(In)consistency of extensions of Higher Order Logic and Type Theory

Herman Geuvers
herman@cs.ru.nl

Radboud University Nijmegen

Abstract. It is well-known, due to the work of Girard and Coquand, that adding polymorphic domains to higher order logic, HOL, or its type theoretic variant λHOL, renders the logic inconsistent. This is known as Girard's paradox, see [7]. But there is also a another presentation of higher order logic, in its type theoretic variant called λPREDω, to which polymorphic domains can be added safely, Both λHOL and λPREDω are well-known type systems and in this paper we study why λHOL with polymorphic domains is inconsistent and why nd λPREDω with polymorphic domains remains consistent. We do this by describing a simple model for the latter and we show why this can not be a model of the first.

1 Introduction

We study extensions of higher order logic **HOL**in the context of typed lambda calculi. It is known that extensions of higher order logic with polymorphic domains are inconsistent. This was established by Girard [16] and later this result was refined by Coquand [10] who showed that quantification over the collection of all domains wasn't needed to obtain the inconsistency. On the other hand, there are systems like the Calculus of Constructions (**CC**, [9]), which are consistent extensions of higher order logic in which we have polymorphic types. It is not so easy to relate **CC** directly to **HOL**, because in **CC** there is no syntactic distinction between domains and propositions (and therefore between set objects and proof objects). In this paper we therefore study the addition of polymorphic types in the context of a system of higher order order logic, presented as a (isomorphic) type theory following the Curry Howard formulas-as-types isomorphism. See [19] for an overview of formulas-as-types.

We present two isomorphic type systems for higher order logic, λ**HOL** and λ**PRED**ω, and we show why in the first case, the addition of polymorphic sets leads to inconsistency, and in the second case it does not. This is done by describing a model for λ**HOL** (and therefore for λ**PRED**ω), and to see how that can be extended to a model for λ**PRED**ω^+: higher order logic with polymorphism.

The model construction that we use is a variation of models described in [14, 21]. It uses sets of untyped terms as the interpretation of types and is closely

related to a saturated sets model. However, we will not make this connection precise.

The main contribution of the paper is a clarification of the fact that $\lambda\mathbf{PRED}\omega$ with polymorphic sets is consistent. The clue is that the "standard" model for higher order logic, where arrow types are interpreted as set theoretic function spaces does not work anymore if one adds polymorphic types (Reynolds [20] result), but that one can shift the interpretation of types one level lower, interpreting the arrow type $\sigma\to\tau$ as the collection of terms (from a combinatory algebra) that map terms of σ to terms of τ. This is basically the Tait [22] construction for saturated sets.

2 Higher Order Logic as a Pure Type System

2.1 Higher order predicate logic

Definition 1. *The language of* **HOL** *is defined as follows.*

1. *The set of* domains*, D is defined by*

$$D ::= \mathrm{Base} \mid \Omega \mid D\to D,$$

   where Base *represents a* basic domain *(we assume that there are countably many basic domains) and Ω represents the* domain of propositions*.*
2. *For every $\sigma \in D$, the set of* terms of domain σ*, Term_σ is inductively defined as follows. (As usual we write $t : \sigma$ to denote that t is a term of domain σ.)*
   (a) *the constants $c_1^\sigma, c_2^\sigma, \ldots$ are in Term_σ,*
   (b) *the variables $x_1^\sigma, x_2^\sigma, \ldots$ are in Term_σ,*
   (c) *if $\varphi : \Omega$ and x^σ is a variable, then $(\forall x^\sigma.\varphi) : \Omega$,*
   (d) *if $\varphi : \Omega$ and $\psi : \Omega$, then $(\varphi \Rightarrow \psi) : \Omega$,*
   (e) *if $M : \sigma\to\tau$ and $N : \sigma$, then $(MN) : \tau$,*
   (f) *if $M : \tau$ and x^σ is a variable, then $(\lambda x^\sigma.M) : \sigma\to\tau$.*
3. *The set of terms of* **HOL**, Term*, is defined by* $\mathrm{Term} := \cup_{\sigma\in D}\mathrm{Term}_\sigma$.
4. *The set of formulas of* **HOL**, **form**, *is defined by* $\mathbf{form} := \mathrm{Term}_\Omega$.

We adapt the well-known notions of *free* and *bound* variable, substitution, β-reduction and β-conversion to the terms of this system. The λ-abstraction is both used for defining functions of higher type, like $\lambda f^{(\sigma\to\sigma)\to\sigma}.f(\lambda x^\sigma.x) : (\sigma\to\sigma)\to\sigma\to\sigma$, and for *comprehension*. Comprehension is the *axiom scheme* $\exists X(\forall \boldsymbol{x}(X\boldsymbol{x} \leftrightarrow \varphi)$, where $\boldsymbol{x}$ is the sequence of free variables of the formula φ. Comprehension holds, because we can always take $X := \lambda\boldsymbol{x}.\varphi$.

There are no 'product' domains $(\sigma \times \sigma)$ in our logic. We represent functions of higher arity by *currying*: a binary function on σ is represented as a term in the domain $\sigma\to(\sigma\to\sigma)$. A predicate is represented as a function to Ω, following the idea (probably due to Church; it appears in [6]) that a predicate can be seen as a function that takes a value as input and returns a formula. So, a binary relation over σ is represented as a term in the domain $\sigma\to(\sigma\to\Omega)$. (If $R : \sigma\to(\sigma\to\Omega)$ and $t, q : \sigma$, then $((Rt)q) : \Omega$.) The logical connectives are just implication and

universal quantification. Due to the fact that we have *higher order* universal quantification, we can constructively express all other quantifiers using just $\Rightarrow$ and $\forall$. See [11] for more details.

We fix the usual notational conventions that outside brackets are omitted and that in the domains we omit the brackets by letting them associate to the right, so $\sigma{\to}\sigma{\to}\Omega$ denotes $\sigma{\to}(\sigma{\to}\Omega)$. In terms we omit brackets by associating them to the left, so Rtq denotes $(Rt)q$.

The derivation rules of **HOL** are given in a natural deduction style.

Definition 2. *The notion of* provability, $\Gamma \vdash \varphi$, *for* Γ *a finite set of formulas (terms of domain* **form***) and* φ *a formula, is defined inductively as follows.*

(*axiom*)	$\dfrac{}{\Gamma \vdash \varphi}$	*if* $\varphi \in \Gamma$
($\Rightarrow$ -*introduction*)	$\dfrac{\Gamma \cup \varphi \vdash \psi}{\Gamma \vdash \varphi \Rightarrow \psi}$	
($\Rightarrow$ -*elimination*)	$\dfrac{\Gamma \vdash \varphi \quad \Gamma \vdash \varphi \Rightarrow \psi}{\Gamma \vdash \psi}$	
($\forall$-*introduction*)	$\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x^\sigma.\varphi}$	*if* $x^\sigma \notin \mathrm{FV}(\Gamma)$
($\forall$-*elimination*)	$\dfrac{\Gamma \vdash \forall x^\sigma.\varphi}{\Gamma \vdash \varphi[t/x^\sigma]}$	*if* $t : \sigma$
(*conversion*)	$\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \psi}$	*if* $\varphi =_\beta \psi$

Remark 1. The rule (conversion) is an operationalization of the comprehension axiom. The rule says that we don't want to distinguish between β-equal propositions.

2.2 Extension with polymorphic domains

Extending higher order logic with polymorphic domains make the system inconstsitent. This extension amounts to the system U^-. Allowing also quantification over all domains yields teh system U. Both systems were defined in [16] and it was shown there that U is inconsistent, which became known as *Girard's paradox*. Later it was shown by [10] and [18] that U^- is also inconsistent. We now define these systems.

Definition 3. *The set of* domains *of* U^- D_U *is defined by*

$$D ::= \mathrm{Base} \mid \mathrm{Var}^D \mid \Omega \mid D{\to}D \mid \Pi_A.D$$

where Var^D *is a set of variables ranging over* D_U *and* $A \in \mathrm{Var}^D$.

For $\sigma \in D_U$, *the set of* terms of domain σ in U^-, $\mathrm{Term}^{U^-}_\sigma$ *is inductively defined as follows.*

1. *if* $t : \Pi_A.\tau$ *and* $\sigma \in D_U$, *then* $t\sigma : \tau[\sigma/A]$
2. *if* $t : \tau$, *then* $\lambda_A.t : \Pi_A.\tau$

The derivation rules for U^- *are the same as for* **HOL**.

The system U *is the extension of* U^-, *where the terms are extended as follows.*

3. *if* $\varphi : \Omega$, *then* $\forall_A.\varphi : \Omega$

The additional derivation rules for U *are:*

$$\boxed{\begin{array}{l}(\forall_2\text{-}introduction)\ \dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \forall_A.\varphi} \quad \text{if } A \notin \mathrm{FV}(\Gamma) \\ \\ (\forall_2\text{-}elimination)\ \dfrac{\Gamma \vdash \forall_A.\varphi}{\Gamma \vdash \varphi[\sigma/A]}\ \text{if } \sigma \in D_U\end{array}}$$

The systems U and U^- are inconsistent, which can be phrased as "higher order logic with polymorphic domains is inconsistent". However, the Calculus of Constructions [9] contains both higher order logic *and* polymorphic domains but it is still consistent. How to understand this seeminlgly paradoxical situation will be explained in this paper, using a model of higher order logic, defined as a typed λ calculus.

The Calculus of Constructions is a type theory where no distinction between objects and proofs is made. The Curry-Howard formulas as types embedding gives an embedding of higher order logic into **CC**, but it is not conservative. However, **CC** is consistent and contains higher order logic, so it must be possible extends **HOL** with polymorphic domains in a consistent way. To see that, we define **PRED**ω, which is a variant of **HOL** and its extension **PRED**ω^+, which is higher order logic with polymorphic domains.

Definition 4. *The set of* domains *of* **PRED**ω D_ω *is defined by*

$$\begin{aligned} D_w &::= D_s \mid D_p \\ D_p &::= \Omega \mid D \to D_p \\ D_s &::= B \mid D \to D_s \end{aligned}$$

The rules for terms and the derivation rules are the same as for **HOL**.

The system **PRED**ω^+ *is the extension of* **PRED**ω, *where the domains* D_s *are as follows.*

$$D_s ::= \mathrm{Base} \mid \mathrm{Var}^D \mid D \to D_s \mid \Pi_A.D_s$$

where Var^D *is a set of variables ranging over* D_s *and* $A \in \mathrm{Var}^D$.

For $\sigma \in D$, *the set of* terms of type σ in U^-, $\mathrm{Term}^{\mathbf{PRED}\omega^+}_\sigma$ *is inductively defined as follows.*

1. *if* $t : \Pi_A.\tau$ *and* $\sigma \in D_s$, *then* $t\sigma : \tau[\sigma/A]$
2. *if* $t : \tau$ *and* $\tau \in D_s$, *then* $\lambda_A.t : \Pi_A.\tau$

The derivation rules for $\mathbf{PRED}\omega^+$ *are the same as for* $\mathbf{PRED}\omega$.

It can be shown that $\mathbf{PRED}\omega$ is isomorphic to **HOL**. The system $\mathbf{PRED}\omega^+$ is in flavor very close to U^-, both being extensions of **HOL** with polymorphic domains. However, $\mathbf{PRED}\omega^+$ is consistent.

2.3 Pure Type Systems for higher order logic

In type theory, one interprets formulas and proofs via the well-known 'formulas-as-types' and 'proofs-as-terms' embedding, originally due to Curry, Howard and de Bruijn. (See [17, 5].) Under this interpretation, a formula is viewed as the type of its proofs. It turns out that one can define a typed λ-calculus $\lambda\mathbf{HOL}$ that represents **HOL** in a very precise way. What *very precise* means will not be defined here, but see e.g. [2] or [12]. In this section we briefly introduce the general framework of Pure Type Systems or PTSs. These were first introduced by [3] and [23], under different names and with slightly different definitions, as a generalization of the so called λ-*cube*, see [2]. The reason for defining the class of PTSs is that many known systems are (or better: can be seen as) PTSs. Here we will focus on higher order logic seen as a PTS.

Definition 5. *For* $\mathcal{S}$ *a set (the set of sorts),* $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ *(the set of axioms)and* $\mathcal{R} \subset \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ *(the set of rules), the* Pure Type System $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ *is the typed lambda calculus with the following deduction rules.*

$(sort)$	$\vdash s_1 : s_2$	*if* $(s_1, s_2) \in \mathcal{A}$
(var)	$\dfrac{\Gamma \vdash T : s}{\Gamma, x{:}T \vdash x : T}$	*if* $x \notin \Gamma$
$(weak)$	$\dfrac{\Gamma \vdash T : s \quad \Gamma \vdash M : U}{\Gamma, x{:}T \vdash M : U}$	*if* $x \notin \Gamma$
(Π)	$\dfrac{\Gamma \vdash T : s_1 \quad \Gamma, x{:}T \vdash U : s_2}{\Gamma \vdash \Pi x{:}T.U : s_3}$	*if* $(s_1, s_2, s_3) \in \mathcal{R}$
(λ)	$\dfrac{\Gamma, x{:}T \vdash M : U \quad \Gamma \vdash \Pi x{:}T.U : s}{\Gamma \vdash \lambda x{:}T.M : \Pi x{:}T.U}$	
(app)	$\dfrac{\Gamma \vdash M : \Pi x{:}T.U \quad \Gamma \vdash N : T}{\Gamma \vdash MN : U[N/x]}$	
$(conv_\beta)$	$\dfrac{\Gamma \vdash M : T \quad \Gamma \vdash U : s}{\Gamma \vdash M : U}$	$T =_\beta U$

If $s_2 \equiv s_3$ in a triple $(s_1, s_2, s_3) \in \mathcal{R}$, we write $(s_1, s_2) \in \mathcal{R}$. In the derivation rules, the expressions are taken from the set of pseudo-terms $\mathcal{T}$ *defined by*

$$\mathcal{T} ::= \mathcal{S} \mid \mathcal{V} \mid (\Pi \mathcal{V}{:}\mathcal{T}.\mathcal{T}) \mid (\lambda \mathcal{V}{:}\mathcal{T}.\mathcal{T}) \mid \mathcal{T}\mathcal{T}.$$

The pseudo-term T is legal *if there is a context Γ and a pseudo-term U such that $\Gamma \vdash T : U$ or $\Gamma \vdash U : T$ is derivable. The set of legal terms of $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ is denoted by* $\mathrm{Term}(\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R}))$.

By convention we write $A \to B$ for $\Pi x{:}A.B$ if $x \notin \mathrm{FV}(B)$.

It is instructive to define an example PTS to see how flexible the notion is. In the following, we describe a PTS by just listing the sort, the axioms and the rules in a box. For higher order logic **HOL** this amounts to the following λ**HOL**.

λ**HOL**
$\mathcal{S}$ $\star, \Box, \Delta$
$\mathcal{A}$ $\star : \Box, \Box : \Delta$
$\mathcal{R}$ $(\star,\star), (\Box,\Box), (\Box,\star)$

The formulas-as-types interpretation from higher order predicate logic **HOL** into λ**HOL** maps a formula to a type and a derivation (in natural deduction) of formula φ to a typed λ-term (of the type associated with φ):

$$\begin{array}{|c|}\hline \\ \Sigma \\ \\ \hline \\ \psi \end{array} \mapsto [\![\Sigma]\!] : (\!|\psi|\!)$$

where $(\!|-|\!)$ denotes the interpretation of formulas as types and $[\![-]\!]$ denotes the interpretation of derivations as λ-terms. In a derivation, we use expressions from the logical language (e.g. to instantiate the $\forall$), which may contain free variables, constants and domains. In type theory, in order to make sure that all terms are well-typed, the basic items (like variables and domains) have to be declared explicitly in the context. Also, a derivation will in general contain non-discharged assumptions $(\varphi_1, \ldots, \varphi_n)$ that will appear as variable declarations $(z_1 : \varphi_1, \ldots, z_n : \varphi_n)$ in the type theoretic context. So the general picture is this.

$$\begin{array}{c} \varphi_1 \ldots \varphi_n \\ \begin{array}{|c|}\hline \\ \Sigma \\ \\ \hline \\ \psi \end{array} \end{array} \mapsto \Gamma_\Sigma, z_1 : \varphi_1, \ldots, z_n : \varphi_n \vdash [\![\Sigma]\!] : (\!|\psi|\!),$$

where Γ_Σ is the context that declares all domains, constants and free variables that occur in Σ.

The system λ**HOL** is stratified in the sense that one can alternatively define it "layer by layer", starting from the terms of type Δ, then the terms of type

A where $A : \Delta$ etcetera. We introduce some naming and notation conventions for $\lambda\mathbf{HOL}$. Some of these depend on properties of $\lambda\mathbf{HOL}$ that we do not prove here.

Remark 2. 1. There is only one term of type Δ, and that is $\Box$.
2. A term of type $\Box$ is called a *kind*. Typical kind names are $\sigma, \tau, \ldots$. So a kind is a term σ with $\Gamma \vdash \sigma : \Box$ for some Γ. All kinds are of the shape $\sigma_1 \to \ldots \to \star$ or $\sigma_1 \to \ldots \to A$ with A a variable of type $\Box$.
3. A term of type a kind is called a *constructor*. Typical constructor names are $P, Q, \ldots$. So a constructor is a term P with $\Gamma \vdash P : \sigma$, where σ is a kind (so $\Gamma \vdash \sigma : \Box$) for some Γ. If $\sigma = \star$, then we call P a *type*. Typical type names are $\varphi, \psi, \ldots$. So a type is a term φ with $\Gamma \vdash \varphi : \star$ for some Γ.
4. An *object* is a term of a type, so a p with $\Gamma \vdash p : \varphi$ where φ is a type, for some Γ. Typical object names are $p, q, \ldots$.

Calling the kinds σ conforms with the use of these names in **HOL**, where the domains were called σ. A domain in **HOL** corresponds with a kind in $\lambda\mathbf{HOL}$.

Remark 3. There are three "ways" of introducing a variable in $\lambda\mathbf{HOL}$. For each of these cases we take the variables from a specific subset of Var and we use specific notations for these variables. So we assume Var to be the disjoint subset of Var^{Δ}, $\mathrm{Var}^{\Box}$ and $\mathrm{Var}^{\star}$. The three cases are:

1. $A : \Box$; we take these variables, the *kind variables*, from Var^{Δ} and we use A as a typical name for such a variable.
2. $v : \sigma$ with $\sigma : \Box$; we take these variables, the *constructor variables*, from $\mathrm{Var}^{\Box}$ and we use α as a typical name for such a variable.
3. $v : \varphi$ with $\varphi : \star$; we take these variables, the *object variables*, from $\mathrm{Var}^{\star}$ and we use x as a typical name for such a variable.

Here is a visual way of thinking of these classes of terms

	Δ	Δ	
	..	..	
	$\Box$	$\Box$	
	..	..	
kinds	σ	$\star$	a special kind
	..	..	
constructors	P	φ	types (a special case of constructors)
		..	
		p	objects

The systems U^- and U can easily be seen as a Pure Type System as follows.

Definition 6. *The systems $\lambda\mathbf{U}^-$ and $\lambda\mathbf{U}$ are defined by adding to $\lambda\mathbf{HOL}$ respectively the rule $(\Delta, \Box)$ and the rules $(\Delta, \Box)$ and $(\Delta, \star)$.*

Another way of looking at higher order logic is the system **PRED**ω. This can be turned into a PTS as follows (originally due to [3]).

λPREDω
$\mathcal{S}$ $\mathsf{Set}, \mathsf{Type}^s, \mathsf{Prop}, \mathsf{Type}^p$
$\mathcal{A}$ $\mathsf{Set} : \mathsf{Type}^s, \mathsf{Prop} : \mathsf{Type}^p$
$\mathcal{R}$ $(\mathsf{Set}, \mathsf{Set}), (\mathsf{Set}, \mathsf{Type}^p), (\mathsf{Type}^p, \mathsf{Type}^p), (\mathsf{Prop}, \mathsf{Prop}),$ $(\mathsf{Set}, \mathsf{Prop}), (\mathsf{Type}^p, \mathsf{Prop})$

It can be formally shown that λ**HOL** and λ**PRED**ω are isomorphic. We will come to that later. We can also view λ**PRED**ω in a stratified way and we could have introduced in in that way. This would be very close to starting off from the domains, then the terms and then the proofs, as we did for **PRED**ω.

	Type^s			
	..			
	Set	Type^p	Type^p	
	..	..	..	
sets	σ	K	Prop	kinds (Prop is a special kind)
	..	..	..	
set objects	t	P	φ	constructors (a type is a special constr.)
			..	
			p	proof objects

We can also add polymorphic types to λ**PRED**ω, which amounts to the system λ**PRED**ω^+

Definition 7. *The systems* λ**PRED**ω^+ *is defined by adding to* λ**PRED**ω *the rule* $(\mathsf{Type}^s, \mathsf{Set})$.

We introduce two other known type systems as PTSs: **F**ω of [16]) and the Calculus of Constructions, **CC** of [9].

CC
$\mathcal{S}$ $\star, \Box$
$\mathcal{A}$ $\star : \Box$
$\mathcal{R}$ $(\star, \star), (\star, \Box), (\Box, \star), (\Box, \Box)$

Fω
$\mathcal{S}$ $\star, \Box$
$\mathcal{A}$ $\star : \Box$
$\mathcal{R}$ $(\star, \star), (\Box, \Box), (\Box, \star)$

In view of higher order predicate logic, one can understand CC as the system obtained by smashing the sorts Prop and Set into one, $\star$. Hence, higher order

predicate logic can be done inside the Calculus of Constructions. We describe the map from λPREDω to **CC** later in detail.

The system **F**ω is known to be consistent. As a consequence, λ**HOL** is consistent: if we map all kind variables to $\star$, then the rules are preserved, so we have an embedding of λ**HOL** into **F**ω, where $\bot$ $(:= \Pi\alpha : \star.\alpha)$ is mapped to itself. As $\bot$ is not inhabited in **F**ω, it is not inhabited in λ**HOL**.

One can sometimes relate results of two different systems by defining an embedding between them. There is one very simple class of embeddings between PTSs.

Definition 8. *For $T = \lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ and $T' = \lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ PTSs, a* PTS-morphism *from T to T' is a mapping $f : \mathcal{S} \to \mathcal{S}'$ that preserves the axioms and rules. That is, for all $s_1, s_2 \in \mathcal{S}$, if $(s_1, s_2) \in \mathcal{A}$ then $(f(s_1), f(s_2)) \in \mathcal{A}'$ and if $(s_1, s_2, s_3) \in \mathcal{R}$ then $(f(s_1), f(s_2), f(s_3)) \in \mathcal{R}'$.*

A PTS-morphism f from $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ to $\lambda(\mathcal{S}', \mathcal{A}', \mathcal{R}')$ extends immediately to a mapping f on pseudo-terms and contexts. Moreover, this mapping preserves reduction in a faithful way: $M \to_\beta N$ iff $f(M) \to_\beta f(N)$. We have the following property.

Proposition 1. *For T and T' PTSs and f a PTS-morphism from T to T', if $\Gamma \vdash M : A$ in T, then $f(\Gamma) \vdash f(M) : f(A)$ in T'.*

Not all PTSs are Strongly Normalizing. We have the following well-known theorem.

Theorem 1. *The Calculus of Constructions,* **CC***, is Strongly Normalizing.*

The proof can e.g. be found in [15], [13], [3].

As a consequence we find that many other PTSs are Strongly Normalizing as well. This comprises all the sub-systems of **CC** and also all systems T for which there is a PTS-morphism from T to **CC**. (Note that a PTS-morphism preserves infinite reduction paths.)

Corollary 1. *The following PTSs are all Strongly Normalizing. All subsystems of* **CC***; λPRED; λPREDω.*

Well-known example of PTSs that are not Strongly Normalizing are λ**U** and λ**U**$^-$.

As a matter of fact, we now have two formalizations of higher order predicate logic as a PTS: λ**HOL** and λPREDω. We employ the notion of PTS-morphism to see that they are equivalent. From λPREDω to λ**HOL**, consider the PTS-morphism f given by

$$\begin{aligned} f(\mathsf{Prop}) &= \star, \\ f(\mathsf{Set}) &= \Box, \\ f(\mathsf{Type}^p) &= \Box, \\ f(\mathsf{Type}^s) &= \Delta. \end{aligned}$$

One verifies immediately that f preserves $\mathcal{A}$ and $\mathcal{R}$, hence we have

$$\Gamma \vdash_{\lambda\mathrm{PRED}\omega} M : A \implies f(\Gamma) \vdash_{\lambda\mathbf{HOL}} f(M) : f(A).$$

The inverse of f can almost be described as a PTS-morphism, but not quite. Define the PTS-morphism g from λPREDω to λ**HOL** as follows.

$$\begin{aligned} g(\star) &= \mathsf{Prop}, \\ g(\Box) &= \mathsf{Set}, \\ g(\Delta) &= \mathsf{Type}^s \end{aligned}$$

(In λ**HOL** the sort Δ can not appear in a context nor in a term on the left side of the ':'.) We extend g to derivable judgments of λ**HOL** in the following way.

$$\begin{aligned} g(\Gamma \vdash M : A) &= g(\Gamma) \vdash g(M) : g(A), \text{ if } A \neq \mathsf{Type}^p, \\ g(\Gamma \vdash M : \mathsf{Type}^p) &= g(\Gamma) \vdash g(M) : \mathsf{Set}, \text{ if } M \equiv \cdots \to \alpha, (\alpha \text{ a variable}), \\ g(\Gamma \vdash M : \mathsf{Type}^p) &= g(\Gamma) \vdash g(M) : \mathsf{Type}^p, \text{ if } M \equiv \cdots \to \mathsf{Prop}. \end{aligned}$$

By easy induction one proves that g preserves derivations. Furthermore, $f(g(\Gamma \vdash M : A)) = \Gamma \vdash M : A$ and $g(f(\Gamma \vdash M : A)) = \Gamma \vdash M : A$. Hence, λPREDω and λ**HOL** are equivalent systems. This equivalence implies that the system λ**HOL** is Strongly Normalizing as well. But we already knew that, because it is also a consequence of the embedding of λ**HOL** into **F**ω and the fact that **F**ω is Strongly Normalizing.

3 The model construction

3.1 Combinatory Algebras

To model the set of pseudo-terms of type theories we can use *combinatory algebras* ca, or variants of combinatory algebras, like *partial combinatory algebras* pca or *conditionally partial combinatory algebras* c-pca. We list the important notions used in this paper. Most of the definitions in this section are taken from [1] and [4].

Definition 9. *A combinatory algebra (ca) is an applicative structure* $\mathcal{A} = \langle\ \mathbf{A}\ ,\ .\ ,\ \mathbf{k}\ ,\ \mathbf{s}\ ,\ =_A\ \rangle$ *with distinguished elements* $\mathbf{k}$ *and* $\mathbf{s}$ *satisfying*

$$(\mathbf{k}.x).y =_A x\ , \qquad ((\mathbf{s}.x).y).z =_A (x.z).(y.z)$$

The application (.) is usually not written.

Definition 10. *The set of terms over* $\mathcal{A}$ *(notation* $\mathcal{T}(\mathcal{A})$*) is defined as follows.*

$$\mathcal{T} ::= \mathrm{Var} \mid \mathbf{A} \mid \mathcal{T}\mathcal{T}$$

Every ca is *combinatory complete*, i.e., for every $T \in \mathcal{T}(\mathcal{A})$ with $\mathrm{FV}(T) \subset \{x\}$, there exists an $f \in \mathbf{A}$ such that

$$f \cdot a =_A T[a/x] \qquad \forall a \in \mathbf{A}.$$

Such an element f will be denoted by $\lambda x.T$ in the sequel. It is well-known that one can define λ as the standard abstraction λ^* with the help of the combinators $\mathbf{k}$ and $\mathbf{s}$ as follows, by induction on the structure of terms.

Definition 11.

$$\begin{aligned} \lambda x.P &:= \mathbf{k}P \text{ if } x \notin \mathrm{FV}(P) \\ \lambda x.x &:= \mathbf{skk} \\ \lambda x.PQ &:= \mathbf{s}(\lambda x.P)(\lambda x.Q) \end{aligned}$$

In the following we could restrict ourselves completely to the combinatory algebra $(\Lambda, \cdot, \lambda xy.x, \lambda xyz.xz(yz), =_\beta)$, where Λ is the set of (open, untyped) λ-terms and $\cdot$ denotes application, which we usually don't write. But the constructions apply more generally to other combinatory algebras, as long as they are *weakly extensional*, i.e. if the following holds

$$\mathcal{A} \models \forall x(T_1 = T_2) \rightarrow \lambda x.T_1 = \lambda x.T_2.$$

The combinatory algebra CL (combinatory logic) is not weakly extensional. The combinatory algebra $(\Lambda, \cdot, \lambda xy.x, \lambda xyz.xz(yz), =_\beta)$ is weakly extensional, if we take for $\lambda x.T$ just $\lambda x.T$ or the definable $\lambda - .-$ as given above. It is well-known that if we take for Λ the set of *closed* lambda terms, the ca is not weakly extensional. Another interesting example of a weakly extensional ca is $\mathbf{\Lambda}(C)$, the set of open λ_C-terms (i.e. lambda-terms over some constant set C) modulo βc-equality, where the c-equality rule is defined by $cN =_c c$, $\lambda v.c =_c c$ (for all $c \in C$ and $N \in \Lambda_C$).

3.2 The Model for λHOL

The notion of λ**HOL**-*structure* and the interpretations of the typable terms of are explained informally in the next paragraphs.

The typable terms of λ**HOL** are mapped into a (set-theoretical) hierarchical structure (called λ**HOL**-structure) according to their classification as objects, types, constructors, or kinds. The kinds of λ**HOL** are interpreted as sets from a *predicative structure* N, so Type^p is interpreted as N. Predicative structures are closed under set-theoretical function space construction. The impredicative universe Prop is interpreted as a collection P of subsets of the underlying ca. We call this collection *polystructure* and its elements *polysets*. P itself is an element of N and is closed under non-empty intersections and a function space construction (to be defined). Constructors are interpreted as elements of $\bigcup_{X \in \mathsf{N}} X$ ($\bigcup \mathsf{N}$ in short). Their interpretations are called *poly-functionals*. In particular, types are mapped to polysets.

Definition 12. *A* polyset structure *over the weakly extensional combinatory algebra $\mathcal{A}$ is a collection $\mathsf{P} \subseteq \wp(\mathbf{A})$ such that*

1. $\mathbf{A} \in \mathsf{P}$,
2. P *is closed under arbitrary non-empty intersection* $\bigcap$*: if* $I \neq \emptyset$ *and* $\forall i \in I(X_i \in \mathsf{P})$*, then* $\bigcap_{i\in I} X_i \in \mathsf{P}$.
3. P *is closed under* function space*, i.e. if* $X, Y \in \mathsf{P}$*, then* $X{\to_0}Y \in \mathsf{P}$*, where* $X{\to_0}Y$ *is defined as*

$$\{a \in \mathbf{A} \mid \forall t \in X(a \cdot t \in Y)\}.$$

The elements of a polyset structure are called polysets.

Example 1. 1. We obtain the *full polyset structure* over the weca $\mathcal{A}$ if we take $\mathsf{P} = \wp(\mathbf{A})$.
2. The *simple polyset structure* over the weca $\mathcal{A}$ is obtained by taking $\mathsf{P} = \{\emptyset, \mathbf{A}\}$. It is easily verified that this is a polyset structure.
3. Given the weca $\mathbf{\Lambda}(C)$ as defined in Example **??** (so C is a set of constants), we define the *polyset structure generated from* C by

$$\mathsf{P} := \{X \subseteq \mathbf{\Lambda}(C) \mid X = \emptyset \vee C \subseteq X\}.$$

4. Given the weca $\mathcal{A}$ and a set $C \subseteq \mathbf{A}$ such that $\forall a, b \in \mathbf{A}(a \cdot b \in C \implies a \in C$, we define the *power polyset structure of* C by

$$\mathsf{P} := \{X \subseteq \mathbf{A} \mid X \subseteq C \vee X = \mathbf{A}\}.$$

5. The *degenerate polyset structure* is $\mathsf{P} := \{\mathbf{A}\}$, in which all types are interpreted as $\mathbf{A}$, so in this structure there are no empty types.

The function space of a polyset structure will be used to interpret types of the form $\varphi{\to}\psi$, where both φ and ψ are types. The intersection will be used to interpret types of the form $\Pi\alpha{:}\sigma.\varphi$, where σ is a kind and φ is a type. To interpret types we need a *predicative structure*.

Definition 13. *For* P *a polyset structure, the* predicative structure over P *is the collection of sets* N *defined inductively by*

1. $\mathsf{P} \in \mathsf{N}$,
2. *If* $X, Y \in \mathsf{P}$*, then* $X{\to_1}Y \in \mathsf{N}$*, where* $\to_1$ *denotes the set-theoretic function space.*

Definition 14. *If $\mathcal{A}$ is a weakly extensional combinatory algebra,* P *a polyset structure over $\mathcal{A}$ and* N *the predicative structure over* P*, then we call the tuple* $\langle \mathcal{A}, \mathsf{P}, \mathsf{N}\rangle$ *a* λ**HOL**-model.

Remark 4. It is possible to vary on the notions of polystructure and predicative structure by requiring closure under dependent function spaces (in P and/or N). In that case we obtain models that can interpret dependent types. For details we refer to [21] or [14].

We now define the interpretation function $[\![-]\!]$, which maps kinds to elements of N, constructors to elements of $\bigcup\mathsf{N}$ (and types to elements of P, which is a subset of $\bigcup\mathsf{N}$) and objects to elements of the combinatory algebra $\mathcal{A}$. All these interpretations are parameterized by *valuations*, assigning values to the free variables (declared in the context).

Definition 15. *A* variable valuation *is a map from* $\mathrm{Var}^{\Delta} \cup \mathrm{Var}^{\Box} \cup \mathrm{Var}^{\star}$ *to* $\mathsf{N} \cup \bigcup\mathsf{N} \cup \mathbf{A}$ *that consists of the union of an* object variable valuation $\rho_0 : \mathrm{Var}^{\star} \to \mathbf{A}$*, a* constructor variable valuation $\rho_1 : \mathrm{Var}^{\Box} \to \bigcup\mathsf{N}$ *and a* kind variable valuation $\rho_2 : \mathrm{Var}^{\Box} \to \mathsf{N}$.

Definition 16. *For ρ a variable valuation, we define the map $(\![-]\!)_\rho$ on the set of well-typed objects as follows. (We leave the model implicit.)*

$$\begin{aligned}
(\![x]\!)_\rho &:= \rho(x),\\
(\![tq]\!)_\rho &:= (\![t]\!)_\rho \cdot (\![q]\!)_\rho, \text{ if } q \text{ is an object},\\
(\![tQ]\!)_\rho &:= (\![t]\!)_\rho, \text{ if } Q \text{ is a constructor},\\
(\![\lambda x{:}\varphi.t]\!)_\rho &:= \lambda v.(\![t]\!)_{\rho(x:=v)}, \text{ if } \varphi \text{ is a type},\\
(\![\lambda \alpha{:}\sigma.t]\!)_\rho &:= (\![t]\!)_\rho, \text{ if } \sigma \text{ is a kind}.
\end{aligned}$$

Definition 17. *For ρ a variable valuation, we define the maps $\mathcal{V}(-)_\rho$ and $[\![-]\!]_\rho$ respectively from kinds to N and from constructors to $\bigcup\mathsf{N}$ as follows. (We leave the model implicit.)*

$$\begin{aligned}
\mathcal{V}(\star)_\rho &:= \mathsf{P},\\
\mathcal{V}(A)_\rho &:= \rho(A), \text{ if } A \text{ is a kind variable},\\
\mathcal{V}(\sigma{\to}\tau)_\rho &:= \mathcal{V}(\sigma)_\rho {\to_1} \mathcal{V}(\tau)_\rho,\\
[\![\alpha]\!]_\rho &:= \rho(\alpha),\\
[\![\Pi\alpha{:}\sigma.\varphi]\!]_\rho &:= \bigcap_{a\in\mathcal{V}(\sigma)_\rho} [\![\varphi]\!]_{\rho(\alpha:=a)}, \text{ if } \sigma \text{ is a kind},\\
[\![\varphi{\to}\psi]\!]_\rho &:= [\![\varphi]\!]_\rho {\to_0} [\![\psi]\!]_\rho, \text{ if } \varphi,\psi \text{ are a types},\\
[\![PQ]\!]_\rho &:= [\![P]\!]_\rho([\![Q]\!]_\rho),\\
[\![\lambda\alpha{:}\sigma.P]\!]_\rho &:= \lambda\!\!\lambda a \in \mathcal{V}(\sigma)_\rho.[\![P]\!]_{\rho(\alpha:=a)}.
\end{aligned}$$

Definition 18. *For Γ a λ**HOL**-context, ρ a variable valuation, we say that ρ* fulfills *Γ, notation $\rho \models \Gamma$, if for all $A \in \mathrm{Var}^{\Delta}$, $x \in \mathrm{Var}^{\star}$ and $\alpha \in \mathrm{Var}^{\Box}$, $A \in \Box \in \Gamma \Rightarrow \rho(A) \in \mathsf{N}$, $\alpha : \sigma \in \Gamma \Rightarrow \rho(\alpha) \in \mathcal{V}(\sigma)_\rho$ and $x : \varphi \in \Gamma \Rightarrow \rho(x) \in [\![\varphi]\!]_\rho$.*

It is (implicit) in the definition that $\rho \models \Gamma$ only if for all declarations $x{:}\varphi \in \Gamma$, $[\![\sigma]\!]_\rho$ is defined (and similarly for $\alpha{:}\sigma \in \Gamma$).

Definition 19. *The notion of* truth in a $\lambda\mathbf{HOL}$-model*, notation* $\models^{\mathcal{S}}$ *and of* truth*, notation* $\models$ *are defined as follows. For* Γ *a context,* t *an object,* φ *a type,* P *a constructor and* σ *a kind of* $\lambda\mathbf{HOL}$*,*

$$\Gamma \models^{\mathcal{S}} t : \varphi \textit{ if } \forall \rho [\rho \models \Gamma \Rightarrow (\!|t|\!)_\rho \in [\![\varphi]\!]_\rho],$$
$$\Gamma \models^{\mathcal{S}} P : \sigma \textit{ if } \forall \rho [\rho \models \Gamma \Rightarrow [\![P]\!]_\rho \in \mathcal{V}(\sigma)_\rho].$$

Quantifying over the class of all $\lambda\mathbf{HOL}$*-models, we define, for* M *an object or a constructor of* $\lambda\mathbf{HOL}$*,*

$$\Gamma \models M : T \;\; \textit{if} \;\; \Gamma \models^{\mathcal{S}} M : T \textit{ for all } \lambda\mathbf{HOL}\textit{-models } \mathcal{S}.$$

Soundness states that if a judgment $\Gamma \vdash M : T$ is derivable, then it is true in all models. It is proved 'model-wise', by induction on the derivation in $\lambda\mathbf{HOL}$.

Theorem 2 (Soundness). *For* Γ *a context,* M *an object or a constructor and* T *a type or a kind of* $\lambda\mathbf{HOL}$*,*

$$\Gamma \vdash M : T \Rightarrow \Gamma \models M : T.$$

Example 2. Let $\mathcal{A}$ be a weca.

1. The *full* $\lambda\mathbf{HOL}$*-model* over $\mathcal{A}$ is $\mathcal{S} = \langle \mathcal{A}, \mathsf{P}, \mathsf{N} \rangle$, where P is the full polyset structure over $\mathcal{A}$ (as defined in Example 1).
2. The *simple* $\lambda\mathbf{HOL}$*-model* over $\mathcal{A}$ is $\mathcal{S} = \langle \mathcal{A}, \mathsf{P}, \mathsf{N} \rangle$, where P is the simple polyset structure over $\mathcal{A}$. (So $\mathsf{P} = \{\emptyset, \mathbf{A}\}$.)
3. The simple $\lambda\mathbf{HOL}$-model over the degenerate $\mathcal{A}$ is also called the *proof-irrelevance model* or *PI-model* for $\lambda\mathbf{HOL}$.
4. For C a set of constants, the $\lambda\mathbf{HOL}$*-model generated from* C is defined by $\mathcal{S} = \langle \mathbf{\Lambda}(C), \mathsf{P}, \mathsf{N} \rangle$, where P is the polyset structure generated from C.

4 Extending the model construction

4.1 Extensions of λHOL

The model for $\lambda\mathbf{HOL}$ can be extended to other type theories. First of all we remark that the rule $(\Delta, \star)$ can easily be interpreted by putting

$$[\![\Pi A{:}\Box.\varphi]\!]_\rho := \bigcap_{W \in \mathsf{N}} [\![\varphi]\!]_{\rho(A:=W)}.$$

This can be interpreted in any model, so the extension of $\lambda\mathbf{HOL}$ with the rule $(\Delta, \star)$ is consistent.

The rule $(\Delta, \Box)$ makes $\lambda\mathbf{HOL}$ inconsistent. This can be observed in the model, because the only possible interpretation in N for $\Pi A{:}\Box.\sigma$ would be

$$\mathcal{V}(\Pi A{:}\Box.\sigma)_\rho := \bigcap_{W \in \mathsf{N}} \mathcal{V}(\sigma)_{\rho(A:=W)},$$

which would only make sense if N were also a polyset structure. (If N were set theoretic, $\mathcal{V}(\Pi A{:}\Box.\sigma)_\rho$ would just be empty.) But this can only be achieved by taking $\mathsf{P} := \{\mathbf{A}\}$, the degenerate polyset structure. (See 1.) then $\mathsf{N} := \{\{\mathbf{A}\}\}$, which can be seen as a polyset structure and is then closed under $\to_0$. In this model all types are interpreted as a non-empty set ($\mathbf{A}$), which conforms with the fact that $\lambda\mathbf{U}^-$ is inconsistent.

4.2 λPREDω and extensions

As λ**HOL** is isomorphic to λ**PRED**ω, we also have a model for λ**PRED**ω. As we want to vary on the type theory λ**PRED**ω, we make the model construction for λ**PRED**ω precise here. As a model we just take the definition of λ**HOL**-model as given in Definition 14.

Definition 20. *A* variable valuation *is a map from* $\mathrm{Var}^{\mathsf{Type}^s} \cup \mathrm{Var}^{\mathsf{Type}^p} \cup \mathrm{Var}^{\mathsf{Set}} \cup \mathrm{Var}^{\mathsf{Prop}}$ *to* $\mathsf{N} \cup \bigcup\mathsf{N} \cup \mathbf{A}$ *that consists of the union of an* proof object variable valuation $\rho_0 : \mathrm{Var}^{\mathsf{Prop}} \to \mathbf{A}$*, a* constructor variable valuation $\rho_{1a} : \mathrm{Var}^{\mathsf{Type}^p} \to \bigcup\mathsf{N}$*, a* set object variable valuation $\rho_{1b} : \mathrm{Var}^{\mathsf{Set}} \to \bigcup\mathsf{N}$ *and a* set variable valuation $\rho_2 : \mathrm{Var}^{\mathsf{Type}^s} \to \mathsf{N}$.

Definition 21. *For ρ a variable valuation, we define the map $(\!|-|\!)_\rho$ on the set of well-typed proof objects of λ**PRED**ω as follows. (We leave the model implicit.)*

$$\begin{aligned}
(\!|x|\!)_\rho &:= \rho(x),\\
(\!|tq|\!)_\rho &:= (\!|t|\!)_\rho \cdot (\!|q|\!)_\rho, \text{ if } q \text{ is a proof object},\\
(\!|tQ|\!)_\rho &:= (\!|t|\!)_\rho, \text{ if } Q \text{ is a constructor or a set object},\\
(\!|\lambda x{:}\varphi.t|\!)_\rho &:= \lambda v.(\!|t|\!)_{\rho(x:=v)}, \text{ if } \varphi \text{ is a type},\\
(\!|\lambda\alpha{:}U.t|\!)_\rho &:= (\!|t|\!)_\rho, \text{ if } U \text{ is a kind or a set}.
\end{aligned}$$

Definition 22. *For ρ a variable valuation, we define the maps $\mathcal{V}(-)_\rho$ and $[\![-]\!]_\rho$ respectively from kinds of λ**PRED**ω to N and from constructors and set objects of λ**PRED**ω to $\bigcup\mathsf{N}$ as follows. (We leave the model implicit.)*

$$\begin{aligned}
\mathcal{V}(\mathsf{Prop})_\rho &:= \mathsf{P},\\
\mathcal{V}(A)_\rho &:= \rho(A), \text{ if } A \in \mathrm{Var}^{\mathsf{Type}^s},\\
\mathcal{V}(\sigma{\to}\tau)_\rho &:= \mathcal{V}(\sigma)_\rho {\to_1} \mathcal{V}(\tau)_\rho,\\
[\![\alpha]\!]_\rho &:= \rho(\alpha), \text{ if } \alpha \in \mathrm{Var}^{\mathsf{Type}^p} \cup \mathrm{Var}^{\mathsf{Set}},\\
[\![\Pi\alpha{:}U.\varphi]\!]_\rho &:= \bigcap_{a \in \mathcal{V}(U)_\rho} [\![\varphi]\!]_{\rho(\alpha:=a)}, \text{ if } U \text{ is a kind or a set},\\
[\![\varphi{\to}\psi]\!]_\rho &:= [\![\varphi]\!]_\rho {\to_0} [\![\psi]\!]_\rho, \text{ if } \varphi, \psi \text{ are a types},\\
[\![PQ]\!]_\rho &:= [\![P]\!]_\rho([\![Q]\!]_\rho), \text{ if } Q \text{ is a constructor or a set object},\\
[\![\lambda\alpha{:}U.P]\!]_\rho &:= \lambda\!\!\lambda a \in \mathcal{V}(U)_\rho.[\![P]\!]_{\rho(\alpha:=a)} \text{ if } U \text{ is a kind or a set}.
\end{aligned}$$

For Γ a $\lambda\mathbf{PRED}\omega$-context, ρ a variable valuation, the notion of ρ *fulfills* Γ ($\rho \models \Gamma$), is the similar to the one for $\lambda\mathbf{HOL}$:
$A \in \mathsf{Set} \in \Gamma \Rightarrow \rho(A) \in \mathsf{N}$, $\alpha : \sigma \in \Gamma \Rightarrow \rho(\alpha) \in \mathcal{V}(\sigma)_\rho$ (for σ a set), $\alpha : K \in \Gamma \Rightarrow \rho(\alpha) \in \mathcal{V}(K)_\rho$ (for K a kind) and $x : \varphi \in \Gamma \Rightarrow \rho(x) \in [\![\varphi]\!]_\rho$.

The notion of *truth* is the same as for $\lambda\mathbf{HOL}$ models (Definition 19) and we also have a soundness result, like Theorem 2.

To compare the situation fro $\lambda\mathbf{HOL}$ and $\lambda\mathbf{PRED}\omega$, we can take a look at the two figures 1 and 2. The first describes how the different "levels" of $\lambda\mathbf{HOL}$ and $\lambda\mathbf{PRED}\omega$ are interpreted in the model. (Forget about the part of dashed arrows for now.) The second describes how the function spaces are interpreted in the model. Again omit the dashed arrows and the part that is not between the two dashed lines on the right.

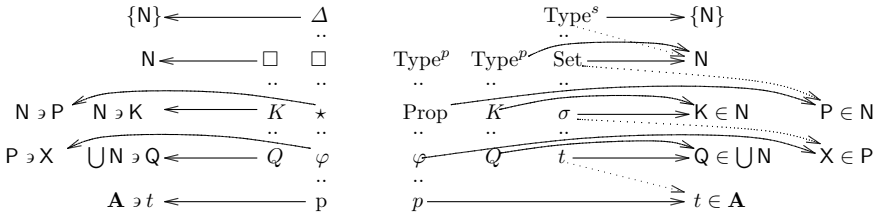

Fig. 1. Interpretation of the different object, types and sorts

We now want to look at $\lambda\mathbf{PRED}\omega^+$, the extension of $\lambda\mathbf{PRED}\omega$ with polymorphic kinds (higher order logic with polymorphic domains). In this system we have types of the form $\Pi A{:}\mathsf{Set}.A{\to}A : \mathsf{Set}$ and the system is known to be consistent. According to the $\lambda\mathbf{PRED}\omega$ semantics, we would have to put

$$\mathcal{V}(\Pi A{:}\mathsf{Set}.A{\to}A)_\rho \in \mathsf{N},$$

but then we would have to interpret the Π either as a set-theoretic dependent function space (which is not possible due to cardinality reasons) or as an intersection, and then N would have to be a polyset structure as well. The latter would amount to a model in which all types are interpreted as non-empty sets, which is not what we want.

The solution is to "shift" the interpretation of the sort Set one level down, interpreting Set as P and $\sigma : \mathsf{Set}$ as a polyset. Then the interpretation of $\Pi A{:}\mathsf{Set}.A{\to}A$ will be as follows.

$$[\![\Pi A{:}\mathsf{Set}.A{\to}A]\!]_\rho = \bigcap_{X \in \mathsf{P}} X{\to_0}X \in \mathsf{P}.$$

This amounts to the dashed arrows in the Figures 1 and 2. In order to define this interpretation we have to extend the notions of polyset structure

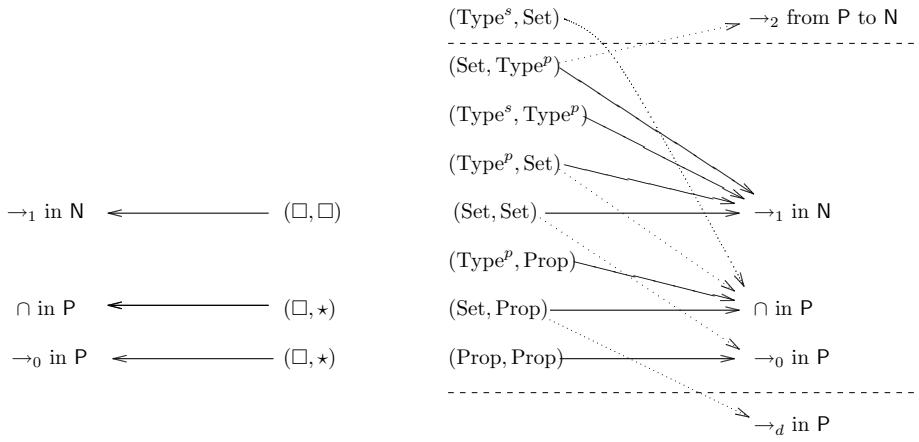

Fig. 2. Interpretation of the different (dependent) function spaces

and predicative structure a bit. As we now have *dependent types* at the polyset level, we need a *dependent function type* in P. Moreover, we have type dependent functions from polysets to predicative set.

Definition 23. *A* polyset structure for $\lambda\mathbf{PRED}\omega^+$ *is a polyset structure that is moreover* closed under dependent function spaces*, ($\to_p$ in Figure 2): if $F : X \to \mathsf{P}$ is a function such that*

$$t =_{\mathbf{A}} q \Rightarrow F(t) = F(q),$$

then P *also contains*

$$\textstyle\prod_{\mathbf{A}}(X, F) := \{f \in \mathbf{A} \mid \forall n \in X(f \cdot n \in F(n))\}$$

For convenience $\prod_{\mathbf{A}}(X, F)$ will be denoted by $\prod_{\mathbf{A}} x \in X.F(x)$. Like in type theory, if F is a constant function on X, say $F(x) = Y$, then we denote $\prod_{\mathbf{A}}(X, F)$ is just the function space $X \to_0 Y$, which is defined as $\{f \in \mathbf{A} \mid \forall n \in X(f \cdot n \in Y)\}$.

Definition 24. *A* predicative structure for $\lambda\mathbf{PRED}\omega^+$ over P *is a predicative structure that is moreover* closed under function spaces from P to N*, ($\to_2$ in Figure 2): if $X \in \mathsf{P}$ and $\mathsf{K} \in \mathsf{N}$, then the following is also in* N

$$X \to_2 \mathsf{K} := \{h \mid \forall t, q \in X, t =_{\mathbf{A}} q \Rightarrow h(t) = h(q) \in \mathsf{K}\}.$$

Note that the elements of $X \to_2 \mathsf{K}$ are set-theoretic functions.

We now make the model construction for $\lambda\mathbf{PRED}\omega^+$ precise. As a model we just take the definition of $\lambda\mathbf{HOL}$-model, where the polyset structure and the predicative structure are as in Definitions **??**, 24. The interpretations will be such that they conform with the dashed arrows in Figures 1 and 2.

Definition 25. *A* variable valuation for $\lambda\mathbf{PRED}\omega^+$ *is a map from* $\mathrm{Var}^{\mathsf{Type}^s} \cup \mathrm{Var}^{\mathsf{Type}^p} \cup \mathrm{Var}^{\mathsf{Set}} \cup \mathrm{Var}^{\mathsf{Prop}}$ *to* $\mathsf{N} \cup \bigcup \mathsf{N} \cup \mathbf{A}$ *that consists of the union of an* proof object variable valuation $\rho_0 : \mathrm{Var}^{\mathsf{Prop}} \to \mathbf{A}$*, a* constructor variable valuation $\rho_{1a} : \mathrm{Var}^{\mathsf{Type}^p} \to \bigcup \mathsf{N}$*, a* set object variable valuation $\rho_{1b} : \mathrm{Var}^{\mathsf{Set}} \to \mathbf{A}$ *and a* set variable valuation $\rho_2 : \mathrm{Var}^{\mathsf{Type}^s} \to \mathsf{P}$.

Definition 26. *For ρ a variable valuation for $\lambda\mathbf{PRED}\omega^+$, we define the map $(\!|-|\!)_\rho$ on the set of well-typed proof objects* and set objects *of $\lambda\mathbf{PRED}\omega$ as follows.*

$$
\begin{aligned}
(\!|x|\!)_\rho &:= \rho(x), \ \textit{if } x \in \mathrm{Var}^{\mathsf{Prop}} \cup \mathrm{Var}^{\mathsf{Set}},\\
(\!|tq|\!)_\rho &:= (\!|t|\!)_\rho \cdot (\!|q|\!)_\rho, \ \textit{if } q \textit{ is a proof object or set object},\\
(\!|tQ|\!)_\rho &:= (\!|t|\!)_\rho, \ \textit{if } Q \textit{ is a constructor or a set},\\
(\!|\lambda x{:}U.t|\!)_\rho &:= \lambda v (\!|t|\!)_{\rho(x:=v)}, \ \textit{if } U \textit{ is a type or a set},\\
(\!|\lambda \alpha{:}K.t|\!)_\rho &:= (\!|t|\!)_\rho, \ \textit{if } K \textit{ is a kind or } \mathsf{Set}.
\end{aligned}
$$

Definition 27. *For ρ a variable valuation, we define the maps $\mathcal{V}(-)_\rho$ and $[\![-]\!]_\rho$ respectively from kinds of $\lambda\mathbf{PRED}\omega^+$ and $\{\mathsf{Set}\}$ to N and from constructors and*

sets of $\lambda\mathbf{PRED}\omega^+$ *to* $\bigcup\mathsf{N}$ *as follows.*

$$
\begin{aligned}
\mathcal{V}(\mathsf{Prop})_\rho &:= \mathsf{P},\\
\mathcal{V}(\mathsf{Set})_\rho &:= \mathsf{P},\\
\mathcal{V}(K_1{\to}K_2)_\rho &:= \mathcal{V}(K_1)_\rho{\to_1}\mathcal{V}(K_2)_\rho,\ \textit{if } K_1 \textit{ is a kind},\\
\mathcal{V}(\sigma{\to}K)_\rho &:= [\![\sigma]\!]_\rho{\to_2}\mathcal{V}(K)_\rho,\ \textit{if } \sigma \textit{ is a set},\\
[\![A]\!]_\rho &:= \rho(A),\ \textit{if } A \in \mathrm{Var}^{\mathsf{Type}^s},\\
[\![\alpha]\!]_\rho &:= \rho(\alpha),\ \textit{if } \alpha \in \mathrm{Var}^{\mathsf{Type}^p},\\
[\![\Pi A{:}\mathsf{Set}.\sigma]\!]_\rho &:= \bigcap_{X\in\mathsf{P}} [\![\sigma]\!]_{\rho(A:=X)},\\
[\![K{\to}\sigma]\!]_\rho &:= [\![\sigma]\!]_\rho,\ \textit{if } K \textit{ is a kind},\\
[\![\Pi\alpha{:}K.\varphi]\!]_\rho &:= \bigcap_{a\in\mathcal{V}(K)_\rho} [\![\varphi]\!]_{\rho(\alpha:=a)},\ \textit{if } K \textit{ is a kind},\\
[\![\Pi\alpha{:}\sigma.\varphi]\!]_\rho &:= \prod_{\mathbf{A}} t \in [\![\sigma]\!]_\rho.[\![\varphi]\!]_{\rho(\alpha:=t)},\ \textit{if } \sigma \textit{ is a set},\\
[\![U{\to}T]\!]_\rho &:= [\![U]\!]_\rho{\to_0}[\![T]\!]_\rho,\ \textit{if } T,U \textit{ are types or sets},\\
[\![PQ]\!]_\rho &:= [\![P]\!]_\rho([\![Q]\!]_\rho),\ \textit{if } Q \textit{ is a constructor},\\
[\![Pt]\!]_\rho &:= [\![P]\!]_\rho((\!|t|\!)_\rho),\ \textit{if } t \textit{ is a set object},\\
[\![\lambda\alpha{:}K.P]\!]_\rho &:= \lambda\!\!\lambda a \in \mathcal{V}(K)_\rho.[\![P]\!]_{\rho(\alpha:=a)}\ \textit{if } K \textit{ is a kind},\\
[\![\lambda\alpha{:}\sigma.P]\!]_\rho &:= \lambda\!\!\lambda a \in [\![\sigma]\!]_\rho.[\![P]\!]_{\rho(\alpha:=a)}\ \textit{if } \sigma \textit{ is a set}.
\end{aligned}
$$

Similar to $\lambda\mathbf{HOL}$ and $\lambda\mathbf{PRED}\omega$ there is a soundness result for $\lambda\mathbf{PRED}\omega^+$, saying that, if the valuation ρ *fulfill* the context Γ, then if $\Gamma \vdash P : K$ (K a kind or Set), then $[\![P]\!]_\rho \in \mathcal{V}(K)_\rho$ and if $\Gamma \vdash t : T$ (T a type or a set), then $(\!|t|\!)_\rho \in [\![T]\!]_\rho$.

As a consequence of the model construction, $\lambda\mathbf{PRED}\omega^+$ is consistent, but we already knew that (because of the embedding into **CC**). It is noteworthy that the model for $\lambda\mathbf{PRED}\omega^+$ is very different from the model for $\lambda\mathbf{PRED}\omega$. This is no surprise, because we know from [20] that polymorphism is no set-theoretic, so a $\lambda\mathbf{PRED}\omega$ model does not extend to a $\lambda\mathbf{PRED}\omega^+$ model in a direct way.

To illustrate this further we consider the following $\lambda\mathbf{PRED}\omega$ context

$$\Gamma := B : \mathsf{Set}, E : B{\to}\mathsf{Prop}, \varepsilon : \mathsf{Prop}{\to}B, h : \Pi\alpha{:}\mathsf{Prop}E(\varepsilon\alpha) = \alpha.$$

. Here, $=$ denotes the Leibniz equality on Prop: $\varphi = \psi := \Pi P{:}\mathsf{Prop}{\to}\mathsf{Prop}.P\,\varphi{\to}P\,\psi$. This context was considered by Coquand in [8] as a context of **CC**, so $\Gamma := B : \star, E : B{\to}\star, \varepsilon : \star{\to}B, h : \Pi\alpha{:}\star\, E(\varepsilon\alpha) = \alpha$. It was shown that Γ is inconsistent, because one can embed $\lambda\mathbf{U}^-$ into it. Here we use Γ to show the difference between $\lambda\mathbf{PRED}\omega$ and $\lambda\mathbf{PRED}\omega^+$.

Lemma 1. *Γ is consistent in $\lambda\mathbf{PRED}\omega$.*

Proof. Take a model in which $\emptyset \in \mathsf{P}$ and take the valuation ρ as follows: $\rho(B) := \mathsf{P}$, $\rho(E) = \rho(\varepsilon)$ is the identity, $\rho(h) := \lambda x.x$. Then $\rho \models \Gamma$ and $[\![\Pi\alpha{:}\mathsf{Prop}.\alpha]\!]_\rho = \emptyset$, so Γ is consistent.

Lemma 2. *Γ is inconsistent in $\lambda\mathbf{PRED}\omega^+$.*

It is instructive to first look at the interpretation of Γ in a $\lambda\mathbf{PRED}\omega^+$ model. Suppose $\rho(B) = \mathsf{B}$. Then $\mathcal{V}((\,)_\rho B{\to}\mathsf{Prop}) = \mathsf{B}{\to_2}\mathsf{P}$ and $[\![\mathsf{Prop}{\to}B]\!]_\rho = \mathsf{B}$. So for a valuation ρ to fulfill Γ, we need that $\rho(\varepsilon) \in \mathsf{B}$ and $\rho(E) \in \mathsf{B}{\to_2}\mathsf{P}$ such that $\rho(E)\rho(\varepsilon) = X$ for any $X \in \mathsf{P}$. This is only possible if $\mathsf{P} = \{\mathbf{A}\}$, the degenerate polyset structure in which all polysets are non-empty.

We now give the proof of the Lemma, which basically follows Coquand's proof in [8] (but Coquand's proof is for **CC**).

Proof. We embed $\lambda\mathbf{U}^-$ into the context Γ of $\lambda\mathbf{PRED}\omega^+$ as follows.

$$\begin{array}{ll} \overline{\Delta} := \mathsf{Type}^s & \overline{\Pi A{:}\Box.\sigma} := \Pi A{:}\mathsf{Type}^s.\overline{\sigma} \\ \overline{\Box} := \mathsf{Set} & \overline{\sigma{\to}\tau} := \overline{\sigma}{\to}\overline{\tau} \\ \overline{\star} := B & \overline{\Pi\alpha{:}\sigma.\varphi} := \varepsilon(\Pi\alpha{:}\overline{\sigma}.\overline{\varphi} \\ & \overline{\varphi{\to}\psi} := \varepsilon(E\overline{\varphi}{\to}E\overline{\psi}) \end{array}$$

Now one can prove the following

$$\begin{array}{l} \Gamma' \vdash_{\lambda\mathbf{U}^-} M : T \Rightarrow \Gamma, \overline{\Gamma'} \vdash_{\lambda\mathbf{PRED}\omega^+} \overline{M} : \overline{T} \text{ if } T : \Box, \Delta \\ \Gamma' \vdash_{\lambda\mathbf{U}^-} M : T \Rightarrow \exists N[\Gamma, \overline{\Gamma'} \vdash_{\lambda\mathbf{PRED}\omega^+} N : E\overline{T}] \text{ if } T : \star \end{array}$$

Therefore $\Gamma \vdash N : E(\overline{\Pi\alpha{:}\star.\alpha})$ for some N. But $E(\overline{\Pi\alpha{:}\star.\alpha}) = E(\varepsilon(\Pi\alpha{:}B.E\alpha)) = \Pi\alpha{:}B.E\alpha$, so we have a term of $\Pi\alpha{:}B.E\alpha$. Taking $\varepsilon\bot$ for α, we have $E(\varepsilon\bot)$ and therefore $\bot$. □

References

1. Henk P. Barendregt. *The lambda calculus, its syntax and semantics*. North-Holland, Amsterdam, second, revised edition, 1984.
2. Henk P. Barendregt. Lambda calculi with types. In D.M. Gabbai Samson Abramski and T.S.E. Maiboum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, Oxford, 1992.
3. S. Berardi. *Type dependence and constructive mathematics*. PhD thesis, Mathematical Institute, University of Turino, Italy, 1990.
4. Inge Bethke and Jan Willem Klop. Collapsing partial combinatory algebras. In *100*, page 17. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 30 1995.
5. N. G. de Bruijn. The mathematical language AUTOMATH. volume 25 of *Lecture Notes in Mathematics*, pages 29–61. Springer, Berlin, 1970.
6. A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 1940.
7. Th. Coquand. An analysis of girard's paradox. In *Logic in Computer Science*, pages 227–236. IEEE, 1986.
8. Th. Coquand. Metamathematical investigations of a Calculus of Constructions. In P. Odifreddi, editor, *Logic and computer science*, pages 91–122. Academic Press, London, 1990. The APIC SERIES , vol. 31.

9. Th. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76:96–120, 1988.
10. Thierry Coquand. A new paradox in type theory. In D. Prawitz, B. Skyrms, and D. Westerståhl, editors, *Proceedings 9th Int. Congress of Logic, Methodology and Philosophy of Science, Uppsala, Sweden, 7–14 Aug 1991*, volume 134, pages 555–570. North-Holland, Amsterdam, 1994.
11. D. van Dalen. *Logic and structure*. Springer, Berlin, second edition, 1983.
12. Herman Geuvers. *Logics and Type Systems*. PhD thesis, University of Nijmegen, Netherlands, 1993.
13. Herman Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In P. Dybjer, B. Nordström, and J. Smith, editors, *Selected Papers 2nd Int. Workshop on Types for Proofs and Programs, TYPES'94, Båstad, Sweden, 6–10 June 1994*, volume 996, pages 14–38. Springer-Verlag, Berlin, 1995.
14. Herman Geuvers. Induction is not derivable in second order dependent type theory. *Lecture Notes in Computer Science*, 2044:166–??, 2001.
15. J. H. Geuvers and M.-J. Nederhof. A modular proof of strong normalization for the Calculus of Constructions. *JoFP*, 1(2):155–189, 1991.
16. J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.
17. W. A. Howard. The formulae-as-types notion of construction. In J. R. Hindley and J. P. Seldin, editors, *Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, London, 1980. Dedicated to Haskell B. Curry on the occasion of his 80th birthday.
18. T. Hurkens. A simplification of girard's paradox. In *TLCA '95: Proceedings of the Second International Conference on Typed Lambda Calculi and Applications*, pages 266–278, London, UK, 1995. Springer-Verlag.
19. Pawel Urzyczyn Morten Heine Sørensen. *Lectures on the Curry-Howard isomorphism*, volume 149 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam, second, revised edition, 2006.
20. John C. Reynolds. Polymorphism is not set-theoretic. In G. Kahn, David B. McQueen, and Gordon D. Plotkin, editors, *Semantics of datatypes*, volume 173 of *Lecture Notes in Computer Science*, pages 145–156. Springer-Verlag, New York, 1984.
21. Milena Stefanova and Herman Geuvers. A simple model construction for the calculus of constructions. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs, Int. Workshop, Torino*, volume 1158 of *LNCS*, pages 249–264, 1996.
22. W.W. Tait. Intensional interpretation of functionals of finite type I. *JSL*, 32:198–212, 1967.
23. Jan Terlouw. Een nadere bewijstheoretische analyse van gegeneraliseerde systemen van termen en typen, 1989. Manuscript, University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands.