa_1.\, T'\rrbracket_{(\sigma,\rho)[X\mapsto\mathcal{T}\widehat{D}\llbracket U\rrbracket_{\sigma,\rho}]}$. □

Lemma 18 **($\widehat{D}$ respects type equality).** *If $\Xi \vdash T = T' : \kappa$ and $(\sigma, \rho) \in \mathcal{T}\widehat{D}^{\Xi}_{\Theta}$ then $\widehat{D}[\![T]\!]_{\sigma,\rho} = \widehat{D}[\![T']\!]_{\sigma,\rho}$.*

Proof. By induction on $\Xi \vdash T = T' : \kappa$, using Lemma 17. $\square$

Lemma 19 **(Realizability of type constructors).** *$\rightarrow \Vdash^{\star\rightarrow\star\rightarrow\star} \rightarrow$ and $\forall^{\kappa} \Vdash^{(\kappa\rightarrow\star)\rightarrow\star} \forall^{\kappa}$.*

Proof. Assume $\mathcal{A} \in \widehat{D}^{A}_{\Xi}$ and $\mathcal{B} \in \widehat{D}^{B}_{\Xi}$ with $A \Vdash^{\star} \mathcal{A}$ and $B \Vdash^{\star} \mathcal{B}$. We show $A \rightarrow B \Vdash^{\star} \mathcal{A} \rightarrow \mathcal{B}$.

$$\underline{A \rightarrow B} \subseteq \overline{A} \rightarrow \underline{B} \subseteq \mathcal{A} \rightarrow \mathcal{B} \subseteq \underline{A} \rightarrow \overline{B} \subseteq \overline{A \rightarrow B}.$$

Assume $\mathcal{F} \in \widehat{D}^{F:\kappa\rightarrow\star}_{\Xi}$ with $F \Vdash^{\kappa\rightarrow\star} \mathcal{F}$. We show $\forall^{\kappa} F \Vdash^{\star} \forall^{\kappa}(F, \mathcal{F})$. Let $X \notin \mathsf{dom}(\Xi)$. Then $\underline{X} \in \widehat{D}^{X:\kappa}_{\Xi,X:\kappa}$ and $X \Vdash^{\kappa} \underline{X}$.

$$\begin{array}{lll}
\underline{\forall^{\kappa} F} & \subseteq \bigcap_{G \in \mathcal{T}^{\kappa}_{\Xi}} G.\underline{F \cdot G} & \\
& \subseteq \bigcap_{G \Vdash^{\kappa} \mathcal{G}} G.\underline{F \cdot G} & \subseteq \bigcap_{G \Vdash^{\kappa} \mathcal{G}} G.\mathcal{F}(G, \mathcal{G}) \\
= \forall^{\kappa}(F, \mathcal{F}) & \subseteq X.\mathcal{F}(X, \underline{X}) & \subseteq X.\overline{F \cdot X} \\
\subseteq \overline{\forall^{\kappa} F} & &
\end{array}$$

$\square$

Theorem 12 **(Realizability).** *$\mathcal{T}\widehat{D}$ is a type structure with application $(F, \mathcal{F}) \cdot (G, \mathcal{G}) = (F \cdot G, \mathcal{F}(G, \mathcal{G}))$ and evaluation $\mathcal{T}\widehat{D}[\![T]\!]_{\sigma,\rho} = (\mathcal{T}[\![T]\!]_{\sigma}, \widehat{D}[\![T]\!]_{\sigma,\rho})$. $\mathcal{C}$ is a type substructure of $\mathcal{T}\widehat{D}$.*

Proof. By Def. 27 and Lemma 19, $\mathcal{C}$ is induced, hence, by Theorem 2, it is a type structure. $\square$

Corollary 3. *If $T \in \mathsf{Ty}^{\star}_{\Xi}$ and $\sigma \Vdash^{\Xi} \rho$ then $\underline{\mathcal{T}[\![T]\!]_{\sigma}} \subseteq \overline{\mathcal{T}[\![T]\!]_{\sigma}}$.*

Proof. By the theorem, $\mathcal{T}[\![T]\!]_{\sigma} \Vdash^{\star} \widehat{D}[\![T]\!]_{\sigma,\rho}$. $\square$

Theorem 13 **(Fundamental theorem of typing).** *Let D be an object structure over $\mathcal{T}$ and $\underline{\mathcal{C}}, \overline{\mathcal{C}} \in \widehat{D}$ a type candidate space. Let $\mathcal{S} \subseteq \mathcal{C}$ be a type substructure of the associated realizability type structure $\mathcal{C}$. Then the family $E^{\Xi \vdash (A,\mathcal{A})}_{(\Delta,\Phi)} := \mathcal{A}_{\Delta}$ is an object substructure of D reindexed by $\pi_1 : \mathcal{S} \rightarrow \mathcal{T}$.*

Proof. Note that $E^{\Xi \vdash (A,\mathcal{A})}_{(\Delta,\Phi)} \subseteq D^{\Xi \vdash A}_{\Delta}$.

1. E respects type equality. For $\Xi \vdash T = T' : \star$ and $(\sigma, \rho) \in \mathcal{S}^{\Xi}_{\Theta}$ we have $E^{\Xi \vdash \mathcal{S}[\![T]\!]_{\sigma,\rho}}_{\Delta,\Phi} = (\widehat{D}[\![T]\!]_{\sigma,\rho})_{\Delta} = (\widehat{D}[\![T']\!]_{\sigma,\rho})_{\Delta} = E^{\Xi \vdash \mathcal{S}[\![T']\!]_{\sigma,\rho}}_{\Delta,\Phi}$ since $\widehat{D}$ respects type equality.
2. Object application is well-defined. Let $(A, \mathcal{A}), (B, \mathcal{B}) \in \mathcal{S}^{\star}_{\Xi}$ and $f \in E^{\Xi \vdash (A,\mathcal{A}) \rightarrow (B,\mathcal{B})}_{\Delta,\Phi} = (\mathcal{A} \rightarrow_{\widehat{D}} \mathcal{B})_{\Delta}$ and $d \in E^{\Xi \vdash (A,\mathcal{A})}_{\Delta,\Phi} = \mathcal{A}_{\Delta}$. By definition of $\rightarrow_{\widehat{D}}$, $f \cdot d \in \mathcal{B}_{\Delta} = E^{\Xi \vdash (B,\mathcal{B})}_{\Delta,\Phi}$.

3. Type application is well-defined. Let $(F, \mathcal{F}) \in \mathcal{S}_{\Xi}^{\kappa \to \star}$, $(G, \mathcal{G}) \in \mathcal{S}_{\Xi}^{\kappa}$, and $d \in E_{\Delta,\Phi}^{\Xi \vdash \forall^\kappa (F,\mathcal{F})} = (\forall^\kappa \mathcal{F})_\Delta = (\bigcap_{G \Vdash^\kappa \mathcal{G}} G.\mathcal{F}(G,\mathcal{G}))_\Delta = \bigcap_{G \Vdash^\kappa \mathcal{G}} (G.\mathcal{F}(G,\mathcal{G}))_\Delta$. Since $G \Vdash \mathcal{G}$, $d \in (G.\mathcal{F}(G,\mathcal{G}))_\Delta$, hence, $d \cdot_E (G,\mathcal{G}) = d \cdot_D G \in \mathcal{F}(G,\mathcal{G}) = E_{\Delta,\Phi}^{\Xi \vdash (F,\mathcal{F}) \cdot (G,\mathcal{G})}$.
4. Evaluation is well-defined. We show $E(\!|t|\!)_\eta^{\sigma,\rho} \in E_{\Delta,\Phi}^{\Theta \vdash S[\![T]\!]_{\sigma,\rho}}$ for $(\Delta, \Phi) \in \mathcal{S}_\Theta^{\mathsf{cxt}}$, $(\sigma, \rho) \in \mathcal{S}_\Theta^{\Xi}$, and $\eta \in E_{\Delta,\Phi}^{\Theta \vdash S[\![\Gamma]\!]_{\sigma,\rho}}$ by induction on $\Xi; \Gamma \vdash t : T$. By definition of E, it is sufficient to show $D(\!|t|\!)_\eta^\sigma \in (\widehat{D}[\![T]\!]_{\sigma,\rho})_\Delta$ for $\eta \in (\widehat{D}[\![\Gamma]\!]_{\sigma,\rho})_\Delta$.
Case
$$\frac{\Xi \vdash \Gamma}{\Xi; \Gamma \vdash x : \Gamma(x)}$$
We have $D(\!|x|\!)_\eta^\sigma = \eta(x) \in (\widehat{D}[\![\Gamma(x)]\!]_{\sigma,\rho})_\Delta$.
Case
$$\frac{\Xi \vdash T : \kappa \to \star \qquad \Xi, X{:}\kappa; \Gamma \vdash t : T\,X}{\Xi; \Gamma \vdash \Lambda X{:}\kappa.\, t : \forall^\kappa T}\ X \notin \mathsf{dom}(\Xi)$$
To show $D(\!|\Lambda X : \kappa.\, t|\!)_\eta^\sigma \in (\widehat{D}[\![\forall^\kappa T]\!]_{\sigma,\rho})_\Delta$, assume arbitrary $(G, \mathcal{G}) \in \mathcal{S}_\Theta^\kappa$, let $\sigma' := \sigma[X \mapsto G]$ and $\rho' := \rho[X \mapsto \mathcal{G}]$, and show $D(\!|\Lambda X : \kappa.\, t|\!)_\eta^\sigma \cdot_E (G,\mathcal{G}) \in (\widehat{D}[\![T]\!]_{\sigma,\rho}(G,\mathcal{G}))_\Delta$. Let $d := D(\!|\Lambda X : \kappa.\, t|\!)_\eta^\sigma \cdot_E (G,\mathcal{G}) = D(\!|t|\!)_\eta^{\sigma'}$ and note that $\mathcal{S}[\![X]\!]_{\sigma',\rho'} = (G,\mathcal{G})$. Since $(\sigma', \rho') \in \mathcal{S}_\Theta^{\Xi, X:\kappa}$, by induction hypothesis $d \in (\widehat{D}[\![T\,X]\!]_{\sigma',\rho'})_\Delta = (\widehat{D}[\![T]\!]_{\sigma,\rho}(G,\mathcal{G}))_\Delta$, because $X \notin FV(T)$ and Lemma 17.
Case
$$\frac{\Xi; \Gamma \vdash t : \forall^\kappa T \qquad \Xi \vdash U : \kappa}{\Xi; \Gamma \vdash t\,U : T\,U}$$
Let $(F, \mathcal{F}) := \mathcal{S}[\![T]\!]_{\sigma,\rho}$ and $(G, \mathcal{G}) := \mathcal{S}[\![U]\!]_{\sigma,\rho}$. By induction hypothesis $d := D(\!|t|\!)_\eta^\sigma \in (\widehat{D}[\![\forall^\kappa T]\!]_{\sigma,\rho})_\Delta = (\forall_{\widehat{D}}^\kappa (F,\mathcal{F}))_\Delta$. Since $G \Vdash \mathcal{G}$, $d \cdot_E (G,\mathcal{G}) = d \cdot_D G \in \mathcal{F}(G,\mathcal{G})_\Delta = (\widehat{D}[\![T\,U]\!]_{\sigma,\rho})_\Delta$.
Case
$$\frac{\Xi; \Gamma \vdash t : T \qquad \Xi \vdash T = T' : \star}{\Xi; \Gamma \vdash t : T'}$$
Clear, since $\widehat{D}$ respects type equality. □

7.2 Soundness of NbE for Objects

Definition 29 (Term-like object structure). *An object structure D over $\mathcal{T}$ is* term-like *if:*

1. *There exists a Kripke family of maps* $\mathsf{var}^{\Xi \vdash \Delta} \in (x \in \mathsf{dom}(\Delta)) \to D_\Delta^{\Xi \vdash \Delta(x)}$ *indexed by $\Delta \in \mathcal{T}_\Xi^{\mathsf{cxt}}$.*
In the presence of var, *we can define the* neutral objects *inductively as follows:*
   - $\mathsf{var}^{\Xi \vdash \Delta}(x) \in D_\Delta^{\Xi \vdash \Delta(x)}$ *is neutral.*
   - *If $e \in D_\Delta^{\Xi \vdash A \to B}$ is neutral and $d \in D_\Delta^{\Xi \vdash A}$ then $e \cdot d \in D_\Delta^{\Xi \vdash B}$ is neutral.*
   - *If $e \in D_\Delta^{\Xi \vdash \forall^\kappa F}$ is neutral and $G \in \mathcal{T}_\Xi^\kappa$, then $e \cdot G \in D_\Delta^{\Xi \vdash F \cdot G}$ is neutral.*

2. *There exists a Kripke family of* computable *maps*

$$\begin{aligned}\mathsf{view}^{\Xi\vdash B}_{\Delta} \in\ & D^{\Xi\vdash B}_{\Delta} \\ & \to \mathsf{ObjVar} \\ & +\ \bigcup_{A\in\mathcal{T}^{\star}_{\Xi}} D^{\Xi\vdash A\to B}_{\Delta}\times D^{\Xi\vdash A}_{\Delta} \\ & +\ \bigcup_{F\in\mathcal{T}^{\kappa\to\star}_{\Xi}} D^{\Xi\vdash\forall^{\kappa}F}_{\Delta}\times\{G\in\mathcal{T}^{\kappa}_{\Xi}\mid F\cdot G=B\} \\ & +\ \{\bot\}\end{aligned}$$

such that:
- *If* $\mathsf{view}(f)=x$ *then* $f=\mathsf{var}(x)$.
- *If* $\mathsf{view}(f)=(e,d)$ *then* $f=e\cdot d$ *and* e *is neutral.*
- *If* $\mathsf{view}(f)=(e,G)$ *then* $f=e\cdot G$ *and* e *is neutral.*
- *If* $\mathsf{view}(f)=\bot$ *then* f *is not neutral.*

If D is a term-like structure over $\mathcal{T}$, we may write $D(\!|t|\!)^{\rho}$ for $D(\!|t|\!)^{\rho}_{\mathsf{var}_D}$. If $\mathcal{T}$ is also term-like we may further abbreviate $D(\!|t|\!)^{\mathsf{Var}_{\mathcal{T}}}$ to $D(\!|t|\!)$.

Lemma 20 (Injectivity in a term-like object structure). *In a term-like object structure,* var *and neutral application are injective.*

Definition 30 (Object reification). *Given a term-like type structure $\mathcal{T}$ and a term-like object structure D over $\mathcal{T}$, we define the relations*

$$\begin{array}{ll}\Xi;\Delta\vdash d\searrow v\Uparrow A & d \textit{ reifies to } v \textit{ at type } A,\\ \Xi;\Delta\vdash e\searrow u\Downarrow A & e \textit{ reifies to } u\textit{, inferring type } A,\end{array}$$

(where $d,e\in D^{\Xi\vdash A}_{\Delta}$ and v,u are syntactical objects) inductively by the following rules:

$$\frac{}{\Xi;\Delta\vdash x\searrow x\Downarrow\Delta(x)}\qquad\frac{\Xi;\Delta\vdash e\searrow u\Downarrow A\to B\qquad\Xi;\Delta\vdash d\searrow v\Uparrow A}{\Xi;\Delta\vdash e\,d\searrow u\,v\Downarrow B}$$

$$\frac{\Xi;\Delta\vdash e\searrow u\Downarrow\forall^{\kappa}F\qquad\Xi\vdash G\searrow V\Uparrow\kappa}{\Xi;\Delta\vdash e\,G\searrow u\,V\Downarrow F\cdot G}\qquad\frac{\Xi,X\!:\!\kappa;\Delta\vdash f\cdot X\searrow v\Uparrow F\cdot X}{\Xi;\Delta\vdash f\searrow\Lambda X\!:\!\kappa.\,v\Uparrow\forall^{\kappa}F}$$

$$\frac{\Xi;\Delta,x\!:\!A\vdash f\cdot x\searrow v\Uparrow B\qquad\Xi\vdash A\searrow U\Uparrow\star}{\Xi;\Delta\vdash f\searrow\lambda x\!:\!U.\,v\Uparrow A\to B}\qquad\frac{\Xi;\Delta\vdash e\searrow u\Downarrow H}{\Xi;\Delta\vdash e\searrow u\Uparrow H}\ H\textit{ neutral}$$

As for types, object reification is deterministic.

Note that we cannot say now in which $\mathsf{Obj}^{\Xi\vdash T}_{\Gamma}$ the objects u and v live. The conjecture is those Γ,T with $\Xi\vdash\Delta\searrow\Gamma$ and $\Xi\vdash A\searrow T\Uparrow\star$. However, this does not follow directly from the definition, it is a consequence of Thm. 15. (Not even if we suppose $(\Delta,\Gamma)\in\mathcal{S}^{\mathsf{cxt}}_{\Xi}$ and $(A,T)\in\mathcal{S}^{\star}_{\Xi}$, because of the $\forall$-elimination rule.)

Def. and Lem. 14 (Glueing type candidate space) *Let $\mathcal{S}\subseteq\mathcal{T}\times\mathsf{Ty}$ a glueing candidate, $\Vdash_{\mathsf{Gl}}\mathcal{S}$. For $(A,T)\in\mathcal{S}^{\star}_{\Xi}$ we define the Kripke families $\underline{\mathsf{gl}}^{\Xi\vdash(A,T)},\overline{\mathsf{gl}}^{\Xi\vdash(A,T)}\in\widehat{D\times\mathsf{Obj}}^{(A,T)}_{\Xi}$ by*

$$\begin{aligned}\overline{\mathsf{gl}}^{\Xi\vdash(A,T)}_{(\Delta,\Gamma)} &:= \{(d,t)\mid\Xi;\Delta\vdash d\searrow v\Uparrow A\textit{ and }\Xi;\Gamma\vdash t=v:T\textit{ for some }v\},\\ \underline{\mathsf{gl}}^{\Xi\vdash(A,T)}_{(\Delta,\Gamma)} &:= \{(e,t)\mid\Xi;\Delta\vdash e\searrow u\Downarrow A\textit{ and }\Xi;\Gamma\vdash t=u:T\textit{ for some }u\}.\end{aligned}$$

gl *is a type candidate space.*

Proof. We show $\underline{\mathsf{gl}}^{\varXi \vdash A,T} \to \overline{\mathsf{gl}}^{\varXi \vdash B,U} \subseteq \overline{\mathsf{gl}}^{\varXi \vdash A\to B, T\to U}$.

$(f,v) \in (\underline{\mathsf{gl}}^{\varXi \vdash A,T} \to \overline{\mathsf{gl}}^{\varXi \vdash B,U})_{\varDelta,\varGamma}$	assumption
$\varDelta' = \varDelta, x\!:\!A$ and $\varGamma' = \varGamma, x\!:\!T$	
$\varDelta' \vdash x \searrow x \Downarrow A$ and $\varGamma' \vdash x = x : T$	by def.
$(x,x) \in \underline{\mathsf{gl}}^{\varXi \vdash A,T}_{\varDelta',\varGamma'}$	by def.
$(f \cdot x, v\,x) \in \overline{\mathsf{gl}}^{\varXi \vdash B,U}_{\varDelta',\varGamma'}$	from assumption
$\varDelta' \vdash f \cdot x \searrow w \Uparrow B$	and
$\varGamma' \vdash v\,x = w : U$	for some w
$\varDelta' \vdash A \searrow T' \Uparrow \star$	and
$\varGamma' \vdash T = T' : \star$	since $(A,T) \in \mathcal{S}^{\star}_{\varXi}$
$\varDelta \vdash f \searrow \lambda x\!:\!T'.\,w \Uparrow A \to B$	and
$\varGamma \vdash v = \lambda x\!:\!T'.\,w : T \to U$	since $x \notin \mathsf{dom}(\varGamma)$
$(f,v) \in \overline{\mathsf{gl}}^{\varXi \vdash (A,T)\to(B,U)}_{\varDelta,\varGamma}$	by def.□

Theorem 15 (Soundness of NbE for objects). *Let D be a term-like object structure over a term-like type structure $\mathcal{T}$. If $\varXi;\varGamma \vdash t : T$ then there is a long normal form v such that $\varXi;\mathcal{T}[\![\varGamma]\!]_{\mathsf{Var}} \vdash D(\!|t|\!)^{\mathsf{Var}}_{\mathsf{var}} \searrow v \Uparrow \mathcal{T}[\![T]\!]_{\mathsf{Var}}$ and $\varXi;\varGamma \vdash t = v : T$.*

Proof. Consider the type candidate space $\underline{\mathsf{gl}}, \overline{\mathsf{gl}} \in \widehat{D \times \mathsf{Obj}}$ for the glueing type structure G. Recall that

$$\mathsf{gl}^{\kappa}_{\varXi} = \{((F,T),\mathcal{F}) \in \mathsf{G}^{\kappa}_{\varXi} \times \widehat{D \times \mathsf{Obj}}^{(F,T):\kappa}_{\varXi} \mid (F,T) \Vdash^{\kappa}_{\mathsf{gl}} \mathcal{F}\}.$$

By the fundamental theorem of typing, $E^{\varXi \vdash ((A,T),\mathcal{A})}_{((\varDelta,\varGamma),\varPhi)} := \mathcal{A}_{(\varDelta,\varGamma)}$ is an object substructure of $D \times \mathsf{Obj}$ reindexed by $\pi_1 : \mathsf{gl} \to \mathsf{G}$. Since $(X,X) \in \mathsf{G}^{\varXi(X)}_{\varXi}$ and $(X,X) \Vdash^{\varXi(X)}_{\mathsf{gl}}$ $\underline{\mathsf{gl}}^{\varXi \vdash (X,X)}$ we have a variable embedding $\mathsf{Var}_{\mathsf{gl}}(X) = ((X,X), \underline{\mathsf{gl}}^{\varXi \vdash (X,X)}) \in \mathsf{gl}^{\varXi(X)}_{\varXi}$ into type structure gl. Also, since $\varXi;\varDelta \vdash x \searrow x \Downarrow \varDelta(x)$ and $\varXi;\varGamma \vdash x = x : \varGamma(x)$ we have $(x,x) \in \underline{\mathsf{gl}}^{\varXi \vdash (\varDelta(x),\varGamma(x))}_{(\varDelta,\varGamma)}$. The setting of $\varDelta$ such that $\mathsf{G}[\![\varGamma]\!] = (\varDelta,\varGamma)$ implies $(x,x) \in \underline{\mathsf{gl}}^{\varXi \vdash \mathsf{G}[\![\varGamma(x)]\!]}_{\mathsf{G}[\![\varGamma]\!]} \subseteq (\widehat{D \times \mathsf{Obj}}[\![\varGamma(x)]\!])_{\mathsf{G}[\![\varGamma]\!]} = E^{\varXi \vdash \mathsf{gl}[\![\varGamma(x)]\!]}_{\mathsf{gl}[\![\varGamma]\!]}$. Hence, setting $\mathsf{var}_E^{\varXi \vdash \mathsf{gl}[\![\varGamma]\!]_{\mathsf{Var}_{\mathsf{gl}}}}(x) = (x,x)$ we have $\mathsf{var}_E^{\varXi \vdash [\![\varGamma]\!]} \in E^{\varXi \vdash \mathsf{gl}[\![\varGamma]\!]}_{\mathsf{gl}[\![\varGamma]\!]}$.

It follows that $E(\!|t|\!) = E(\!|t|\!)^{\mathsf{Var}_{gl}}_{\mathsf{var}_E} \in E^{\varXi \vdash \mathsf{gl}[\![T]\!]}_{\mathsf{gl}[\![\varGamma]\!]} = (\widehat{D \times \mathsf{Obj}}[\![T]\!])_{\mathsf{G}[\![\varGamma]\!]} \subseteq \overline{\mathsf{gl}}^{\varXi \vdash \mathsf{G}[\![T]\!]}$. With $d := D(\!|t|\!)$ we have $(d,t) = (D \times \mathsf{Obj})(\!|t|\!) = E(\!|t|\!)$, thus, $\varXi;\varDelta \vdash d \searrow v \Uparrow \mathcal{T}[\![T]\!]$ and $\varXi;\varGamma \vdash t = v : T$ for some v. □

8 Object Groupoids

Definition 31 (Object groupoid). *An object structure D over a type structure $\mathcal{T}$ is an* object groupoid *if each $D_{\Delta}^{\Xi\vdash A}$ is a groupoid and*

1. $\mathsf{app}(f,d)^{-1} = \mathsf{app}(f^{-1},d^{-1})$*,*
2. $\mathsf{TyApp}(f,G)^{-1} = \mathsf{TyApp}(f^{-1},G)$*, and*
3. *if $f * f'$ and $d * d'$ are defined, then $\mathsf{app}(f,d) * \mathsf{app}(f',d') = \mathsf{app}(f * f', d * d')$ and $\mathsf{TyApp}(f,G) * \mathsf{TyApp}(f',G) = \mathsf{TyApp}(f * f', G)$.*

Definition 32 (Groupoidal at higher kinds). *Let D be a object groupoid over $\mathcal{T}$. Then $\mathcal{A} \in \widehat{D}^{A:\star}$ is* ($\star$-)groupoidal *if $\mathcal{A}_\Delta$ is a subgroupoid for each Δ. $\mathcal{F} \in \widehat{D}^{F:\kappa\to\kappa'}$ is ($\kappa \to \kappa'$-)groupoidal if $\mathcal{F}(G,\mathcal{G})$ is (κ'-)groupoidal for all $G \in \mathcal{T}_{\Xi}^{\kappa}$ and (κ-)groupoidal $\mathcal{G} \in \widehat{D}_{\Xi}^{G:\kappa}$. Let*

$$\begin{aligned} \widetilde{D}_{\Xi}^{F:\kappa} &:= \{\mathcal{F} \in \widehat{D}_{\Xi}^{F:\kappa} \mid \mathcal{F} \text{ is } \kappa\text{-groupoidal}\}, \\ \mathcal{T}\widetilde{D}_{\Xi}^{\kappa} &:= \{(F,\mathcal{F}) \in \mathcal{T}_{\Xi}^{\kappa} \times \mathcal{T}\widetilde{D}_{\Xi}^{F:\kappa}\}. \end{aligned}$$

Lemma 21 (Function space is groupoidal). *If $\mathcal{A} \in \widehat{D}^{\Xi\vdash A}$ and $\mathcal{B} \in \widehat{D}^{\Xi\vdash B}$ are groupoidal, so is $\mathcal{A} \to_{\widehat{D}} \mathcal{B} \in \widehat{D}^{\Xi\vdash A\to B}$.*

Proof. First, we show that $(\mathcal{A} \to \mathcal{B})_\Delta$ is closed under inversion. Assume $f \in (\mathcal{A} \to \mathcal{B})_\Delta$. To show $f^{-1} \in (\mathcal{A} \to \mathcal{B})_\Delta$, assume arbitrary $\Delta' \leq \Delta$ and $d \in \mathcal{A}_{\Delta'}$ and show $f^{-1} \cdot d \in \mathcal{B}_{\Delta'}$. Since $d^{-1} \in \mathcal{A}_{\Delta'}$ we have $f \cdot d^{-1} \in \mathcal{B}_{\Delta'}$, hence, $\mathcal{B}_{\Delta'} \ni (f \cdot d^{-1})^{-1} = f^{-1} \cdot d$.

Then, we show that $(\mathcal{A} \to \mathcal{B})_\Delta$ is closed under composition. Assume $f, f' \in (\mathcal{A} \to \mathcal{B})_\Delta$ and $f * f'$ defined. Further, assume $\Delta' \leq \Delta$ and $d \in \mathcal{A}_{\Delta'}$ arbitrary and show $(f * f') \cdot d \in \mathcal{B}_{\Delta'}$. Note that $d = d * d^{-1} * d$, hence, it suffices to show $(f \cdot d) * (f' \cdot (d^{-1} * d)) \in \mathcal{B}_{\Delta'}$. Yet this holds, since $f \cdot d \in \mathcal{B}_{\Delta'}$ and $f' \cdot (d^{-1} * d) \in \mathcal{B}_{\Delta'}$ (because $d^{-1} * d \in \mathcal{A}_{\Delta'}$).

Lemma 22 (Type abstraction is groupoidal). *If $\mathcal{A} \in \widehat{D}^{\Xi\vdash F\cdot G}$ is groupoidal, so is $G.\mathcal{A} \in \widehat{D}^{\Xi\vdash\forall^\kappa F}$.*

Proof. Assume $f, f' \in (G.\mathcal{A})_\Delta$ and $f * f'$ defined. Since $f \cdot G \in \mathcal{A}_\Delta$, also $(f \cdot G)^{-1} = f^{-1} \cdot G \in \mathcal{A}_\Delta$. Further, $(f \cdot G) * (f' \cdot G) = (f * f') \cdot G \in \mathcal{A}_\Delta$, hence $f * f' \in (G.\mathcal{A})_\Delta$.

Corollary 4 (Impredicative quantification is groupoidal). $\forall^\kappa \in \widehat{D}^{\Xi\vdash\forall^\kappa:(\kappa\to\star)\to\star}$ *is $(\kappa \to \star) \to \star$-groupoidal.*

Theorem 16 (Fund. thm. of kinding for type groupoids). *Let D be an object groupoid over $\mathcal{T}$. Then $\mathcal{T}\widetilde{D}$ is a type substructure of $\mathcal{T}\widehat{D}$.*

Proof. By Lemma 21 and Cor. 4, $\to_{\mathcal{T}\widehat{D}} \in \mathcal{T}\widetilde{D}_{\Xi}^{\star\to\star\to\star}$ and $\forall^{\kappa}_{\mathcal{T}\widehat{D}} \in \mathcal{T}\widetilde{D}_{\Xi}^{(\kappa\to\star)\to\star)}$. By definition of *groupoidal*, $(F,\mathcal{F}) \cdot_{\mathcal{T}\widehat{D}} (G,\mathcal{G}) = (F \cdot G, \mathcal{F}(G,\mathcal{G})) \in \mathcal{T}\widetilde{D}_{\Xi}^{\kappa'}$ for all $(F,\mathcal{F}) \in \mathcal{T}\widetilde{D}_{\Xi}^{\kappa\to\kappa'}$ and $(G,\mathcal{G}) \in \mathcal{T}\widetilde{D}_{\Xi}^{\kappa}$, thus, application is well-defined.

It remains to show that interpretation is well-defined. Let $\Xi \vdash T : \kappa$ and $(\sigma,\rho) \in \mathcal{T}\widetilde{D}_{\Theta}^{\Xi}$. One easily shows $\mathcal{T}\widehat{D}[\![T]\!]_\rho \in \mathcal{T}\widetilde{D}_{\Theta}^{\kappa}$ by induction on $\Xi \vdash T : \kappa$.

Definition 33 (Square object groupoid). *Let D be an object structure over $\mathcal{T}$. Then ${}^2D^{\Xi\vdash A}_{\Delta} = D^{\Xi\vdash A}_{\Delta} \times D^{\Xi\vdash A}_{\Delta}$ with*

$$\begin{array}{ll} (d, d')^{-1} & = (d', d) \\ (d_1, d_2) * (d_2, d_3) & = (d_1, d_3) \end{array}$$

is a type groupoid over $\mathcal{T}$.

8.1 Fundamental theorem of object equality

Theorem 17 (Fundamental theorem of object equality). *Assume a type structure $\mathcal{T}$ that respects type equality and a combinatory object structure D over $\mathcal{T}$. Let $\underline{\mathcal{C}}, \overline{\mathcal{C}} \in \widetilde{{}^2D}$ be a type candidate space and $\mathcal{C}$ its associated realizability type structure. If $\Xi; \Gamma \vdash t = t' : T$ and $(\sigma, \rho) \in \mathcal{C}^{\Xi}_{\Theta}$ and $(\eta, \eta') \in \widetilde{{}^2D}[\![\Gamma]\!]_{\sigma,\rho}$ then $(D(\!|t|\!)^{\sigma}_{\eta}, D(\!|t'|\!)^{\sigma}_{\eta'}) \in \widetilde{{}^2D}[\![T]\!]_{\sigma,\rho}$.*

Proof. By induction on $\Xi; \Gamma \vdash t = t' : T$. We show a few representative cases.

Case
$$\frac{\Xi, X\!:\!\kappa; \Gamma \vdash t : T \qquad \Xi \vdash U : \kappa}{\Xi; \Gamma \vdash (\Lambda X\!:\!\kappa.\, t)\, U = t[U/X] : T}$$

Let $(G, \mathcal{G}) := \mathcal{T}\widetilde{{}^2D}[\![U]\!]_{\sigma,\rho} \in \mathcal{C}^{\kappa}_{\Theta}$ and $(\sigma', \rho') := (\sigma, \rho)[X \mapsto (G, \mathcal{G})] \in \mathcal{C}^{\Xi, X:\kappa}_{\Theta}$. By induction hypothesis, $(d, d') := ((\!|t|\!)^{\sigma'}_{\eta}, (\!|t|\!)^{\sigma'}_{\eta'}) \in \widetilde{{}^2D}[\![T]\!]_{\sigma',\rho'}$. This entails the goal, since $\widetilde{{}^2D}[\![T[U/X]]\!]_{\sigma,\rho} = \widetilde{{}^2D}[\![T]\!]_{\sigma',\rho'}$ and $((\!|(\Lambda X : \kappa.\, t)\, U|\!)^{\sigma}_{\eta}, (\!|t[U/X]|\!)^{\sigma}_{\eta'}) = (d, d')$ by the combinatory laws.

Case
$$\frac{\Xi; \Gamma \vdash t : \forall^{\kappa} T}{\Xi; \Gamma \vdash \Lambda X\!:\!\kappa.\, t\, X = t : \forall^{\kappa} T}\; X \notin \mathsf{FV}(t)$$

Let $\mathcal{F} := \widetilde{{}^2D}[\![T]\!]_{\sigma,\rho}$, $d := (\!|t|\!)^{\sigma}_{\eta}$ and $d' := (\!|t|\!)^{\sigma}_{\eta'}$. Let further $\hat{d} := (\!|\Lambda X : \kappa.\, t\, X|\!)^{\sigma}_{\eta}$. We have to show $(\hat{d}, d') \in \forall^{\kappa} \mathcal{F}$. Assume arbitrary $(G, \mathcal{G}) \in C^{\kappa}_{\Theta}$. It is sufficient to show $(\hat{d}, d') \cdot G \in \mathcal{F}(G, \mathcal{G})$. Note that $\hat{d} \cdot G = (\!|t\, X|\!)^{\sigma[X \mapsto G]}_{\eta} = (\!|t|\!)^{\sigma[X \mapsto G]}_{\eta} \cdot G = d \cdot G$, since $X \notin \mathsf{FV}(t)$ and D is combinatory. The desired $(d \cdot G, d' \cdot G) \in \mathcal{F}(G, \mathcal{G})$ now follows from the induction hypothesis $(d, d') \in \forall^{\kappa} \mathcal{F}$.

Case
$$\frac{\Xi; \Gamma \vdash t = t' : \forall^{\kappa} T \qquad \Xi \vdash U = U' : \kappa}{\Xi; \Gamma \vdash t\, U = t'\, U' : T\, U}$$

Let $d := (\!|t|\!)^{\sigma}_{\eta}$ and $d' := (\!|t'|\!)^{\sigma}_{\eta'}$. Let further $(G, \mathcal{G}) := (\mathcal{T}\widetilde{{}^2D}[\![U]\!]_{\sigma,\rho}) = (\mathcal{T}\widetilde{{}^2D}[\![U']\!]_{\sigma,\rho})$, since $\mathcal{T}$ (hence also $\mathcal{T}\widetilde{{}^2D}$) respects type equality. First, observe that

$$\begin{array}{ll} (\!|t\, U|\!)^{\sigma}_{\eta} & = d \cdot G \\ (\!|t'\, U'|\!)^{\sigma}_{\eta} & = d' \cdot G \\ \widetilde{{}^2D}[\![T\, U]\!]_{\sigma,\rho} & = \widetilde{{}^2D}[\![T]\!]_{\sigma,\rho}(G, \mathcal{G}) \end{array}$$

By induction hypothesis,

$$(d, d') \in \widetilde{{}^2D}[\![\forall^{\kappa} T]\!]_{\sigma,\rho} = \bigcap_{(G,\mathcal{G}) \in \mathcal{C}^{\kappa}_{\Theta}} G.(\widetilde{{}^2D}[\![T]\!]_{\sigma,\rho}(G, \mathcal{G})).$$

By the fundamental theorem of kinding, $(G, \mathcal{G}) \in \mathcal{C}^{\kappa}_{\Theta}$, hence $(d, d') \cdot G = (d \cdot G, d' \cdot G) \in \widetilde{{}^2D}[\![T]\!]_{\sigma,\rho}(G, \mathcal{G})$, as expected.

Case

$$\frac{\Xi; \Gamma \vdash t = t' : T \qquad \Xi \vdash T = T' : \star}{\Xi; \Gamma \vdash t = t' : T'}$$

Trivial, since $\widetilde{{}^2D}$ respects type equality.

8.2 Completeness of Nbe for objects

Definition 34 (Type candidate space for completeness). *Let D be a term-like object structure over term-like $\mathcal{T}$. We define the Kripke families $\underline{\mathsf{per}}^{\Xi \vdash A}, \overline{\mathsf{per}}^{\Xi \vdash A} \in \widetilde{{}^2D}^{A:\star}_{\Xi}$ by*

$$\begin{aligned}
\overline{\mathsf{per}}^{\Xi \vdash A}_{\Delta} &:= \{(d, d') \in {}^2D^{\Xi \vdash A}_{\Delta} \mid \Xi; \Delta \vdash d \searrow v \Uparrow A \textit{ and} \\
&\qquad \Xi; \Delta \vdash d' \searrow v \Uparrow A \textit{ for some } v\} \\
\underline{\mathsf{per}}^{\Xi \vdash A}_{\Delta} &:= \{(e, e') \in {}^2D^{\Xi \vdash A}_{\Delta} \mid \Xi; \Delta \vdash e \searrow u \Downarrow A \textit{ and} \\
&\qquad \Xi; \Delta \vdash e' \searrow u \Downarrow A \textit{ for some } u\}
\end{aligned}$$

Note that composition $(d_1, d_2) * (d_2, d_3) = (d_1, d_3)$ is well-defined in $\underline{\mathsf{per}}^{\Xi \vdash A}_{\Delta}$ and $\overline{\mathsf{per}}^{\Xi \vdash A}_{\Delta}$ since reification is deterministic.

Lemma 23. $\underline{\mathsf{per}}, \overline{\mathsf{per}}$ *form a type candidate space for ${}^2D$.*

Theorem 18 (Completeness of NbE for objects). *Let D_0 be a term-like object structure over reifyable $\mathcal{T}_0$. If $\Xi; \Gamma \vdash t = t' : T$ then there is a long normal form v such that $\Xi; \Delta \vdash (\!|t|\!) \searrow v \Uparrow A$ and $\Xi; \Delta \vdash (\!|t'|\!) \searrow v \Uparrow A$ where $\Delta := \mathcal{T}[\![\Gamma]\!]$ and $A := \mathcal{T}[\![T]\!]$.*

Proof. The term structure $\mathcal{T} := \mathcal{T}_0/\mathsf{P}$ is reifyable and respects type equality. Let D be defined as D_0 reindexed by the representation function $\mathcal{T} \to \mathcal{T}_0$. Consider the type structure

$$\mathsf{per}^{\kappa}_{\Xi} = \{(F, \mathcal{F}) \in \mathcal{T}^{\kappa}_{\Xi} \times \widetilde{{}^2D}^{F:\kappa}_{\Xi} \mid F \Vdash^{\kappa}_{\mathsf{per}} \mathcal{F}\}.$$

The variable embedding for per is given by $\mathsf{Var}_{\Xi} = (X \mapsto (X, \underline{\mathsf{per}}^{\Xi \vdash X})) \in \mathsf{per}^{\Xi}_{\Xi}$.

$$E^{\Xi \vdash (A, \mathcal{A})}_{(\Delta, \Phi)} := \mathcal{A}_{\Delta}$$

is an object subgroupoid of ${}^2D$ reindexed by $\pi_1 : \mathsf{per} \to \mathcal{T}$. Its variable embedding is given by

$$\begin{aligned}
\mathsf{var}^{\Xi \vdash \Delta}_{E} &= (\mathsf{var}^{\Xi \vdash \pi_1 \circ \Delta}_{D}, \mathsf{var}^{\Xi \vdash \pi_1 \circ \Delta}_{D}) = (x \mapsto (x, x)) \\
&\in \underline{\mathsf{per}}^{\Xi \vdash \Delta}_{\Delta} \subseteq E^{\Xi \vdash (\Delta, \Phi)}_{(\Delta, \Phi)}.
\end{aligned}$$

By the fundamental theorem of object equality we obtain $(D(\!|t|\!), D(\!|t'|\!)) \in \widetilde{{}^2D}[\![T]\!] \subseteq \overline{\mathsf{per}}^{\Xi \vdash A}_{\Delta}$, thus, $\Delta \vdash (\!|t|\!) \searrow v \Uparrow A$ and $\Delta \vdash (\!|t'|\!) \searrow v \Uparrow A$ for some v.

9 Conclusion

We have developed type and object structures, which are sorted applicative structures on type and object level, in order to facilitate generic model constructions for System F^{ω}—which are an alternative to categorical semantics [See87] and Bruce-Meyer-Mitchell models [BM84]. Using special instances of kind candidate spaces we have shown soundness and completeness of an abstract normalization by evaluation algorithm for types. We have gone on to show soundness and completeness of NbE for objects.

We seek to extend NbE to the Calculus of Constructions, using ideas from this work. Due to dependency, type and object levels cannot be defined in sequence, but must be defined simultaneously; this seems to be the main remaining technical difficulty.

Acknowledgments. This work is dedicated to Otto Forster who taught me basic algebraic vocabulary in his excellent lectures on linear algebra in 1994/95 at the Ludwig-Maximilians-University Munich, and to Thierry Coquand who introduced me to abstract semantics of type theory via lambda and PER models.

References

[Abe08] Andreas Abel. Weak $\beta\eta$-normalization and normalization by evaluation for System F. In *LPAR'08*, volume 5330 of *LNAI*, pages 497–511. Springer, 2008.

[ACD07] Andreas Abel, Thierry Coquand, and Peter Dybjer. Normalization by evaluation for Martin-Löf Type Theory with typed equality judgements. In *LICS'07*, pages 3–12. IEEE CS Press, 2007.

[ACD08] Andreas Abel, Thierry Coquand, and Peter Dybjer. Verifying a semantic $\beta\eta$-conversion test for Martin-Löf type theory. In *MPC'08*, volume 5133 of *LNCS*, pages 29–56. Springer, 2008.

[ADHS01] Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Philip J. Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *LICS'01*, pages 303–310. IEEE CS Press, 2001.

[AHS96] Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Reduction-free normalisation for a polymorphic system. In *LICS'96*, pages 98–106. IEEE CS Press, 1996.

[AR08] Andreas Abel and Dulma Rodriguez. Syntactic metatheory of higher-order subtyping. In *CSL'08*, volume 5213 of *LNCS*, pages 446–460. Springer, 2008.

[Bar84] Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North Holland, Amsterdam, 1984.

[Bar08] Freiric Barral. *Decidability for non-standard conversions in lambda-calculus*. PhD thesis, Ludwig-Maximilians-University Munich, 2008.

[BCF04] Vincent Balat, Roberto Di Cosmo, and Marcelo P. Fiore. Extensional normalisation and type-directed partial evaluation for typed lambda calculus with sums. In *POPL'04*, pages 64–76. ACM, 2004.

[BJS07] Frédéric Blanqui, Jean-Pierre Jouannaud, and Pierre-Yves Strub. Building decision procedures in the Calculus of Inductive Constructions. In *CSL'07*, volume 4646 of *LNCS*, pages 328–342. Springer, 2007.

[Bla05] Frédéric Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.

[BM84] Kim B. Bruce and Albert R. Meyer. The semantics of second order polymorphic lambda calculus. In *Semantics of Data Types*, volume 173 of *LNCS*, pages 131–144. Springer, 1984.

[BS91] Ulrich Berger and Helmut Schwichtenberg. An inverse to the evaluation functional for typed λ-calculus. In *LICS'91*, pages 203–211. IEEE CS Press, 1991.

[CAM07] James Chapman, Thorsten Altenkirch, and Conor McBride. Epigram reloaded: a standalone typechecker for ETT. In *TFP'05*, volume 6 of *Trends in Functional Programming*, pages 79–94. Intellect, 2007.

[CD97] Thierry Coquand and Peter Dybjer. Intuitionistic model constructions and normalization proofs. *MSCS*, 7(1):75–94, 1997.

[Coq96] Thierry Coquand. An algorithm for type-checking dependent types. In *MPC'95*, volume 26 of *SCP*, pages 167–177. Elsevier, 1996.

[CWC07] Jacek Chrzaszcz and Daria Walukiewicz-Chrzaszcz. Towards rewriting in Coq. In *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *LNCS*, pages 113–131. Springer, 2007.

[Dan99] Olivier Danvy. Type-directed partial evaluation. In *Partial Evaluation Summer School '98*, volume 1706 of *LNCS*, pages 367–411. Springer, 1999.

[Dyb00] Peter Dybjer. A general formulation of simultaneous inductive-recursive definitions in type theory. *JSL*, 65(2):525–549, 2000.

[Gir72] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur.* Thèse de Doctorat d'État, Université de Paris VII, 1972.

[GL02] Benjamin Grégoire and Xavier Leroy. A compiled implementation of strong reduction. In *ICFP'02*, volume 37 of *SIGPLAN Notices*, pages 235–246. ACM, 2002.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in TCS*. CUP, 1989.

[INR07] INRIA. *The Coq Proof Assistant, Version 8.1.* INRIA, 2007. http://coq.inria.fr/.

[Nor07] Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers, Göteborg, Sweden, 2007.

[Pol94a] Randy Pollack. *The Theory of LEGO*. PhD thesis, University of Edinburgh, 1994.

[Pol94b] Robert Pollack. Closure under alpha-conversion. In *TYPES'93*, volume 806 of *LNCS*, pages 313–332. Springer, 1994.

[See87] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *JSL*, 52(4):969–989, 1987.

[Tai67] William W. Tait. Intensional interpretations of functionals of finite type I. *JSL*, 32(2):198–212, 1967.

[WCPW03] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgements and properties. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2003.