On the Church-Rosser Property for Expressive Type Systems and its Consequences for their Metatheoretic Study *

Herman Geuvers†
Faculty of Mathematics and Computer Science
Technological University of Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Benjamin Werner‡
Department of Computer Science,
Cornell University,
Ithaca NY, 14 853, USA

Projet Coq,
INRIA–Rocquencourt BP 105,
78 153 LE CHESNAY cedex, France

Abstract

We consider two alternative definitions for the conversion rule in Pure Type Systems. We study the consequences of this choice for the metatheory and point out the related implementation issues. We relate two open problems by showing that if a PTS *allows the construction of a fixed point combinator, then Church-Rosser for $\beta\eta$-reduction fails. We present a new formalization of Russell's paradox in a slight extension of Martin-Löf's inconsistent theory with Type:Type and show that the resulting term leads to a fix-point construction. The main consequence is that the corresponding system is non-confluent. This example shows that in some typed λ-calculi, the Church-Rosser proof for the $\beta\eta$-reduction is not purely combinatorial anymore, as in pure λ-calculus, but relies on the normalization and thus the logical consistency of the system.*

1 Introduction

This paper deals with the syntactic study of expressive Type Systems, i.e. typed λ-calculi which can be used to model and formalize mathematical reasonning. We present new results concerning two open problems: the validity of the Church-Rosser property for Pure Type Systems with η-conversion and the existence of a fixed-point operator in inconsistent Type Theories, especially in Martin-Löf's original system with Type:Type.

While these results might seem quite specialized, we will show how they actually shed light on the complex links between three essential features of Type Systems; namely itemdepth ¿3 toodeep

the exact definition of the system, especially the choice of the *conversion* or *type equality rule*

the metatheoretic study of the system, i.e. the manner and the order of proof of the the system's fundamental properties (normalization, confluence, consistency ...)

the implementation issues, or more precisely, the possibility of building an efficient and realistic computer implementation of the considered theory.

There are several different ways to describe a system of typed λ calculus, like, for example, the polymorphic λ-calculus or the Calculus of Constructions. Each of these descriptions of the syntax has its own advantages and disadvantages, depending on what one wants to do with the system: implement it, study its denotational semantics or study its meta-theory, for example. It is important to know that all these different descriptions define the same system. (That is, they all have the same set of derivable typing judgements.) For a system like the polymorphic λ-calculus - which does not incorporate type dependency - this question is easily solved. For systems with dependent types, like the Calculus of Constructions or Martin-Löf style type theories, the question of equivalence of the different descriptions of the syntax is rather more subtle. Here we sketch the situation by giving two formulations of the collection of Pure Type Systems, the original one which we call *syntactical* and is better suited for implementation and syntactic meta-theory, and a version with a typed conversion rule which we call *semantical* because it is better suited for model construction and denotational semantics.

We will show how to prove the equivalence between the two versions of a system. The proof relies very heavily on the confluence of reduction. It is well-known that if one restricts oneself to β-conversion, confluence holds and hence the two presentations are equivalent. However, if one considers $\beta\eta$-reduction (and conversion), the situation is much more subtle because all known proofs of confluence of $\beta\eta$ for systems with dependent types rely on the normalization property. In other words, normalization has to be

*This research was partially supported by ESPRIT Basic Research Action "TYPES".

†herman@info.win.tue.nl
‡Benjamin.Werner@inria.fr

proven independently, and *before* confluence. And this makes the already complicated normalization proof much more tedious. Furthermore, for non-normalizing systems, confluence of $\beta\eta$ is certainly not immediate and, as we will show, may even be not true. This last point is important, because it indicates that in the general case there is no *combinatorial* proof of the Church-Rosser property which we could carry on independently of the *logical* normalization property.

To prove this last point, we start by showing that, if a typed λ-calculus (of a certain rather general form) allows the construction of a fixed point operator, then confluence for $\beta\eta$-reduction cannot hold. It is not yet known whether one can type a fixed point operator in any Pure Type System, but we give a small and natural extension of the system in which 'Type' is itself a type, where a fixed point combinator can be typed. To do so, we start by defining what it means to formalize Russell's paradox in a typed λ-calculus and we give a necessary condition for this formalization to yield a typing of a fixed point operator. We finally propose a new formalization of Russell's paradox which fullfills that condition. Hence, for the considered system, $\beta\eta$-reduction is not confluent.

2 Syntactical versus Semantical versions of typed λ calculi

We now describe the syntactical version of *Pure Type Systems*, as it can be found in [Barendregt 1992], [Geuvers and Nederhof 1991] and the variant with a $\beta\eta$-conversion rule. For these systems the Church-Rosser property (for $\beta\eta$-reduction) has been discussed in [Geuvers 1992] (and proved for Pure Type Systems that are functional and normalizing.) This Church-Rosser property will play an important role in the proof of equivalence of the syntactical and the semantical versions of Pure type Systems. The typing rules of a Pure Type System define how to derive judgements of the form $\Gamma \vdash M : A$, where Γ is a sequence of declarations of variables to types and M and A are terms. Usually this is done by first giving the set of so called *pseudoterms* over a specific base set, from which the typing rules then select the *typable* (or *legal*) terms. For $\mathcal{S}$ some set, the set of pseudoterms over $\mathcal{S}$, T, is defined by

$$\mathsf{T} ::= \mathcal{S} \mid \mathsf{Var} \mid (\Pi \mathsf{Var}{:}\mathsf{T}.\mathsf{T}) \mid (\lambda \mathsf{Var}{:}\mathsf{T}.\mathsf{T}) \mid \mathsf{TT},$$

where Var is a countable set of expressions, called variables. (The dependency of T on $\mathcal{S}$ is usually left implicit.) Both Π and λ bind variables and hence we have the usual notions of *free variable* and *bound variable*. On T we have the usual notions of β-reduction ($\longrightarrow_\beta$) and η-reduction ($\longrightarrow_\eta$). We adopt from the untyped λ calculus the conventions of denoting the transitive reflexive closure of $\longrightarrow_\beta$ by $\twoheadrightarrow_\beta$ and the transitive symmetric closure of $\twoheadrightarrow_\beta$ by $=_\beta$ (and similar for $\longrightarrow_\eta$ and $\longrightarrow_{\beta\eta} \equiv \longrightarrow_\beta \cup \longrightarrow_\eta$).

Definition 2.1 *A* Pure Type System with $\beta\eta$-conversion *($\mathsf{PTS}_{\beta\eta}$) is given by a set $\mathcal{S}$, a set $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ and a set $\mathcal{R} \subset \mathcal{S} \times \mathcal{S} \times \mathcal{S}$. The* PTS *that is given by $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ is denoted by $\lambda_{\beta\eta}(\mathcal{S}, \mathcal{A}, \mathcal{R})$ and is the typed lambda calculus with the following deduction rules.*

$$(sort) \quad \vdash s_1 : s_2 \quad \text{if } (s_1, s_2) \in \mathcal{A}$$

$$(var) \quad \frac{\Gamma \vdash A : s}{\Gamma, x{:}A \vdash x : A}$$

$$(weak) \quad \frac{\Gamma \vdash A : s \quad \Gamma \vdash M : C}{\Gamma, x{:}A \vdash M : C}$$

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

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

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

$$(conv_{\beta\eta}) \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} \quad A =_{\beta\eta} B$$

In the rules (var) and (weak) it is always assumed that the newly declared variable is fresh, that is, it has not yet been declared in Γ. If $s_2 \equiv s_3$ in a triple $(s_1, s_2, s_3) \in \mathcal{R}$, we write $(s_1, s_2) \in \mathcal{R}$. The equality in the conversion rule ($conv_{\beta\eta}$) is the $\beta\eta$-equality on the set of pseudoterms T.
The elements of $\mathcal{S}$ are called sorts, *the elements of $\mathcal{A}$ (usually written as $s_1 : s_2$) are called* axioms *and the elements of $\mathcal{R}$ are called* rules.
A Pure Type System with β-conversion *(PTS_β) is also given by a triple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$, now denoted by $\lambda_\beta(\mathcal{S}, \mathcal{A}, \mathcal{R})$. The only difference with a $\mathsf{PTS}_{\beta\eta}$ is that a PTS_β has a β-conversion rule:*

$$(conv_\beta) \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} \quad A =_\beta B$$

In [Geuvers 1993] a lot of meta-theoretic properties are proved for these $\mathsf{PTS}_{\beta\eta}$s; in particular:

Proposition 2.2 (Subject Reduction for β, $\mathbf{SR}_\beta$) *If $\Gamma \vdash M : A$ and $M \twoheadrightarrow_\beta M'$ then $\Gamma \vdash M' : A$.*

An important subclass of all PTSs is the class of *functional* ones.

Definition 2.3 *A* PTS $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ *is* functional *if the relation $\mathcal{A}$ is a function from $\mathcal{S}$ to $\mathcal{S}$ and the relation $\mathcal{R}$ is a function from $\mathcal{S} \times \mathcal{S}$ to $\mathcal{S}$. (That is, if $s : s'$, $s : s'' \in \mathcal{A}$, then $s' \equiv s''$ and if $(s_1, s_2, s_3), (s_1, s_2, s_3') \in \mathcal{R}$ then $s_3 \equiv s_3'$.)*

Definition 2.4 (Strengthening)
We say that strengthening *holds for a $\mathsf{PTS}_{\beta\eta}$, if when $\Gamma_1, x{:}A, \Gamma_2 \vdash M : B$ and $x \notin FV(\Gamma_2, M, B)$, then $\Gamma_1, \Gamma_2 \vdash M : B$.*

The meta-theory for $\mathsf{PTS}_{\beta\eta}$s is not yet understood in its full generality. For example, it is not clear whether *strengthening* and its consequence, *subject reduction for* η, hold for all these systems. However the following condition is sufficient for proving these properties for most well-known systems:

Definition 2.5 *A* $\mathsf{PTS}_{\beta\eta}$ *satisfies* $\beta\eta$-preservation of sorts *if* $\Gamma \vdash A : s$, $\Gamma \vdash B : s'$ *and* $A =_{\beta\eta} B$ *together imply that* $s \equiv s'$.

Proposition 2.6 (Subject Reduction for η, $\mathbf{SR}_\eta$) *For a* $\mathsf{PTS}_{\beta\eta}$*s verifying 2.5, if* $\Gamma \vdash M : A$ *and* $M \twoheadrightarrow_\eta M'$ *then* $\Gamma \vdash M' : A$.

It can easily be shown that all systems of Barendregt's cube, including the Calculus of Constructions, more generally all functional normalizing PTSs, as well as the non-normalizing systems $\lambda U^-, \lambda U$ and $\lambda *$ verify 2.5 and thus subject reduction. As already pointed out, the main difficulty is the Church-Rosser property which, up to now, can only be proven for normalizing systems [Geuvers 1992]:

Theorem 2.7 (Church-Rosser for $\beta\eta$) *In a functional and normalizing* $\mathsf{PTS}_{\beta\eta}$, *if* $\Gamma \vdash M, N : A$ *and* $M =_{\beta\eta} N$, *then* $\exists P[M \twoheadrightarrow_{\beta\eta} P,\ N \twoheadrightarrow_{\beta\eta} P]$. *(By subject reduction, this* P *is then also of type* A *in* Γ.*)*

It should be clear that Church-Rosser is a very desirable property for a $\mathsf{PTS}_{\beta\eta}$. As a matter of fact, Church-Rosser together with subject reduction imply that the system in itself is *sound*:

Definition 2.8 *A* $\mathsf{PTS}_{\beta\eta}$ *is* sound *if the following holds. If* $\Gamma \vdash A, B : s$ *and* $A =_{\beta\eta} B$, *then there is a* $\beta\eta$*-expansion/reduction path from* A *to* B *that remains inside the set of typable terms**.

In a sense, soundness of a $\mathsf{PTS}_{\beta\eta}$ says that, although the system was defined by making use of the set of pseudoterms, the pseudoterms that are not typable do not play an essential role in the system. Of course, this is what we want: the set T was just introduced for obtaining a rather smooth definition of the system, but we really do not want to attach any meaning to the pseudoterms that are not typable. This also implies that one might construct a denotational semantics for the system, without having to give an interpretation for those terms that are not typable. Note however that normalization proofs are much easier if one can disposes of the soundness condition in the first place.

For the reasons mentioned above, and because soundness basically says that the relation $=_{\beta\eta}$ can be read as a *typed* equality, it is very tempting to impose soundness in the definition of the system. This leads us to the semantical version of PTSs:

*A $\beta\eta$-expansion/reduction path from A to B is a sequence of terms $C_0, C_2, \ldots, C_n$ such that $C_0 \equiv A$, $C_n \equiv B$ and $\forall i < n[C_i \longrightarrow_{\beta\eta} C_{i+1} \vee C_{i+1} \longrightarrow_{\beta\eta} C_i]$.

Definition 2.9 *The* semantical *version of a* PTS $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$) *has the following typing rules. First we have the rules (sort), (weak), (var),* (Π), (λ), *and (app) as for ordinary* PTS*s. (To denote that we are in a semantical version we write* $\vdash_=$ *in the rules.) The conversion rule is a* typed *one:*

$$(conv^{=}_{\beta\eta}) \quad \frac{\Gamma \vdash_= M : A \quad \Gamma \vdash_= A = B : s}{\Gamma \vdash_= M : B}$$

The judgement $\Gamma \vdash_= A = B : s$ *is generated by the rules*

$$(\beta) \quad \frac{\Gamma \vdash_= \lambda x{:}A.M : \Pi x{:}C.D \quad \Gamma \vdash_= N : C}{\Gamma \vdash_= (\lambda x{:}A.M)N = M[N/x] : D[N/x]}$$

$$(\eta) \quad \frac{\Gamma \vdash_= \lambda y{:}A.My : B}{\Gamma \vdash_= \lambda y{:}A.My = M : B} \quad \text{if } y \notin FV(M).$$

There are rules (refl), (sym) and (trans) to make it an equivalence relation and furthermore there are rules ($\Pi_=$), ($\lambda_=$) *and* ($app_=$) *to make the equality compatible with abstraction and application. Finally, the rules* ($weak_=$) *and* ($conv_=$) *make the equality relation is closed under extensions of the context and replacing the type by an equal type. We give some of these rules as illustration. (For the rule* ($\Pi_=$), *assume that* $(s_1, s_2, s_3) \in \mathcal{R}$.*)*

$$(refl) \quad \frac{\Gamma \vdash_= M : A}{\Gamma \vdash_= M = M : A}$$

$$(\Pi_=) \quad \frac{\Gamma \vdash_= A = A' : s_1 \quad \Gamma, x{:}A \vdash_= B = B' : s_2}{\Gamma \vdash_= \Pi x{:}A.B = \Pi x{:}A'.B' : s_3}$$

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

$$(conv_=) \quad \frac{\Gamma \vdash_= M = M' : A \quad \Gamma \vdash_= A = B : s}{\Gamma \vdash_= M = M' : B}$$

The main problem with these systems, is that it seems much more difficult to prove some very basic properties, especially subject-reduction. More precisely, there is no known combinatorial proof of subject-reduction (especially for η). This last property is quite important in the case of an implementation, when we need an efficient algorithm for type-checking. We first describe one method, which is to prove the equivalence with the syntactical version after having done the meta-theory of the latter. We discuss alternatives at the end of this section.

The main equivalence result is:

Theorem 2.10 *For* S *a* $\mathsf{PTS}_{\beta\eta}$ *for which strengthening and Church-Rosser hold,*

$$\left.\begin{array}{l}\Gamma \vdash M : A \\ \Gamma \vdash N : A \\ M =_{\beta\eta} N\end{array}\right\} \Longleftrightarrow \left\{\begin{array}{l}\Gamma \vdash_= M : A \\ \Gamma \vdash_= N : A \\ \Gamma \vdash_= M = N : A.\end{array}\right.$$

It is this result which validates for example the model construction made by Streicher [Streicher 1991, Hofmann and Streicher] for the Calculus of Constructions. The proof, even if not too surprising is not totally trivial, and relatively long. Therefore we only give an outline of the proof and omit most of the details.

The proof of the implication from right to left is straightforward by proving the following lemma.

Lemma 2.11 *If strengthening holds for a* $\mathsf{PTS}_{\beta\eta}$*, then*

$$\begin{aligned} \Gamma \vdash_= M : A &\implies \Gamma \vdash M : A, \\ \Gamma \vdash_= M = N : A &\implies \Gamma \vdash M, N : A \ \& \ M =_{\beta\eta} N. \end{aligned}$$

Proof Note that if $\Gamma \vdash_= M = N : A$, then $M =_{\beta\eta} N$ (as pseudoterms). Furthermore, if $\Gamma_1, x{:}B, \Gamma_2 \vdash N : C$ and $\Gamma_1 \vdash B' : s$ with $B =_{\beta\eta} B'$, then $\Gamma_1, x{:}B', \Gamma_2 \vdash N : C$. The proof is now by simultaneous induction on the derivation. (Strengthening is only needed to prove the soundness of the (η)-rule, especially to show that, if $\Gamma \vdash \lambda y{:}A.My : B$, then $\Gamma \vdash M : B$.) $\Box$

A proof of the implication from left to right in Theorem 2.10 amounts to constructing a proof of $\Gamma \vdash_= M = N : A$ from the equality $M =_{\beta\eta} N$, using the fact that M and N are typable. This means that one has to find a path from M to N which remains inside the typable terms. It should be clear that subject reduction and confluence for the syntactical version are essential properties for this proof. Therefore, we can only show the equivalence for Pure Type Systems for which strengthening (and hence subject reduction for η) and confluence for $\beta\eta$ hold.

To prove the equivalence of S and $S_=$ we introduce a slightly different version of S, which is better suited to show the inclusion of S in $S_=$.

Definition 2.12 *For* S *a* $\mathsf{PTS}_{\beta\eta}$*, the system* S_+ *is defined by replacing the (app)-rule by the* restricted application rule *(app*$^+$*) and the (conv*$_{\beta\eta}$*)-rule by the* one-step-conversion rule *(1-conv*$_{\beta\eta}$*). These two rules are as follows.*

$$(app^+)\ \frac{\Gamma \vdash_+ M : \Pi x{:}A.B \quad \Gamma \vdash_+ N{:}A \quad \Gamma \vdash_+ B[N/x]{:}s}{\Gamma \vdash_+ MN : B[N/x]}$$

$$(1\text{-}conv_{\beta\eta})\ \frac{\Gamma \vdash_+ M : A \quad \Gamma \vdash_+ B : s}{\Gamma \vdash_+ M : B} \quad A \stackrel{1}{=}_{\beta\eta} B$$

where $A \stackrel{1}{=}_{\beta\eta} B$ *stands for one-step-conversion, that is,* $A \longrightarrow_{\beta\eta} B$ *or* $B \longrightarrow_{\beta\eta} A$*.*

It is not difficult to see that the restricted application rule does not really put any restriction on the system. (If $\Gamma \vdash \Pi x{:}A.B : s$ and $\Gamma \vdash N : A$, then $B[N/x]$ is always a typable term.) It is mainly there to take care that for every derivation of $\Gamma \vdash M : A$ there is a shorter derivation of $\Gamma \vdash A : s$ for some sort s, if A is not itself a sort. We state this lemma explicitly. (The proof is by induction on derivations.)

Sublemma 2.13 *Suppose* $A \notin \mathcal{S}$*. If there is a derivation of* $\Gamma \vdash_+ M : A$ *of length* k*, then there is a derivation of* $\Gamma \vdash_+ A : s$ *of length* $\leq k$ *for some* $s \in \mathcal{S}$*.*

The one-step-conversion rule can be a real restriction if $\mathrm{CR}_{\beta\eta}$ is not true in the system. (In S_+, two types have the same inhabitants only if they are equal via a path through the typable terms.) We shall therefore assume in the following that $\mathrm{CR}_{\beta\eta}$ holds for the system that we are considering.

Lemma 2.14 *If* S *is a* $\mathsf{PTS}_{\beta\eta}$ *for which strengthening holds and* $CR_{\beta\eta}$*, then*

$$\Gamma \vdash M : A \iff \Gamma \vdash_+ M : A.$$

Proof Both implications are by induction on the derivation. The implication from right to left is straightforward. The implication from left to right really uses the fact that $S \models \mathrm{CR}_{\beta\eta}$ and that subject reduction holds for β- and η-reduction. (SR_η follows from strengthening. See [Geuvers 1993] for details.) $\Box$

Now, if we prove that S_+ is a subsystem of $S_=$ we are done. This is the most difficult part of the proof of equivalence of S and $S_=$ (Theorem 2.10). To give the proof we need the notion of *simultaneous reduction*, denoted by $\Rightarrow_{\beta\eta}$.

Definition 2.15 *Simultaneous* $\beta\eta$*-reduction,* $\Rightarrow_{\beta\eta}$*, is defined on the set of pseudoterms* T *by* $x \Rightarrow_{\beta\eta} x$*, and if* $M \Rightarrow_{\beta\eta} M'$*,* $N \Rightarrow_{\beta\eta} N'$ *and* $A \Rightarrow_{\beta\eta} A$*, then* $MN \Rightarrow_{\beta\eta} M'N'$*,* $\lambda x{:}A.M \Rightarrow_{\beta\eta} \lambda x{:}A'.M'$ *and* $(\lambda x{:}A.M)N \Rightarrow_{\beta\eta} M'[N'/x]$*.*

We have the following (expected) properties for a simultaneous reduction:

$$\begin{aligned} \text{If } M \Rightarrow_{\beta\eta} N, &\quad \text{then } M \twoheadrightarrow_{\beta\eta} N, \\ \text{If } M \longrightarrow_{\beta\eta} N, &\quad \text{then } M \Rightarrow_{\beta\eta} N. \end{aligned}$$

The simultaneous reduction $\Rightarrow_{\beta\eta}$ immediately extends to simultaneous reduction on contexts by letting $x_1{:}A_1, \ldots, x_n{:}A_n \Rightarrow_{\beta\eta} x_1{:}A'_1, \ldots, x_n{:}A'_n$ if $A_i \Rightarrow_{\beta\eta} A'_i$ for all $i \leq n$.

Lemma 2.16 *For* S *a* $\mathsf{PTS}_{\beta\eta}$ *for which strengthening holds,*

$$\left.\begin{array}{r} \Gamma \vdash_+ M : A \\ \Gamma \Rightarrow_{\beta\eta} \Gamma' \\ M \Rightarrow_{\beta\eta} N \end{array}\right\} \implies \left\{\begin{array}{l} \Gamma' \vdash_= M : A \\ \Gamma' \vdash_= N : A \\ \Gamma' \vdash_= M = N : A. \end{array}\right.$$

Proof The proof is by induction on the *length* of a derivation of $\Gamma \vdash_+ M : A$, using Sublemma 2.13. $\Box$

This completes the proof of Theorem 2.10.

As mentioned above, the direct study of semantical systems is less well understood. Altenkirch [Altenkirch 1993] proves decidability of typing for a semantical Calculus of Constructions, but

because he gives no proof of SR_η, the resulting algorithm has to check the correctness of each reduction to the detriment of efficiency[2]. This might lso be problematic for justifying proof-search algorithms, like [Dowek 1993a]. Another possibility is the approach of [Coquand 1991], where the reducibility method is used not only for proving normalization, but also SR_η, which allows the derivation of an efficient type-checking algorithm; the counter-part is that the reducibility proof is more complicated. In this case, subject-reduction becomes a logical property. It would be interesting to apply this method to a more complex system like the Calculus of Constructions.

3 Fixed point operator and Confluence for $\beta\eta$

It is not at all obvious that all PTSs (the syntactical versions) satisfy confluence for $\beta\eta$. If one considers β-reduction only, the problem of confluence is relatively easy, because we have confluence for β-reduction on the pseudoterms (the set T), that is for all $M, N, P \in \mathsf{T}$, if $M \twoheadrightarrow_\beta N$ and $M \twoheadrightarrow_\beta P$, then there is a Q such that $N \twoheadrightarrow_\beta Q$ and $P \twoheadrightarrow_\beta Q$. Then, by subject reduction for β (saying that, if $\Gamma \vdash M : A$ and $M \twoheadrightarrow_\beta N$, then $\Gamma \vdash N : A$), one concludes that confluence holds for the typed terms:

$$\left.\begin{array}{r}\Gamma \vdash M, N : A \\ M =_\beta N\end{array}\right\} \Longrightarrow \exists Q \left\{\begin{array}{l}\Gamma \vdash Q : A \\ M \twoheadrightarrow_\beta Q \\ N \twoheadrightarrow_\beta Q\end{array}\right.$$

Notation 3.1 *This latter property is abbreviated to* CR_β*, where* $\mathrm{CR}_{\beta\eta}$ *of course denotes the same property for the* $\beta\eta$ *case. We write* $S \models CR_{\beta\eta}$ *to denote that* $\beta\eta$*-reduction is confluent in a Type System* S *and* $S \not\models CR_{\beta\eta}$ *to denote the negation of that.*

When one considers $\beta\eta$-reduction, this proof breaks down on the fact that $\beta\eta$-reduction is not Church-Rosser on T: if $A \neq_{\beta\eta} B$, then $\lambda x{:}A.(\lambda y{:}B.y)x \longrightarrow_\beta \lambda x{:}A.x$ and $\lambda x{:}A.(\lambda y{:}B.y)x \longrightarrow_\eta \lambda y{:}B.y$ and these two terms have no common reduct [3]. In [Geuvers 1992] this counter-example was used in a positive way, in the form of the following lemma.

[2] Actually, Altenkirch mainly uses a slight variation of this presentation of semantical systems, but which remains quite similar. Other possible presentations include the one of [Streicher 1991] (semantical) and of [Dowek et al. 1993] (syntactical). In all these cases however, it remains quite easy to show the equivalence between the different syntactical (resp. semantical) presentations.

[3] A tempting possibility is therefore to use a presentation based on untyped abstractions ($\lambda x.t$ instead of $\lambda x : T.t$). In [Giannini et al. 1993] the 'cube of typed λ-calculi' (containing the Calculus of Constructions) is defined in this *Curry-style*. Actually such a presentation makes some sense, but jeopardizes the decidability of typing (see [Dowek 1993b]). It is therefore not very useful. Furthermore, it allows terms to be typed that should not be typable, see [Liquori et al. 93] for an example.

Lemma 3.2 (Domain Lemma) *If* $C[\lambda x{:}A.M]$ *and* B *are in* T *(i.e.* C *is a pseudoterm with subterm* $\lambda x{:}A.M$*), then*

$$C[\lambda x{:}A.M] =_{\beta\eta} C[\lambda x{:}B.M]$$

Proof $C[\lambda y{:}B.(\lambda x{:}A.M)y] \longrightarrow_\beta C[\lambda x{:}B.M]$ and $C[\lambda y{:}B.(\lambda x{:}A.M)y] \longrightarrow_\eta C[\lambda x{:}A.M]$ where y is any variable not occurring free in A or M. □

We will now show that, if a $\mathsf{PTS}_{\beta\eta}$ (satisfying some extra requirements) has a fixed point operator, then confluence for $\beta\eta$ does not hold. The proof of this fact uses the Domain Lemma 3.2. Let, for the rest of this section, S be a $\mathsf{PTS}_{\beta\eta}$ that contains a specific sort $*$ and furthermore axioms $s_0 : *$ and $* : s_1$ and rules $(*,*)$, $(s_1,*)$, $(*,s_0)$ and (s_0,s_0) for some sorts s_0, s_1. We now want to look at the situation that S has a fixed point operator.

Definition 3.3 *S has a fixed point operator if there is a closed term* Fix *of type* $\Pi\alpha{:}*.(\alpha{\to}\alpha){\to}\alpha$ *such that* $\mathsf{Fix}f = f(\mathsf{Fix}f)$.

Theorem 3.4 *If S has a fixed point operator, then* $S \not\models CR_{\beta\eta}$.

Proof Let $\mathsf{Fix} : \Pi\alpha{:}*.(\alpha{\to}\alpha){\to}\alpha$ be the fixed point operator in S. Let furthermore β, γ and δ be distinct (type) variables. (So $\beta, \gamma, \delta : s_0$) Now, define

$$\begin{array}{rcl} A & \equiv & \mathsf{Fix}s_0(\lambda\epsilon : s_0.\epsilon{\to}(\beta{\to}\beta){\to}\delta) : s_0, \\ B & \equiv & \mathsf{Fix}s_0(\lambda\epsilon : s_0.\epsilon{\to}(\gamma{\to}\gamma){\to}\delta) : s_0. \end{array}$$

Then $A =_\beta A{\to}(\beta{\to}\beta){\to}\delta$ and $B =_\beta B{\to}(\gamma{\to}\gamma){\to}\delta$. Now define

$$\begin{array}{rcl} M & \equiv & \lambda y{:}A.yy : A{\to}(\beta{\to}\beta){\to}\delta(= A), \\ N & \equiv & \lambda y{:}B.yy : B{\to}(\gamma{\to}\gamma){\to}\delta(= B). \end{array}$$

To construct two $\beta\eta$-convertible terms of the same type that have no common reduct, define

$$\begin{array}{rcl} M_0 & \equiv & MM(\lambda z{:}\beta.z) : \delta, \\ N_0 & \equiv & NN(\lambda z{:}\gamma.z) : \delta. \end{array}$$

The terms M_0 and N_0 only differ in their domains (the type labels in the λ abstractions), for the rest they are exactly the same. Hence M_0 and N_0 are $\beta\eta$-convertible, due to the Domain Lemma 3.2. Furthermore, the part $\lambda z{:}\beta.z$ in M_0 and the part $\lambda z{:}\gamma.z$ in N_0 are not affected by a possible reduction (they remain in position), so M_0 and N_0 have no common reduct. □

Remark 3.5 *The above result holds also for* any extension of S. *This means in particular, that if we add arbitrary recursive types to a polymorphic* λ*-calculus (like Constructions,* F_ω *or even* F*) we immediately break* $CR_{\beta\eta}$ [4].

[4] The system F does not strictly respect the above conditions, but adding a primitive fix-point contruct over types makes the sort s_0 superfluous in the proof. More precisely, in F we have $* : s_1$ and as rules $(*,*)$ and $(s_1,*)$. Then for $\beta, \gamma, \delta : *$, one can define $A \equiv \mu\alpha{:}*.\alpha{\to}(\beta{\to}\beta){\to}\delta) : *$ and $B \equiv \mu\alpha{:}*.\alpha{\to}(\gamma{\to}\gamma){\to}\delta) : *$, where μ represents the primitive fixed point construct.

In case the system S has type dependency, it is also possible to construct typable terms Q, Q_1 and Q_2 such that $Q \longrightarrow_\beta Q_1$, $Q \longrightarrow_\eta Q_2$ and Q_1 and Q_2 have no common reduct. Say we have the rule (s_0, s_1). Take M_0 and N_0 as in the proof of Theorem 3.4 and let P be a variable of type $\delta \to *$. Then $\lambda x{:}PM_0.(\lambda y{:}PN_0.y)x : PM_0 \to PM_0$ and $\lambda x{:}PM_0.(\lambda y{:}PN_0.y)x \longrightarrow_\beta \lambda x{:}PM_0.x$ and $\lambda x{:}PM_0.(\lambda y{:}PN_0.y)x \longrightarrow_\eta \lambda y{:}PN_0.y$.

One example of a system for which Theorem 3.4 applies is $\lambda *$ where the universe of types is itself a type. (Take both s_0 and s_1 to be $*$.) It is not known whether $\lambda *$ has a fixed point operator, hence we can not conclude from Theorem 3.4 that confluence for $\beta\eta$ is false. In the next section we discuss a slight extension of $\lambda *$ and show how to type a fixed point operator in it.

4 Fixed point operator and the Russell paradox

Our aim is now to exhibit a fixed point combinator in some Type Theory as "reasonable" as possible, in order to show that the Church-Rosser property relies on normalization in the general case. A fixed point combinator can be used to inhabit any type, and thus corresponds to the formalization of some mathematical paradox. There are various studies of the computational behavior of paradoxical terms, especially in $\lambda *$; [Meyer and Reinhold 1986] first thought they could build a fixed point on the top of the proof of Girard's paradox (see [Coquand 1986]), but [Howe 87] then showed that this only lead to a weaker class of so-called *looping combinators*. This is a family of combinators $Y_0, Y_1, Y_2, \ldots$, all of type $\Pi\alpha{:}*.(\alpha \to \alpha) \to \alpha$ such that

$$Y_n \alpha f = f(Y_{n+1} \alpha f).$$

The work of [Howe 87] was later extended by [Coquand and Herbelin 1992], who gave a general procedure for turning a proof of inconsistency into a looping combinator. So far nobody has succeeded in exhibiting a real fixed point operator.

We now continue by pointing out that Russell's paradox, because of its "flip-flop" structure, is a good candidate for a formalization of a fixed point operator. At the end of this section we shall give another criterion for the existence of a fixed point combinator, which is the existence of an *enumerator*.

Definition 4.1 *Let S be a pure type system that contains a specific sort $*$ (the sort of propositions) and furthermore an axiom $* : s_1$ and rules $(*, *)$, $(s_1, *)$ and (s_1, s_1) for some sort s_1 (the sort of domains).*

1. *A* representation of the Russell paradox in S *consists of a triple $(V, \epsilon, \mathsf{comp})$ such that*

$$\begin{aligned} &\vdash \quad V : s_1, \\ &\vdash \quad \epsilon : V \to V \to *, \\ &\vdash \quad \mathsf{comp} : (V \to *) \to V, \end{aligned}$$

such that the type

$$\Pi P{:}V \to * .\Pi x{:}V.Px \leftrightarrow \epsilon x(\mathsf{comp}P)$$

is inhabited in the empty context.

2. *A* strong representation of the Russell paradox in S *consists of a tuple $(V, \epsilon, \mathsf{comp}, F, G)$ such that $(V, \epsilon, \mathsf{comp})$ is a representation of the Russell paradox in S and*

$$\begin{aligned} P{:}V \to *, x{:}V \quad &\vdash \quad F : \epsilon x(\mathsf{comp}P) \to Px, \\ P{:}V \to *, x{:}V \quad &\vdash \quad G : Px \to \epsilon x(\mathsf{comp}P), \end{aligned}$$

such that

$$F(Gz) = z \text{ for arbitrary } z.$$

So F and G together establish the proof of $\Pi P{:}V \to * .\Pi x{:}V.Px \leftrightarrow \epsilon x(\mathsf{comp}P)$.

In this definition and also in the rest of this paper we freely use the logical connectives $\leftrightarrow$ and $\&$ between types. They are formally defined by letting $A \,\&\, B \equiv \Pi\alpha{:}*.(A \to B \to \alpha) \to \alpha$, $A \leftrightarrow B \equiv (A \to B) \,\&\, (B \to A)$. It is well-known that the logical rules for the connectives can be obtained from these definitions.

In all the following, let α be a variable of type $*$; we write $\overset{\alpha}{\neg} T$ for $T \to \alpha$. The intuition is that, to show inconsistency, it suffices to construct a term of type α in the context $\alpha : *$. This is done by 'proving' both T and $\overset{\alpha}{\neg} T$ for some type T. Furthermore, to construct a fixed point combinator it suffices to construct a term M of type α in the context $\alpha : *, f : \alpha \to \alpha$ with the property that $M = fM$.

The following lemma motivates Definition 4.1:

Lemma 4.2 *If $(V, \epsilon, \mathsf{comp})$ is a representation of the Russell paradox in S, then there is a closed term R of type V in S for which both $\overset{\alpha}{\neg}(\epsilon RR)$ and $\overset{\alpha}{\neg}\overset{\alpha}{\neg}(\epsilon RR)$ hold (in the context $\alpha : *$.)*

Proof Let $(V, \epsilon, \mathsf{comp})$ be a representation of the Russell paradox in S. So, $\Pi P{:}V \to *.\Pi x{:}V.Px \leftrightarrow \epsilon x(\mathsf{comp}P)$ is inhabited. Define Russell's paradoxical "set" (the set of all elements that do not contain themself as an element) as:

$$R \equiv \mathsf{comp}(\lambda x{:}V.\ \overset{\alpha}{\neg}(\epsilon xx)) : V.$$

Then $\epsilon RR \equiv \epsilon R(\mathsf{comp}(\lambda x{:}V.\ \overset{\alpha}{\neg}(\epsilon xx)))$, which implies $(\lambda x{:}V.\ \overset{\alpha}{\neg}(\epsilon xx))R$, which is equal to $\overset{\alpha}{\neg}(\epsilon RR)$. Hence, ϵRR implies $\overset{\alpha}{\neg}(\epsilon RR)$, and so $\overset{\alpha}{\neg}(\epsilon RR)$ holds. Now, $\overset{\alpha}{\neg}(\epsilon RR)$ is equal to $(\lambda x{:}V.\ \overset{\alpha}{\neg}(\epsilon xx))R$, which implies ϵRR, so $\overset{\alpha}{\neg}\overset{\alpha}{\neg}(\epsilon RR)$ holds. □

The Lemma immediately implies that a system that has a representation of the Russell paradox is inconsistent. It is interesting to see what a term M with $\alpha : * \vdash M : \alpha$ would look like in that case. Let therefore $(V, \epsilon, \mathsf{comp})$ be a representation of the Russell paradox in S with the terms F and G such that

$$\begin{aligned} P{:}V \to *, x{:}V \quad &\vdash \quad F : \epsilon x(\mathsf{comp}P) \to Px, \\ P{:}V \to *, x{:}V \quad &\vdash \quad G : Px \to \epsilon x(\mathsf{comp}P). \end{aligned}$$

To keep our terms short we define F_0 and G_0 by

$$\begin{aligned} F_0 &\equiv F[\lambda x{:}V.\,\overset{\alpha}{\neg}(\epsilon x x)/P, R/x] : (\epsilon RR)\to\overset{\alpha}{\neg}(\epsilon RR), \\ G_0 &\equiv G[\lambda x{:}V.\,\overset{\alpha}{\neg}(\epsilon x x)/P, R/x] : \overset{\alpha}{\neg}(\epsilon RR)\to(\epsilon RR). \end{aligned}$$

Then terms of type $\overset{\alpha}{\neg}(\epsilon RR)$ and $\overset{\alpha}{\neg}\overset{\alpha}{\neg}(\epsilon RR)$ are

$$\begin{aligned} \alpha:* &\vdash \lambda p{:}\epsilon RR.F_0pp : \overset{\alpha}{\neg}(\epsilon RR), \\ \alpha:* &\vdash \lambda q{:}\overset{\alpha}{\neg}(\epsilon RR).q(G_0q) : \overset{\alpha}{\neg}\overset{\alpha}{\neg}(\epsilon RR). \end{aligned}$$

So, a term of type α is

$$\alpha:*\vdash(\lambda q{:}\overset{\alpha}{\neg}(\epsilon RR).q(G_0q))(\lambda p{:}\epsilon RR.F_0pp):\alpha.$$

A term of type $-$ is now constructed by abstracting over α.

So, we see that if we have a representation of the Russell paradox in S, then a term of type $-$ can be constructed that is quite close to Ω. To really be able to type Ω (that is, find types A and B such that $(\lambda x{:}A.xx)(\lambda x{:}B.xx)$ is typable in S) we need some specific requirements from the representation of the Russell paradox. It suffices that the triple $(V,\epsilon,\mathsf{comp})$ can be chosen in such a way that $\epsilon x(\mathsf{comp}P)=Px$ for arbitrary x and P. (Then $\overset{\alpha}{\neg}(\epsilon RR)=\epsilon RR$ and for the terms F and G above one can just take the identity and so $F_0=G_0=\mathsf{Id}_{\epsilon RR}$.) It is possible to do with less than $\epsilon x(\mathsf{comp}P)=Px$ to get a typing for Ω. In fact it suffices to have a strong representation of the Russel paradox.

Proposition 4.3 *If there is a strong representation of the Russell paradox in S, then there is a fixed point operator in S. (That is, a closed term Fix of type $\Pi\alpha{:}*.(\alpha\to\alpha)\to\alpha$ such that $\mathsf{Fix}\,\alpha f=f(\mathsf{Fix}\,\alpha f)$.)*

Proof Let the tuple $(V,\epsilon,\mathsf{comp},F,G)$ be a strong representation of the Russell paradox in S. Let α be a variable of type $*$ and f a variable of type $\alpha\to\alpha$. Define R, F_0 and G_0 as above, so

$$\begin{aligned} R &\equiv \mathsf{comp}(\lambda x{:}V.(\epsilon xx)\to\alpha), \\ F_0 &\equiv F[\lambda x{:}V.\,\overset{\alpha}{\neg}(\epsilon xx)/P, R/x], \\ G_0 &\equiv G[\lambda x{:}V.\,\overset{\alpha}{\neg}(\epsilon xx)/P, R/x]. \end{aligned}$$

Then $F_0\circ G_0=\mathsf{Id}_{\overset{\alpha}{\neg}\epsilon RR}$ and define

$$\begin{aligned} \omega_F &\equiv \lambda p{:}\epsilon RR.f(F_0pp) : \overset{\alpha}{\neg}\epsilon RR, \\ \omega_G &\equiv \lambda q{:}\overset{\alpha}{\neg}\epsilon RR.f(q(G_0q)) : \overset{\alpha}{\neg}\overset{\alpha}{\neg}\epsilon RR. \end{aligned}$$

Hence $\omega_G\omega_F:\alpha$ and

$$\begin{aligned} \omega_G\omega_F &\twoheadrightarrow_\beta f(\omega_F(G_0\omega_F)) \\ &\twoheadrightarrow_\beta f(f(F_0(G_0\omega_F)(G_0\omega_F))) \\ &= f(f(\omega_F(G_0\omega_F))),\ (\text{using } F_0\circ G_0=\mathsf{Id}) \\ &= f(\omega_G\omega_F). \end{aligned}$$

So, the term

$$\mathsf{Fix}\equiv\lambda\alpha{:}*.\lambda f{:}\alpha\to\alpha.\omega_G\omega_F$$

is a fixed point operator. □

Obviously, every inconsistent Pure Type System S has a representation of the Russell paradox. (Provided that S contains a sort of propositions $*$, an axiom $*:s_1$ and rules $(*,*)$, $(s_1,*)$ and (s_1,s_1) for some sort s_1.) This is the case because, due to inconsistency, for *any* triple $(V,\epsilon,\mathsf{comp})$ with $\vdash V:s_1$, $\vdash\epsilon:V\to V\to*$, $\vdash\mathsf{comp}:(V\to*)\to V$, the type $\Pi P{:}V\to*.\Pi x{:}V.Px\leftrightarrow\epsilon x(\mathsf{comp}P)$ is inhabited in the empty context: just apply the term of the type $-(\equiv\Pi\alpha{:}*.\alpha)$ to $\Pi P{:}V\to*.\Pi x{:}V.Px\leftrightarrow\epsilon x(\mathsf{comp}P)$. Note that this will *never* generate a strong representation of the Russel paradox, because the term of type $-$ has no head-normal form.

It is not known whether there are Pure Type Systems that have a strong representation of the Russell paradox. There is however a slight extension of the Pure Type System $\lambda*$ in which there is a strong representation of the Russell paradox, and hence a fixed point operator. This extension will be discussed in the next section. The system $\lambda*$ has another nice property, which is that the existence of a fixed point operator is equivalent to the existence of a strong representation of the Russell paradox:

Proposition 4.4 *The* PTS *$\lambda*$, or any of its extensions, has a strong representation of the Russell paradox iff it has a fixed point combinator.*

Proof The implication from left to right holds by Proposition 4.3. For the implication from right to left, let $\mathsf{Fix}:\Pi\alpha.(\alpha\to\alpha)\to\alpha$ be a fixed point operator in $\lambda*$. Define $V\equiv\mathsf{Fix}(\lambda\alpha{:}*.\alpha\to*)$, so $V=V\to*$. Now we can take $\epsilon\equiv\lambda xy{:}V.yx : V\to V\to*$ and $\mathsf{comp}\equiv\mathsf{Id}_V : V\to V\to*$. This yields a strong representation of the Russell paradox, because

$$\begin{aligned} P{:}V\to*, x{:}V &\vdash \mathsf{Id}_{Px}:Px\to\epsilon x(\mathsf{comp}P), \\ P{:}V\to*, x{:}V &\vdash \mathsf{Id}_{\epsilon x(\mathsf{comp}P)}:\epsilon x(\mathsf{comp}P)\to Px. \end{aligned}$$ □

5 A type system that does have a Fixed point operator

In this section, let S be a PTS that contains a sort $*$, an axiom $*:s_1$ and rules $(*,*)$, $(s_1,*)$, (s_1,s_1) and $(*,s_1)$ for some sort s_1. So, in addition to what is required in Definition 4.1, S must have the rule $(*,s_1)$. This rule introduces type dependency into the system: If $A:*$ then we can have $P:A\to*$ and hence $Pt:*$ for $t:A$, so the type Pt depends on a term t (note that, for example, the Calculus of Constructions and $\lambda*$ satisfy these requirements). We extend S by the following constants and typing rules:

$$\begin{aligned} \overset{eq}{=} &: \Pi A{:}*.A\to A\to* \\ \mathsf{refl} &: \Pi A{:}*.\Pi x{:}A.x\overset{eq}{=}_A x \\ \mathsf{elim} &: \Pi A{:}*.\Pi x,y{:}A.\Pi P{:}A\to*.Px\to(x\overset{eq}{=}_A y)\to Py \\ \mathrm{K} &: \Pi A{:}*.\Pi x{:}A.\Pi P{:}(x\overset{eq}{=}_A x)\to*. \\ &\qquad P(\mathsf{refl}_Ax)\to\Pi q{:}x\overset{eq}{=}_A x.Pq \end{aligned}$$

where (as in the following), $x \stackrel{eq}{=}_A y$ is infix notation for $(\stackrel{eq}{=}\ A\ x\ y)$ and refl_A, elim_A and K_A stand respectively for $\mathsf{refl}\ A$, $\mathsf{elim}\ A$ and $\mathrm{K}\ A$. Furthermore, we extend the conversion relation by:

$$\begin{array}{rcl} \mathsf{elim}_A xyPq(\mathsf{refl}_A z) & \longrightarrow_{=} & q \\ \mathrm{K}_A xPq(\mathsf{refl}_B y) & \longrightarrow_{K} & q \end{array}$$

An important remark is that all the rules and reduction not involving K can be obtained by using the usual Leibniz equality, i.e. defining

$$x \stackrel{eq}{=}_A y \equiv \Pi P : A \rightarrow * .Px \rightarrow Py.$$

It is not very difficult to show that Leibniz equality is an equivalence relation and furthermore that if two terms are convertible, then they are Leibniz-equal. The proof of the fact that each term is equal to itself is $\mathsf{refl}_A \equiv \lambda x{:}A.\lambda P{:}A \rightarrow * .\lambda q{:}Px.q$

Two terms are Leibniz-equal if the same 'properties' hold for them, viewing a property of terms of type A as a function of type $A \rightarrow *$. This is formally expressed by the equality elimination rule. (The term refl is the equality introduction rule.)

$$\begin{array}{rcl} \mathsf{elim}_A & \equiv & \lambda x, y{:}A.\lambda P{:}A \rightarrow * .\lambda z{:}Px.\lambda q{:}(x \stackrel{eq}{=}_A y).qPz \\ & : & \Pi x, y{:}A.\Pi P{:}A \rightarrow * .Px \rightarrow (x \stackrel{eq}{=}_A y) \rightarrow Py. \end{array}$$

We also have $\mathsf{elim}_A xxPq(\mathsf{refl}_A x) \twoheadrightarrow_\beta q$.

If two terms are Leibniz-equal, they need not be convertible. (The reverse is obviously the case.) So, if the types A and B (of type $*$) are Leibniz-equal, they need not have the same inhabitants. This can be seen as a drawback, but it is the price one has to pay for the typing to be decidable.

The addition of the K axiom has been suggested by [Streicher 1993]. This axiom has been proven to be extremely useful in practice and now appears as a natural extension to theories like the Calculus of Constructions. It is in particular necessary for dealing with equality over family of types. This is because in a system with K-axiom, it can be shown that all equality proofs are equal:

Lemma 5.1 *In a type system that includes K, the type*

$$\Pi x, y{:}A.\Pi p, q{:}(x \stackrel{eq}{=}_A y).p \stackrel{eq}{=}_{x \stackrel{eq}{=}_A y} q$$

is inhabited.

Proof A term of this type can be constructed by first constructing a term M (using K) such that

$$A{:}*, x{:}A \vdash \Pi p, q{:}(x \stackrel{eq}{=}_A x).p \stackrel{eq}{=}_{x \stackrel{eq}{=}_A x} q.$$

Then define $R : A \rightarrow *$ by

$$R \equiv \lambda y{:}A.\Pi p, q{:}(x \stackrel{eq}{=}_A y).p \stackrel{eq}{=}_{x \stackrel{eq}{=}_A y} q.$$

There is a term of type Rx in the context $A{:}*, x{:}A$, hence we can construct a term N such that

$$A{:}*, x{:}A, r : x \stackrel{eq}{=}_A y \vdash N : Ry$$

But then there is a term of type

$$\Pi x, y{:}A.\Pi p, q{:}(x \stackrel{eq}{=}_A y).p \stackrel{eq}{=}_{x \stackrel{eq}{=}_A y} q \ \square$$

The consequence of K mentioned in the Lemma can also be frased in the following way. If P is a polymorphic predicate on equality proofs, then this predicate holds for all equality proofs iff P holds for refl. More precisely, there is a term of type

$$\begin{array}{l} \Pi P{:}(\Pi x, y{:}A.(x \stackrel{eq}{=}_A y) \rightarrow *). \\ (\Pi x{:}A.Pxx(\mathsf{refl}_A x)) \rightarrow (\Pi x, y{:}A.\Pi q{:}(x \stackrel{eq}{=}_A y).Pxyq). \end{array}$$

What is essential to point out, is that this extension with K does not jeopardize the previous metatheoretical results; in particular it preserves the Church-Rosser property in the case of normalizing systems (Theorem 2.7) and it preserves subject-reduction (Lemma 2.6) (all the proofs of [Geuvers 1993] go through). Therefore, the system $\lambda * +\mathrm{K}$ is a good candidate for illustrating the fact that Church-Rosser might not hold, for a, "at first sight" interesting Type Theory [5].

Proposition 5.2 *In $\lambda * +K$, there is a strong representation of the Russell paradox; hence $\lambda * +K \not\models CR_{\beta\eta}$*

Proof Let $\stackrel{eq}{=}$ be the equality for which we have the K-axiom. Now, define V, ϵ and comp as follows.

$$\begin{array}{rcl} V & \equiv & \Pi\alpha{:}*.\alpha \rightarrow * \quad (:*), \\ \epsilon & \equiv & \lambda x, y{:}V.yVx \quad (:V \rightarrow V \rightarrow *), \\ \mathsf{comp} & \equiv & \lambda P{:}V \rightarrow * .\lambda\alpha{:}* .\lambda x{:}\alpha.\Pi q{:}V \stackrel{eq}{=}_* \alpha. \\ & & \mathsf{elim}_* V\alpha(\lambda\beta.\beta \rightarrow *)Pqx \quad (:(V \rightarrow *) \rightarrow V). \end{array}$$

For comp, note that, if $q : V \stackrel{eq}{=}_* \alpha$ and $Q : * \rightarrow *$, then any inhabitant of QV can be transformed into an inhabitant of $Q\alpha$. (This is done by applying $\mathsf{elim}_* V\alpha Qq$ to the inhabitant of QV.) So, if we take $Q \equiv \lambda\beta{:}*.\beta \rightarrow *$, then the term P of type QV $(\equiv V \rightarrow *)$ can be transformed into a term of type $Q\alpha$ $(\equiv \alpha \rightarrow *)$. The latter term is then applied to x.

Now, the terms F and G are constructed as follows. Let $P{:}V \rightarrow *, x{:}V$. Then

$$\begin{array}{rcl} \epsilon x(\mathsf{comp}P) & = & \mathsf{comp}PVx \\ & = & \Pi q{:}V \stackrel{eq}{=}_* V.\mathsf{elim}_* VV(\lambda\beta{:}* .\beta \rightarrow *)qPx, \end{array}$$

[5] There exist other formalizations of Russell's paradox; in particular, T. Streicher had independently developed one in $\lambda * +\mathrm{K}$. They seem however more complicated and, to our knowledge, do not lead to a fixed point operator. Another remark is that this fixed-point construction can also be done in the Calculus of Constructions with strong sums [Coquand 1986] extended with K.

so for $z : \epsilon x(\mathsf{comp}P)$, we have that $z(\mathsf{refl}_* V) : \mathsf{elim}_* VV(\lambda\beta{:}*.\beta{\to}*)(\mathsf{refl}_* V)Px = Px$. Now take $F \equiv \lambda z{:}\epsilon x(\mathsf{comp}P).z(\mathsf{refl}_* V)$ and $G \equiv$ $\lambda z{:}Px.\mathrm{K}_* V(\lambda q{:}V \stackrel{eq}{=}_* V.\mathsf{elim}_* VV(\lambda\beta{:}*.\beta{\to}*)Px)z$. Then indeed, $P{:}V{\to}*, x{:}V \vdash F : \epsilon x(\mathsf{comp}P){\to}Px$ and $P{:}V{\to}*, x{:}V \vdash G : Px{\to}\epsilon x(\mathsf{comp}P)$. Also $F(Gz) \twoheadrightarrow_\beta$ $\mathrm{K}_* V(\lambda q{:}V \stackrel{eq}{=}_* V.q(\lambda\beta{:}*.\beta{\to}*)Px)z(\mathsf{refl}_* V)$ $\longrightarrow_K z$, for arbitrary z. □

Another way to describe this fact is to say that the syntactical and semantical versions of $\lambda*{+}\mathrm{K}$ are not equivalent or that $\lambda*{+}\mathrm{K}$ is not sound.

To end this section we want to describe another criterion for the existence of a fixed point combinator. The construction we give here is largely due to [van Draanen 1994], who described it in the framework of the simply typed lambda calculus. In a nutshell the result says that, if there is an emuerator in the system S, then there is a fixed point opertor in the system S.

To describe what it means to have an enumerator in S, we assume that we have a coding of all the closed terms as natural numbers $\#$. So, if M is a closed term in S, then $\#M \in \mathbf{N}$. Furthermore we assume that we have a fixed closed type $N : *$, the type of numerals, and for $n \in \mathbf{N}$ we let $\ulcorner n \urcorner$ be the representation of the number n as a (closed) term of type N. (So, for example in $\lambda*$, one could have $N \equiv \Pi\alpha{:}*.\alpha{\to}(\alpha{\to}\alpha){\to}\alpha$ and $\ulcorner n \urcorner \equiv \lambda\alpha{:}*.\lambda x{:}\alpha.\lambda f{:}\alpha{\to}\alpha.f^n(x)$, the polymorphic Church numerals.)

Definition 5.3 *An* enumerator *in S is a family of combinators E_σ, for every closed type σ, such that*

$$\begin{array}{rcl} E_\sigma & : & N{\to}\sigma, \\ E_\sigma\ulcorner\#M\urcorner & = & M, \textit{ for all closed terms } M \textit{ of type } \sigma. \end{array}$$

(The equality is the conversion of the system.)

Lemma 5.4 ([van Draanen 1994]) *If there is a (family of) enumerators, then there is a family of fixed point combinators. That is, for every closed type σ there is a term $Y_\sigma : (\sigma{\to}\sigma){\to}\sigma$ such that $Y_\sigma f = f(Y_\sigma f)$.*

Proof Take

$$A_\sigma :\equiv \lambda n{:}N.\lambda f{:}\sigma{\to}\sigma.f(E_\tau nnf),$$

where $\tau :\equiv N{\to}(\sigma{\to}\sigma){\to}\sigma$ and E_τ is the enumerator for the type τ. Take

$$Y_\sigma :\equiv A_\sigma\ulcorner\#A_\sigma\urcorner.$$

Then

$$\begin{array}{rcl} Y_\sigma f & = & f(E_\tau\ulcorner\#A_\sigma\urcorner\ulcorner\#A_\sigma\urcorner f) \\ & = & f(A_\sigma\ulcorner\#A_\sigma\urcorner f) \\ & = & f(Y_\sigma f). \ \square \end{array}$$

So, the fixed point combinator Y_σ is a close relative to the Turing Fixed point combinator $\Theta \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy))$. Now in $\lambda*$ we have the following.

Lemma 5.5 *If there is a fixed point combinator for the type $\Pi\alpha{:}*.(\alpha{\to}\alpha){\to}\alpha$, then there is a polymorphic fixed point combinator $Y : \Pi\alpha{:}*.(\alpha{\to}\alpha){\to}\alpha$.*

Proof Write σ for the type $\Pi\alpha{:}*.(\alpha{\to}\alpha){\to}\alpha$ and let $Y_\sigma : (\sigma{\to}\sigma){\to}\sigma$ be the fixed point combinator for σ. Take

$$G \equiv \lambda g{:}\sigma.\lambda\alpha{:}*.\lambda f{:}\alpha{\to}\alpha.f(g\alpha f).$$

Then $G : \sigma{\to}\sigma$ and G has a fixed point $Y_\sigma G$, for which we have

$$Y_\sigma G\alpha f = G(Y_\sigma G)\alpha f = f(Y_\sigma G\alpha f),$$

so $Y_\sigma G$ is a polymorphic fixed point combinator. □

Corollary 5.6 *If $\lambda*$ has enumerators, then it has a polymorphic fixed point operator.*

In fact, it suffices to have an enumerator for the type $N{\to}(\sigma{\to}\sigma){\to}\sigma$, where σ is the type of the polymorphic fixed point operator.

It is obvious that a type of numerals N and a coding $\#$ exist, but it is not clear how the typable terms E_σ should be constructed.

6 Discussion and conclusion

We have motivated the importance of the Church-Rosser property for typed λ-calculi and related it to the existence of a fixed point combinator. By constructing such a fixed point in a relatively natural extension of $\lambda*$, we have shown that Church-Rosser might actually be false for non-normalizing systems. Therefore, a direct confluence proof for $\mathsf{PTSs}_{\beta\eta}$ seems much less likely to exist, and, what is more important, would not be extremely useful, as we now know it could not apply for more general systems including extensions like a reasonable equality. The main practical conclusion seems to be that a clear trade-off has to be made between the power of the studied Type System, its implementability, and the tediousness of the normalization proof. For example, by working with a semantical version of the Calculus of Constructions, Altenkirch is able to achieve an extremely elegant normalization and consistency proof; but he cannot show that his system is equivalent to the implementable syntactical version without dropping η-conversion because he lacks the necessary Church-Rosser proof. On the other hand, it is possible to do all the metatheory for the syntactical version, but having to deal with untyped conversions complicates some parts of the proof [Werner 1994].

Referencesmkboth ReferencesReferences

[Altenkirch 1993] T. Altenkirch, Constructions, Inductive Types and Strong Normalization, Ph.D. thesis, University of Edinburgh, 1993.

[Barendregt 1984] H.P. Barendregt, *The lambda calculus: its syntax and semantics*, revised edition. Studies in Logic and the Foundations of Mathematics, North Holland.

[Barendregt 1992] H.P. Barendregt, Typed lambda calculi. In *Handbook of Logic in Computer Science*, eds. Abramski et al., Oxford Univ. Press.

[van Benthem Jutting 199+] L.S. van Benthem Jutting, Typing in Pure Type Systems, *Information and Computation*.

[Berardi 1990] S. Berardi, Type dependence and constructive mathematics, Ph.D. thesis, Universita di Torino, Italy.

[de Bruijn 1980] N.G. de Bruijn, A survey of the project Automath, In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, eds. J.P. Seldin, J.R. Hindley, Academic Press, New York, pp 580-606.

[Coquand 1986] Th. Coquand, An analysis of Girard's paradox, *Proceedings of the first symposium on Logic in Computer Science, Cambridge Mass.*, IEEE, pp 227-236.

[Coquand 1990] Th. Coquand, Metamathematical investigations of a calculus of constructions. In *Logic and Computer Science*, ed. P.G. Odifreddi, APIC series, vol. 31, Academic Press, pp 91-122.

[Coquand 1991] Th. Coquand, An algorithm for testing conversion in type theory, *Logical Frameworks*, eds. G. Huet and G. Plotkin, Cambridge University Press, 1991.

[Coquand 199+] Th. Coquand, A new paradox in type theory, to appear in *Proceedings of the 9th International Congress of Logic, Methodology and Philosophy of Science, Uppsala, Sweden 1991.*

[Coquand and Herbelin 1992] Th. Coquand and H. Herbelin, An Application of A-translation to the existence of families of looping combinators in inconsistent Type Systems, to appear in *Journal of Functional Programming.*

[Coquand and Huet 1985] Th. Coquand and G. Huet, Constructions: a higher order proof system for mechanizing mathematics. *Proceedings of EUROCAL '85, Linz*, LNCS 203.

[Dowek et al. 1991] G. Dowek, A. Felty, H. Herbelin, G. Huet, Ch. Paulin-Mohring, B. Werner, The Coq proof assistant version 5.6, user's guide. INRIA Rocquencourt - CNRS ENS Lyon.

[Dowek et al. 1993] G. Dowek, G. Huet and B. Werner, On the Definition of the η-long Normal Form in Type Systems of the Cube. Informal Proceedings of the BRA Types workshop, H. Geuvers Ed., Nijmegen, 1993.

[Dowek 1993a] G. Dowek, A Complete Proof Synthesis Method for the Cube of Type Systems, *Journal of Logic and Computation*, vol 3 (3), pp 287-315, 1993.

[Dowek 1993b] G. Dowek, The undecidability of typing in the lambda-pi-calculus, in *Typed Lambda Calculi and Applications, LNCS 664*, eds. M. Bezem and J.F. Groote, pp 139-146.

[van Draanen 1994] J.-P. van Draanen, Personal Communication.

[Geuvers and Nederhof 1991] J.H. Geuvers and M.J. Nederhof, A modular proof of strong normalisation for the calculus of constructions. *Journal of Functional Programming*, vol 1 (2), pp 155-189.

[Geuvers 1992] J.H. Geuvers, The Church-Rosser property for $\beta\eta$-reduction in typed lambda calculi. In *Proceedings of the seventh annual symposium on Logic in Computer Science, Santa Cruz, Cal.*, IEEE, pp 453-460.

[Geuvers 1993] J.H. Geuvers, Logics and Type Systems, PhD. Thesis, Katholieke Universiteit Nijmegen, Netherlands.

[Giannini et al. 1993] P. Giannini, F. Honsell, S. Ronchi della Rocca, Type inference: some results, some problems, *Fundamenta Informaticae*, 19, 1,2, 1993, pp 87-126.

[Girard 1972] J.-Y. Girard, Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur. Ph.D. thesis, Université Paris VII, France.

[Hofmann and Streicher] M. Hofmann and T. Streicher, A groupoid model refutes uniqueness of identity types, this volume, 1994.

[Howe 87] D.J. Howe, The computational behavior of Girard's paradox, Proccedings of LICS 1987.

[Liquori et al. 93] L. Liquori, S. Ronchi della Rocca, S. van Bakel and P. Urzyczyn, Comparing Cubes, manuscript, University of Turin, Italy.

[Luo and Pollack 1992] Z. Luo, R. Pollack, Lego proof development system: User's Manual, Dept. of Computer Science, University of Edinburgh, April 1992.

[Meyer and Reinhold 1986] A.R. Meyer, M.B. Reinhold, Type is not a type, Proceedings of POPL 1986.

[Streicher 1991] T. Streicher, *Semantics of type theory : correctness, completeness and independence results*, Birkhauser, 1991.

[Streicher 1993] T. Streicher, Investigations into Intensional Type Theory, Habilitationsschrift, University of Münich, 1993.

[Werner 1994] B. Werner, Une Théorie des Constructions Inductives, Thèse de Doctorat, Université Paris VII, 1994.