On the relation between Church-style typing and Curry-style typing[*]

Jonathan P. Seldin
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
jonathan.seldin@uleth.ca
http://www.cs.uleth.ca/∼seldin

September 5, 2008

Abstract

There are two versions of type assignment in λ-calculus: Church-style, in which the type of each variable is fixed, and Curry-style (also called "domain free"), in which it is not. As an example, in Church-style typing, $\lambda x : A . x$ is the identity function on type A, and it has type $A \to A$ but not $B \to B$ for a type B different from A. In Curry-style typing, $\lambda x . x$ is a general identity function with type $C \to C$ for *every* type C.

In this paper, I will show how to interpret in a Curry-style system every Pure Type System (PTS) in the Church-style without losing any typing information. I will also prove a kind of comservative extension result for this interpretation, a result which implies that for most consistent PTSs of the Church-style, the corresponding Curry-style system is consistent. (This generalizes some unpublished work with Garrel Pottinger.) I will then show how to interpret in a system of the

[*]This work was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada. Some of the work in this paper was joint work with Garrel Pottinger.

Church-style (a modified PTS, stronger than a PTS) every PTS-like system in the Curry style.

Keywords: Church-style typing, Curry-style typing, domain-full typing, domain-free typing

1 Introduction

There are two main styles of type theory in λ-calculus: the Church-style, in which each abstraction indicates the type of the variable, as in

$$\lambda x : A \,.\, M,$$

and the Curry-style, in which no such type is given:

$$\lambda x \,.\, M.$$

These two styles of typing are often called the *domain-full* and the *domain-free* styles respectively. These styles are compared and discussed in [2].

Remark 1 Barthe and Sørensen [2] distinguish between domain-free systems, which they regard as Church-style systems with the types of the bound variables omitted, and what they think of as the Curry view, in which typing rules assign types to terms that already exist in the pure λ-calculus. In this they are following Barendregt [1, Definition 4.1.7], who identifies as the Curry-version of $\lambda 2$ a system in which the rules for $\forall$ are given as follows:

$$(\forall\text{-elimination}) \qquad \frac{\Gamma \vdash M : (\forall \alpha \,.\, \sigma)}{\Gamma \vdash M : [\tau/\alpha]\sigma}$$

$$(\forall\text{-introduction}) \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M : (\forall \alpha \,.\, \sigma)} \qquad \begin{array}{c}\textit{Condition:}\\ \alpha \notin \mathrm{FV}(\Gamma).\end{array}$$

But these two rules seem to me to be closer to the ideas of the intersection type systems than to any of the systems that interested Curry. As should be clear from [5, Chapter 14], Curry was basically interested in the system usually called $\lambda\!\rightarrow$, and the basic characteristic of his version is that $\lambda x \,.\, x$ in λ-calculus and I in a system of combinators can have any type of the form $\alpha \rightarrow \alpha$, which Curry wrote $\mathsf{F}\alpha\alpha$. This is what Curry called *functionality.* He also suggested what he called *generalized functionality*, in which the constant F was replaced by G, where $\mathsf{G}\alpha\beta$ is the type we now write $(\Pi x : \alpha \,.\, \beta x)$. I treated a basic form of generalized functionality in Curry's style in my paper [14]. On the other hand, Church's typing is probably best exemplified by his simple type theory of [4], which is characterized by the presence of the type of each bound variable, as in $\lambda x^{\alpha} \,.\, M$. Hence, to be historically accurate, I think it is better to identify the Curry-style with domain-free type systems.

In a Curry-style system, there are terms like $\lambda y \,.\, yy$ that have no types. But there is a sense in which this is also true in a Church-style system: $\lambda y : A \,.\, yy$ is a perfectly good pseudoterm of a Church-style system, but it will not have a type in many of the usual systems. Perhaps the vocabulary used may disguise the similarity here: a pseudoterm in a Church-style system corresponds to a term in a Curry-style system.

There is one standard interpretation of a Church-style system in a corresponding Curry-style system: The function Erase, which simply deletes the domains from a formula, so that

$$\text{Erase}(\lambda x : A \,.\, M) \equiv \lambda x \,.\, M.$$

(The formal definition of Erase is given in Remark 4 below.) Erase has been used estensively to relate Church-style systems and Curry-style systems. For example, Erase and modifications of Erase are used by Steffen van Bakel et al [16] to compare Church-style PTSs (which they call typed systems) and Curry-style PTSs (which they call type assignment systems). But using Erase to interpret a Church-style PTS in a Curry-style PTS causes some type information to be lost. One might think that this information can be restored from the type $(\Pi x : A \,.\, B)$ of an abstraction term $(\lambda x \,.\, M)$ by mapping this to $(\lambda x : A \,.\, B)$. But as is shown in [16, Example 3.5, Theorem 3.6], this is too simple and may not work properly for some systems.

In this paper, I propose to show how to interpret a system of each style in an appropriate system of the other without this kind of loss of typing information. In one direction, the direction from Church-style to Curry-style, the interpretation is defined by allowing an abstraction of the form $(\lambda x : A \,.\, M)$ to be an abbreviation for a term of the Curry-style PTS, so the information about the type of the bound variable in the λ-abstraction is not lost. This interpretation extends previous work with Garrel Pottinger [13],[1] which carried through the interpretation for three systems from the Barendregt cube:

[1]But the reader will not need knowledge ot [13] to understand the present paper.

$\lambda \rightarrow$, $\lambda 2$, λC, and its extension, the system ECC of Luo [8, 9]. The Curry-style PTS into which the Church-style system is interpreted is, in general, stronger than the Church-style PTS interpreted in it; for one thing, the term interpreting $(\lambda x : A . M)$ β-reduces to $(\lambda x . M)$, and this does not correspond to any feature of the Church-style PTS. However, it is not too much stronger, and to show this I prove a kind of conservative extension result for the Curry-style system over the Church-style system, a result from which it follows that for most consistent PTSs of the Church-style, the corresponding Curry-style system is also consistent.

In the other direction, the Church-style system into which the Curry-style PTS is interpreted is not a PTS, but is obtained from a PTS by the addition of a rule. The idea here is to provide a dummy type A to be the "domain" of a Curry-style abstraction term $(\lambda x . M)$, so that the Church-style abstraction which interprets this Curry-style abstraction is $(\lambda x : \mathsf{A} . M)$. This dummy type A does not have any sort as its type, so it can only play a very limited role in the Church-style system, but to make the interpretation work a rule is must be added to the Church-style PTS to allow an inference from $\Gamma \vdash (\lambda x : B . M) : (\Pi x : B . C)$ to $\Gamma \vdash (\lambda x : \mathsf{A} . M) : (\Pi x : B . C)$. This rule corresponds in a sense to the fact that in the Curry-style system, the term interpreting $\lambda x : B . M$ β-reduces to $(\lambda x . M)$.

I would like to thank Martin Bunder, Roger Hindley, Garrel Pottinger, and the anonymous referees for their helpful comments and suggestions.

2 Basic definitions

I will assume that the reader is familiar with the basic definitions and notation of [1] and [7]. The systems considered in this paper are defined from the pure syntax for pseudoterms

$$M \longrightarrow x|c|MM|\lambda x : M . M|(\Pi x : M . M),$$

for a Church-style system or from the pure syntax for terms

$$M \longrightarrow x|c|MM|\lambda x . M|(\Pi x : M . M)$$

for a Curry-style system. The reduction we consider will be β-reduction. For a Church-style system a β-contraction will be

$$(\beta_{\mathrm{Ch}}) \qquad (\lambda x : A . M)N \rhd_{\mathrm{Ch}} [N/x]M,$$

whereas for a Curry-style system, it will be

$$(\beta_{\mathrm{Cu}}) \qquad\qquad (\lambda x \,.\, M)N \rhd_{\mathrm{Cu}} [N/x]M.$$

Remark 2 It is relatively easy to add η-reductions. For the Church-style syntax, this is done by adding the following contractions:

$$(\eta_{\mathrm{Ch}}) \qquad (\lambda x : A \,.\, Mx) \rhd_{\eta\mathrm{Ch}} M \qquad \text{provided that } x \notin \mathrm{FV}(M).$$

For the Curry-style syntax, this is the following:

$$(\eta_{\mathrm{Cu}}) \qquad (\lambda x \,.\, Mx) \rhd_{\eta\mathrm{Cu}} M \qquad \text{provided that } x \notin \mathrm{FV}(M).$$

For the Curry-style syntax, doing this causes no problems, but for the Church-style syntax, the Church-Rosser Theorem fails, as the following example due to Nederpelt [10, p. 71] shows: Let x, y, and z be distinct variables. Then

$$\lambda x : y \,.\, (\lambda x : z \,.\, x)x \rhd_{\beta_{\mathrm{Ch}}} \lambda x : y \,.\, x$$

and

$$\lambda x : y \,.\, (\lambda x : z \,.\, x)x \rhd_{\eta} \lambda x : z \,.\, x,$$

and the terms $\lambda x : y \,.\, x$ and $\lambda x : z \,.\, x$ are distinct terms in normal form.

There are some systems in the Church-style in which the Church-Rosser Theorem does hold for $\beta\eta$-reduction, as proved for so-called "functional" PTSs by Herman Geuvers in [6]. But the result does not hold for all PTSs.

For this reason, β-reduction is usually treated separately from $\beta\eta$-reduction in work on typed λ-calculi, and they will be treated separately in this paper as well.

A *pseudo-context* is a finite, ordered sequence $x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$, where the variables $x_1, x_2, \ldots x_n$ are all distinct. Typing judgements will all have the form

$$\Gamma \vdash M : A,$$

which says that $M : A$, where M and A are pseudoterms (or terms in a Curry-style system), can be derived from the pseudocontext Γ by the typing rules of the system; in this case, M and A are called *legal expressions* or *legal terms* and Γ is a *legal context.*

Definition 1 A *pure type system*, or *PTS* is determined by a *specification*, which consists of a triple $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ where

1. $\mathcal{S}$ is a set of constants called *sorts*;
2. $\mathcal{A}$ is a set of *axioms* of the form
$$c : s,$$
where c is a constant and s is a sort;
3. $\mathcal{R}$ is a set of *rules* of the form
$$(s_1, s_2, s_3),$$
where $s_1, s_2, s_3 \in \mathcal{S}$. A rule of the form (s_1, s_2, s_2) is often written (s_1, s_2).

The typing rules for a *Church-style* PTS, $\lambda S = \lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ are as follows:

(axiom) $$\vdash\ c : s$$ *Condition:* $c : s \in \mathcal{A}$;

(start) $$\frac{\Gamma\ \vdash\ A : s}{\Gamma, x : A\ \vdash\ x : A}$$ *Condition:* $s \in \mathcal{S}$ and $x \notin \mathrm{FV}(\Gamma)$;

(weak) $$\frac{\Gamma\ \vdash\ M : B \quad \Gamma\ \vdash\ A : s}{\Gamma, x : A\ \vdash\ M : B}$$ *Condition:* $s \in \mathcal{S}$ and $x \notin \mathrm{FV}(\Gamma)$;

(prod) $$\frac{\Gamma\ \vdash\ A : s_1 \quad \Gamma, x : A\ \vdash\ B : s_2}{\Gamma\ \vdash\ (\Pi x : A\ .\ B) : s_3}$$ *Condition:* $(s_1, s_2, s_3) \in \mathcal{R}$;

(appl) $$\frac{\Gamma\ \vdash\ M : (\Pi x : A\ .\ B) \quad \Gamma\ \vdash\ N : A}{\Gamma\ \vdash\ MN : [N/x]B}$$;

(abstr$_{\mathrm{Ch}}$) $$\frac{\Gamma, x : A\ \vdash\ M : B \quad \Gamma\ \vdash\ (\Pi x : A\ .\ B) : s}{\Gamma\ \vdash\ (\lambda x : A\ .\ M) : (\Pi x : A\ .\ B)}$$ *Condition:* $s \in \mathcal{S}$;

(conv) $$\frac{\Gamma\ \vdash\ M : B \quad \Gamma\ \vdash\ B' : s \quad B =_\beta B'}{\Gamma\ \vdash\ M : B'}$$ *Condition:* $s \in \mathcal{S}$.

The typing rules for a *Curry-style* PTS, $CS = C(\mathcal{S}, \mathcal{A}, \mathcal{R})$, are the same except that the rule (abstr$_{\mathrm{Ch}}$) is replaced by

$$(\mathrm{abstr}_{\mathrm{Cu}}) \qquad \frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash (\Pi x : A \,.\, B) : s}{\Gamma \vdash (\lambda x \,.\, M) : (\Pi x : A \,.\, B)} \qquad \textit{Condition: } s \in \mathcal{S}.$$

Remark 3 It is not hard to see from this definition that if a variable x occurs on the right of a provable statement of the form $\Gamma \vdash M : A$, then x is one of the variables to which a type is assigned in Γ. It follows that if a term $\lambda x : A \,.\, M$ is assigned a type by a provable statement, then $x \notin \mathrm{FV}(A)$. From now on we will assume that in any term of the form $\lambda x : A \,.\, M$, $x \notin \mathrm{FV}(A)$.

We will need the following sharpened form of the Generation Lemma, which is proved in [3, Lemma 3.7]:

Lemma 1 *If $\Gamma \vdash M : A$, then one of the following conditions holds:*

1. *M is a constant c, in which case $(c : A) \in \mathcal{A}$ (in which case A is a sort), or there is a pseudoterm A' such that $A' =_\beta A$ and $(c : A') \in \mathcal{A}$ and there is a sort s such that $\Gamma \vdash A : s$.*

2. *M is a variable x, in which case $x : A$ is in Γ or there is a pseudoterm A' such that $A' =_\beta A$ and $x : A'$ is in Γ and there is a sort s such that $\Gamma \vdash A : s$.*

3. *$M \equiv (\Pi x : B \,.\, C)$, in which case there are sorts s_1, s_2, s_3 such that $\Gamma \vdash B : s_1$ and $\Gamma, x : B \vdash C : s_2$, and $(s_1, s_2, s_3) \in \mathcal{R}$ and either $A \equiv s_3$ or $A =_\beta s_3$ or there is a sort s such that $\Gamma \vdash A : s$.*

4. *$M \equiv (\lambda x : B \,.\, N)$, in which case there are a pseudoterm C and a sort s_3 such that $\Gamma \vdash (\Pi x : B \,.\, C) : s_3$ and $\Gamma, x : B \vdash N : C$ and $A \equiv (\Pi x : B \,.\, C)$ or else $A =_\beta (\Pi x : B \,.\, C)$ and there is a sort s such that $\Gamma \vdash A : s$.*

5. *$M \equiv PQ$, in which case there are pseudoterms B and C such that $\Gamma \vdash P : (\Pi x : B \,.\, C)$ and $\Gamma \vdash Q : B$ and either $A \equiv [N/x]C$ or else $A =_\beta [N/x]C$ and there is a sort s such that $\Gamma \vdash A : s$.*

In each case the derivations of judgements of the form $\Gamma \vdash R : D$ have shorter length than that of $\Gamma \vdash M : A$, where the length of a derivation is the total number of steps in the derivation. (The two steps $\Gamma, x : B \vdash C : s_2$ in 3 and $\Gamma, x : B \vdash M : C$ in 4 may not have shorter derivations.)

Remark 4 Note that the formal definition of Erase is by induction on the structure of the pseudoterms of the Church-style syntax as follows:

1. If x is a variable, then $\text{Erase}(x) \equiv x$.
2. If c is a constant, then $\text{Erase}(c) \equiv c$.
3. $\text{Erase}(MN) \equiv (\text{Erase}(M)\text{Erase}(N))$.
4. $\text{Erase}(\lambda x : A . M) \equiv (\lambda x . \text{Erase}(M))$.
5. $\text{Erase}(\Pi x : A . B) \equiv (\Pi x : \text{Erase}(A) . \text{Erase}(B))$.

Luo's *extended calculus of constructions*, ECC, is the PTS determined by the following sets:

$$\begin{array}{lcl}
\mathcal{S} & = & \{\star\} \cup \{\Box_n : n \text{ a nonnegative integer}\} \\
\mathcal{A} & = & \{\star : \Box_0\} \cup \{\Box_n : \Box_{n+1} \ : n \text{ a nonnegative integer}\} \\
\mathcal{R} & = & \{(\star,\star,\star), (\star,\star,\Box_n), (\Box_n,\star,\star), (\Box_n,\star,\Box_m) : 0 \le n \le m\} \\
 & & \cup\{(\star,\Box_n,\Box_m) : n \le m\} \cup \{(\Box_n,\Box_m,\Box_r) : 0 \le n \le r \text{ and } 0 \le m \le r\}.
\end{array}$$

3 Church-style to Curry-style

The basic idea of this interpretation is due to Garrel Pottinger [11, §9], who proposed using a constant Label so that in the Curry-style syntax

$$(\lambda x : A . M)$$

is an abbreviation for

$$\mathsf{Label} A(\lambda x . M).$$

By the analogy with (β_{Ch}), we will want

$$\mathsf{Label} A(\lambda x . M)N \rhd_{\text{Cu}} [N/x]M.$$

This suggests that Label should have the reduction rule

$$\mathsf{Label}XYZ \rhd YZ,$$

as proposed by Pottinger in [11, §9].[2] This would suggest, in turn, that we define Label to be $\lambda xyz\,.\,yz$. However, the second argument of Label will always be replaced by an abstraction term when we use it, and this will give us

$$\begin{array}{rll} (\lambda xyz\,.\,yz)A(\lambda x\,.\,M) & \rhd_\beta & \lambda z\,.\,(\lambda x\,.\,M)z \\ & \rhd_\beta & \lambda z\,.\,[z/x]M \\ & \rhd_\alpha & \lambda x\,.\,M. \end{array}$$

Thus, Label can be defined by

$$\mathsf{Label} \equiv \lambda xy\,.\,y. \tag{1}$$

With this definition, we can define a function to interpret the Church-style syntax in Curry-style syntax as follows:

Definition 2 The function $-^{\mathrm{Cu}}$ from the Church-style syntax to the Curry-style syntax is defined as follows by induction on the structure of the pseudoterms of the Church-style syntax:

1. If x is a variable, $x^{\mathrm{Cu}} \equiv x$,
2. If c is a constant, $c^{\mathrm{Cu}} \equiv c$,
3. $(MN)^{\mathrm{Cu}} \equiv M^{\mathrm{Cu}}N^{\mathrm{Cu}}$,
4. $(\lambda x : A\,.\,M)^{\mathrm{Cu}} \equiv \mathsf{Label}A^{\mathrm{Cu}}(\lambda x\,.\,M^{\mathrm{Cu}})$, where Label is defined by (1),
5. $(\Pi x : A\,.\,B)^{\mathrm{Cu}} \equiv (\Pi x : A^{\mathrm{Cu}}\,.\,B^{\mathrm{Cu}})$.

The following lemmas are easily proved by induction:

Lemma 2 *For every pseudoterm of the Church-style syntax,* $\mathrm{FV}(M^{\mathrm{Cu}}) = \mathrm{FV}(M)$*.*

[2]Pottinger used "ϕ" for "Label".

Lemma 3 *For pseudoterms M and N of the Church-style syntax,* $([N/x]M)^{\mathrm{Cu}} \equiv [N^{\mathrm{Cu}}/x]M^{\mathrm{Cu}}$.

Lemma 4 *If*

$$M =_{\beta_{\mathrm{Ch}}} N$$

then

$$M^{\mathrm{Cu}} =_{\beta_{\mathrm{Cu}}} N^{\mathrm{Cu}}.$$

For a pseudocontext

$$\Gamma \equiv x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$$

define

$$\Gamma^{\mathrm{Cu}} \equiv x_1 : A_1^{\mathrm{Cu}}, x_2 : A_2^{\mathrm{Cu}}, \ldots, x_n : A_n^{\mathrm{Cu}}.$$

In order to make use of Label, it will be necessary for it to have a type. As is clear from the proof of Theorem 1, we will need the type for Label when we have sorts s_1 and s_2 and a context Γ such that

$$\Gamma \vdash A : s_1 \qquad \text{and} \qquad \Gamma \vdash (\Pi x : A \,.\, B) : s_2. \tag{2}$$

The typing it requires two stages:

1. Typing $\lambda y \,.\, y$. The type of $\lambda y \,.\, y$ will have to be obtained as the conclusion of the rule (abstr$_{\mathrm{Cu}}$), and we will want to deduce

   $$\Gamma \vdash (\lambda y \,.\, y) : (\Pi x : A \,.\, B) \to (\Pi x : A \,.\, B) \tag{3}$$

   Let us call sorts s_1 and s_2 for which when we have (2) we also have (3) I-*complete* sorts.

   In order to deduce (3) by (abstr$_{\mathrm{Cu}}$), we will need a sort s_3 for which there is a rule in $\mathcal{S}$ of the form (s_2, s_2, s_3), and we will have

   $$\Gamma \vdash (\Pi x : A \,.\, B) \to (\Pi x : A \,.\, B) : s_3. \tag{4}$$

   If the rules are all of the form (s, s') as in the Barendregt λ-cube, then we will need to have the rule $(s_2, s_2) \in \mathcal{R}$, and we will have $s_3 \equiv s_2$. If the PTS in question is singly sorted ([1, Definition 5.2.19]), so that the unicity of types property holds ([1, Lemma 5.2.21]), then it is impossible for $(\Pi x : A \,.\, B)$ to have more than one sort as a type, and these conditions are necessary and sufficient for the sorts s_1 and s_2 to be I-complete. If the PTS is not singly sorted, so that the unicity of types property does not hold, then these conditions will only be sufficient, and may not be necessary. The systems of the λ-cube, which are singly sorted, the PTSs in which both $\star$ and $\Box$ are I-complete are $\lambda\rightarrow$, $\lambda\underline{\omega}$, $\lambda\omega$, $\lambda\mathrm{P}\underline{\omega}$, and $\lambda\mathrm{C}$. In the PTS ECC, which is not in the λ-cube, all sorts are I-complete. (Note that ECC is not singly sorted.)

2. Typing $\lambda xy\,.\,y$. To finish, we will need to deduce

$$\Gamma\ \vdash\ \mathsf{Label} : (\Pi u : s_1\,.\,(\Pi x : u\,.\,B) \to (\Pi x : u\,.\,B)). \qquad (5)$$

This will have to be the conclusion of an inference by (abstr$_{\mathrm{Cu}}$) whose premises are

$$\Gamma, u : s_1\ \vdash\ (\lambda y\,.\,y) : (\Pi x : u\,.\,B) \to (\Pi x : u\,.\,B)$$

and

$$\Gamma\ \vdash\ (\Pi u : s_1\,.\,(\Pi x : u\,.\,B) \to (\Pi x : u\,.\,B)) : s_4$$

for some sorts s_4 and s_5 for which $(s_5, s_3, s_4) \in \mathcal{R}$ and $s_1 : s_5 \in \mathcal{A}$. The first of these premises will hold in any PTS for which the sorts involved are I-complete, but not necessarily otherwise. Also, there will clearly not be sorts s_4 and s_5 satisfying the second premise if s_1 is a topsort, a sort s for which there is no other sort s' such that $s : s' \in \mathcal{A}$. And even if s_1 is not a topsort, in which case there will be an s_5 such that $s_1 : s_5 \in \mathcal{A}$, there may be no s_4 for which there is a rule $(s_5, s_3, s_4) \in \mathcal{R}$. Let us call a sort s_1 of a pair s_1, s_2 of I-complete sorts for which there are sorts s_4 and s_5 for which the second premise holds whenever (2) holds and $s_1 : s_5 \in \mathcal{A}$ a Label-complete sort. clearly, if a sort is a topsort, then the sort is not Label-complete.

If sorts s_1 and s_2 are I-complete and the sort s_1 is Label-complete, then it is possible to deduce (5) when (2) hold.

Note that if all rules in $\mathcal{R}$ have the form (s, s'), then we will have $s_3 \equiv s_2$, and what we will need for sorts s_1 and s_2 to be I-complete and s_1 to be Label-complete is $(s_5, s_2) \in \mathcal{R}$. This will give us $s_4 \equiv s_2$.

Since $\Box$ is a topsort in every system of the λ-cube, it is not Label-complete for any of those PTSs. Of the PTSs of the λ-cube for which both $\star$ and $\Box$ are I-complete, those in which the sort $\star$ is Label-complete are $\lambda\omega$ and $\lambda\mathrm{C}$. In ECC, every sort is Label-complete.

For sorts of PTSs which are not Label-complete, we will need to arrange for Label to be usable by adding a rule:

$$\mathsf{Label} \quad \frac{\Gamma \vdash A : s_1 \qquad \Gamma \vdash (\Pi x : A \,.\, B) : s_2}{\Gamma \vdash \mathsf{Label}A : ((\Pi x : A \,.\, B) \rightarrow (\Pi x : A \,.\, B))}$$

Condition: $s_1, s_2 \in \mathcal{S}$ and either s_1 and s_2 are not I-complete or they are I-complete but s_1 is not Label-complete.

Definition 3 Given a PTS $\lambda S \equiv \lambda(\mathcal{S}_\lambda, \mathcal{A}_\lambda, \mathcal{R}_\lambda)$ in the Church-style, let CS be defined from λS by replacing rule (abstr$_{\mathrm{Ch}}$) by rule (abstr$_{\mathrm{Cu}}$) and adding Rule Label above.

Note that rule Label is not needed if λS is ECC. The coresonding system CS is just obtained by replacing rule (abstr$_{\mathrm{Ch}}$) by rule (abstr$_{\mathrm{Cu}}$).

By the above argument, we have

Lemma 5 *In every PTS of the Curry-style CS, if (2) hold, then*

$$\Gamma \vdash \mathsf{Label}A : ((\Pi x : A \,.\, B) \rightarrow (\Pi x : A \,.\, B)). \tag{6}$$

Remark 5 In a PTS of the Curry-style for which not all pairs of sorts are I-complete, the Subject-Reduction Theorem does not hold, since (6) holds but

$$\mathsf{Label}A \equiv (\lambda xy \,.\, y)A \rhd_\beta (\lambda y \,.\, y)$$

and (3) does not hold. If every pair of sorts of CS is I-complete, then the Subject-Reduction Theorem does hold.

Remark 6 In some PTSs, it may be the case that in addition to (6) it is possible to deduce

$$\Gamma \vdash \mathsf{Label}C((\Pi x : A \,.\, B) \rightarrow (\Pi x : A \,.\, B)), \tag{7}$$

where C differs from A, and may even be in a different sort. For example, it is possible to derive in the Curry-style version of $\rightarrow$C

$$\vdash (\lambda xy \,.\, y) : \mathsf{N} \rightarrow ((\mathsf{N} \rightarrow \mathsf{N}) \rightarrow (\mathsf{N} \rightarrow \mathsf{N})),$$

where[3]

$$\mathsf{N} \equiv (\Pi u : \star \, . \, (u \to u) \to (u \to u));$$

and from this follows

$$\vdash \ \mathsf{Label} N : ((N \to N) \to (N \to N)).$$

But this is an example of (7) instead of (6); an example of (6) would have to be

$$\vdash \ \mathsf{Label}(N \to N) : ((N \to N) \to (N \to N)).$$

Let us call a derivation in a Curry-style PTS *Church compatible* if no formula of the form (7) occurs in it. By starting at the bottom of a proof and at each step replacing the deduction of (7) by a deduction of (6), it is possible to prove the following lemma.

Lemma 6 *Every derivation in a Curry-style PTS can be converted into a Church compatible derivation with the same premises and conclusion.*

Now we can prove the main "soundness" theorem of this interpretation.

Theorem 1 *If*

$$\Gamma \ \vdash \ M : A \tag{8}$$

holds in λS, then

$$\Gamma^{\mathrm{Cu}} \ \vdash \ M^{\mathrm{Cu}} : A^{\mathrm{Cu}} \tag{9}$$

holds in CS.

Proof By induction on the derivation of (8). The interesting case is for (abstr$_{\mathrm{Ch}}$): in this case, $\Gamma \ \vdash \ M : A$ is the conclusion of an inference by (abstr$_{\mathrm{Ch}}$), so $M \equiv \lambda x : B \, . \, N$ and $A \equiv (\Pi x : B \, . \, C)$, and the premises are

$$\Gamma, x : B \ \vdash \ N : C, \qquad \text{and} \qquad \Gamma \ \vdash \ (\Pi x : B \, . \, C) : s$$

in λS for some $s \in \mathcal{S}$. By the second of these and Lemma 1, there is a sort s_1 such that, in λS,

$$\Gamma \ \vdash \ B : s_1.$$

By the induction hypothesis, we have in CS

$$\Gamma^{\mathrm{Cu}} \ \vdash \ B^{\mathrm{Cu}} : s_1,$$

[3]See [15, §9] to see the role of N in interpreting arithmetic in $\to$C.

and we also have in CS

$$\Gamma^{\mathrm{Cu}}, x : B^{\mathrm{Cu}} \;\vdash\; N^{\mathrm{Cu}} : C^{\mathrm{Cu}} \qquad \text{and} \qquad \Gamma^{\mathrm{Cu}} \;\vdash\; (\Pi x : B^{\mathrm{Cu}} \,.\, C^{\mathrm{Cu}}) : s.$$

By the last two of these and rule (abstr$_{\mathrm{Cu}}$), we have

$$\Gamma^{\mathrm{Cu}} \;\vdash\; (\lambda x \,.\, N^{\mathrm{Cu}}) : (\Pi x : B^{\mathrm{Cu}} \,.\, C^{\mathrm{Cu}}).$$

in CS. Since we also have

$$\Gamma^{\mathrm{Cu}} \;\vdash\; B^{\mathrm{Cu}} : s_1,$$

we have, by Lemma 5,

$$\Gamma^{\mathrm{Cu}} \;\vdash\; \mathsf{Label} B^{\mathrm{Cu}} : (\Pi x : B^{\mathrm{Cu}} \,.\, C^{\mathrm{Cu}}) \to (\Pi x : B^{\mathrm{Cu}} \,.\, C^{\mathrm{Cu}}),$$

and by (appl) we get

$$\Gamma^{\mathrm{Cu}} \;\vdash\; \mathsf{Label} B^{\mathrm{Cu}}(\lambda x \,.\, N^{\mathrm{Cu}}) : (\Pi x : B^{\mathrm{Cu}} \,.\, C^{\mathrm{Cu}}),$$

which is the desired conclusion. ■

Remark 7 The last step of the first part of the proof shows that if

$$\Gamma \;\vdash\; (\lambda x \,.\, M) : (\Pi x : A \,.\, B) \tag{10}$$

in CS, then

$$\Gamma \;\vdash\; \mathsf{Label} A(\lambda x \,.\, M) : (\Pi x : A \,.\, B) \tag{11}$$

is also derivable in CS.

Note that if every pair of sorts in CS is l-complete, then there is an inference from (11) to (10) by the Subject-Reduction Theorem [2, Theorem 17], since

$$\begin{array}{rll} \mathsf{Label} A(\lambda x \,.\, M) & \rhd_{\mathrm{Cu}} & (\lambda uvw \,.\, vw) A(\lambda x \,.\, M) \\ & \rhd_{\mathrm{Cu}} & \lambda w \,.\, (\lambda x \,.\, M) w \\ & \rhd_{\mathrm{Cu}} & \lambda w \,.\, [w/x] M \\ & \rhd_{\mathrm{Cu}} & \lambda x \,.\, M. \end{array}$$

This means that in CS, (10) and (11) are equivalent in these systems.

Remark 8 One might ask why it is not possible to ensure the typing of Label by adding sorts to $\mathcal{S}$ and rules to $\mathcal{R}$ in defining CS from λS. The answer is that it is possible, but in general it would make CS much stronger than λS, and in many cases it would make CS inconsistent in the sense that every type is inhabited. The systems of the λ-cube that would be interpreted in inconsistent Curry-style systems this way are λP2, $\lambda\omega$, $\lambda\mathrm{P}\underline{\omega}$, and λC, all of which are interpreted in systems that include λU of [1, Definition 5.5.1] and $\lambda\underline{\omega}$, which is interpreted in a system containing λU^- of [1, after the proof of Theorem 5.5.26]. The following theorem, which is something of a conservative extension result, shows that for most PTSs, if λS is consistent, then so is CS.

Theorem 2 *If*

$$\Gamma^{\mathrm{Cu}} \vdash M^{\mathrm{Cu}} : B \tag{12}$$

in CS, then there is a term A in the Church-style syntax such that $B =_\beta A^{\mathrm{Cu}}$ and

$$\Gamma \vdash M : A \tag{13}$$

in λS. Furthermore, if

$$\Gamma^{\mathrm{Cu}} \vdash B : s \tag{14}$$

in CS for some $s \in \mathcal{S}$, then there is a term in the Church-style syntax such that $B =_\beta A^{\mathrm{Cu}}$ and

$$\Gamma \vdash A : s \tag{15}$$

in λS.

Proof By induction on the proof of (12) or (14). By Lemma 6, we may assume that the derivation of (12) or (14) is Church compatible. The last rule of (12) or (14) cannot be (abstr$_{\mathrm{Cu}}$), since its conclusion does not have the right form. The interesting case is that for rule (appl), in which case $M^{\mathrm{Cu}} \equiv N_1 N_2$. By Definition 2, there are two subcases.

Subcase 1. $M^{\mathrm{Cu}} \equiv \mathsf{Label} C^{\mathrm{Cu}}(\lambda x \,.\, N^{\mathrm{Cu}})$. Then by the assumption that the derivation of (12) or (14) is Church compatible, $B \equiv (\Pi x : C^{\mathrm{Cu}} \,.\, D)$. Then by Lemma 5, (12) must be derived from

$$\Gamma^{\mathrm{Cu}}, x : C^{\mathrm{Cu}} \vdash N^{\mathrm{Cu}} : D, \tag{16}$$

where $x \notin \mathrm{FV}(\Gamma^{\mathrm{Cu}})$, and

$$\Gamma^{\mathrm{Cu}} \vdash (\Pi x : C^{\mathrm{Cu}} \,.\, D) : s, \tag{17}$$

where $s \in \mathcal{S}$. By (16) and the induction hypothesis, there is E such that $D \equiv E^{\mathrm{Cu}}$ and

$$\Gamma, x : C \ \vdash \ N : E \tag{18}$$

holds in λS. Also,

$$(\Pi x : C^{\mathrm{Cu}} \,.\, D) =_\beta (\Pi x : C^{\mathrm{Cu}} \,.\, E^{\mathrm{Cu}}) =_\beta (\Pi x : C \,.\, E)^{\mathrm{Cu}} \equiv B,$$

so that if $A \equiv (\Pi x : C \,.\, E)$, then $B \equiv A^{\mathrm{Cu}}$. By (17) and the induction hypothesis,

$$\Gamma \ \vdash \ (\Pi x : C \,.\, E) : s \tag{19}$$

holds in λS. By (18), (19), and rule ($\mathrm{appl}_{\mathrm{Ch}}$), in λS,

$$\Gamma \ \vdash \ (\lambda x : C \,.\, N) : (\Pi x : C \,.\, E)$$

or

$$\Gamma \ \vdash \ (\lambda x : C \,.\, N) : A. \tag{20}$$

Since $(\lambda x : C \,.\, N)^{\mathrm{Cu}} \equiv \mathsf{Label}C^{\mathrm{Cu}}(\lambda x \,.\, N^{\mathrm{Cu}}) \equiv M$, (20) is (13).

Subcase 2. $M^{\mathrm{Cu}} \equiv M_1^{\mathrm{Cu}} M_2^{\mathrm{Cu}}$, and the premises for the inference by (appl) in CS are

$$\Gamma^{\mathrm{Cu}} \ \vdash \ M_1^{\mathrm{Cu}} : (\Pi x : D \,.\, F) \tag{21}$$

and

$$\Gamma^{\mathrm{Cu}} \ \vdash \ M_2^{\mathrm{Cu}} : D, \tag{22}$$

where $B \equiv [M_2/x]F$. By (22) and the induction hypothesis, there is G such that $D =_\beta G^{\mathrm{Cu}}$ and

$$\Gamma \ \vdash \ M_2 : G \tag{23}$$

in λS. By (21) and (conv),

$$\Gamma^{\mathrm{Cu}} \ \vdash \ M_1^{\mathrm{Cu}} : (\Pi x : G^{\mathrm{Cu}} \,.\, F),$$

so by the induction hypothesis there is H such that $H^{\mathrm{Cu}} =_\beta (\Pi x : G^{\mathrm{Cu}} \,.\, F)$. It follows by Definition 4 that there is K such that $F =_\beta K^{\mathrm{Cu}}$ and

$$\Gamma \ \vdash \ M_1 : (\Pi x : G \,.\, K) \tag{24}$$

in λS. If we set $A \equiv [M_2/x]K$, then $B =_\beta A^{\mathrm{Cu}}$ and by (appl)

$$\Gamma \ \vdash \ M : [M_2/x]K$$

in λS, which is (13). ■

Let us call a PTS *topsort grounded* if, whenever s is a topsort, it is inhabited by another sort: i.e., if whenever $s \in \mathcal{S}$ is a topsort then there is $s' \in \mathcal{S}$ such that $(s', s) \in \mathcal{A}$. Every PTS of the λ-cube is toposrt grounded, as is ECC (vacuously).

Note that by Lemma 1, if $\Gamma \vdash M : A$ in any PTS, then either $\Gamma \vdash A : s$ for some $s \in \mathcal{S}$ or else A converts to a topsort.

Corollary 1 *If λS is topsort grounded and consistent, then CS is consistent.*

Proof The proof will be by contraposition. Note that since λS is topsort grounded, then every topsort in λS is inhabited. Assume that CS is inconsistent. We need to prove that every type which does not convert to a topsort is inhabited. Since CS is inconsistent, every type of CS is inhabited, so, in particular, for every sort $s \in \mathcal{S}$, there is a closed type M such that

$$\vdash M : (\Pi x : s \,.\, x)$$

in CS. By Lemma 6, we may assume that the derivation is Church compatible. Starting with the bottom of this derivation, for every subdeduction of the form

$$\Gamma \vdash (\lambda y \,.\, P) : (\Pi x : A \,.\, B) \tag{25}$$

whose conclusion is not the minor premise for an inference by (appl) whose conclusion is

$$\Gamma \vdash \mathsf{Label}A(\lambda y \,.\, P) : (\Pi x : A \,.\, B), \tag{26}$$

use Lemma 5 to expand the subproof to a proof of (26). When this process is completed, the result will be proof in CS of

$$\vdash N^{\mathrm{Cu}} : (\Pi x : s \,.\, x).$$

By Theorem 2 and the fact that $(\Pi x : s \,.\, x)^{\mathrm{Cu}} \equiv (\Pi x : s \,.\, x)$, it follows that there is a proof in λS of

$$\vdash N : (\Pi x : s \,.\, x).$$

From this and rule (appl), it follows that in λS,

$$y : s \vdash Ny : y.$$

Hence, if C is any type in s,

$$\vdash\ NC : C,$$

which proves that every type in s is inhabited. Now every type in λS which is not convertible to a topsort is in some sort, so this proves that every type which is not convertible to a topsort is inhabited, which proves that λS is inconsistent. ■

It appears that for topsort grounded PTSs, the main difference between λS and CS is that in the latter, it is possible to derive

$$\Gamma\ \vdash\ (\lambda x : A\ .\ M) : (\Pi x : B\ .\ C)$$

where $B \neq_\beta A$, whereas this is not possible in λS.

Remark 9 If subtyping is present, $\mathsf{Label}A(\lambda x\ .\ M)$ is, in a sense, a restriction of $(\lambda x\ .\ M)$ to the type A as domain, and $(\lambda x\ .\ M)$ is a kind of universal function, whose domain consists of all possible values N of x which have a type that allows $[N/x]M$ to be typed. The equivalence of (10) and (11) means that if all pairs of sorts are I-complete, Curry-style typing identifies functions with their restrictions. At first this might seem surprising, but if we stop to think about it, I think it makes sense: λ-calculus involves *uniform* definitions of functions as rules. In Church-style typing, the domain in an abstraction is given explicitly in the abstraction term; it is the type A in $\lambda x : A\ .\ M$. However, in Curry-style typing, no such domain is given in $\lambda x\ .\ M$. In a Church-style PTS with subtyping, the specification of the domain A in $(\lambda x : A\ .\ M)$ can represent a restriction of a function as a distinct term not convertible to $(\lambda x : B\ .\ M)$ for a type B which is not convertible to A, but there is no way to use the Curry-style syntax to represent a restriction this way.

Remark 10 If we can find a way to add η-reduction to a Church-style system with subtyping without losing the Church-Rosser Theorem, then we will have a Church-style system in which functions cannot be distinguished from their restrictions; see [7, Remark 13.77]. For suppose B is a subtype of C, and suppose that within a certain context, $M : C \to D$. Then the subtyping relation gives us

$$(\lambda u : B\ .\ u) : (B \to C),$$

so that if $x \notin \mathrm{FV}(M)$,

$$(\lambda x : B \,.\, M((\lambda u : B \,.\, u)x))$$

represents the restriction of M to domain type B. Now we have by (β_{Ch}),

$$(\lambda x : B \,.\, M((\lambda u : B \,.\, u)x)) \rhd (\lambda x : B \,.\, Mx),$$

so $(\lambda x : B \,.\, Mx)$ also represents the restriction of M to domain type B. But if we then apply a contraction by (η_{Ch}), we contract $(\lambda x : B \,.\, Mx)$ to M, thus identifying M with its restriction. Thus, to have a system with subtyping in which functions can be distinguished from their restrictions, it is necessary to use a Church-style syntax and use only β-reduction.

This seems to show the importance of the distinction between β-reduction and $\beta\eta$-reduction for type theory.

Remark 11 In his paper [12], Garrel Pottinger introduces a variety of the λ-calculus, which he calls the *multivariate λ-calculus*, in which a term of the form $\lambda x_1 x_2 \ldots x_n \,.\, M$ is not an abbreviation for repeated abstraction, but is a term which can only become the head of a redex if it is followed by n arguments. Thus in that calculus, $(\lambda xyz \,.\, xz(yz))MN$ is not a redex, but $(\lambda xyz \,.\, xz(yz))MNP$ is. If the multivariate λ-calculus is used, then Label would be a term which requires two arguments to make a redex.

In the multivariate λ-calculus, the reduction is β-reduction, not $\beta\eta$-reduction. The reason for this is that η-reduction collapses multivariate abstractions to regular abstractions, since for a multivariate abstract

$$(\lambda x_1 x_2 \ldots x_n \,.\, M)$$

and variables $u_1, u_2, \ldots, u_n$ which do not occur bound or free in our term, we have

$$\begin{aligned}
&\lambda u_1 \,.\, \lambda u_2 \,.\, \ldots \,.\, \lambda u_n \,.\, (\lambda x_1 x_2 \ldots x_n \,.\, M)u_1 u_2 \ldots u_n \\
&\quad \rhd_\beta \quad \lambda u_1 \,.\, \lambda u_2 \,.\, \ldots \lambda u_n \,.\, [u_1/x_1, u_2/x_2, \ldots, u_n/x_n]M \\
&\quad \rhd_\alpha \quad \lambda x_1 \,.\, \lambda x_2 \,.\, \ldots \,.\, \lambda x_n \,.\, M
\end{aligned}$$

and

$$\begin{aligned}
&\lambda u_1 \,.\, \lambda u_2 \,.\, \ldots \,.\, \lambda u_n \,.\, (\lambda x_1 x_2 \ldots x_n \,.\, M)u_1 u_2 \ldots u_n \\
&\quad \rhd_\eta \quad \lambda u_1 \,.\, \lambda u_2 \,.\, \ldots \lambda u_{n-1} \,.\, (\lambda x_1 x_2 \ldots x_n \,.\, M)u_1 u_2 \ldots u_{n-1} \\
&\quad \vdots \\
&\quad \rhd_\eta \quad (\lambda x_1 x_2 \ldots x_n \,.\, M).
\end{aligned}$$

This tells us that if η-reduction is introduced into the multivariate λ-calculus in connection with the use with Label in the previous paragraph, then the distinction between functions and their restrictions would be lost, just as it is with η in the Church-style syntax as shown in Remark 10.

4 Curry-style to Church-style

An interpretation in the reverse direction requires a type to use as the domain in the Church-style syntax to interpret the Curry-style abstract $(\lambda x \,.\, M)$. For this purpose, let us introduce a new atomic constant A, so that the interpretation of $(\lambda x \,.\, M)$ in the Church-style system will be $(\lambda x : \mathsf{A} \,.\, M)$. Then we will need to add the following rule to the system:

$$(\mathsf{A}\lambda) \qquad \frac{\Gamma \;\vdash\; (\lambda x : B \,.\, M) : (\Pi x : B \,.\, C)}{\Gamma \;\vdash\; (\lambda x : \mathsf{A} \,.\, M) : (\Pi x : B \,.\, C).}$$

This rule corresponds to the inference in the Curry-style system from (11) to (10) provided that we think of the Church-style term $(\lambda x : B \,.\, M)$ as being interpreted by the translation $-^{\mathrm{Cu}}$, and is necessary for the proof of the Theorem 3 below.

We can now define the mapping from the Curry-style syntax to the Church-style syntax:

Definition 4 The function $-^{\mathrm{Ch}}$ from the Curry-style syntax to the Church-style syntax is defined by induction on the structure of the pseudoterms:

1. $x^{\mathrm{Ch}} \equiv x$,
2. $c^{\mathrm{Ch}} \equiv c$,
3. $(MN)^{\mathrm{Ch}} \equiv M^{\mathrm{Ch}} N^{\mathrm{Ch}}$,
4. $(\lambda x \,.\, M)^{\mathrm{Ch}} \equiv (\lambda x : \mathsf{A} \,.\, M^{\mathrm{Ch}})$,
5. $(\Pi x : B \,.\, C)^{\mathrm{Ch}} \equiv (\Pi x : B^{\mathrm{Ch}} \,.\, C^{\mathrm{Ch}})$.

The following lemmas follow easily by induction:

Lemma 7 *If M is a term of the Curry-style syntax, then* $\mathrm{FV}(M^{\mathrm{Ch}}) = \mathrm{FV}(M)$.

Lemma 8 *If M and N are terms of the Curry-style syntax, then* $([N/x]M)^{\mathrm{Ch}} \equiv [N^{\mathrm{Ch}}/x]M^{\mathrm{Ch}}$.

Lemma 9 *If*

$$M \rhd_{\beta_{\mathrm{Cu}}} N$$

then

$$M^{\mathrm{Ch}} \rhd_{\beta_{\mathrm{Ch}}} N^{\mathrm{Ch}}.$$

Definition 5 Given a PTS CS in the Curry-style, define the system $\lambda S'$ by first defining λS to be the Church-style PTS with the same specification as CS. Then add the typing rule $(\mathsf{A}\lambda)$ given above.

Note that the system $\lambda S'$ is not a PTS. Note also that there are no postulates which make it possible to deduce $\vdash \mathsf{A} : s$ for any sort s. This means that in $\lambda S'$, the only place in which A can occur is in the domain of an abstraction. In particular, in $\lambda S'$, we cannot introduce assumptions of the form $x : \mathsf{A}$ into any legal context and we cannot prove a result of the form $\Gamma \vdash M : \mathsf{A}$.

Theorem 3 *If*

$$\Gamma \vdash M : B$$

in CS, then

$$\Gamma^{\mathrm{Ch}} \vdash M^{\mathrm{Ch}} : B^{\mathrm{Ch}}$$

in $\lambda S'$, and conversely.

The proof is by induction on the proof of $\Gamma \vdash M : B$. The interesting case is that for $(\mathrm{abstr}_{\mathrm{Cu}})$, and that case is straightforward using the induction hypothesis and rule $(\mathsf{A}\lambda)$. For the converse, note that in a term of the form M^{Ch}, every abstraction has the form $(\lambda x : \mathsf{A} \,.\, M)$.

The proof of the theorem does not depend on how the type A is interpreted, but it might be worth considering how that interpretation should be carried out. Since $(\lambda x : \mathsf{A} \,.\, M)$ is the way the Curry-style abstraction $(\lambda x \,.\, M)$ is interpreted, this suggests that we want to interpret $(\lambda x : B \,.\, M)$ for B not convertible to A, as a restriction of $(\lambda x : \mathsf{A} \,.\, M)$. With this idea for an interpretation, we might think of rule $(\mathsf{A}\lambda)$ as making the system, which is based on the Church-style syntax, more like a system based on the Curry-style syntax. This idea for an interpretation suggests that our intended interpretation of A is as a type including all terms in all other types.

The system $\lambda S'$ as defined above does not include any postulates to formalize this interpretation: all we have in $\lambda S'$ is an alternative to adding domain-free abstraction terms to the Church-style syntax. If we wanted to add such a postulate, we might consider adding the following rule:

(AI) $\dfrac{\Gamma \vdash M : B}{\Gamma \vdash M : \mathsf{A}}$ *Condition:* B is any legal type.

It might appear that this significantly strengthens the system. However, unless we add an axiom of the form $\mathsf{A} : s$ for some sort s, it is impossible to prove that a type of the form $(\Pi x : \mathsf{A} \,.\, B)$ has any sort as its type, so a type of this form cannot be the type of a conclusion of (abstr$_{\mathrm{Ch}}$). Hence, it does not appear that adding rule (AI) would have that much effect on the system. In particular, appears that adding rules (AI) and ($\mathsf{A}\lambda$) to a system satisfying strong normalization would preserve strong normalization.

Another possibility would be to interpret A as the type ω that is the type of every term (or pseudoterm in a Church-style syntax, but with ω present as a type, every pseudoterm in a Church-style system is a legal term), which is the way this type is used in intersection type systems. Then instead of rule (AI), we would add the rule

(ωI) $\dfrac{}{\Gamma \vdash M : \omega}$ *Condition:* M is any pseudoterm of the Church-style semantics,

This would automatically give a type to every pseudoterm of the Church-style semantics. Since the Curry-style system with which we are starting is a PTS, whose postulates exclude a rule like (ωI), this really would strengthen the system. And this rule clearly does not satisfy strong normalization.

Note that adding postulates to interpret A as a type is not necessary to the interpretation of the Curry-style PTS in a Church-style system.

5 Open Questions

The following questions arise naturally from the above interpretations, but are so far unanswered:

1. In the Church-style syntax, we might consider adding η-contractions and also the contraction scheme

$$(\lambda x : B \,.\, M) \rhd (\lambda x : \mathsf{A} \,.\, M). \tag{27}$$

This is roughly equivalent to adding Curry-style abstractions to the Church-style syntax and then adding the contraction scheme

$$(\lambda x : B \,.\, M) \rhd (\lambda x \,.\, M).$$

In conjunction with this extended reduction, the typing rule $(\mathsf{A}\lambda)$ would preserve the Subject-Reduction Theorem. Furthermore, the contraction scheme (27) is the analogue for the Church-style syntax of a valid reduction in the Curry-style syntax under the interpretation of this paper. But would the Church-Rosser property hold for this reduction?

2. If we interpret A as the type ω as suggested above, then the normal form theorem fails. In intersection type systems with this type, it can be proved that any term that has a type in which ω does not occur has a normal form. Would this be true here?

3. Suppose that instead of PTSs we consider the more liberal versions of PTSs of [3]. How are the results of this paper affected?

References

[1] H. Barendregt. Lambda calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Oxford University Press, 1992.

[2] G. Barthe and M. H. Sørensen. Domain-free pure type systems. *Journal of Functional Programming*, 10(5):412–452, September 2000.

[3] M. Bunder and W. Dekkers. Pure type systems with more liberal rules. *Journal of Symbolic Logic*, 66:1561–1580, 2001.

[4] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[5] H. Curry, J. Hindley, and J. Seldin. *Combinatory Logic*, volume 2. North-Holland Publishing Company, Amsterdam and London, 1972.

[6] H. Geuvers. the church-rosser property for $\beta\eta$-reduction in typed λ-calculi. In *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 453–460. IEEE Computer Society, IEEE Computer Society Press, 1992.

[7] J. Hindley and J. Seldin. *Lambda-Calculus and Combinators, an Introduction.* Cambridge University Press, 2008.

[8] Z. Luo. ECC, an extended calculus of constructions. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science, June 1989, Asilomar, California, U.S.A.*, 1989.

[9] Z. Luo. *An Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, 1990.

[10] R. P. Nederpelt. *Strong Normalization in a Typed Lambda Calculus with Lambda Structured Types.* PhD thesis, Technical University of Eindhoven, 1973.

[11] G. Pottinger. Ulysses: Logical and computational foundations of the primitive inference engine. Technical Report TR 11-8, ORA Corporation., January 1988.

[12] G. Pottinger. A tour of the multivariate lambda calculus. In J. M. Dunn and A. Gupta, editors, *Truth or Consequences: Essays in Honor of Nuel Belnap*, pages 209–229. Kluwer Academic Publishers, Dordrecht, Boston, and London, 1990.

[13] G. Pottinger and J. P. Seldin. Interpreting Church-style typed λ-calculus in Curry-style type assignment. Former title, "Note on η-Reduction and Labelling Bound Variables in Typed λ-Calculus." Unpublished.

[14] J. Seldin. Progress report on generalized functionality. *Annals of Mathematical Logic*, 17:29–59, 1979.

[15] J. Seldin. On the proof theory of Coquand's calculus of constructions. *Annals of Pure and Applied Logic*, 83:23–101, 1997.

[16] S. van Bakel, L. Liquori, S. R. della Rocca, and P. Urzyczyn. Comparing cubes of typed and type assignment systems. *Annals of Pure and Applied Logic*, 86(3):267–303, 1997.