HAL open science

Conservativity of embeddings in the lambda-Pi calculus modulo rewriting (long version)

Ali Assaf

To cite this version:

Ali Assaf. Conservativity of embeddings in the lambda-Pi calculus modulo rewriting (long version). 2013. hal-01084165v1

HAL Id: hal-01084165

https://hal.science/hal-01084165v1

Preprint submitted on 18 Nov 2014 (v1), last revised 20 Apr 2015 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Conservativity of embeddings in the lambda-Pi calculus modulo rewriting

Ali Assaf

June 5, 2014

Abstract

The lambda-Pi calculus can be extended with rewrite rules to embed any other functional pure type system. The normalization and conservativity properties of the embedding is an open problem. In this paper, we show that the embedding is conservative. We define an inverse translation into a pure type system completion and show that the completion is conservative using the reducibility method. This result further justifies the use of the lambda-Pi calculus modulo rewriting as a logical framework.

1 Introduction

The $\lambda\Pi$-*calculus modulo rewriting* is an extension of the $\lambda\Pi$-calculus with rewrite rules. Through the Curry-de Bruijn-Howard correspondence, it can express the proofs of various logics. Cousineau and Dowek [5] introduced a general embedding of *functional pure type systems* (FPTS) in the $\lambda\Pi$-calculus modulo rewriting: for any FPTS λS, they contructed the system $\lambda\Pi/S$ using appropriate rewrite rules, and defined two translation functions $|M|$ and $\|A\|$ that translate the terms and types of λS to $\lambda\Pi/S$. This embedding is complete[1], in the sense that it preserves typing: if $\Gamma \vdash_{\lambda S} M : A$ then $\|\Gamma\| \vdash_{\lambda\Pi/S} |M| : \|A\|$. The converse property, called *conservativity*, was shown to hold in some cases: assuming $\lambda\Pi/S$ is strongly normalizing, if there is a term N such that $\|\Gamma\| \vdash_{\lambda\Pi/S} N : \|A\|$ then there is a term M such that $\Gamma \vdash_{\lambda S} M : A$.

Not much is known about normalization in $\lambda\Pi/S$. Cousineau and Dowek showed that if $\lambda\Pi/S$ is strongly normalizing, then so is λS, but this was not enough to show the conservativity of the embedding, even if λS is itself strongly normalizing. As a result, the proof of conservativity relied on the assumption that $\lambda\Pi/S$ terminates. This result is insufficient if one wants to consider the $\lambda\Pi$-calculus modulo rewriting as a general logical framework for defining logics and expressing proofs in those logics, as proposed in [3, 4].

Consider the PTS λHOL that corresponds to higher order logic:

$$\begin{array}{rcl} \mathcal{S} & = & Prop, Type, Kind \\ \mathcal{A} & = & (Prop : Type), (Type : Kind) \\ \mathcal{R} & = & (Prop, Type), (Type, Prop), \\ & & (Type, Type), (Kind, Kind) \end{array}$$

[1] This property is called *soundness* instead of *completeness* in the original paper [5].

This PTS is strongly normalizing, and therefore consistent. A polymorphic variant of λHOL is specified by $U^- = HOL + (Kind, Type)$. It turns out that λU^- is inconsistent: there is a term ω such that $\vdash_{\lambda U^-} \omega : \Pi\alpha : Prop.\, \alpha$ and which is not normalizing [1]. We motivate the need for conservativity with the following example.

Example 1.1. The polymorphic identity function $I = \lambda\alpha{:}Type.\, \lambda x{:}\alpha.\, x$ is not well-typed in λHOL, but it is well-typed in λU^-:

$$\vdash_{\lambda U^-} I : \Pi\alpha{:}Type.\, \alpha \to \alpha$$

$$\vdash_{\lambda U^-} (\Pi\alpha{:}Type.\, \alpha \to \alpha) : Type$$

However, the translation $|I| = \lambda\alpha{:}u_{Type}.\, \lambda x{:}\varepsilon_{Type}\, \alpha.\, x$ *is* well-typed in $\lambda\Pi/HOL$:

$$\vdash_{\lambda\Pi/HOL} |I| : \Pi\alpha{:}u_{Type}.\, \varepsilon_{Type}\, \alpha \to \varepsilon_{Type}\, \alpha$$

$$\vdash_{\lambda\Pi/HOL} (\Pi\alpha{:}u_{Type}.\, \varepsilon_{Type}\, \alpha \to \varepsilon_{Type}\, \alpha) : Type$$

It seems that $\lambda\Pi/HOL$, just like λU^-, allows more functions than λHOL, even though the type of $|I|$ is not the translation of a λHOL type. Is that enough to make $\lambda\Pi/HOL$ inconsistent?

In this paper we show that $\lambda\Pi/S$ is conservative in all cases, even when λS is *not* normalizing. After identifying the main difficulties, we characterize a *completion* [11, 10] S^* containing S, and define an inverse translation from $\lambda\Pi/S$ to λS^*. We then prove that λS^* is a conservative extension of λS using the reducibility method [12, 7].

2 Pure type systems

Pure type systems [1] are a general class of typed λ-calculi parametrized by a specification.

Definition 2.1. A PTS *specification* is a triple $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ where

- S is a set of of symbols called *sorts*
- $\mathcal{A} \subseteq \mathcal{S} \times S$ is a set of *axioms* of the form $(s_1 : s_2)$
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ is a set of *rules* of the form (s_1, s_2, s_3)

Definition 2.2. We write (s_1, s_2) as a short-hand for the rule (s_1, s_2, s_2). The specification S is *functional* if the relations $\mathcal{A}$ and $\mathcal{R}$ are functional, that is $(s_1, s_2) \in \mathcal{A}$ and $(s_1, s_2') \in \mathcal{A}$ imply $s_2 = s_2'$, and $(s_1, s_2, s_3) \in \mathcal{R}$ and $(s_1, s_2, s_3') \in \mathcal{R}$ imply $s_3 = s_3'$.

Definition 2.3. Given a PTS specification $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ and an infinite set of variables $\mathcal{V}$, the abstract syntax of λS is defined by the following grammar:

$$\begin{array}{lrcl} \text{(terms)} & \mathcal{T} & ::= & \mathcal{S} \mid \mathcal{V} \mid \mathcal{T}\,\mathcal{T} \mid \lambda\mathcal{V}{:}\mathcal{T}.\, \mathcal{T} \mid \Pi\mathcal{V}{:}\mathcal{T}.\, \mathcal{T} \\ \text{(contexts)} & \mathcal{C} & ::= & \cdot \mid \mathcal{C}, \mathcal{V} : \mathcal{T} \end{array}$$

$$\textsc{Empty}\quad \frac{}{WF_{\lambda S}(\cdot)} \qquad \textsc{Declaration}\quad \frac{\Gamma \vdash_{\lambda S} A : s \qquad x \notin \Gamma}{WF_{\lambda S}(\Gamma, x : A)} \qquad \textsc{Sort}\quad \frac{WF_{\lambda S}(\Gamma) \qquad (s_1 : s_2) \in \mathcal{A}}{\Gamma \vdash_{\lambda S} s_1 : s_2}$$

$$\textsc{Variable}\quad \frac{WF_{\lambda S}(\Gamma) \qquad (x : A) \in \Gamma}{\Gamma \vdash_{\lambda S} x : A} \qquad \textsc{Application}\quad \frac{\Gamma \vdash_{\lambda S} M : \Pi x{:}A.\, B \qquad \Gamma \vdash_{\lambda S} N : A}{\Gamma \vdash_{\lambda S} M\, N : [N/x]B}$$

$$\textsc{Abstraction}\quad \frac{\Gamma, x : A \vdash_{\lambda S} M : B \qquad \Gamma \vdash_{\lambda S} \Pi x{:}A.\, B : s}{\Gamma \vdash_{\lambda S} \lambda x{:}A.\, M : \Pi x{:}A.\, B}$$

$$\textsc{Product}\quad \frac{\Gamma \vdash_{\lambda S} A : s_1 \qquad \Gamma, x : A \vdash_{\lambda S} B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash_{\lambda S} \Pi x{:}A.\, B : s_3}$$

$$\textsc{Conversion}\quad \frac{\Gamma \vdash_{\lambda S} M : A \qquad \Gamma \vdash_{\lambda S} B : s \qquad A \equiv_\beta B}{\Gamma \vdash_{\lambda S} M : B}$$

Figure 1: Typing rules of λS

We use lower case letters $x, y, \alpha, \beta, \ldots$ to denote variables, uppercase letters such as $M, N, A, B, \ldots$ to denote terms, and uppercase letters such as $\Gamma, \Delta, \Sigma, \ldots$ to denote contexts. We write $A \to B$ for $\Pi x{:}A.\, B$ when $x \notin B$.

The typing rules of λS are presented in Figure 1. We write $\Gamma \vdash M : A$ instead of $\Gamma \vdash_{\lambda S} M : A$ when the context is unambiguous. We say that M is a Γ-*term* when $WF(\Gamma)$ and $\Gamma \vdash M : A$ for some A. We say that A is a Γ-*type* when $WF(\Gamma)$ and either $\Gamma \vdash A : s$ or $A = s$ for some $s \in \mathcal{S}$. We write $\Gamma \vdash M : A : s$ as a shorthand for $\Gamma \vdash M : A$ and $\Gamma \vdash A : s$.

Example 2.4. The following well-known systems can be described as functional pure type systems using the same set of sorts $\mathcal{S} = Type, Kind$ and the same set of axioms $\mathcal{A} = (Type : Kind)$.

- Simply-typed λ-calculus ($\lambda \to$):
  $\mathcal{R} = (Type, Type)$
- System F ($\lambda 2$):
  $\mathcal{R} = (Type, Type), (Kind, Type)$
- $\lambda\Pi$-calculus (λP):
  $\mathcal{R} = (Type, Type), (Type, Kind)$
- Calculus of constructions (λC):
  $\mathcal{R} = (Type, Type), (Kind, Type),$
  $(Type, Kind), (Kind, Kind)$

Example 2.5. Let $I = \lambda\alpha{:}Type.\, \lambda x{:}\alpha.\, x$ be the polymorphic identity function. The term I is not well-typed in the simply typed λ-calculus but it is well-typed

in the Calculus of constructions λC:

$$\vdash_{\lambda C} I : \Pi\alpha{:}Type.\, \alpha \to \alpha$$

The following properties hold for all pure type systems. The proofs can be found in [1].

Proposition 2.6 (Confluence)**.** *If $M_1 \longrightarrow_\beta^* M_2$ and $M_1 \longrightarrow_\beta^* M_3$ then there is a term M_4 such that $M_2 \longrightarrow_\beta^* M_4$ and $M_3 \longrightarrow_\beta^* M_4$.*

Proposition 2.7 (Subject reduction)**.** *If $\Gamma \vdash_{\lambda S} M : A$ and $M \longrightarrow_\beta^* M'$ then $\Gamma \vdash_{\lambda S} M' : A$.*

Proposition 2.8 (Correctness of types)**.** *If $\Gamma \vdash_{\lambda S} M : A$ then either $\Gamma \vdash_{\lambda S} A : s$ or $A = s$ for some $s \in \mathcal{S}$.*

The reason why we don't always have $\Gamma \vdash_{\lambda S} A : s$ is because some sorts do not have an associated axiom, such as $Kind$ in Example 2.4.

Definition 2.9. A sort $s \in \mathcal{S}$ is called a *top-sort* when there is no sort $s' \in \mathcal{S}$ such that $(s : s') \in \mathcal{A}$.

The following proposition is very useful when proving properties about pure type systems with top-sorts.

Proposition 2.10. *If $\Gamma \vdash_{\lambda S} A : s$ and s is a top-sort then either $A = s'$ for some sort $s' \in \mathcal{S}$ or $A = \Pi x{:}B.\, C$ for some terms B, C.*

Finally, we state the following property for functional pure type systems.

Proposition 2.11 (Uniqueness of types)**.** *Let $\mathcal{S}$ be a functional specification. If $\Gamma \vdash_{\lambda S} M : A$ and $\Gamma \vdash_{\lambda S} M : B$ then $A \equiv_\beta B$.*

3 The $\lambda\Pi$-calculus modulo rewriting

The $\lambda\Pi$-*calculus*, also known as LF and as λP, is one of the simplest forms of λ-calculus with dependent types, and corresponds to first-order logic through the Curry-de Bruijn-Howard correspondence. As mentioned in Example 2.4, it can be defined as the functional pure type system λP with the following specification:

$$\begin{array}{rcl} \mathcal{S} & = & Type, Kind \\ \mathcal{A} & = & Type : Kind \\ \mathcal{R} & = & (Type, Type), (Type, Kind) \end{array}$$

The $\lambda\Pi$-*calculus modulo rewriting* extends the $\lambda\Pi$-calculus with rewrite rules. We recall that a rewrite rule is a triple $[\Delta]\ M \rightsquigarrow N$ where Δ is a context and M, N are terms. A set of rewrite rules R induces a reduction relation on terms, written $\longrightarrow_R$, defined as the smallest contextual closure such that if $[\Delta]\ M \rightsquigarrow N \in R$ then $\sigma(M) \longrightarrow_R \sigma(N)$ for any substitution σ of the variables in Δ. The relation $\equiv_R$ is the smallest congruence containing $\longrightarrow_R$, and $\equiv_{\beta R}$ is the smallest congruence containing both $\longrightarrow_\beta$ and $\longrightarrow_R$.

Definition 3.1. A rewrite rule $[\Delta]\ M \rightsquigarrow N$ is well-typed in a context Σ when there is a term A such that $\Sigma, \Delta \vdash_{\lambda\Pi} M : A$ and $\Sigma, \Delta \vdash_{\lambda\Pi} N : A$.

$$\textsc{Empty}\quad \frac{}{WF_{\lambda\Pi/}(\cdot)}$$

$$\textsc{Declaration}\quad \frac{WF_{\lambda\Pi/}(\Gamma) \qquad \Gamma \vdash_{\lambda\Pi/} A : s \qquad x \notin \Sigma, \Gamma}{WF_{\lambda\Pi/}(\Gamma, x : A)}$$

$$\textsc{Sort}\quad \frac{WF_{\lambda\Pi/}(\Gamma) \qquad (s_1 : s_2) \in \mathcal{A}}{\Gamma \vdash_{\lambda\Pi/} s_1 : s_2}$$

$$\textsc{Variable}\quad \frac{WF_{\lambda\Pi/}(\Gamma) \qquad (x : A) \in \Sigma, \Gamma}{\Gamma \vdash_{\lambda\Pi/} x : A}$$

$$\textsc{Abstraction}\quad \frac{\Gamma, x : A \vdash_{\lambda\Pi/} M : B \qquad \Gamma \vdash_{\lambda\Pi/} \Pi x{:}A.\, B : s}{\Gamma \vdash_{\lambda\Pi/} \lambda x{:}A.\, M : \Pi x{:}A.\, B}$$

$$\textsc{Application}\quad \frac{\Gamma \vdash_{\lambda\Pi/} M : \Pi x{:}A.\, B \qquad \Gamma \vdash_{\lambda\Pi/} N : A}{\Gamma \vdash_{\lambda\Pi/} M\, N : [N/x]B}$$

$$\textsc{Product}\quad \frac{\Gamma \vdash_{\lambda\Pi/} A : s_1 \qquad \Gamma, x : A \vdash_{\lambda\Pi/} B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash_{\lambda\Pi/} \Pi x{:}A.\, B : s_3}$$

$$\textsc{Conversion}\quad \frac{\Gamma \vdash_{\lambda\Pi/} M : A \qquad \Gamma \vdash_{\lambda\Pi/} B : s \qquad A \equiv_{\beta R} B}{\Gamma \vdash_{\lambda\Pi/} M : B}$$

Figure 2: Typing rules of $\lambda\Pi/(\Sigma, R)$

Definition 3.2. Let Σ be a well-formed $\lambda\Pi$ context and R a set of rewrite rules that are well-typed in Σ. The $\lambda\Pi$-calculus modulo (Σ, R), written $\lambda\Pi/(\Sigma, R)$, is defined with the same syntax as the $\lambda\Pi$-calculus, but with the typing rules of Figure 2. We write $\lambda\Pi/$ instead of $\lambda\Pi/(\Sigma, R)$ when the context is unambiguous.

Example 3.3. Let Σ be the context

$$(\alpha : Type, c : \alpha, f : \alpha \to Type)$$

and R be the following rewrite rule

$$[\cdot]\ f\, c \rightsquigarrow (\Pi y{:}\alpha.\, f\, y \to f\, y)$$

Then the term

$$\delta = \lambda x{:}f\, c.\, x\, c\, x$$

is well-typed in $\lambda\Pi/(\Sigma, R)$:

$$\vdash_{\lambda\Pi/(\Sigma,R)} \delta : f\, c \to f\, c$$

Note that the term δ would not be well-typed without the rewrite rule, even if we replace $f\, c$ by $(\Pi y{:}\alpha.\, f\, y \to f\, y)$.

4 Embedding FPTS's in the $\lambda\Pi$-calculus modulo

In this section, we present the embedding of functional pure type systems in the $\lambda\Pi$-calculus modulo rewriting as introduced by Cousineau and Dowek [5]. This is done in two steps. First, given a pure type system λS, we construct $\lambda\Pi/S$ by giving an appropriate signature and rewrite system. Second, we define a translation from the terms and types of λS to the terms and types of $\lambda\Pi/S$.

Definition 4.1 (The system $\lambda\Pi/S$)**.** Consider a functional pure type system specified by $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$. Define Σ_S to be the context containing the declarations:

$$\begin{array}{ll} u_s : Type & \forall s \in \mathcal{S} \\ \varepsilon_s : u_s \to Type & \forall s \in \mathcal{S} \\ \dot{s_1} : u_{s_2} & \forall s_1 : s_2 \in \mathcal{A} \\ \dot{\pi}_{s_1 s_1 s_3} : \Pi\alpha\!:\!u_{s_1}.\,(\varepsilon_{s_1}\,\alpha \to u_{s_2}) \to u_{s_3} & \forall (s_1, s_2, s_3) \in \mathcal{R} \end{array}$$

Let R_S be the rewrite system containing the rules

$$[\cdot]\ \varepsilon_{s_2}\,\dot{s_1} \rightsquigarrow u_{s_1}$$

for all $s_1 : s_2 \in \mathcal{A}$, and

$$[\Delta_{s_1 s_2 s_3}]\ \varepsilon_{s_3}\,(\dot{\pi}_{s_1 s_2 s_3}\,A\,B) \rightsquigarrow \Pi x\!:\!(\varepsilon_{s_1}\,A).\,\varepsilon_{s_2}\,(B\,x)$$

for all $(s_1, s_2, s_3) \in \mathcal{R}$, where $\Delta_{s_1 s_2 s_3} = (A : u_{s_1}, B : (\varepsilon_{s_1}\,\alpha \to u_{s_2}))$. The system $\lambda\Pi/S$ is defined as the $\lambda\Pi$-calculus modulo (Σ_S, R_S), that is, $\lambda\Pi/(\Sigma_S, R_S)$.

There are two translations, one from the terms of λS to the terms of $\lambda\Pi/S$, the other from the types of λS to the types of $\lambda\Pi/S$.

Definition 4.2. The translation $|M|_\Gamma$ of Γ-terms and the translation $\|A\|_\Gamma$ of Γ-types are mutually defined as follows.

$$\begin{array}{rcl} |s|_\Gamma & = & \dot{s} \\ |x|_\Gamma & = & x \\ |M\,N|_\Gamma & = & |M|_\Gamma\ |N|_\Gamma \\ |\lambda x\!:\!A.\,M|_\Gamma & = & \lambda x\!:\!\|A\|_\Gamma\,.\ |M|_{\Gamma,x:A} \\ |\Pi x\!:\!A.\,B|_\Gamma & = & \dot{\pi}_{s_1 s_2 s_3}\ |A|_\Gamma\ (\lambda x\!:\!\|A\|_\Gamma\,.\ |B|_{\Gamma,x:A}) \\ & & \text{where } \Gamma \vdash A : s_1 \\ & & \text{and } \Gamma, x : A \vdash B : s_2 \\ & & \text{and } (s_1, s_2, s_3) \in \mathcal{R} \end{array}$$

$$\begin{array}{rcl} \|s\|_\Gamma & = & u_s \\ \|\Pi x\!:\!A.\,B\|_\Gamma & = & \Pi x\!:\!\|A\|_\Gamma\,.\ \|B\|_{\Gamma,x:A} \\ \|A\|_\Gamma & = & \varepsilon_s\ |A|_\Gamma \ \text{ where } \Gamma \vdash A : s \end{array}$$

Note that this definition is redundant but it is well-defined up to $\equiv_{\beta R}$. In particular, because some Γ-types are also Γ-terms, there are two ways to translate them, but they are equivalent:

$$\begin{array}{rcl} \varepsilon_{s_2}\,\dot{s_1} & \equiv_{\beta R} & u_{s_1} \\ \varepsilon_{s_3}\ |\Pi x\!:\!A.\,B|_\Gamma & \equiv_{\beta R} & \Pi x\!:\!\|A\|_\Gamma\,.\ \|B\|_{\Gamma,x:A} \end{array}$$

This definition is naturally extended to well-formed contexts as follows.

$$\begin{array}{rcl} \|\cdot\| & = & \cdot \\ \|\Gamma, x : A\| & = & \|\Gamma\|\,, x : \|A\|_\Gamma \end{array}$$

Example 4.3. The polymorphic identity function of the Calculus of constructions λC is translated as

$$|I| = \lambda\alpha\!:\!u_{Type}.\, \lambda x\!:\!\varepsilon_{Type}\, \alpha.\, x$$

and its type $A = \Pi\alpha\!:\!Type.\, \alpha \to \alpha$ is translated as:

$$|A| = \dot{\pi}_{Kind,Type,Type}\, \dot{Type}\, (\lambda\alpha\!:\!u_{Type}.\, |A_\alpha|)$$

where $A_\alpha = \alpha \to \alpha$ and

$$|A_\alpha| = \dot{\pi}_{Type,Type,Type}\, \alpha\, (\lambda x\!:\!\varepsilon_{Type}\, \alpha.\, \varepsilon_{Type}\, \alpha)$$

The identity function applied to itself is translated as:

$$|I\, A\, I| = |I|\ |A|\ |I|$$

The embedding is complete, in the sense that all the typing relations of λS are preserved by the translation [5].

Theorem 4.4 (Completeness)**.** *For any context* Γ *and terms* M *and* A*, if* $\Gamma \vdash_{\lambda S} M : A$ *then* $\|\Gamma\| \vdash_{\lambda\Pi/S} |M|_\Gamma : \|A\|_\Gamma$.

5 Conservativity

In this section, we prove the converse of the completeness property. One could attempt to prove that if $\|\Gamma\| \vdash_{\lambda\Pi/S} |M|_\Gamma : \|A\|_\Gamma$ then $\Gamma \vdash_{\lambda S} M : A$. However, that would be meaningless because the translation $|M|_\Gamma$ is only defined for well-typed terms. A second attempt would be to define inverse translations $\varphi(M)$ and $\psi(A)$ and prove that if $\Gamma \vdash_{\lambda\Pi/S} M : A$ then $\psi(\Gamma) \vdash_{\lambda S} \varphi(M) : \psi(A)$, but that would not work either because not all terms and types of $\lambda\Pi/S$ correspond to valid terms and types of λS, as was shown in Example 1.1. Therefore the property that we want to prove is: if there is a term N such that $\|\Gamma\| \vdash_{\lambda\Pi/S} N : \|A\|_\Gamma$ then there is a term M such that $\Gamma \vdash_{\lambda S} M : A$.

The main difficulty is that some of these *external* terms can be involved in witnessing valid λS types, as illustrated by the following example.

Example 5.1. Consider the context $nat : Type$. Even though the polymorphic identity function I and its type are not well-typed in λHOL, they can be used in $\lambda\Pi/HOL$ to construct a witness for $nat \to nat$.

$$nat : u_{Type} \vdash_{\lambda\Pi/HOL} (|I|\ nat) : (\varepsilon_{Type}\, nat \to \varepsilon_{Type}\, nat)$$

We can normalize the term $|I|\ nat$ to $\lambda x : \varepsilon_{Type}\, nat.\, x$ which is a term that corresponds to a valid λHOL term: it is the translation of the term $\lambda x\!:\!nat.\, x$. However, as discussed previously, we cannot restrict ourselves to normal terms because we do not know if $\lambda\Pi/S$ is normalizing.

To prove conservativity, we will therefore need to address the following issues:

1. The system $\lambda\Pi/S$ can type more terms than λS.
2. The $\lambda\Pi/S$ terms that inhabit the translation of λS types can be reduced to the translation of λS terms.

First, we will eliminate β-redexes at the level of $Kind$ by reducing $\lambda\Pi/S$ to a subset $\lambda\Pi^-/S$. Then, we will extend λS to a *minimal completion* λS^* that can type more terms than λS, and show that $\lambda\Pi^-/S$ corresponds to λS^* using inverse translations $\varphi(M)$ and $\psi(A)$. Finally, we will show that λS^* terms inhabiting λS types can be reduced to λS terms. The procedure is summarized in the following diagram.

$$\begin{tikzcd}
\lambda\Pi/S \arrow[r, "\beta^*"'] & \lambda\Pi^-/S \arrow[d, "\varphi(M)"', "\psi(A)"] \\
\lambda_S \arrow[u, dotted, "|M|"', "\|A\|"] & \lambda_{S^*} \arrow[l, "\beta^*"]
\end{tikzcd}$$

5.1 Eliminating β-redexes at the level of $Kind$

In $\lambda\Pi/S$, we can have β-redexes at the level of $Kind$ such as $(\lambda x : A.\, u_s)\, M$. These redexes are artificial and are never generated by the forward translation of any PTS. We show here that they can always safely be eliminated.

Definition 5.2. A Γ-term M of type C is at the level of $Kind$ if $\Gamma \vdash C : Kind$. We define $\lambda\Pi^-/S$ terms as the subset of well-typed $\lambda\Pi/S$ terms that do not contain any $Kind$-level β-redexes.

Lemma 5.3. *For any $\lambda\Pi/S$ context Γ and Γ-term M, there is a $\lambda\Pi^-/S$ term M^- such that $M \longrightarrow_\beta^* M^-$.*

Proof. Reducing a $Kind$-level β-redex $(\lambda x\!:\!A.\, B)\, N$ does not create other $Kind$-level β-redexes because N is at the level of $Type$. Therefore, the number of $Kind$-level β-redexes strictly decreases, so any $Kind$-level β-reduction strategy will terminate. $\square$

Example 5.4. The term

$$I_1 = \lambda\alpha\!:\!u_{Type}.\, \lambda x\!:\!\varepsilon_{Type}\, ((\lambda\beta\!:\!u_{Type}.\, \beta)\, \alpha).\, x$$

is in $\lambda\Pi^-/HOL$. The term

$$I_2 = \lambda\alpha\!:\!u_{Type}.\, \lambda x\!:\!((\lambda\beta\!:\!u_{Type}.\, \varepsilon_{Type}\, \beta)\, \alpha).\, x$$

is not in $\lambda\Pi^-/HOL$ but

$$I_2 \longrightarrow_\beta \lambda\alpha\!:\!u_{Type}.\, \lambda x\!:\!\varepsilon_{Type}\alpha.\, x$$

which is in $\lambda\Pi^-/HOL$.

5.2 Minimal completion

In pure type systems, we can sometimes have types that do not have a type, such as top-sorts because there is no associated axiom. Similarly, we can sometimes prove $\Gamma, x : A \vdash_{\lambda S} M : B$ but cannot abstract over x because there is no associated product rule. Completions of pure type systems were originally introduced by Severi [11, 10] to address these issues by injecting λS into a larger pure type system.

Definition 5.5 (Completion)**.** A specification $S' = (\mathcal{S}', \mathcal{A}', \mathcal{R}')$ is a *completion of* S if

1. $\mathcal{S} \subseteq \mathcal{S}'$,$\mathcal{A} \subseteq \mathcal{A}'$, $\mathcal{R} \subseteq \mathcal{R}'$, and
2. for all sorts $s_1 \in \mathcal{S}$, there is a sort $s_2 \in \mathcal{S}'$ such that $(s_1 : s_2) \in \mathcal{A}'$, and
3. for all sorts $s_1, s_2 \in \mathcal{S}'$, there is a sort $s_3 \in \mathcal{S}'$ such that $(s_1, s_2, s_3) \in \mathcal{R}'$.

Notice that all the top-sorts of λS are typable in $\lambda S'$ and that $\lambda S'$ is full, meaning that all products are typable. Not all completions are conservative though, so we define the following completion.

Definition 5.6 (Minimal completion)**.** We define the *minimal completion of* S, written S^*, to be the following specification:

$$\begin{array}{rcl} \mathcal{S}^* & = & \mathcal{S} \cup \{\tau\} \\ \mathcal{A}^* & = & \mathcal{A} \cup \{(s_1 : \tau) \mid s_1 \in \mathcal{S}, (s_1 : s_2) \notin \mathcal{A}\} \\ \mathcal{R}^* & = & \mathcal{R} \cup \{(s_1, s_2, \tau) \mid s_1, s_2 \in \mathcal{S}^*, (s_1, s_2, s_3) \notin \mathcal{R}\} \end{array}$$

where $\tau \notin \mathcal{S}$.

Any well-typed term of λS is also well-typed in λS^*, but just like $\lambda\Pi^-/S$, this system allows more functions than λS.

Example 5.7. The polymorphic identity function is well-typed in λHOL^*.

$$\vdash_{\lambda HOL^*} I : \Pi\alpha{:}Type.\, \alpha \to \alpha$$

$$\vdash_{\lambda HOL^*} \Pi\alpha{:}Type.\, \alpha \to \alpha : \tau$$

Next, we define inverse translations that translate the terms and types of $\lambda\Pi^-/S$ to the terms and types of λS^*.

Definition 5.8 (Inverse translations)**.** The inverse translation of terms $\varphi(M)$ and the inverse translation of types $\psi(A)$ are mutually defined as follows.

$$\begin{array}{rcl} \varphi(\dot{s}) & = & s \\ \varphi(\dot{\pi}_{s_1 s_2 s_3}) & = & \lambda\alpha{:}s_1.\, \lambda\beta{:}(\alpha \to s_2).\, \Pi x{:}\alpha.\, \beta\, x \\ \varphi(x) & = & x \\ \varphi(M\, N) & = & \varphi(M)\, \varphi(N) \\ \varphi(\lambda x{:}A.\, M) & = & \lambda x{:}\psi(A).\, \varphi(M) \end{array}$$

$$\begin{array}{rcl} \psi(u_s) & = & s \\ \psi(\varepsilon_s\, M) & = & \varphi(M) \\ \psi(\Pi x{:}A.\, B) & = & \Pi x{:}\psi(A).\, \psi(B) \end{array}$$

Note that this is only a partial definition, but it is total for $\lambda\Pi^-/S$ terms. In particular, it is an inverse of the forward translation in the following sense.

Lemma 5.9.

1. $\varphi(|M|_\Gamma) \equiv_\beta M$.
2. $\psi(\|A\|_\Gamma) \equiv_\beta A$.

Proof. By induction on M or A. We show the product case $M = \Pi x : A.\, B$. Then

$$\varphi(|M|) = \varphi(\dot{\pi}_{s_1 s_2 s_3})\, \varphi(|A|)\, (\lambda x : \psi(\|A\|).\, \varphi(|B|))$$

By induction hypothesis, $\varphi(|A|) \equiv_\beta A$ and $\varphi(|B|) \equiv_\beta B$. Since $\varphi(\dot{\pi}_{s_1 s_2 s_3}) = \lambda \alpha : s_1.\, \lambda \beta : (\alpha \to s_2).\, \Pi x : \alpha.\, \beta\, x$, we get

$$\begin{array}{rcl} \varphi(|M|) & \equiv_\beta & \Pi x : A.\, (\lambda x : \psi(\|A\|).\, B)\, x \\ & \equiv_\beta & \Pi x : A.\, B \end{array}$$

□

Next we show that the inverse translations preserve typing.

Lemma 5.10.

1. $\varphi([N/x]M) = [\varphi(N)/x]\varphi(M)$
2. $\psi([N/x]A) = [\varphi(N)/x]\psi(A)$

Proof. By induction on M or A. We show the product case $A = \Pi y : B.\, C$. Without loss of generality, $y \neq x$ and $y \notin N$ and $y \notin \varphi(N)$. Then $[N/x]\Pi y : B.\, C = \Pi y : [N/x]B.\, [N/x]C$. By induction hypothesis, $\psi([N/x]B) = [\varphi(N)/x]\psi(B)$ and $\psi([N/x]C) = [\varphi(N)/x]\psi(C)$. Therefore

$$\begin{array}{rcl} \psi([N/x]A) & = & \psi(\Pi y : [N/x]B.\, [N/x]C) \\ & = & \Pi y : \psi([N/x]B).\, \psi([N/x]C) \\ & = & \Pi y : [\varphi(N)/x]\psi(B).\, [\varphi(N)/x]\psi(C) \\ & = & [\varphi(N)/x]\Pi x : \psi(B).\, \psi(C) \\ & = & [\varphi(N)/x]\psi(\Pi x : B.\, C) \end{array}$$

□

Lemma 5.11.

1. If $M \longrightarrow_{\beta R} N$ then $\varphi(M) \longrightarrow_\beta^* \varphi(N)$
2. If $A \longrightarrow_{\beta R} B$ then $\psi(A) \longrightarrow_\beta^* \psi(B)$

Proof. By induction on M or A. We show the case for β-redexes with $M = (\lambda x : A_1.\, M_1)\, N_1$ and $N = [N_1/x]M_1$. Then

$$\varphi(M) = (\lambda x : \psi(A_1).\, \varphi(M_1))\, \varphi(N_1)$$

Therefore $\varphi(M) \longrightarrow_\beta [\varphi(N_1)/x]\varphi(M_1)$ which is equal to $\varphi([N_1/x]M_1)$ by Lemma 5.10. □

Lemma 5.12.

1. If $M \equiv_{\beta R} N$ then $\varphi(M) \equiv_\beta \varphi(N)$
2. If $A \equiv_{\beta R} B$ then $\psi(A) \equiv_\beta \psi(B)$

Proof. Follows from Lemma 5.11. □

Definition 5.13 (Abstraction context)**.** We say that Γ is an *abstraction context* if $\Gamma \vdash_{\lambda\Pi/S} A : Type$ for all $x : A \in \Gamma$. If $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ is an abstraction context, we define $\psi(\Gamma)$ as $(x_1 : \psi(A_1), \ldots, x_n : \psi(A_n))$.

Lemma 5.14. *For any $\lambda\Pi^-/S$ abstraction context Γ and terms M, A:*

1. If $WF_{\lambda\Pi/S}(\Gamma)$ then $WF_{\lambda S^*}(\psi(\Gamma))$.
2. If $\Gamma \vdash_{\lambda\Pi/S} M : A : Type$ then $\psi(\Gamma) \vdash_{\lambda S^*} \varphi(M) : \psi(A)$.
3. If $\Gamma \vdash_{\lambda\Pi/S} A : Type$ then $\psi(\Gamma) \vdash_{\lambda S^*} \psi(A) : s$ for some sort $s \in \mathcal{S}^*$.

Proof. By induction on the derivation. The details of the proof can be found in the Appendix. □

5.3 Reduction to λS

In order to show that λS^* is a conservative extension of λS, we need to prove that β-reduction at the level of τ terminates. A straightforward proof by induction would fail because contracting a τ-level β-redex can create other such redexes. To solve this, we use Girard and Tait's *reducibility method* [7, 12]. The idea is to strengthen the induction hypothesis of the proof by defining a predicate by induction on the type of the term.

Definition 5.15. If $WF_{\lambda S}(\Gamma)$ and $\Gamma \vdash_{\lambda S^*} A : s$ then the predicate $\Gamma \models_S M : A$ is defined as:

- if $s \neq \tau$ or $A = s'$ for some $s' \in \mathcal{S}$ then $\Gamma \models_S M : A$ iff $M \longrightarrow_\beta^* M'$, and $A \longrightarrow_\beta^* A'$ for some M', A' such that $\Gamma \vdash_{\lambda S} M' : A'$,
- if $s = \tau$ and $A = \Pi x\!:\!B.\,C$ for some B, C then $\Gamma \models_S M : A$ iff for all N such that $\Gamma \models_S N : B$, $\Gamma \models_S M\,N : [N/x]C$.

To show that this inductive definition is well-founded, we define the following measure of the height of A.

Definition 5.16. If $WF_{\lambda S}(\Gamma)$ and $\Gamma \vdash_{\lambda S^*} A : s$ then the height of A at the level of τ is defined as:

$$\begin{array}{rcll} \mathcal{H}_\tau(A) & = & 0 & \text{if } s \neq \tau \\ \mathcal{H}_\tau(s') & = & 0 & \text{if } s = \tau \\ \mathcal{H}_\tau(\Pi x\!:\!B.\,C) & = & 1 + max(\mathcal{H}_\tau(B) + \mathcal{H}_\tau(C)) & \text{if } s = \tau \end{array}$$

Proposition 5.17. *If $\Gamma, x : B \vdash_{\lambda S^*} C : \tau$ then $\mathcal{H}_\tau([N/x]C) = \mathcal{H}_\tau(C)$.*

Proof. By induction on C. □

Corollary 5.18. *Definition 5.15 is well-founded.*

Proof. The measure $\mathcal{H}_\tau(A)$ strictly decreases in the definition. □

Lemma 5.19. *If $\Gamma \models_S M : A$ and $M \equiv_\beta M'$ then $\Gamma \models_S M' : A$.*

Proof. By induction on the height of A. □

- If $s \neq \tau$ or $A = s'$ for some $s' \in \mathcal{S}$ then $M \longrightarrow_\beta^* M''$ and $A \longrightarrow_\beta^* A'$ for some M'', A' such that $\Gamma \vdash_{\lambda S} M'' : A'$. By confluence and subject reduction, $M' \longrightarrow_\beta^* M'''$ such that $\Gamma \vdash_{\lambda S} M''' : A'$.

- If $s = \tau$ and $A = \Pi x{:}B.\,C$ for some B, C then for all N such that $\Gamma \models_S N : B$, $\Gamma \models_S M\,N : [N/x]C$. By induction hypothesis, $\Gamma \models_S M'\,N : [N/x]C$. Therefore $\Gamma \models_S M' : \Pi x{:}B.\,C$.

Lemma 5.20. *If $\Gamma \models_S M : A$ and $A \equiv_\beta A'$ then $\Gamma \models_S M : A'$.*

Proof. By induction on the height of A. □

- If $s \neq \tau$ or $A = s'$ for some $s' \in \mathcal{S}$ then $M \longrightarrow_\beta^* M'$ and $A \longrightarrow_\beta^* A''$ for some M', A'' such that $\Gamma \vdash_{\lambda S} M' : A''$. By confluence and conversion, $A' \longrightarrow_\beta^* A'''$ such that $\Gamma \vdash_{\lambda S} M' : A'''$.

- If $s = \tau$ and $A = \Pi x : B.\,C$ for some B, C then for all N such that $\Gamma \models_S N : B$, $\Gamma \models_S M\,N : [N/x]C$. By induction hypothesis, $\Gamma \models_S M\,N : [N/x]C'$. Therefore $\Gamma \models_S M : \Pi x{:}B.\,C'$.

Definition 5.21. If $WF_{\lambda S}(\Gamma')$ and σ is a substitution, then $\Gamma' \models_S \sigma : \Gamma$ if $\Gamma' \models_S \sigma(x) : \sigma(A)$ for all $(x : A) \in \Gamma$.

Lemma 5.22. *If $\Gamma \vdash_{\lambda S^*} M : A : s$ then for any context Γ' and substitution σ such that $WF_{\lambda S}(\Gamma')$ and $\Gamma' \models_S \sigma : \Gamma$, $\Gamma' \models_S \sigma(M) : \sigma(A)$.*

Proof. By induction on the derivation of $\Gamma \vdash_{\lambda S^*} M : A$. The details of the proof can be found in the Appendix. □

Corollary 5.23. *Suppose $WF_{\lambda S}(\Gamma)$ and either $\Gamma \vdash_{\lambda S} A : s$ or $A = s$ for some $s \in \mathcal{S}$. If $\Gamma \vdash_{\lambda S^*} M : A$ then $M \longrightarrow_\beta^* M'$ such that $\Gamma \vdash_{\lambda S} M' : A$.*

Proof. Taking σ as the identity substitution, there are terms M' and A' such that $M \longrightarrow_\beta^* M'$ and $A \longrightarrow_\beta^* A'$ and $\Gamma \vdash_{\lambda S} M' : A'$. If $A = s \in S$ then $A' = s$ and we are done. Otherwise by conversion we get $\Gamma \vdash_{\lambda S} M' : A$. □

We now have all the tools to prove the main theorem.

Theorem 5.24 (Conservativity)**.** *For any λS Γ-type A, if there is a term N such that $\|\Gamma\| \vdash_{\lambda\Pi/S} N : \|A\|_\Gamma$ then there is a term M such that $\Gamma \vdash_{\lambda S} M : A$.*

Proof. By Lemma 5.3, there is a $\lambda\Pi^-/S$ term N^- such that $N \longrightarrow_\beta^* N^-$. By subject reduction, $\|\Gamma\| \vdash_{\lambda\Pi/S} N^- : \|A\|_\Gamma$. By Lemmas 5.14 and 5.9, $\Gamma \vdash_{\lambda S^*} \varphi(N^-) : A$. By Corollary 5.23, there is a term M such that $\varphi(N^-) \longrightarrow_\beta^* M$ and $\Gamma \vdash_{\lambda S} M : A$. □

6 Conclusion

We have shown that $\lambda\Pi/S$ is conservative even when λS is not normalizing. Even though $\lambda\Pi/S$ can construct more functions than λS, it preserves provability. This effect is similar to various conservative extensions of pure type systems such as pure type systems with definitions [11], pure type systems without the Π-condition [10], or predicative polymorphism [9]. Inconsistency in pure type

systems usually does not come from the ability to type more functions, but from the impredicativity caused by assigning a sort to the type of these functions. It is clear that no such impredicativity arises in $\lambda\Pi/S$ because there is no constant $\dot{\pi}_{s_1 s_2 s_3}$ associated to the type of illegal abstractions.

The results of this paper also indirectly imply that $\lambda\Pi/S$ is weakly normalizing when λS is weakly normalizing. The strong normalization of $\lambda\Pi/S$ is still an open problem. The Barendregt-Geuvers-Klop conjecture states that any weakly normalizing PTS is also strongly normalizing [6]. There is evidence that this conjecture is true [2], in which case its proof could be adapted to prove the strong normalization of $\lambda\Pi/S$. Weak normalisation could also be used as an intermediary *quasi-normalisation* result to construct a proof of strong normalisation, as done by Luo [8] for the *extended calculus of constructions* (ECC).

Our proof can be simplified in some cases. A PTS is *complete* when it is a completion of itself. In that case, the construction of S^* is unnecessary. The translations $\varphi(M)$ and $\psi(A)$ translate directly into λS, and Section 5.3 can be omitted. This is the case for example for the calculus of constructions with *infinite type hierarchy* (λC^∞) [11], which is the basis for proof assistants such as Coq and Matita.

Acknowledgements

The author thanks Gilles Dowek and Guillaume Burel for their support and feedback.

References

[1] H. P. Barendregt. Lambda calculi with types. In *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1992.

[2] Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. Weak normalization implies strong normalization in a class of non-dependent pure type systems. *Theoretical Computer Science*, 269(1-2):317–361, 2001.

[3] M. Boespflug and G. Burel. CoqInE: translating the calculus of inductive constructions into the $\lambda\Pi$-calculus modulo. In *Proof Exchange for Theorem Proving - Second International Workshop, PxTP 2012*, pages 44–50, 2012.

[4] M. Boespflug, Q. Carbonneaux, and O. Hermant. The $\lambda\Pi$-calculus modulo as a universal proof language. In *Proof Exchange for Theorem Proving - Second International Workshop, PxTP 2012*, pages 28–43, 2012.

[5] Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-Pi-calculus modulo. In Simona Ronchi Della Rocca, editor, *Typed Lambda Calculi and Applications*, number 4583 in Lecture Notes in Computer Science, pages 102–117. Springer Berlin Heidelberg, 2007.

[6] Herman Geuvers. *Logics and type systems.* PhD thesis, University of Nijmegen, 1993.

[7] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and types.* Cambridge university press Cambridge, 1989.

[8] Z. Luo. ECC, an extended calculus of constructions. In *Fourth Annual Symposium on Logic in Computer Science*, pages 386–395, 1989.

[9] Cody Roux. The structural theory of pure type systems, 2013. unpublished, available at `http://www.andrew.cmu.edu/user/croux/papers/struct-pts-full.pdf`.

[10] Paula Severi. Pure type systems without the Pi-condition. *Proceedings of 7th Nordic Workshop on Programming Theory*, 1995.

[11] Paula Severi and Erik Poll. Pure type systems with definitions. In Anil Nerode and Yu V. Matiyasevich, editors, *Logical Foundations of Computer Science*, number 813 in Lecture Notes in Computer Science, pages 316–328. Springer Berlin Heidelberg, 1994.

[12] W. W. Tait. Intensional interpretations of functionals of finite type I. *The Journal of Symbolic Logic*, 32(2):198–212, June 1967.

A Proof details

Lemma 5.14

Proof. By induction on the derivation.

1. There are 2 cases.

   - $\dfrac{}{WF(\cdot)}$ (Empty)
     Then $WF(\cdot)$ trivially.

   - $\dfrac{WF(\Gamma) \qquad \Gamma \vdash A : Type \qquad x \notin \Sigma, \Gamma}{WF(\Gamma, x : A)}$ (Declaration)
     Then $x \notin \psi(\Gamma)$. By induction hypothesis, $WF(\psi(\Gamma))$ and $\psi(\Gamma) \vdash \psi(A) : s$ for some sort $s \in \mathcal{S}^*$. Therefore $WF(\psi(\Gamma), x : \psi(A))$.

2. There are 4 cases.

   - $\dfrac{WF(\Gamma) \qquad (x : A) \in \Sigma, \Gamma}{\Gamma \vdash x : A}$ (Variable)
     By induction hypothesis, $WF(\psi(\Gamma))$.
     (a) If $x = \dot{s_1}$ then $A = u_{s_2}$ and $(s_1 : s_2) \in \mathcal{A}$. Therefore $\psi(\Gamma) \vdash s_1 : s_2$.
     (b) If $x = \dot{\pi}_{s_1 s_2 s_3}$ then $A = \Pi\alpha : u_{s_1}.\, (\varepsilon_{s_1}\, \alpha \to u_{s_2}) \to u_{s_3}$ and $(s_1, s_2, s_3) \in \mathcal{R}$. Therefore $\psi(\Gamma), \alpha : s_1, \beta : \alpha \to s_2 \vdash \Pi x : \alpha.\, \beta\, x : s_3$, which implies $\psi(\Gamma) \vdash (\lambda\alpha : s_1.\, \lambda\beta : (\alpha \to s_2).\, \Pi x : \alpha.\, \beta\, x) : \Pi\alpha : s_1.\, (\alpha \to s_2) \to s_3$.
     (c) Otherwise $(x : A) \in \Gamma$, so $(x : \psi(A)) \in \psi(\Gamma)$. By induction hypothesis, $WF(\psi(\Gamma))$. Therefore $\psi(\Gamma) \vdash x : \psi(A)$.

   - $\dfrac{\Gamma \vdash M : \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : [N/x]B}$ (Application)
     By induction hypothesis, $\psi(\Gamma) \vdash \varphi(M) : \Pi x : \psi(A).\, \psi(B)$ and $\psi(\Gamma) \vdash \varphi(N) : \psi(A)$. Therefore $\psi(\Gamma) \vdash \varphi(M)\, \varphi(N) : [\varphi(N)/x]\psi(B)$. By Lemma 5.10, $\psi(\Gamma) \vdash \varphi(M)\, \varphi(N) : \psi([N/x]B)$

   - $\dfrac{\Gamma \vdash \Pi x : A.\, B : Type \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B}$ (Abstraction)
     By induction hypothesis, $\psi(\Gamma) \vdash \Pi x : \psi(A).\, \psi(B) : s$ and $\psi(\Gamma), x : \psi(A) \vdash \varphi(M) : \psi(B)$ for some sort $s \in \mathcal{S}^*$. Therefore $\psi(\Gamma) \vdash (\lambda x : \psi(A).\, \varphi(M)) : \Pi x : \psi(A).\, \psi(B)$.

- Conversion
$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : Type \qquad A \equiv_{\beta R} B}{\Gamma \vdash M : B}$$
By induction hypothesis, $\psi(\Gamma) \vdash \varphi(M) : \psi(A)$ and $\psi(\Gamma) \vdash \psi(B) : s$ for some sort $s \in \mathcal{S}^*$. By Lemma 5.10, $\psi(A) \equiv_\beta \psi(B)$. Therefore $\psi(\Gamma) \vdash \varphi(M) : \psi(B)$.

3. There are 4 cases.

- Variable
$$\frac{WF(\Gamma) \qquad (x : Type) \in \Sigma, \Gamma}{\Gamma \vdash x : Type}$$
Since Γ is an abstraction context we must have $x \in \Sigma$, so $x = u_{s_1}$ for some $s_1 \in \mathcal{S}$. By induction hypothesis, $WF(\psi(\Gamma))$. By definition, there is a sort $s_2 \in \mathcal{S}^*$ such that $(s_1 : s_2) \in \mathcal{A}^*$. Therefore $\psi(\Gamma) \vdash s_1 : s_2$.

- Application
$$\frac{\Gamma \vdash M : \Pi x{:}A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : [N/x]B}$$
Since Γ is an abstraction context and $M\, N$ is not a β-redex, we must have $M = \varepsilon_{s_1}$ and $\Pi x : A.\, B = u_{s_1} \to Type$ and $N : u_{s_1}$ for some $s_1 \in \mathcal{S}$. By induction hypothesis, $\psi(\Gamma) \vdash \varphi(N) : s_1$.

- Product
$$\frac{\Gamma \vdash A : Type \qquad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x{:}A.\, B : Type}$$
By induction hypothesis, $\psi(\Gamma) \vdash \psi(A) : s_1$ and $\psi(\Gamma), x : \psi(A) \vdash \psi(B) : s_2$ for some sorts $s_1, s_2 \in \mathcal{S}^*$. By definition, there is a sort $s_3 \in \mathcal{S}^*$ such that $(s_1, s_2, s_3) \in \mathcal{R}^*$. Therefore $\psi(\Gamma) \vdash (\Pi x{:}\psi(A).\, \psi(B)) : s_3$.

- Conversion
$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B : Kind \qquad B \equiv_{\beta R} Type}{\Gamma \vdash A : Type}$$
We must have $B = Type$. By induction hypothesis, $\psi(\Gamma) \vdash \psi(A) : s$ for some sort $s \in \mathcal{S}^*$.

□

Lemma 5.22

Proof. By induction on the derivation of $\Gamma \vdash_{\lambda S^*} M : A$.

- Sort
$$\frac{WF(\Gamma) \qquad (s_1 : s_2) \in \mathcal{A}^*}{\Gamma \vdash s_1 : s_2}$$
Since $s_2 : s$, we must have $s_2 \neq \tau$, so $(s_1 : s_2) \in \mathcal{A}$. Therefore $\Gamma' \vdash_{\lambda S} s_1 : s_2$, which implies $\Gamma' \models_S s_1 : s_2$.

- Variable
$$\frac{WF(\Gamma) \qquad (x : A) \in \Sigma, \Gamma}{\Gamma \vdash x : A}$$
Then $\Gamma' \models_S \sigma(M) : \sigma(A)$ by definition of $\Gamma' \models_S \sigma : \Gamma$.

- Application
$$\frac{\Gamma \vdash M : \Pi x{:}A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : [N/x]B}$$
Without loss of generality, $x \notin \Gamma'$, so $\sigma([N/x]B) = [\sigma(N)/x]\sigma(B)$. By induction hypothesis, $\Gamma' \models_S \sigma(M) : \Pi x : \sigma(A).\, \sigma(B)$ and $\Gamma' \models_S \sigma(N) : \sigma(A)$.

  1. If $\Gamma \vdash_{\lambda S^*} \Pi x{:}A.\, B : s_3 \neq \tau$ then $\Gamma \vdash_{\lambda S^*} A : s_1$ and $\Gamma, x : A \vdash_{\lambda S^*} B : s_2$ for some s_1, s_2 such that $(s_1, s_2, s_3) \in \mathcal{S}$, which also means that $\Gamma \vdash_{\lambda S^*} [N/x]B : s_2 \neq \tau$. By induction hypothesis, $\sigma(M) \longrightarrow_\beta^* M'$, $\sigma(A) \longrightarrow A'$ and $\sigma(B) \longrightarrow B'$ such that$\Gamma' \vdash_{\lambda S^*} M' : \Pi x : A'.\, B'$ and $\sigma(N) \longrightarrow_\beta^* N'$, $\sigma(A) \longrightarrow_\beta^* A''$ such that $\Gamma' \vdash_{\lambda S^*} N' : A''$. By confluence and subject reduction, we can assume $A' = A''$. Therefore $\Gamma' \vdash_{\lambda S^*} M'\, N' : [N'/x]B'$. Since $[N/x]B \longrightarrow_\beta^* [N'/x]B'$, this implies $\Gamma' \models_S M\, N : [N/x]B$.
  2. Otherwise $\Gamma \vdash \Pi x : A.\, B : \tau$. By definition, $\Gamma' \models_S \sigma(M)\, \sigma(N) : [\sigma(N)/x]\sigma(B)$.

- Abstraction
$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x{:}A.\, B : s}{\Gamma \vdash \lambda x{:}A.\, M : \Pi x{:}A.\, B}$$
Without loss of generality, $x \notin \Gamma'$.

  1. If $s \neq \tau$ then by induction hypothesis, $\sigma(A) \longrightarrow_\beta^* A'$ and $\sigma(B) \longrightarrow_\beta^* B'$ such that $\Gamma' \vdash_{\lambda S} \Pi x{:}A'.\, B' : s$. By inversion, $\Gamma' \vdash_{\lambda S} A' : s_1$ for some $s_1 \neq \tau$, so $\Gamma \models_S A : s_1$, which implies $\Gamma', x : A' \models_S \sigma : (\Gamma, x : A)$. By induction hypothesis, $\sigma(M) \longrightarrow_\beta^* M'$ and $\sigma(B) \longrightarrow_\beta^* B''$ such that $\Gamma', x : A' \vdash_{\lambda S} M' : B''$. By confluence and subject reduction, we can assume $B' = B''$. Therefore $\Gamma' \vdash_{\lambda S} (\lambda x{:}A'.\, M') : \Pi x{:}A'.\, B'$, which implies $\Gamma' \models_S (\lambda x{:}A.\, M) : \Pi x{:}A.\, B$.
  2. If $s = \tau$ then for all N such that $\Gamma' \models_S N : \sigma(A)$, we have $\Gamma' \models_S (\sigma, N/x) : (\Gamma, x : A)$. By induction hypothesis, $\Gamma' \models_S (\sigma, N/x)(M) : (\sigma, N/x)(B)$. Since $x \notin \Gamma'$, we have $(\sigma, N/x)(M) = [N/x]\sigma(M)$ and $(\sigma, N/x)(B) = [N/x]\sigma(B)$. Therefore $\Gamma' \models_S [N/x]\sigma(M) : [N/x]\sigma(B)$. By Lemma 5.19, $\Gamma' \models_S ((\lambda x : \sigma(B).\, \sigma(M))\, N) : [N/x]\sigma(B)$. Therefore $\Gamma' \models_S (\lambda x{:}\sigma(B).\, \sigma(M)) : \Pi x{:}A.\, B$.

- Product
$$\frac{\Gamma \vdash_{\lambda S} A : s_1 \qquad \Gamma, x : A \vdash_{\lambda S} B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}^*}{\Gamma \vdash_{\lambda S} \Pi x{:}A.\, B : s_3}$$
Without loss of generality, $x \notin \Gamma'$. Since $s_3 : s$, we must have $s_3 \neq \tau$, so $(s_1, s_2, s_3) \in \mathcal{R}$, which also means $s_1 \neq \tau$ and $s_2 \neq \tau$. By induction hypothesis, $\sigma(A) \longrightarrow_\beta^* A'$ such that $\Gamma' \vdash_{\lambda S} A' : s_1$. This means that

$WF_{\lambda S}(\Gamma', x : A')$ and $\Gamma', x : A' \models_S (\sigma, x/x) : (\Gamma, x : A)$. By induction hypothesis, $\sigma(B) \longrightarrow_\beta^* B'$ such that $\Gamma' \vdash_{\lambda S} B' : s_2$. Therefore $\Gamma' \vdash_{\lambda S} (\Pi x{:}A'.\, B') : s_3$, which implies $\Gamma' \models_S (\Pi x{:}A'.\, B') : s_3$.

- Conversion
$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s \qquad A \equiv_\beta B}{\Gamma \vdash M : B}$$
By induction hypothesis, $\Gamma' \models_S \sigma(M) : \sigma(A)$. Since $A \equiv_\beta B$, we have $\sigma(A) \equiv_\beta \sigma(B)$. By Lemma 5.20, $\Gamma' \models_S \sigma(M) : \sigma(A)$.

$\square$