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

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

June 8, 2004

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. (This generalizes some unpublished work with Garrel Pottinger.) I will then show how to interpret in a system of the Church-style (a modified 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: 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 respec-

[*]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 talk was joint work with Garrel Pottinger.

tively. 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 assigns types to terms that already exist in the pure - la -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:

($\forall$-elimination)

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

($\forall$-introduction)

$$\frac{\Gamma \vdash M : (\forall \alpha \,.\, \sigma)}{\Gamma \vdash M : \sigma}$$

where in the latter rule, $\alpha \notin \mathrm{FV}(\Gamma)$.

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. To be historically accurate, I think it is better to identify the Curry-style with domain-free type systems.

In this paper, I propose to show how to interpret a system of each style in an appropriate system of the other. In one direction, the direction from Church-style to Curry-style, this paper extends previous work with Garrel Pottinger [8], which carried through the interpretation for three systems from the Barendregt cube: $\lambda_{\rightarrow}$, $\lambda 2$, $\lambda \mathrm{C}$, and its extension, the system ECC of Luo [4, 5].

2 Basic definitions

I will assume that the reader is familiar with the basic definitions and notation of [3]. The systems considered in this paper are defined from the pure syntax for pseudoterms

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

for a Church-style system or from the pure syntax

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

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

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

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, can be derived from the context 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 $s_1, s_2, s_3 \in \mathcal{S}$.

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

(axiom) If $c : s \in \mathcal{A}$,

$$\frac{}{\vdash\ c : s}$$

(start) If $s \in \mathcal{S}$ and $x \notin \mathrm{FV}(\Gamma)$,

$$\frac{\Gamma\ \vdash\ A : s}{\Gamma, x : A\ \vdash\ x : A}$$

(weakening) If $s \in \mathcal{S}$ and $x \notin \mathrm{FV}(\Gamma)$,

$$\frac{\Gamma\ \vdash\ M : B \quad \Gamma\ \vdash\ A : s}{\Gamma, x : A\ \vdash\ M : B}$$

(product) If $(s_1, s_2, s_3) \in \mathcal{R}$ and $x \notin \mathrm{FV}(\Gamma)$

$$\frac{\Gamma\ \vdash\ A : s_1 \quad \Gamma, x : A\ \vdash\ B : s_2}{\Gamma\ \vdash\ (\Pi x : A\ .\ B) : s_3}$$

(application)

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

(abstraction$_{\mathrm{Ch}}$) If $s \in \mathcal{S}$ and $x \notin \mathrm{FV}(\Gamma)$,

$$\frac{\Gamma, x : A\ \vdash\ M : B \quad \Gamma\ \vdash\ (\Pi x : A\ .\ B) : s}{\Gamma\ \vdash\ (\lambda x : A\ .\ M) : (\Pi x : A\ .\ B)}$$

(conversion) If $s \in \mathcal{S}$

$$\frac{\Gamma\ \vdash\ M : B \quad \Gamma\ \vdash\ B' : s \quad B =_\beta B'}{\Gamma\ \vdash\ M : B'}$$

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

(abstraction$_{\mathrm{Cu}}$) If $s \in \mathcal{S}$ and $x \notin \mathrm{FV}(\Gamma)$,

$$\frac{\Gamma, x : A\ \vdash\ M : B \quad \Gamma\ \vdash\ (\Pi x : A\ .\ B) : s}{\Gamma\ \vdash\ (\lambda x\ .\ M) : (\Pi x : A\ .\ B)}$$

3 Church-style to Curry-style

The basic idea of this interpretation is due to Garrel Pottinger [7], who proposed using a constant Label so that in the Curry-style syntax $(\lambda x : A\ .\ M)$ is an abbreviation for Label$A(\lambda x\ .\ M)$. By the analogy with (β_{Ch}), we will want

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

This suggests that Label should have the reduction rule

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

which, in turn, suggests that Label can be defined by

$$\mathsf{Label} \equiv \lambda xyz\ .\ yz.$$

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. $(\lambda x : A \,.\, M)^{\mathrm{Cu}} \equiv \mathsf{Label}A^{\mathrm{Cu}}(\lambda x \,.\, M^{\mathrm{Cu}})$,
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 1 *For every pseudoterm of the Church-style syntax,* $\mathrm{FV}(M^{\mathrm{Cu}}) = \mathrm{FV}(M)$.

Lemma 2 *For pseudoterms M and N of the Church-style syntax,* $([N/x]M)^{\mathrm{Cu}} \equiv [N^{\mathrm{Cu}}/x]M^{\mathrm{Cu}}$.

Lemma 3 *If*

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

then

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

For a pseudocontext

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

define

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

Recall that a *topsort* of a PTS is a sort s for which there is no sort s' such that $s : s' \in \mathcal{A}$.

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 as $C(\mathcal{S}_C, \mathcal{A}_C, \mathcal{R}_C)$ where $\mathcal{S}_C$ is obtained from $\mathcal{S}_\lambda$, $\mathcal{A}_C$ from $\mathcal{A}_\lambda$, and $\mathcal{R}_C$ from $\mathcal{R}_\lambda$ as follows:

1. If there is a topsort s which occurs in the first position of a rule in $\mathcal{R}_\lambda$, then a new sort s^T is added to $\mathcal{S}_C$ and for each such topsort s an axiom $s : s^T$ is added to $\mathcal{A}_C$, and

2. If there is a rule $(s_1, s_2, s) \in \mathcal{R}_\lambda$ of λS, then (s, s, s) and (s', s, s), where $s_1 : s' \in \mathcal{A}_C$, are added to $\mathcal{R}_C$. (Here s' may be s^T. If there is more than one sort s' with $s_1 : s' \in \mathcal{A}_C$, then any one may be chosen.)

Theorem 1 *If*

$$\Gamma \vdash M : A$$

in λS, then

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

in CS. If no new rules are added to those of $\mathcal{R}_\lambda$ to get $\mathcal{R}_C$, then the converse also holds.

The proof is by induction on the derivation of $\Gamma \vdash M : A$. The interesting case is for (abstraction), and the additions specified in clauses 1 and 2 of Definition 3 are used to show that Label has the type needed for this clause of the proof, which assures that if

$$\Gamma \vdash (\lambda x \,.\, M) : (\Pi x : A \,.\, B) \qquad (1)$$

in CS, then

$$\Gamma \vdash \mathsf{Label}A(\lambda x \,.\, M) : (\Pi x : A \,.\, B) \qquad (2)$$

is also derivable in CS.

Remark 2 The converse will fail if new rules are added to those of $\mathcal{R}_\lambda$ to get $\mathcal{R}_C$, since the underlying system is made stronger. New rules will have to be added for all the systems of Barendregt's λ-cube. However, no new rules need to be added in the case of Luo's ECC [4, 5]. ECC is really an extension of a PTS, but it is enough like a PTS that the above result holds.

Remark 3 Note that the inference from (2) to (1) follows by the Subject-Reduction Theorem [2, Theorem 17], since

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

This means that in CS, (1) and (2) are equivalent.

If subtyping is present, $(\lambda x\ .\ M)$ is, in a sense, a restriction of $(\lambda x\ .\ M)$ to the type A as domain. The equivalence of (1) and (2) means that 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, and Curry-style typing emphasizes this fact.

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 $(\lambda x\ .\ M)$. Since the latter abstract can be applied to *any* term as an argument, the needed type would have to be a type of all terms with types in the Church-style syntax. It would be possible to use the type ω that is often a part of intersection type systems, since every term has this type. However, a system with this type does not have the normal form property. An alternative is to postulate a constant A along with the rule

(AI) If B is any legal type,

$$\frac{\Gamma\ \vdash\ M : B}{\Gamma\ \vdash\ M : \mathsf{A}}$$

But this is still not enough. We also need the rule

($\mathsf{A}\lambda$)

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

This rule formalizes in the Church-style the inference in the Curry-style system from (2) to (1). It also turns out to be necessary for the proof of the Theorem 2 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 4 *If M is a term of the Curry-style syntax, then* $\mathrm{FV}(M^{\mathrm{Ch}}) = \mathrm{FV}(M)$.

Lemma 5 *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 6 *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 two typing rules (AI) and ($\mathsf{A}\lambda$) given above.

Note that the system $\lambda S'$ is not a PTS.

Theorem 2 *If*

$$\Gamma \vdash M : B$$

in CS, then

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

in $\lambda S'$, and conversely.

5 Open Questions

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

1. The above interpretation of the Curry-style system in a Church-style system leaves one problem open: how to deal with η-reduction. This is no problem in the Curry-style semantics, but in the Church-style semantics, CR fails for η-reduction, as the following example due to Nederpelt [6, 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.

   In the Church-style syntax, we might consider extending β_{Ch}-reduction by adding the contraction scheme

   $$(\lambda x : B \,.\, M) \rhd (\lambda x : \mathsf{A} \,.\, M). \qquad (3)$$

   In conjunction with this extended reduction, the typing rule ($\mathsf{A}\lambda$) would preserve the Subject-Reduction Theorem. Furthermore, the contraction scheme (3) is the analogue for the Church-style syntax of a valid reduction in the Curry-style syntax. But would the Church-Rosser property hold for this reduction?

2. It is possible to carry out the interpretation of the Curry-style system in a Church-style interpretation using the type ω of the intersection type discipline instead of A. Then instead of Rule (AI), we have

   (ωI) If M is any pseudoterm of the Church-style semantics,

   $$\overline{\Gamma \vdash M : \omega}$$

   The normal form theorem fails for a system with this typing rule. 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. Is it possible to find pairs of systems of the form $\lambda S'$ and CS such that the functions $-^{\mathrm{Cu}}$ and $-^{\mathrm{Ch}}$ interpret each system in the other without further change?

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] J. R. Hindley and J. P. Seldin. *Introduction to Combinators and λ-Calculus*. Cambridge University Press, 1986.

[4] 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.

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

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

[7] 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.

[8] 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.