Conservativity of Embeddings in the $\lambda\Pi$ Calculus Modulo Rewriting

Ali Assaf[1,2]

1 **Inria, Paris, France**
2 **École polytechnique, Palaiseau, France**

Abstract

The $\lambda\Pi$ calculus can be extended with rewrite rules to embed any functional pure type system. In this paper, we show that the embedding is conservative by proving a relative form of normalization, thus justifying the use of the $\lambda\Pi$ calculus modulo rewriting as a logical framework for logics based on pure type systems. This result was previously only proved under the condition that the target system is normalizing. Our approach does not depend on this condition and therefore also works when the source system is not normalizing.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases $\lambda\Pi$ calculus modulo rewriting, pure type systems, logical framework, normalization, conservativity

Digital Object Identifier 10.4230/LIPIcs.TLCA.2015.31

1 Introduction

The $\lambda\Pi$ *calculus modulo rewriting* is a logical framework that extends the $\lambda\Pi$ calculus [10] with rewrite rules. Through the Curry-de Bruijn-Howard correspondence, it can express properties and proofs of various logics. Cousineau and Dowek [6] introduced a general embedding of *functional pure type systems* (FPTS), a large class of typed λ-calculi, in the $\lambda\Pi$ calculus modulo rewriting: for any FPTS λS, they constructed the system $\lambda\Pi/S$ using appropriate rewrite rules, and defined two translation functions $|M|$ and $\|A\|$ that translate respectively the terms and the types of λS to $\lambda\Pi/S$. This embedding is complete, in the sense preserves typing: if $\Gamma \vdash_{\lambda S} M : A$ then $\|\Gamma\| \vdash_{\lambda\Pi/S} |M| : \|A\|$. From the logical point of view, it preserves provability. The converse property, called *conservativity*, was only shown partially: 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 $\Gamma \vdash_{\lambda S} M : A$.

1.1 Normalization and conservativity

Not much is known about normalization in $\lambda\Pi/S$. Cousineau and Dowek [6] showed that the embedding preserves reduction: if $M \longrightarrow M'$ then $|M| \longrightarrow^+ |M'|$. As a consequence, if $\lambda\Pi/S$ is strongly normalizing (i.e. every well-typed term normalizes) then so is λS, but the converse might not be true *a priori*. This was not enough to show the conservativity of the embedding, so the proof relied on the unproven assumption that $\lambda\Pi/S$ is normalizing. 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 [4, 5]. Indeed, if the embedding turns out to be inconsistent then checking proofs in the logical framework has very little benefit.

13th International Conference on Typed Lambda Calculi and Applications (TLCA'15).
Editor: Thorsten Altenkirch; pp. 31–44

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Consider the PTS λHOL that corresponds to higher order logic [1]:

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

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

▶ **Example 1.** The polymorphic identity function $I = \lambda\alpha : \mathsf{Type}.\, \lambda x : \alpha.\, x$ is *not* well-typed in λHOL, but it is well-typed in λU^- and so is its type:

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

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

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

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

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

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. Does that make $\lambda\Pi/HOL$ inconsistent?

1.2 Absolute normalization vs relative normalization

One way to answer the question is to prove strong normalization of $\lambda\Pi/S$ by constructing a model, for example in the algebra of *reducibility candidates* [9]. Dowek [7] recently constructed such a model for the embedding of higher-order logic (λHOL) and of the calculus of constructions (λC). However, this technique is still very limited. Indeed, proving such a result is, by definition, at least as hard as proving the consistency of the original system. It requires specific knowledge of λS and the construction of such a model can be very involved, such as for the calculus of constructions with an infinite universe hierarchy (λC^∞).

In this paper, we take a different approach and show that $\lambda\Pi/S$ is conservative in all cases, even when λS is *not* normalizing. Instead of showing that $\lambda\Pi/S$ is strongly normalizing, we show that it is weakly normalizing *relative to* λS, meaning that proofs in the target language can be reduced to proofs in the source language. That way we prove only what is needed to show conservativity, without having to prove the consistency of λS all over again. After identifying the main difficulties, we characterize a *PTS completion* [17, 16] 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* [18].

1.3 Outline

In Section 2, we recall the theory of pure type systems. In Section 3, we present the framework of the λΠ calculus modulo rewriting. In Section 4, we introduce Cousineau and Dowek's embedding of functional pure type systems in the λΠ calculus modulo rewriting. In Section 5, we prove the conservativity of the embedding using the techniques mentioned above. In Section 6, we summarize the results and discuss future work. Some long proofs have been omitted and can be found in the long version of this paper[1].

[1] Available online at arXiv:1504.05038.

$$\textsc{Empty}\ \frac{}{\mathrm{WF}_{\lambda S}(\cdot)} \qquad \textsc{Declaration}\ \frac{\Gamma \vdash_{\lambda S} A : s \qquad x \notin \Gamma}{\mathrm{WF}_{\lambda S}(\Gamma, x : A)} \qquad \textsc{Variable}\ \frac{\mathrm{WF}_{\lambda S}(\Gamma) \qquad (x : A) \in \Gamma}{\Gamma \vdash_{\lambda S} x : A}$$

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

$$\textsc{Abstraction}\ \frac{\Gamma, x : A \vdash_{\lambda S} M : B \qquad \Gamma \vdash_{\lambda S} \Pi x{:}A.\, B : s}{\Gamma \vdash_{\lambda S} \lambda x{:}A.\, M : \Pi x{:}A.\, B} \qquad \textsc{Application}\ \frac{\Gamma \vdash_{\lambda S} M : \Pi x{:}A.\, B \qquad \Gamma \vdash_{\lambda S} N : A}{\Gamma \vdash_{\lambda S} M\, N : B[x \backslash N]}$$

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

Figure 1 Typing rules of λS.

2 Pure type systems

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

▶ **Definition 2.** A PTS *specification* is a triple $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ where

- $\mathcal{S}$ 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)

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 $s_3 = s_3'$. The specification is *full* if for all $s_1, s_2 \in \mathcal{S}$, there is a sort s_3 such that $(s_1, s_2, s_3) \in \mathcal{R}$.

▶ **Definition 3.** Given a PTS specification $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ and a countably infinite set of variables $\mathcal{V}$, the abstract syntax of λS is defined by the following grammar:

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

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 Greek letters such as $\Gamma, \Delta, \Sigma, \ldots$ to denote contexts. The set of free variables of a term M is denoted by $\mathrm{FV}(M)$. We write $A \to B$ for $\Pi x{:}A.\, B$ when $x \notin \mathrm{FV}(B)$.

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 $\mathrm{WF}(\Gamma)$ and $\Gamma \vdash M : A$ for some A. We say that A is a Γ-*type* when $\mathrm{WF}(\Gamma)$ 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 $\Gamma \vdash M : A \wedge \Gamma \vdash A : s$.

▶ **Example 4.** The following well-known systems can all be expressed as functional pure type systems using the same set of sorts $\mathcal{S} = \mathsf{Type}, \mathsf{Kind}$ and the same set of axioms $\mathcal{A} = (\mathsf{Type} : \mathsf{Kind})$:

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

▶ **Example 5.** Let $I = \lambda\alpha : \mathsf{Type}.\, \lambda x : \alpha.\, x$ 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:

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

The following properties hold for all pure type systems [1].

▶ **Theorem 6** (Correctness of types). *If* $\Gamma \vdash_{\lambda S} M : A$ *then* $\mathrm{WF}_{\lambda S}(\Gamma)$ *and either* $\Gamma \vdash_{\lambda S} A : s$ *or* $A = s$ *for some* $s \in \mathcal{S}$*, i.e.* A *is a* Γ*-type.*

The reason why we don't always have $\Gamma \vdash_{\lambda S} A : s$ is that some sorts do not have an associated axiom, such as Kind in Example 4, which leads to the following definition.

▶ **Definition 7** (Top-sorts). 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 property is useful for proving properties about systems with top-sorts.

▶ **Theorem 8** (Top-sort types). *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*.*

▶ **Theorem 9** (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* $M_3 \longrightarrow_\beta^* M_4$*.*

▶ **Theorem 10** (Product compatibility). *If* $\Pi x : A.\, B \equiv_\beta \Pi x : A'.\, B'$ *then* $A \equiv_\beta A'$ *and* $B \equiv_\beta B'$*.*

▶ **Theorem 11** (Subject reduction). *If* $\Gamma \vdash_{\lambda S} M : A$ *and* $M \longrightarrow_\beta^* M'$ *then* $\Gamma \vdash_{\lambda S} M' : A$*.*

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

▶ **Theorem 12** (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* $A \equiv_\beta B$*.*

In the rest of the paper, all the pure type systems we will consider will be functional.

3 The λΠ 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 through the Curry-de Bruijn-Howard correspondence to a minimal first-order logic of higher-order terms. As mentioned in Example 4, it can be defined as the functional pure type system λP with the following specification:

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

$$\textsc{Empty}\ \frac{}{\mathrm{WF}_{\lambda\Pi/}(\cdot)} \qquad \textsc{Declaration}\ \frac{\Gamma \vdash_{\lambda\Pi/} A : s \qquad x \notin \Sigma, \Gamma}{\mathrm{WF}_{\lambda\Pi/}(\Gamma, x : A)} \qquad \textsc{Variable}\ \frac{\mathrm{WF}_{\lambda\Pi/}(\Gamma) \qquad (x : A) \in \Sigma, \Gamma}{\Gamma \vdash_{\lambda\Pi/} x : A}$$

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

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

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

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

The $\lambda\Pi$ *calculus modulo rewriting* extends the $\lambda\Pi$ calculus with rewrite rules. By equating terms modulo a set of rewrite rules R in addition to α and β equivalence, it can type more terms using the conversion rule, and therefore express theories that are more complex. The calculus can be seen as a variant of Martin-Löf's logical framework [13, 11] where equalities are expressed as 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 such that $\mathrm{FV}(N) \subseteq \mathrm{FV}(M)$. 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. We define the relation $\longrightarrow_{\beta R}$ as $\longrightarrow_\beta \cup \longrightarrow_R$, the relation $\equiv_R$ as the smallest congruence containing $\longrightarrow_R$, and the relation $\equiv_{\beta R}$ as the smallest congruence containing $\longrightarrow_{\beta R}$.

▶ **Definition 13.** 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 $\Sigma, \Delta \vdash_{\lambda\Pi} N : A$.

▶ **Definition 14.** 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 15.** Let Σ be the context

$$\alpha : \mathsf{Type}, c : \alpha, f : \alpha \to \mathsf{Type}$$

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 all the occurrences of $f\, c$ in δ by $\Pi y{:}\alpha.\, f\, y \to f\, y$.

The system $\lambda\Pi$ is a pure type system and therefore enjoys all the properties mentioned in Section 2. The behavior of $\lambda\Pi/\,(\Sigma, R)$ however depends on the choice of (Σ, R). In particular, some properties analogous to those of pure type systems depend on the confluence of the relation $\longrightarrow_{\beta R}$.

▶ **Theorem 16** (Correctness of types). *If* $\Gamma \vdash_{\lambda\Pi/} M : A$ *then* $\mathrm{WF}_{\lambda\Pi/}(\Gamma)$ *and either* $\Gamma \vdash_{\lambda\Pi/} A : s$ *for some* $s \in \{\mathsf{Type}, \mathsf{Kind}\}$ *or* $A = \mathsf{Kind}$.

▶ **Theorem 17** (Top-sort types). *If* $\Gamma \vdash_{\lambda\Pi/} A : \mathsf{Kind}$ *then either* $A = \mathsf{Type}$ *or* $A = \Pi x{:}B.\, C$ *for some terms* B, C *such that* $\Gamma, x : B \vdash_{\lambda\Pi/} C : \mathsf{Kind}$.

Assuming $\longrightarrow_{\beta R}$ is confluent, the following properties hold [3].

▶ **Theorem 18** (Product compatibility). *If* $\Pi x : A.\, B \equiv_{\beta R} \Pi x : A'.\, B'$ *then* $A \equiv_{\beta R} A'$ *and* $B \equiv_{\beta R} B'$.

▶ **Theorem 19** (Subject reduction). *If* $\Gamma \vdash_{\lambda\Pi/} M : A$ *and* $M \longrightarrow^*_{\beta R} M'$ *then* $\Gamma \vdash_{\lambda\Pi/} M' : A$.

▶ **Theorem 20** (Uniqueness of types). *If* $\Gamma \vdash_{\lambda\Pi/} M : A$ *and* $\Gamma \vdash_{\lambda\Pi/} M : B$ *then* $A \equiv_{\beta R} B$.

4 Embedding FPTS's in the λΠ 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 [6]. In this embedding, sorts are represented as *universes à la Tarski*, as introduced by Martin-Löf [12] and later developed by Luo [11] and Palmgren [14]. The embedding 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 $\lambda\Pi/S$. The proofs of the theorems in this section can be found in the original paper [6].

▶ **Definition 21** (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 well-formed context containing the declarations:

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

Let R_S be the well-typed 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, $\lambda\Pi/(\Sigma_S, R_S)$.

▶ **Theorem 22** (Confluence). *The relation* $\longrightarrow_{\beta R}$ *is confluent.*

The translation is composed of two functions, 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 23.** The translation $|M|_\Gamma$ of Γ-terms and the translation $\|A\|_\Gamma$ of Γ-types are mutually defined as follows.

$$\begin{array}{rcl}
|s|_\Gamma & = & \dot{s} \\
|x|_\Gamma & = & x \\
|M\,N|_\Gamma & = & |M|_\Gamma\ |N|_\Gamma \\
|\lambda x\!:\!A.\,M|_\Gamma & = & \lambda x\!:\!\|A\|_\Gamma\,.\ |M|_{\Gamma,x:A} \\
|\Pi x\!:\!A.\,B|_\Gamma & = & \dot{\pi}_{s_1s_2s_3}\ |A|_\Gamma\ (\lambda x\!:\!\|A\|_\Gamma\,.\ |B|_{\Gamma,x:A}) \\
 & & \text{where } \Gamma \vdash A : s_1 \\
 & & \text{and } \Gamma, x : A \vdash B : s_2 \\
 & & \text{and } (s_1, s_2, s_3) \in \mathcal{R}
\end{array}$$

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

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 24.** The polymorphic identity function of the Calculus of constructions λC is translated as

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

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

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

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

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

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.

▶ **Theorem 25** (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 too weak 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. 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 26.** 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 $nat \to nat$.

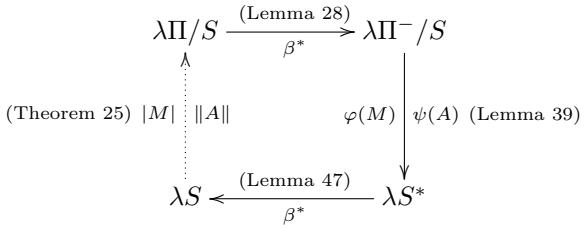
$$nat : u_{\mathsf{Type}} \vdash_{\lambda\Pi/HOL} (|I|\ nat) : (\varepsilon_{\mathsf{Type}}\, nat \to \varepsilon_{\mathsf{Type}}\, nat)$$

We can normalize the term $|I|\ nat$ to $\lambda x : \varepsilon_{\mathsf{Type}}\, nat.\, x$ 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. These terms can be used to construct proofs for the translation of λS types.
3. The $\lambda\Pi/S$ terms that inhabit the translation of λS types can be reduced to the translation of λS terms.

We will proceed as follows. 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.



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 be safely eliminated.

▶ **Definition 27.** A Γ-term M of type C is at the level of Kind (resp. Type) if $\Gamma \vdash C$: Kind (resp. $\Gamma \vdash C$: Type). 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 28.** *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. Indeed, in the $\lambda\Pi$ calculus modulo rewriting the only Kind rule is (Type, Kind, Kind). Therefore $N : A :$ Type. If N reduces to a λ-abstraction then the only redexes it can create are at the level of Type. Therefore, the number of Kind-level β-redexes strictly decreases, so any Kind-level β-reduction strategy will terminate. ◀

▶ **Example 29.** The term

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

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

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

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

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

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

5.2 Minimal completion

To simplify our reducibility proof in the next section, we will translate $\lambda\Pi/S$ back to a pure type system, but since it cannot be λS we will define a slightly larger PTS called λS^* that contains λS and that will be easier to manipulate than $\lambda\Pi/S$.

The reason we need a larger PTS is that we 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 [17, 16] to address these issues by injecting λS into a larger pure type system.

▶ **Definition 30** (Completion [16])**.** 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. These two properties reflect exactly the discrepancy between λS and $\lambda\Pi^-/S$. Not all completions are conservative though, so we define the following completion.

▶ **Definition 31** (Minimal completion)**.** We define the *minimal completion of* S, written S^*, to be the following specification:

$$\begin{aligned}
\mathcal{S}^* &= \mathcal{S} \cup \{\tau\} \\
\mathcal{A}^* &= \mathcal{A} \cup \{(s_1 : \tau) \mid s_1 \in \mathcal{S}, \nexists s2, (s_1 : s_2) \in \mathcal{A}\} \\
\mathcal{R}^* &= \mathcal{R} \cup \{(s_1, s_2, \tau) \mid s_1, s_2 \in \mathcal{S}^*, \nexists s_3, (s_1, s_2, s_3) \in \mathcal{R}\}
\end{aligned}$$

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

We add a new top-sort τ and axioms $s : \tau$ for all previous top-sorts s, and complete the rules to obtain a PTS full. The new system is a completion by Definition 30 and it is minimal in the sense that we generically added the smallest number of sorts, axioms, and rules so that the result is guaranteed to be conservative. 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 32.** The polymorphic identity function is well-typed in λHOL^*.

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

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

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

▶ **Definition 33** (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 34.** *For any* Γ*-term* M *and* Γ*-type* A*,*
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 where $M = \Pi x : A.\, B$. By induction hypothesis, $\varphi(|A|) \equiv_\beta A$ and $\varphi(|B|) \equiv_\beta B$. Therefore

$$\begin{array}{rcl}
\varphi(|M|) & = & (\lambda\alpha.\, \lambda\beta.\, \Pi x\!:\!\alpha.\, \beta\, x)\ \varphi(|A|)\, (\lambda x.\, \varphi(|B|)) \\
 & \longrightarrow_\beta^* & \Pi x\!:\!\varphi(|A|).\, \varphi(|B|) \\
 & \equiv_\beta & \Pi x\!:\!A.\, B
\end{array}$$

◀

Next we show that the inverse translations preserve typing.

▶ **Lemma 35.**
1. $\varphi(M[x\backslash N]) = \varphi(M)[x\backslash\varphi(N)]$
2. $\psi(A[x\backslash N]) = \psi(A)[x\backslash\varphi(N)]$

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 $\Pi y\!:\!B.\, C[x\backslash N] = \Pi y\!:\!B[x\backslash N].\, C[x\backslash N]$.

By induction hypothesis, $\psi(B[x\backslash N]) = \psi(B)[x\backslash\varphi(N)]$ and $\psi(C[x\backslash N]) = \psi(C)[x\backslash\varphi(N)]$. Therefore

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

◀

▶ **Lemma 36.**
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 base cases.
- Case $M = (\lambda x{:}A_1.\,M_1)\,N_1$, $N = M_1[x\backslash N_1]$. Then $\varphi(M) = (\lambda x{:}\psi(A_1).\,\varphi(M_1))\,\varphi(N_1)$. Therefore $\varphi(M) \longrightarrow_\beta \varphi(M_1)[x\backslash\varphi(N_1)]$ which is equal to $\varphi(M_1[x\backslash N_1])$ by Lemma 35.
- Case $A = \varepsilon_s\,\dot{s}$, $B = u_s$. Then $\psi(A) = s = \psi(B)$.
- Case $A = \varepsilon_{s_1}\,(\dot{\pi}_{s_1s_2s_3}\,A_1\,B_1)$, $B = \Pi x{:}\varepsilon_{s_1}\,A_1.\,\varepsilon_{s_2}\,(B_1\,x)$. Then

$$\begin{array}{rcl}\psi(A) & = & (\lambda\alpha.\,\lambda\beta.\,\Pi x{:}\alpha.\,\beta\,x)\;\varphi(A_1)\,\varphi(B_1) \\ & \longrightarrow_\beta^* & \Pi x{:}\varphi(A_1).\,\varphi(B_1)\,x \\ & = & \psi(\Pi x{:}A_1.\,B_1\,x)\end{array}$$

◀

▶ **Lemma 37.**
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 36. ◀

Because the forward translation of contexts does not introduce any type variable, we define the following restriction on contexts.

▶ **Definition 38** (Object context)**.** We say that Γ is an *object context* if $\Gamma \vdash_{\lambda\Pi/S} A : \mathsf{Type}$ for all $x : A \in \Gamma$. If $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ is an object context, we define $\psi(\Gamma)$ as $(x_1 : \psi(A_1), \ldots, x_n : \psi(A_n))$.

▶ **Lemma 39.** *For any* $\lambda\Pi^-/S$ *object context* Γ *and terms* M, A*:*
1. *If* $\mathrm{WF}_{\lambda\Pi/S}(\Gamma)$ *then* $\mathrm{WF}_{\lambda S^*}(\psi(\Gamma))$.
2. *If* $\Gamma \vdash_{\lambda\Pi/S} M : A : \mathsf{Type}$ *then* $\psi(\Gamma) \vdash_{\lambda S^*} \varphi(M) : \psi(A)$.
3. *If* $\Gamma \vdash_{\lambda\Pi/S} A : \mathsf{Type}$ *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 long version of this paper. ◀

5.3 Reduction to λS

In order to show that λS^* is a conservative extension of λS, we 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 adapt Tait's *reducibility method* [18]. The idea is to strengthen the induction hypothesis of the proof by defining a predicate by induction on the type of the term.

▶ **Definition 40.** The predicate $\Gamma \models_S M : A$ is defined as $\mathrm{WF}_{\lambda S}(\Gamma)$ and $\Gamma \vdash_{\lambda S^*} M : A : s$ for some sort s and:

- 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 $\Gamma \models_S N : B$, $\Gamma \models_S M\, N : C[x \backslash N]$.

Note that recursive definition covers all cases thanks to Theorem 8. To show that it is well-founded, we define the following measure of A.

▶ **Definition 41.** If $\mathrm{WF}_{\lambda S}(\Gamma)$ and $\Gamma \vdash_{\lambda S^*} A : s$ then $\mathcal{H}_\tau(A)$ is defined as:

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

▶ **Lemma 42.** *If $\Gamma, x : B \vdash_{\lambda S^*} C : \tau$ and $\Gamma \vdash_{\lambda S^*} N : B$ then $\mathcal{H}_\tau(C[x \backslash N]) = \mathcal{H}_\tau(C)$.*

Proof. By induction on C. ◀

▶ **Corollary 43.** *Definition 40 is well-founded.*

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

The predicate we defined is compatible with β-equivalence.

▶ **Lemma 44.** *If $\Gamma \models_S M : A$ and $\Gamma \vdash_{\lambda S^*} M' : A$ 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 $\Gamma \models_S N : B$, $\Gamma \models_S M\, N : C[x \backslash N]$. By induction hypothesis, $\Gamma \models_S M'\, N : C[x \backslash N]$. Therefore $\Gamma \models_S M' : \Pi x : B.\, C$. ◀

▶ **Lemma 45.** *If $\Gamma \models_S M : A$ and $\Gamma \vdash_{\lambda S^*} A' : s$ 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 conversion, $\Gamma \vdash_{\lambda S^*} M : A'$, so by subject reduction $\Gamma \vdash_{\lambda S^*} M' : A'$. By confluence, subject reduction, 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 $\Gamma \models_S N : B$, $\Gamma \models_S M\, N : C[x \backslash N]$. By product compatibility, $A' = \Pi x : B'.\, C'$ such that $B \equiv_\beta B'$ and $C \equiv_\beta C'$. By induction hypothesis, $\Gamma \models_S M\, N : C'[x \backslash N]$. Therefore $\Gamma \models_S M : \Pi x : B'.\, C'$. ◀

We extend the definition of the inductive predicate to contexts and substitutions before proving the main general lemma.

▶ **Definition 46.** If $\mathrm{WF}_{\lambda S^*}(\Gamma)$, $\mathrm{WF}_{\lambda S}(\Gamma')$, and σ is a substitution for the variables of Γ, then $\Gamma' \models_S \sigma : \Gamma$ when $\Gamma' \models_S \sigma(x) : \sigma(A)$ for all $(x : A) \in \Gamma$.

▶ **Lemma 47.** *If* $\Gamma \vdash_{\lambda S^*} M : A : s$ *then for any context* Γ' *and substitution* σ *such that* $\mathrm{WF}_{\lambda S}(\Gamma')$ *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 long version of this paper. ◀

▶ **Corollary 48.** *Suppose* $\mathrm{WF}_{\lambda S}(\Gamma)$ *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 49** (Conservativity). *For any* Γ*-type* A *of* λS, *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 28, 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 39 and 34, $\Gamma \vdash_{\lambda S^*} \varphi(N^-) : A$. By Corollary 48, 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 the semantics of λS. This effect is similar to various conservative extensions of pure type systems such as *pure type systems with definitions* [17], *pure type systems without the* Π*-condition* [16], or *predicative (ML) polymorphism* [15]. Inconsistency in pure type systems usually does not come from the ability to type more functions, but from the possible impredicativity caused by assigning a sort to the type of these functions. It is clear that no such effect 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.

One could ask whether the techniques we used are adequate. While the construction of λS^* is not absolutely necessary, we feel that it simplifies the proof and that it helps us better understand the behavior of $\lambda\Pi/S$ by reflecting it back into a pure type system. The relative normalization steps of Section 5.3 correspond to the normalization of a simply typed λ calculus. Therefore, it is not surprising that we had to use Tait's reducibility method. However, 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^∞) [17], which is the basis for proof assistants such as Coq and Matita.

The results of this paper can be extended in several directions. They could be adapted to show the conservativity of other embeddings, such as that of the *calculus of inductive constructions* (CIC) [4]. They also indirectly imply that $\lambda\Pi/S$ is weakly normalizing when λS is weakly normalizing because the image of a λS term is normalizing [6]. The strong normalization of $\lambda\Pi/S$ when λS is strongly normalizing is still an open problem. The Barendregt-Geuvers-Klop conjecture states that any weakly normalizing PTS is also strongly normalizing [8]. There is evidence that this conjecture is true [2], in which case we hope that its proof could be adapted to prove the strong normalization of $\lambda\Pi/S$. Weak normalization could also be used as an intermediary step for constructing models by induction on types in order to prove strong normalization.

Acknowledgments. We thank Gilles Dowek and Guillaume Burel for their support and feedback, as well as Frédéric Blanqui, Raphaël Cauderlier, and the various anonymous referees for their comments and suggestions on previous versions of this paper.

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 Frédéric Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(01):37–92, 2005.
4 M. Boespflug and G. Burel. CoqInE: translating the calculus of inductive constructions into the λΠ-calculus modulo. In *Proof Exchange for Theorem Proving - Second International Workshop, PxTP 2012*, pages 44–50, 2012.
5 M. Boespflug, Q. Carbonneaux, and O. Hermant. The λΠ-calculus modulo as a universal proof language. In *Proof Exchange for Theorem Proving - Second International Workshop, PxTP 2012*, pages 28–43, 2012.
6 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.
7 Gilles Dowek. Models and termination of proof-reduction in the λΠ-calculus modulo theory. arXiv:1501.06522, hal-01101834, 2014.
8 Herman Geuvers. *Logics and type systems*. PhD thesis, University of Nijmegen, 1993.
9 Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse de doctorat, Université Paris VII, 1972.
10 Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, 1993.
11 Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, Inc., New York, NY, USA, 1994.
12 Per Martin-Löf and Giovanni Sambin. *Intuitionistic type theory*, volume 17. Bibliopolis Naples, 1984.
13 Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's type theory*, volume 200. Oxford University Press Oxford, 1990.
14 Erik Palmgren. On universes in type theory. In *Twenty-five years of constructive type theory*, pages 191–204. Oxford University Press, 1998.
15 Cody Roux and Floris van Doorn. The structural theory of pure type systems. In Gilles Dowek, editor, *Rewriting and Typed Lambda Calculi*, number 8560 in Lecture Notes in Computer Science, pages 364–378. Springer International Publishing, 2014.
16 Paula Severi. Pure type systems without the Pi-condition. *Proceedings of 7th Nordic Workshop on Programming Theory*, 1995.
17 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.
18 W. W. Tait. Intensional interpretations of functionals of finite type I. *The Journal of Symbolic Logic*, 32(2):198–212, 1967.