*J. Functional Programming* **7** (6): 613–660, November 1997. Printed in the United Kingdom 


# *Modularity of strong normalization in the algebraic-λ-cube*

FRANCO BARBANERA[1], MARIBEL FERNÁNDEZ[2]

and HERMAN GEUVERS[3]

[1]*Dipartimento di Informatica, Universitá di Torino,*
*Corso Svizzera 185, 10149 Torino, Italy*
(*e-mail:* barba@di.unito.it)
[2]*DMI - LIENS (CNRS URA 1327), École Normale Supérieure,*
*45, rue d'Ulm, 75005 Paris, France*
(*e-mail:* maribel@dmi.ens.fr)
[3]*Faculty of Mathematics and Informatics, Catholic University of Nijmegen,*
*Toernooiveld 1, 6525 ED Nijmegen, The Netherlands*
(*e-mail:* herman@cs.kun.nl)

## Abstract

In this paper we present the algebraic-λ-cube, an extension of Barendregt's λ-cube with first- and higher-order algebraic rewriting. We show that strong normalization is a modular property of all the systems in the algebraic-λ-cube, provided that the first-order rewrite rules are non-duplicating and the higher-order rules satisfy the general schema of Jouannaud and Okada. We also prove that local confluence is a modular property of all the systems in the algebraic-λ-cube, provided that the higher-order rules do not introduce critical pairs. This property and the strong normalization result imply the modularity of confluence.

## Capsule Review

The λ-cube contains various systems intended to be used for the formalization of proofs in order to verify these mechanically. Also, these formal proofs are being used to automatically extract algorithms from them, whenever possible. Algebraic terms can easily be formed inside the λ-cube. For efficiency reasons (ease of formalizing and executability of the resulting proofs), one would like to add rewrite rules on these. This paper studies various ways to do this. Several conditions are formulated that guarantee the preservation of properties like strong normalization and confluence for the systems with the extended reduction relations. Because the added rewrite rules may be of a higher order, these conditions are somewhat involved.

## 1 Introduction

One of the main motivations for the development of computational models is no doubt that of isolating particular aspects of the practice of computing, in order to better investigate them, so allowing either to tune existing programming languages or to devise new ones. However, the study of such models cannot exploit all its possibilities to help the development of actual computing tools unless their

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

interactions and possible (in)compatibilities are also investigated. In this framework, many research efforts have been devoted over the last few years to the study of the interactions between two closely related models of computation: that based on $\beta$-reduction on $\lambda$-terms; and that formalized by means of rewrite rules on algebraic terms. These particular models are relevant for the study of two aspects of programming languages: higher-order programming and data type specification. The combination of these two models has also provided an alternative in the design of new programming languages: the *algebraic functional languages* (Jouannaud and Okada, 1991). These languages allow algebraic definitions of data types and operators (as in equational languages like OBJ) and the definition of higher-order functions (as in functional languages like ML), in a unified framework.

The study of systems based on $\lambda$-calculus and algebraic rewriting has been carried out both in untyped and typed contexts. If no type discipline is imposed on the languages, the interactions between these computational models raise several problems (Klop, 1987; Dougherty, 1992). For typed languages things work out nicely. In Breazu-Tannen and Gallier (1990) and Okada (1989), it is shown that the system obtained by combining a terminating first-order many-sorted term rewrite system with the second-order typed $\lambda$-calculus is again terminating with respect to $\beta$-reduction and the algebraic reductions induced by the rewrite rules, i.e. strong normalization is a *modular* property in this case. The same holds for confluence (Breazu-Tannen and Gallier, 1992). In Jouannaud and Okada (1991), both results are extended to combinations of first- and higher-order rewriting systems with second-order $\lambda$-calculus, under certain conditions on the form of the rewrite rules.

The question that naturally arises is whether such a nice interaction between typed $\lambda$-calculi and algebraic rewriting is independent of the power of the type discipline. More precisely, the question is whether the existing results extend to higher-order type disciplines such as the Calculus of Constructions of Coquand and Huet (1988). Actually, considering only first-order algebraic rewriting, this problem has already been addressed (Barbanera, 1990). A strong restriction was, however, imposed on the definition of the combined system: in the type conversion rule only $\beta$-conversion ($=_\beta$) was considered as equality. So, even if there was a rewrite rule $x+0 \to x$ in the system, two types of the form $P(x)$ and $P(x+0)$ (where $P$ is a type depending on natural numbers) were not considered to be the same. Such a choice was motivated mainly by the essential use of the property of confluence in the proof of modularity of strong normalization.

In this paper we extend the Calculus of Constructions, adding not only first-order but also higher-order algebraic rewriting, and considering in the type conversion rule the $R\beta$-reduction/expansion relation generated by the algebraic reductions together with $\beta$-reduction. Because of the presence of the $R\beta$-reductions, the proof of strong normalization cannot rely on the confluence property, which obviously does not hold in general. Also, other properties of the metatheory of the system, like Subject Reduction, which in the case of the pure Calculus of Constructions are proven using confluence, will have to be proven independently of confluence in this extension.

In fact, the presence of the $R$-reduction relation in the type conversion rule also makes the definition of the system more involved: this rule is used to define the

terms of the system, and one cannot define a notion of $R\beta$-reduction to be used in the rule unless one knows what the terms are. We have then to cope with a circularity, which can be solved in two ways, either by defining the system by levels, starting from the pure Calculus of Constructions, or by defining algebraic rewriting on pseudoterms and using it in the type conversion rule, but taking into account only sets of rewrite rules that have a correct algebraic meaning when, after the definition of legal terms, they can be typed. The second solution, to be discussed more fully later, is the one we have chosen in the present paper where, for the sake of uniformity, we provide a definition of the extension with first- and higher-order (in the sense of Jouannaud and Okada, 1991) algebraic rewriting of all the systems of the so-called $\lambda$-cube (Berardi, 1990; Barendregt, 1991). This extension will be called *algebraic-λ-cube* (*λR-cube* for short).

The main result we prove for the systems of the $\lambda R$-cube is the modularity of the strong normalization property, i.e. we prove that the systems are strongly normalizing in case the first-order algebraic rules are so on algebraic terms (the higher-order rules we will use are strongly normalizing because of their structure).

As stated before, we had to cope with the problem of not having at hand the property of confluence. We solved such a problem by extending some technical results devised in Geuvers (1992) (see also Geuvers, 1993). We prove strong normalization in three steps. By means of a reduction preserving translation, we prove the strong normalization of the extended Calculus of Constructions to be implied by the same property of system $\lambda_{R\omega}$ (the extended typed $\lambda$-calculus of order $\omega$). The strong normalization property of this last system will also be proven by means of a reduction preserving translation, showing it to be implied by the same property of system $\lambda_{\wedge R}$ (a type assignment system for $\lambda$-calculus with intersection types and algebraic rewriting), which in turn was proven strongly normalizing in Barbanera and Fernández (1996). A preliminary version of the strong normalization proof for $\lambda_{R\omega}$ was given in Barbanera and Fernández (1993b).

Finally, we also prove that local confluence is a modular property of the systems of the algebraic-$\lambda$-cube, provided that the higher-order rules do not introduce critical pairs. This, and the previous strong normalization result, imply the modularity of confluence.

This paper is an extended version of Barbanera *et al.* (1994). It is organized as follows. In section 2 we define the $\lambda R$-cube. Section 3 is devoted to the metatheory of the $\lambda R$-cube (among others, the Subject Reduction property is proved). We then outline the skeleton of the strong normalization proof in section 4. The main points of such proof are the topic of sections 5 and 6 (we recall system $\lambda_{\wedge R}$ in Appendix A). In section 7 we prove the modularity of confluence. Section 8 contains the conclusions.

## 2 Adding algebraic rewriting to the λ-cube: the λR-cube

The $\lambda$-cube (Berardi, 1990; Barendregt, 1991) is a coherent collection of eight type systems. Each system is placed on a vertex of the cube in a way that geometrically exploits the possible dependencies between types and terms. Each of the possible

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

directions in the three-dimensional space in which the cube is corresponds to a particular dependency.

The aim of the present section is to define an extension of the $\lambda$-cube with algebraic rewriting features: the algebraic-$\lambda$-cube ($\lambda R$-cube for short). As usual, we will define first the set of pseudoterms of the system and then the inference rules that are used to generate legal terms. The set of pseudoterms of the $\lambda R$-cube includes algebraic constants and types, defined as follows:

*Definition 2.1* (*Algebraic types*)
Let $\mathscr{S}$ be a countable set of *sorts*: $\mathscr{S} = \{s_1, s_2, \ldots\}$. The elements of $\mathscr{S}$ denote *basic algebraic types*. The set $\mathsf{T}_{\mathscr{S}}$ of *algebraic types* over $\mathscr{S}$ is inductively defined as follows:

- $\mathscr{S} \subseteq \mathsf{T}_{\mathscr{S}}$
- $\sigma, \tau \in \mathsf{T}_{\mathscr{S}} \;\Rightarrow\; \sigma \to \tau \in \mathsf{T}_{\mathscr{S}}$

We will call *first-order algebraic types* the elements $\sigma_1 \to \ldots \to \sigma_n \to \sigma \in \mathsf{T}_{\mathscr{S}}$ such that $\sigma, \sigma_i \in \mathscr{S}$ $(1 \leq i \leq n)$.

*Definition 2.2* (*Signature*)
A *signature* (over $\mathscr{S}$) is a set $\mathscr{F}$ of (typed) *function symbols*, such that

$$\mathscr{F} = \bigcup_{\tau \in \mathsf{T}_{\mathscr{S}}} \mathscr{F}_\tau,$$

where $\mathscr{F}_\tau$ denotes the set of (Curried) function symbols whose functionality (type) is $\tau$. We assume $\mathscr{F}_\tau \cap \mathscr{F}_{\tau'} = \emptyset$ if $\tau \not\equiv \tau'$. Each function symbol $f$ in $\mathscr{F}$ is assumed to have an arity which, when necessary, will be denoted by superscripts $(f^n)$[1].

A function symbol $f^m$ is called *first-order* if it has a first-order algebraic type $s_1 \to \ldots \to s_n \to s$ and $n \equiv m$. Function symbols which are not first-order will be called *higher-order*. We denote by $\Sigma$ the set of all first-order function symbols in $\mathscr{F}$ and by $f, g, \ldots$ its generic elements. Capital letters $F, G, \ldots$ denote instead generic higher-order function symbols. When it is clear from the context, we use $f$ to denote a generic (first- or higher-order) element of $\mathscr{F}$.

We now add function symbols and sorts to the syntax of the pure $\lambda$-cube:

*Definition 2.3* (*Pseudoterms*)
The eight systems of the $\lambda R$-cube are based on a set $T$ of *pseudoterms* defined by the following grammar:

$$T ::= \mathbf{x} \mid \mathbf{f} \mid \mathbf{s} \mid \star \mid \Box \mid (T\,T) \mid \lambda \mathbf{x}\!:\!T.T \mid \Pi \mathbf{x}\!:\!T.T$$

[1] The arity of a function symbol cannot be deduced from its type, it must be given when defining a signature (if a function symbol has a type $\sigma_1 \to \ldots \to \sigma_n \to s$, with $s \in \mathscr{S}$, its arity can at most be $n$). Arities will serve to distinguish first- and higher-order function symbols.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

where **f** and **s** range over $\mathscr{F}$ and $\mathscr{S}$, respectively, **x** ranges over the category of variables (divided into two groups: $Var^{\star}$ and $Var^{\Box}$ ) and $\star$ and $\Box$ are special constants, the former informally denoting the set of *types*, the latter that of *kinds*[2].

The notions of *bound variable* and *free variable* are defined as usual, as well as that of substitution. The notation

$$M[N_1/x_1, \ldots, N_m/x_m]$$

will be used to denote the simultaneous substitution of the terms $N_i$ for the variables $x_i$ $(i = 1, \ldots, m)$ in the term $M$. A generic substitution will be often denoted by $\varphi$. In such a case $M\varphi$ will denote its application to the term $M$.

If $M$ is a pseudoterm then $Var(M)$, $FV(M)$ will denote the set of variables and the set of free variables of $M$, respectively. Just as in the untyped $\lambda$-calculus, terms that only differ from each other in their bound variables will be identified: we work *modulo* α*-conversion*.

$\vec{M}$ will stand for a sequence $M_1 \ldots M_n$; $\Pi\vec{x}{:}\vec{A}$ for $\Pi x_1{:}A_1 \ldots \Pi x_n{:}A_n$. The length of a sequence $\vec{M}$ will be denoted by $|\vec{M}|$. Expressions of the form $\Pi x{:}A.B$ will, as usual, be denoted by $A \to B$ if $x \notin FV(B)$. Such a notation, together with the fact that function symbols are Curried, makes it possible to see the elements of $\mathsf{T}_{\mathscr{S}}$ as pseudoterms.

To be able to define algebraic reduction rules, we consider the following subset of pseudoterms.

*Definition 2.4* (*Algebraic pseudoterms*)

(i) The set of *algebraic pseudoterms* is defined by the following grammar

$$A ::= \mathbf{x} \mid \mathbf{f} \mid (AA).$$

(ii) A function symbol $f^n$ is said to be *saturated* in an algebraic pseudoterm $t$ if any occurrence of it appears in subterms of $t$ of the form $ft_1 \ldots t_m$ with $m \equiv n$ (note that we omit brackets as usual, since application associates to the left). An algebraic pseudoterm is saturated if any function symbol occurring in it is so.

(iii) A *first-order algebraic pseudoterm* $t$ is a saturated algebraic pseudoterm such that

1. any $f \in \mathscr{F}$ occurring in $t$ is first-order
2. there is no subterm of $t$ of the form $xP$, where $x$ is a variable and $P$ is any pseudoterm.

(iv) A *higher-order algebraic pseudoterm* is an algebraic pseudoterm which is not first-order.

Given the algebraic pseudoterms we can define a set of algebraic rewrite pseudorules.

[2] In the literature, $\star$ and $\Box$ are usually called *sorts*, a word that we reserve for the base types of algebraic rewrite systems.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Definition 2.5* (*Algebraic rewrite pseudorules*)

(i) An *algebraic rewrite pseudorule* $r$ is a pair of algebraic pseudoterms $\langle t, t' \rangle$ such that
   1. $t$ is not a variable;
   2. $FV(t') \subseteq FV(t)$;
   3. $t$ and $t'$ are saturated.

(ii) A *first-order rewrite pseudorule* is a rewrite pseudorule $\langle t, t' \rangle$ such that both $t$ and $t'$ are first-order algebraic pseudoterms.

(iii) A *higher-order rewrite pseudorule* is a rewrite pseudorule which is not first-order.

A rewrite pseudorule will be denoted as usual by $r : t \to t'$. Given a set $R$ of rewrite pseudorules, we denote by $FOR$ and $HOR$ the subsets of first-order and higher-order pseudorules of $R$, respectively.

We cannot speak of types in algebraic rewrite rules yet, since types are well-formed terms of the $\lambda R$-cube, and to define well-formed terms we need to consider algebraic reductions in the type conversion rule which, together with the other term formation rules, will define the set of well-formed terms. In a sense, in order to avoid a circular definition, the 'algebraic correctness' of the algebraic pseudorules can be checked only *a posteriori*.

Given a set $R$ of algebraic reduction pseudorules, we can also define on pseudoterms, besides the usual $\beta$-reduction, $R$-reductions.

*Definition 2.6* (*Reductions on pseudoterms*)

(i) The $\beta$-reduction relation on pseudoterms ($\to_\beta$) is defined as the compatible closure of the notion of reduction

$$(\lambda x{:}A.B)C \to_\beta B[C/x].$$

(ii) The reduction relation induced by $R$ on pseudoterms ($\to_R$) is defined as follows:
$M \to_R N$ iff there exist $r : t \to t' \in R$, a context $C[\,]$ and a substitution $\varphi$ such that $M \equiv C[t\varphi]$ and $N \equiv C[t'\varphi]$.

(iii) We define $\to_{R\beta}$ as the union of $\to_R$ and $\to_\beta$: $M \to_{R\beta} N \iff (M \to_R N \vee M \to_\beta N)$.

As usual, $\twoheadrightarrow_\beta$, $\twoheadrightarrow_R$ and $\twoheadrightarrow_{R\beta}$ denote the reflexive and transitive closure of $\to_\beta$, $\to_R$ and $\to_{R\beta}$, respectively. The relations of $\beta$-, $R$-, and $R\beta$-conversion, i.e. the least equivalence relations generated by the reduction relations, will be denoted by $=_\beta$, $=_R$, and $=_{R\beta}$, respectively.

The relation $\to_\beta$ on pseudoterms satisfies the Church–Rosser property (Barendregt, 1986). It is obvious that, in general, this property does not hold for $\to_{R\beta}$.

We now present the set of typing rules that define the legal terms of the system, but first we introduce some terminology and notation.

An *assignment* is an expression of the form $M : N$, where $M$ and $N$ are pseudoterms. A *declaration* is an expression of the form $x{:}A$, where $x$ is a variable and $A$ a pseudoterm.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

A *pseudocontext* is a finite sequence $\langle x_1:A_1,\ldots,x_n:A_n,x:B\rangle$ of declarations.

If $\Gamma = \langle x_1:A_1,\ldots,x_n:A_n\rangle$ then $\Gamma,x:B$ denotes the pseudocontext $\langle x_1:A_1,\ldots,x_n:A_n,x:B\rangle$, and $\Gamma_{|V}$ denotes the restriction of $\Gamma$ to the variables in $V$.

$\Gamma' \subseteq \Gamma$ ($\Gamma'$ *is subcontext* of $\Gamma$) iff $x:A$ in $\Gamma' \Rightarrow x:A$ in $\Gamma$. $\Gamma \twoheadrightarrow_\beta \Gamma'$ ($\Gamma =_\beta \Gamma'$) iff $\Gamma = x_1:A_1,\ldots,x_n:A_n$, $\Gamma' = x_1:A'_1,\ldots,x_n:A'_n$ and $A_i \twoheadrightarrow_\beta A'_i$ ($A_i =_\beta A'_i$) for $1 \le i \le n$. A context $\Gamma = \langle x_1:A_1,\ldots,x_n:A_n\rangle$ is called *algebraic* if $A_i \in \mathsf{T}_{\mathscr{S}}$ $(1 \le i \le n)$.

A generic system of the $\lambda R$-cube will be denoted by $\lambda_{R-}$. A *statement* in a system $\lambda_{R-}$ is an expression of the form $\Gamma \vdash_{\lambda_{R-}} M : A$, where $\Gamma$ is a pseudocontext and $M$ and $A$ pseudoterms.

We show now, for each system of the $\lambda R$-cube, how to generate its *legal statements*. Legal *terms* and *contexts* will then be the pseudoterms and pseudocontexts contained in legal statements. Instead of 'the statement $\Gamma \vdash_{\lambda_{R-}} M : A$ is legal', from now on we shall just write $\Gamma \vdash_{\lambda_{R-}} M : A$. The legal statements of a system $\lambda_{R-}$ are those generated by the following general axioms and rules, and by particular specific rules (instantiations, depending on the system, of the parametric rule (Π)).

*General axioms and rules*

(ax) $\quad \vdash \star : \Box$

(alg1) $\quad \vdash s : \star$ $\qquad$ for any $s \in \mathscr{S}$

(alg2) $\quad \vdash f : \sigma$ $\qquad$ for any $f \in \mathscr{F}_\sigma$

(var) $\quad \dfrac{\Gamma \vdash A : p}{\Gamma, x{:}A \vdash x : A}$ $\qquad$ if $p \in \{\star,\Box\}, x \in Var^p$ and $x \notin \Gamma$

(weak) $\quad \dfrac{\Gamma \vdash A : p \quad \Gamma \vdash M : C}{\Gamma, x{:}A \vdash M : C}$ $\qquad$ if $p \in \{\star,\Box\}, x \in Var^p$ and $x \notin \Gamma$

($\lambda$) $\quad \dfrac{\Gamma, x{:}A \vdash M : B \quad \Gamma \vdash \Pi x{:}A.B : p}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}$ $\qquad$ if $p \in \{\star,\Box\}$

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

(red$_{R\beta}$) $\quad \dfrac{\Gamma \vdash M : A \quad \Gamma \vdash B : p}{\Gamma \vdash M : B}$ $\qquad$ if $A \twoheadrightarrow_R B, A \twoheadrightarrow_\beta B, B \twoheadrightarrow_R A$ or $B \twoheadrightarrow_\beta A$

*Specific rules*

(Π) $\quad \dfrac{\Gamma \vdash A : p_1 \quad \Gamma, x{:}A \vdash B : p_2}{\Gamma \vdash \Pi x{:}A.B : p_2}$ $\quad p_1, p_2 \in \{\star,\Box\}$

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

The specific rules, which characterize the different systems of the cube, are all introduction rules, obtained by considering particular $p_1, p_2 \in \{\star, \Box\}$ in the parametric rule ($\Pi$). Given particular $p_1, p_2 \in \{\star, \Box\}$, the corresponding ($\Pi$)-rule will be called $(p_1, p_2)$.

As for the pure $\lambda$-cube, the specific term-formation rules have the following informal meaning:

$(\star, \star)$ allows forming terms depending on terms,
$(\star, \Box)$ allows forming types depending on terms,
$(\Box, \star)$ allows forming terms depending on types,
$(\Box, \Box)$ allows forming types depending on types.

Recall that in the $\lambda R$-cube the reduction relation $\to_{R\beta}$ used in rule ($\mathrm{red}_{R\beta}$) has not been defined on terms, but on pseudoterms. To do otherwise would have led to a circularity, since rule ($\mathrm{red}_{R\beta}$) itself is used to define what a (legal) term is. Moreover, unlike in the pure $\lambda$-cube, the reduction-expansion relation has been used instead of the conversion relation. This is motivated by the fact that a side condition like $A =_{R\beta} B$ could be proved by means of a chain of reductions-expansions where pseudoterms that are not legal terms are also present. The use of conversion causes no trouble at all in the pure $\lambda$-cube, since, by the Church–Rosser and Subject Reduction properties of $\to_\beta$, we can always get $A =_\beta B$ by means of a chain containing only legal terms. As stated before, we cannot rely, instead, on Church–Rosser for the pseudoterms (and terms) of the $\lambda R$-cube.

The absence, in general, of the Church–Rosser property for $\to_{R\beta}$ makes also difficult to prove some properties easily provable in the pure cube, like Subject Reduction for $\beta$-reduction alone.

*Definition 2.7*
We use $A \overset{c}{=}_\beta B$ to denote that terms $A$ and $B$ are convertible via $\beta$-reductions and $\beta$-expansions that remain inside the set of well-typed terms. Similarly for $A \overset{c}{=}_R B$ and $A \overset{c}{=}_{R\beta} B$. To be precise, $A \overset{c}{=}_{R\beta} B$ means that there are well-typed terms $E_1, \ldots, E_n$ such that

$$A \twoheadrightarrow_{\rho_1} E_1 \twoheadleftarrow_{\rho_2} E_2 \twoheadrightarrow_{\rho_3} E_3 \cdots E_n \twoheadrightarrow_{\rho_{n+1}} B,$$

where each of the $\rho_i$ is either $R$ or $\beta$. Note that, for example, the terms on the reduction path from $A$ to $E_1$ need not be well-typed. (But we want to prove that they are, of course.)

Given a set $R$ of algebraic reduction pseudorules, the reduction relation $\to_R$ on terms of the $\lambda R$-cube not always makes sense. First, because the left-hand side and the right-hand side of a rule could have different types in the same context. Besides, because in the presence of higher-order rules, a situation like the following could arise. Assume that $f \in \mathscr{F}_{s_2 \to s_3 \to s}$, $a \in \mathscr{F}_{s_3}$, $h \in \mathscr{F}_{s_3 \to s}$, and consider the following rewrite pseudorule:

$$r : f(Xx)a \to ha \tag{1}$$

Now the following terms could be legal in the $\lambda R$-cube:

$$\Gamma \vdash_{\lambda_{R-}} M : P(f((\lambda Y : \star . y)x)a), \qquad \Gamma \vdash_{\lambda_{R-}} P(ha) : \star, \tag{2}$$

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

where $y{:}s_2, x{:}\star \in \Gamma$. By the rewrite rule $r$ above, we have $P(f((\lambda Y{:}\star.y)x)a) \to_R P(ha)$ and hence, from (2), by applying rule $(\mathrm{red}_{R\beta})$, we can infer $\Gamma \vdash_R M : P(ha)$. This, however, would have no sense. In fact, in the context $\Gamma$ the term $f((\lambda Y:\star.y)x)a$ has no algebraic meaning, since the term $\lambda Y:\star.y$ has no algebraic interpretation.

To overcome this problem, and have well-typed algebraic rewrite rules, we restrict the set of the algebraic rewriting rules that can be used to extend the $\lambda$-cube. In particular, we will consider rewrite rules whose 'algebraicability' is independent from systems and contexts.

*Definition 2.8* (*Cube-embeddability*)

(i) An algebraic pseudoterm $t$ is *cube-embeddable* if it is a legal term and there exist $\Gamma$ and $\sigma$, unique and algebraic, such that for any $\Gamma'$, $A$ and system $\lambda_{R-}$:

$$\Gamma' \vdash_{\lambda_{R-}} t : A \;\Rightarrow\; (\Gamma'_{|FV(t)} \overset{c}{=}_{R\beta} \Gamma) \;\&\; (A \overset{c}{=}_{R\beta} \sigma).$$

(ii) An algebraic rewrite pseudorule $r : t \to t'$ is *cube-embeddable* if $t$ is cube-embeddable, and $\Gamma \vdash_{\lambda_{R-}} t' : \sigma$, where $\Gamma$ and $\sigma$ are the unique algebraic context and type of $t$. $R$ is cube-embeddable if all its rules are.

*Definition 2.9* (*Algebraic terms and rewrite rules*)
Cube-embeddable algebraic pseudoterms and cube-embeddable algebraic rewrite pseudorules will be called *algebraic terms* and *algebraic rules*, respectively.

A variable $x$ in an algebraic term will be called first-order if $\tau$ $(\in \mathsf{T}_{\mathscr{S}})$ is first-order, where $x{:}\tau \in \Gamma$ and $\Gamma$ is the unique algebraic context of the above definition; $x$ will be called higher-order if $\tau$ is higher-order.

We now give a simple syntactical condition implying cube-embeddability.

*Definition 2.10*
$\mathbf{C}(t)$ is the predicate on algebraic pseudoterms defined as follows:
$\mathbf{C}(t) \iff$ $t$ is a legal term and for any $x \in FV(t)$ there exists a subterm $fP_1 \ldots P_k$ of $t$ such that $f \in \mathscr{F}$ and $P_j \equiv x$ for some $1 \le j \le k$.

It is straightforward to check that such a condition subsumes condition 1 of Definition 2.5(i), since no single variable can satisfy it.

By the Stripping Lemma and Lemma 3.9 for the $\lambda R$-cube (see the next section), and the definition of $\mathbf{C}(t)$, it is easy to check the following:

*Fact 2.11*
Let $t$ be an algebraic pseudoterm: $\mathbf{C}(t) \;\Rightarrow\; t$ is cube-embeddable.

We can now formally define the $\lambda R$-cube.

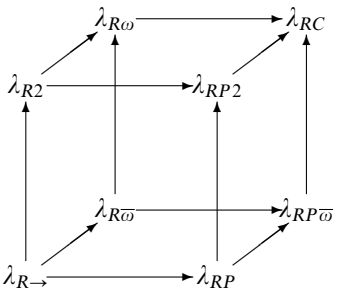
*Definition 2.12* (*The λR-cube*)
Given a cube-embeddable set of algebraic rewrite pseudorules $R$, the *algebraic cube of typed λ-calculi* ($\lambda R$-cube) is defined as the set of the type systems $\lambda_{R\to}$, $\lambda_{R2}$, $\lambda_{RP}$, $\lambda_{R\overline{\omega}}$, $\lambda_{RP2}$, $\lambda_{R\omega}$, $\lambda_{RP\overline{\omega}}$ and $\lambda_{RP\omega}$ (also called $\lambda_{RC}$) defined by the General Axioms and

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

rules above and, respectively, by the following Specific Rules:

$$
\begin{array}{lcl}
\lambda_{R\to} & = & \{\ (\star,\star)\ \} \\
\lambda_{R2} & = & \{\ (\star,\star),\quad (\Box,\star)\ \} \\
\lambda_{RP} & = & \{\ (\star,\star),\quad (\star,\Box)\} \\
\lambda_{R\overline{\omega}} & = & \{\ (\star,\star),\quad (\Box,\Box)\ \} \\
\lambda_{RP2} & = & \{\ (\star,\star),\quad (\Box,\star),\quad (\star,\Box)\} \\
\lambda_{R\omega} & = & \{\ (\star,\star),\quad (\Box,\star),\quad (\Box,\Box)\ \} \\
\lambda_{RP\overline{\omega}} & = & \{\ (\star,\star),\quad (\star,\Box),\quad (\Box,\Box)\ \} \\
\lambda_{RP\omega} & = & \{\ (\star,\star),\quad (\Box,\star),\quad (\star,\Box),\quad (\Box,\Box)\ \}
\end{array}
$$

We can draw a picture of the $\lambda$-cube as follows.




Some systems of the $\lambda R$-cube are already present in the literature. In particular, when the rules in $R$ are first-order, $\lambda_{R\to}$ is the system studied in Breazu-Tannen (1988) and Okada (1989), while $\lambda_{R2}$ is equivalent to the system defined by Breazu-Tannen and Gallier (1991). The systems of Jouannaud and Okada (1991) correspond instead to $\lambda_{R\to}$ and $\lambda_{R2}$. We have already mentioned in the introduction which results were proved for these systems.

*Definition 2.13*
Given a system $\lambda_{R\_}$, we define

$$
\begin{array}{lllll}
\text{(i)} & \mathsf{Context}(\lambda_{R\_}) & = & \{\Gamma \mid \Gamma \vdash_{R\lambda\_} A : B \ \text{ for some pseudoterms } \ A, B\} \\
\text{(ii)} & \mathsf{Term}(\lambda_{R\_}) & = & \{A \mid \Gamma \vdash_{R\lambda\_} A : B \\
 & & & \quad \text{for some pseudoterm } \ B \ \text{ and } \Gamma \in \mathsf{Context}(\lambda_{R\_})\} \\
\text{(iii)} & \mathsf{Kind}(\lambda_{R\_}) & = & \{A \mid \Gamma \vdash_{R\lambda\_} A : \Box \ \text{ for some } \ \Gamma \in \mathsf{Context}(\lambda_{R\_})\} \\
\text{(iv)} & \mathsf{Constr}(\lambda_{R\_}) & = & \{A \mid \Gamma \vdash_{R\lambda\_} A : B : \Box \\
 & & & \quad \text{for some } \ \Gamma \in \mathsf{Context}(\lambda_{R\_}), B \in \mathsf{Term}(\lambda_{R\_})\} \\
\text{(v)} & \mathsf{Type}(\lambda_{R\_}) & = & \{A \mid \Gamma \vdash_{R\lambda\_} A : \star \ \text{ for some } \ \Gamma \in \mathsf{Context}(\lambda_{R\_})\} \\
\text{(vi)} & \mathsf{Object}(\lambda_{R\_}) & = & \{A \mid \Gamma \vdash_{R\lambda\_} A : B : \star \\
 & & & \quad \text{for some } \ \Gamma \in \mathsf{Context}(\lambda_{R\_}), B \in \mathsf{Term}(\lambda_{R\_})\}
\end{array}
$$

Then, according to this definition, *kinds* are those terms that can be typed with $\Box$, *constructors* the ones that can be typed with a kind, *types* the constructors that can be typed with $\star$, and *objects* the terms that can be typed with a type.

The replacement of rule ($\mathrm{red}_{R\beta}$) for the rule (*conv*) of the pure $\lambda$-cube is only needed for systems with dependent types: the following lemma (which will be proved

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

at the end of section 3), shows that such a replacement is useless for systems without the $(\star, \Box)$-rule, i.e. without dependent types.

*Lemma 2.14*
Let $A, M \in \mathsf{Term}(\lambda_{R-})$, and let $\lambda_{R-}$ be any system of the $\lambda R$-cube without the $(\star, \Box)$-rule. $A \in \mathsf{Constr}(\lambda_{R-})$ & $M$ subexpression of $A \Rightarrow M \in \mathsf{Kind}(\lambda_{R-}) \vee M \in \mathsf{Constr}(\lambda_{R-})$.

*Lemma 2.15*
Let $\lambda_{R-}$ be a system of the $\lambda R$-cube without the $(\star, \Box)$-rule. Then in $\lambda_{R-}$ the rule (*conv*) is equivalent to the rule ($red_{R\beta}$).

*Proof*
Consequence of Lemma 2.14 and the Church–Rosser property of the pure $\lambda$-cube. □

*Definition 2.16* (*Reductions on terms*)
We define the reduction relations on terms of the $\lambda R$-cube simply by restricting to the set of terms the reduction relations on pseudoterms defined in Def.2.6.

We have defined the algebraic extension of the $\lambda$-cube with Curried rewriting. Although the usual presentation of term rewriting systems is not Curried (see Dershowitz and Jouannaud, 1990), it is straightforward to show that Currying establishes the expected relation between both presentations: a rule $r : q \to q'$ of a many-sorted uncurried term rewriting system, becomes $c(r) : curry(q) \to curry(q')$ in our framework. It has to be noted, however, that in a Curried system there are unsaturated terms which have no meaning in the context of the usual term rewriting systems. As we will see later, this fact, far from being a drawback, provides flexibility and expressive power to the $\lambda R$-cube. Moreover, our strong normalization result is formulated for rewrite rules that are terminating on the set of first-order algebraic terms, *which coincide in both presentations*. So we do not loose any generality by using Curried rewriting; on the contrary, as a consequence of our main theorem, we obtain a result of preservation of termination under Currying for standard many-sorted term rewriting systems (preservation of strong normalization under Currying for untyped systems was shown by Kennaway *et al.* (1996), whereas preservation of confluence was proven by Kahrs (1995)).

### *2.1 Strong normalization for the λR-cube*

As stated in the introduction, we are interested in the strong normalization property for the systems of the $\lambda R$-cube. However, if unrestricted terminating higher-order rewrite rules are considered it can easily be shown that this property fails. For example, let $HOR$ be the set

$$\{r : f(Xx)xX \to f(Xx)(Xx)X\}.$$

For such terminating set of rules, even the simplest system of the $\lambda R$-cube, $\lambda_{R\to}$, is not strongly normalizing, as can be seen by the following derivation

$$f((\lambda y{:}s.y)x)x(\lambda y{:}s.y) \to_r f((\lambda y{:}s.y)x)((\lambda y{:}s.y)x)(\lambda y{:}s.y) \to_\beta$$
$$\to_\beta f((\lambda y{:}s.y)x)x(\lambda y{:}s.y).$$

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

Then one has necessarily to restrict the notion of higher-order rule in order to get strongly normalizing systems. Following Jouannaud and Okada (1991), we consider higher-order rules that always terminate on algebraic terms thanks to their structure: a generalization of primitive recursion called *general schema*.

Higher-order rewrite rules satisfying the general schema are of wide use in the practice of higher-order rewriting and can be considered as definitions of new functionals of a language.

We will use the notation $t[\vec{v}]$ to indicate that $t$ is a term and $v_1, \ldots, v_n$ are subterms of $t$. So $t[\vec{v}]$ is the same as $t$; such notation only makes appear explicitly some of the subterms of $t$.

*Definition 2.17* (*The general schema* (*Jouannaud and Okada, 1991*))

(i) A higher-order rewrite rule $r : t \to t'$ *satisfies the general schema w.r.t.* $R$ if it is of the form

$$F\ \vec{l}[\vec{X}, \vec{x}]\ \vec{Y}\ \to v[(F\ \vec{q_1}[\vec{X}, \vec{x}]\ \vec{Y}), \ldots, (F\ \vec{q_m}[\vec{X}, \vec{x}]\ \vec{Y})]$$

where $\vec{X}$ and $\vec{Y}$ are sequences of higher-order variables and $\vec{x}$ is a sequence of first-order variables, and

1. $\vec{X} \subseteq \vec{Y}$;[3]
2. $F$ is a higher-order function symbol that can appear neither in the sequences of terms $\vec{l}, \vec{q_1}, \ldots, \vec{q_m}$, nor in the rules of $R$, and its occurrences in $v$ are only the ones explicitly indicated;
3. $v$ is any algebraic term;
4. $\vec{l}, \vec{q_1}, \ldots, \vec{q_m}$ are sequences of terms of sort type;
5. $\forall i \in \{1..m\}$, $\vec{q_i} \lhd_{mul} \vec{l}$ where $\lhd$ denotes strict subterm ordering and $\lhd_{mul}$ its multiset extension, defined as usual. (If $<$ is a partial ordering on $S$, then the ordering $<_{mul}$ on multisets of elements of $S$ is the transitive closure of the replacement of an element with any finite number, including zero, of elements that are smaller under $<$.)

(ii) A set $HOR = \{l_i \to r_i\}_{i \in I}$ of higher-order rewrite rules *satisfies the general schema* (w.r.t. $FOR$) if each rule $l_i \to r_i \in HOR$ satisfies the general schema w.r.t. $FOR \cup \{l_j \to r_j\}_{j<i}$. This implies that there is no mutual recursion in $HOR$.

*Remark 2.18*

It would be possible to consider more powerful higher-order rules allowing $v$ to be a term in $\lambda_{R\to}$, or $\lambda_{R\omega}$. To do so, however, even if it would not modify the proof of strong normalization, would make the definition of the $\lambda R$-cube more involved The definition of the $\lambda R$-cube containing such extended higher-order rewrite rules can be found in Barbanera *et al.* (1994).

Notice that some of the conditions given in the definition of the general schema can be loosened. Condition (i).1 could be removed by reasoning on a transformed

[3] Notice that this condition implies $\mathbf{C}(F\ \vec{l}[\vec{X}, \vec{x}]\ \vec{Y})$ and hence that $F\ \vec{l}[\vec{X}, \vec{x}]\ \vec{Y}$ is cube-embeddable.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

version of $F$, while condition (i).2 could be removed by introducing product types and packing mutually recursive definitions in the same product.

As said above, although restricted, the general schema is interesting from a practical point of view: it allows the introduction of functional constants of higher-order types by primitive recursion on a first-order data structure.

Let us show some examples.

*Example 2.19*

Consider the signature of lists, with constructors cons and nil. The function append (that concatenates two lists) can be defined by a set $FOR$ of first-order rules (Jouannaud and Okada, 1991):

$\mathsf{append}\ \mathsf{nil}\ l \to l$
$\mathsf{append}\ (\mathsf{cons}\ x\ l)\ l' \to \mathsf{cons}\ x\ (\mathsf{append}\ l\ l')$
$\mathsf{append}\ (\mathsf{append}\ l\ l')\ l'' \to \mathsf{append}\ l\ (\mathsf{append}\ l'\ l'')$

The functional map, which applies a function to all the elements of a list, can be defined using two higher-order rules:

$\mathsf{map}\ \mathsf{nil}\ X \to \mathsf{nil}$
$\mathsf{map}\ (\mathsf{cons}\ x\ l)\ X \to \mathsf{cons}\ (X\ x)\ (\mathsf{map}\ l\ X)$

Here, append is defined algebraically (the third rule establishes the associativity of append on lists) while the higher-order function map is defined recursively on the structure of lists. Its definition satisfies the general schema.

We will show another example using lists:

*Example 2.20*

foldr is a very useful higher-order function, whose informal meaning is the following: Let $\langle x_1, \ldots, x_n \rangle$ denote the list containing the elements $x_1, \ldots, x_n$, then

$$\mathsf{foldr}\ a\ \langle x_1, \ldots, x_n \rangle\ f = f x_1 (f x_2 (\ldots (f x_n a) \ldots))$$

where $f$ is a function and $a$ is a constant.

It is easy to define foldr by a set of higher-order rules satisfying the general schema:

$\mathsf{foldr}\ a\ \mathsf{nil}\ X \to a$
$\mathsf{foldr}\ a\ (\mathsf{cons}\ x\ l)\ X \to X\ x\ (\mathsf{foldr}\ a\ l\ X)$

Now, using foldr, and assuming that $+$, $\times$, 0 and 1 are already defined, we can define the (unary) functions sum and product:

$\mathsf{sum}\ l \to \mathsf{foldr}\ 0\ l\ +$
$\mathsf{product}\ l \to \mathsf{foldr}\ 1\ l\ \times$

The function sum adds the elements of a list of numbers, while product multiplies them. Moreover, assume that append is defined as in the previous example, then we can define the (unary) function concat

$\mathsf{concat}\ x \to \mathsf{foldr}\ \mathsf{nil}\ x\ \mathsf{append}$

The function concat concatenates a list of lists into one long list.

The higher-order rewrite rules defining foldr, sum, product and concat satisfy the

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

general scheme, then, as a consequence of the 'main theorem' that we will prove later, the union of the above defined rewrite systems is strongly normalizing.

We have seen that unrestricted higher-order rewrite rules could prevent the strong normalization property to hold and have proposed a restriction. This, however, is not still sufficient to get modularity of strong normalization, even if we only consider reductions on algebraic terms. It is in fact possible to code Toyama's example of non-termination (Toyama, 1987), as shown below.

*Example 2.21*
Consider the following sets of rewrite rules:

$$FOR = \{r_1 : f01x \to fxxx\}$$
$$HOR = \{r_2 : FX \to 1, r_3 : FX \to 0\}$$

where $F$ is a higher-order function symbol, $f$ is a first-order function symbol, $0, 1$ are first-order constants and $X, x$ are variables. $R$ is terminating and $HOR$ satisfies the general schema, nonetheless there exists an infinite reduction sequence:

$$f(FX)(FX)(FX) \to_{r_3} f0(FX)(FX) \to_{r_2} f01(FX) \to_{r_1}$$
$$\to_{r_1} f(FX)(FX)(FX) \to_{r_3} \ldots$$

Confluence of $HOR$ does not suffice to restore the strong normalization property. This can be seen by coding the example of Barendregt and Klop (see Toyama, 1987).

The cause of non-termination in the example is that rule $r_1$ is *duplicating*, i.e. there are more occurrences of the variable $x$ in its right-hand side than in its left-hand side. We will show that the restriction of $FOR$ to *non-duplicating* rules (also called *conservative*), together with the general schema condition for $HOR$, imply the modularity of Strong Normalization in the $\lambda R$-cube.

*Definition 2.22*

(i) A rewrite rule $r : t \to t'$ is *non-duplicating* if, for any variable, the number of its occurrences in $t$ is greater than or equal to the number of its occurrences in $t'$.

(ii) A set of rewrite rules is non-duplicating if each of them is so.

*Example 2.23*
The first-order system defining the function append in Example 2.19 is non-duplicating.

The restriction to non-duplicating rules appeared first in the work of Rusinowitch (1987), who studied modularity of termination of unions of first-order term rewriting systems.

We can now state our main result.

*Theorem 2.24* (*Main Theorem*)
Let $R$ be a cube-embeddable set of rewrite rules such that

1. $FOR$ is non-duplicating and strongly normalizing on first-order algebraic terms;

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

2. $HOR$ satisfies the general schema (w.r.t. $FOR$).

Then the systems of the $\lambda R$-cube are strongly normalizing w.r.t. $\rightarrow_{R\beta}$.

The restriction to non-duplicating first-order rules is not needed if we consider only first-order rules.

The rest of the paper will be devoted to the proof of the main theorem. Since all the systems of the $\lambda R$-cube are subsystems of $\lambda_{RC}$, the proof of the main theorem will be given for $\lambda_{RC}$, and hence from now on every notion we shall refer to (if not otherwise stated) will be of $\lambda_{RC}$.

## 3 Metatheory of the $\lambda R$-cube

In this section we present the syntactical properties of $\lambda_{RC}$ that will be used in the proof of the main theorem. The proofs of some of them are straightforward extensions of the corresponding proofs for the $\lambda$-cube, and will be omitted. Complete proofs will be given instead for those properties, like subject reduction, that require the development of some technical machinery.

*Proposition 3.1* (*Substitution*)
For $\Gamma_1, x{:}A, \Gamma_2$ a context, $M$, $B$ and $N$ terms,

$$\left.\begin{array}{r}\Gamma_1, x{:}A, \Gamma_2 \vdash M : B \\ \Gamma_1 \vdash N : A\end{array}\right\} \Rightarrow \Gamma_1, \Gamma_2[N/x] \vdash M[N/x] : B[N/x].$$

Note this also implies that, if $\Gamma, x{:}A \vdash M, N : C$ with $M \stackrel{c}{=}_{R\beta} N$ and $\Gamma \vdash Q : A$, then $M[Q/x] \stackrel{c}{=}_{R\beta} N[Q/x]$.

*Lemma 3.2* (*Thinning for the $\lambda R$-cube*)
For $\Gamma$ and $\Gamma'$ contexts, $M$ and $N$ terms, we have the following.

$$\Gamma \vdash M : N \;\&\; \Gamma \subset \Gamma' \Rightarrow \Gamma' \vdash M : N.$$

*Lemma 3.3* (*Stripping for the $\lambda R$-cube*)
For $\Gamma$ a context, $M$, $N$ and $R$ terms, we have the following:

(i) $\Gamma \vdash p : R$, with $p \in \{\star, \Box\}$ $\Rightarrow$ $p \equiv \star$, $R \stackrel{c}{=}_{R\beta} \Box$,
(ii) $\Gamma \vdash s : R$, with $s \in \mathscr{S}$ $\Rightarrow$ $R \stackrel{c}{=}_{R\beta} \star$,
(iii) $\Gamma \vdash x : R$, with $x \in \mathsf{Var}$ $\Rightarrow$ $R \stackrel{c}{=}_{R\beta} A$ with $x : A \in \Gamma$ for some term $A$,
(iv) $\Gamma \vdash f : R$, with $f \in \mathscr{F}_\tau$ $\Rightarrow$ $R \stackrel{c}{=}_{R\beta} \tau$
(v) $\Gamma \vdash \Pi x{:}M.N : R$ $\Rightarrow$ $\Gamma \vdash M : p_1, \Gamma, x{:}M \vdash N : p_2$ and $R \stackrel{c}{=}_{R\beta} p_2$ for some $p_1, p_2 \in \{\star, \Box\}$, and rule $(p_1, p_2)$
(vi) $\Gamma \vdash \lambda x{:}M.N : R$ $\Rightarrow$ $\Gamma, x{:}M \vdash N : B, \Gamma \vdash \Pi x{:}M.B : p$ and $R \stackrel{c}{=}_{R\beta} \Pi x{:}M.B$ for some term $B$ and $p \in \{\star, \Box\}$,
(vii) $\Gamma \vdash MN : R$ $\Rightarrow$ $\Gamma \vdash M : \Pi x{:}A.B, \Gamma \vdash N : A$ with $R \stackrel{c}{=}_{R\beta} B[N/x]$ for some terms $A$ and $B$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Proof*
The proof is easy. We can go up in the derivation tree until we reach the point where the term has been formed. In doing this we only pass through applications of the conversion or weakening rule. At the point where the term has been formed, we distinguish the seven different cases above, according to the form of the term, and we easily check that the conclusions are satisfied. □

*Lemma 3.4* (*Correctness of types*)
For $\Gamma$ a context, $M$ and $A$ terms,

$$\Gamma \vdash M : A \Rightarrow \exists p \in \{\star, \Box\}[A \equiv p \vee \Gamma \vdash A : p].$$

*Proof*
The proof can be given by analysing the derivation tree of $\Gamma \vdash M : A$, like in the proof of 3.3, but also by induction on the derivation of $\Gamma \vdash M : A$. We follow the second option, which gives the shortest proof. The only case that has some interest is when the last rule is (app).

(app)

$$\frac{\Gamma \vdash P : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash PN : B[N/x]}$$

Then $\Gamma \vdash \Pi x{:}A.B : p$ by induction, and hence by Stripping (Lemma 3.3), $\Gamma, x{:}A \vdash B : p'$ for some $p' \in \{\star, \Box\}$. Now by Substitution (Proposition 3.1), we conclude that $\Gamma \vdash B[N/x] : p'$. □

*Lemma 3.5*
(i) $\Gamma \vdash \Pi u{:}C.\Box : D$ is not possible,
(ii) $N \in \mathsf{Term}$ with $N \stackrel{c}{=}_{R\beta} p \in \{\star, \Box\}$, then $N \equiv p$.
(iii) $M \equiv \Pi \vec{x}{:}\vec{A}.\star$ and $M$ typable in $\Gamma \Leftrightarrow \Gamma \vdash M : \Box$.

*Proof*
(i) First note that $\Box$ does not have a type. This immediately proves the thesis.
(ii) If $N \stackrel{c}{=}_{R\beta} p$ with $N \not\equiv p$ then necessarily $N' \to_R p \in \{\star, \Box\}$ or $N' \to_\beta p \in \{\star, \Box\}$ for some $N'$. The first case is not possible for typing reasons. In the second case, $N' \equiv (\lambda x{:}A.M)Q$, with $p$ occurring as subterm of $M$ or $Q$ in applied or abstracted form. Now, by stripping this term we find that $\Box$ has a type, which is not the case. Then $N \equiv p$.
(iii) ($\Leftarrow$) is the most interesting implication; the proof uses the second item. First one proves that $\Gamma \vdash PQ : \Box$ is impossible and then one proves $\Gamma \vdash \Pi x{:}C.D : \Box \Rightarrow \Gamma, x{:}C \vdash D : \Box$ and we are done. Suppose $\Gamma \vdash PQ : \Box$. Then $\Gamma \vdash P : \Pi x{:}A.B$ with $B[Q/x] \stackrel{c}{=} \Box$, hence $B[Q/x] \equiv \Box$. But then $B \equiv \Box$ or $B \equiv x$ and $Q \equiv \Box$, which are both easily falsified. Suppose now that $\Gamma \vdash \Pi x{:}C.D : \Box$. Then $\Gamma, x{:}C \vdash D : p$ with $p \stackrel{c}{=} \Box$, hence $p \equiv \Box$. For ($\Rightarrow$) one uses the Stripping Lemma: part (i) if $\vec{x}$ is an empty sequence, and part (v) otherwise. □

*Lemma 3.6*
$\Gamma \vdash \Pi x{:}A.B : C \Rightarrow C \in \{\star, \Box\}$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Proof*
$C \overset{c}{=}_{R\beta} \star$ or $C \overset{c}{=}_{R\beta} \Box$ by Stripping ($v$). But then, by Lemma 3.5 ($ii$), $C \equiv \star$ or $C \equiv \Box$.
□

*Lemma 3.7*

$$\left.\begin{array}{c}\Gamma \vdash M : \Box \\ N \in \mathsf{Term} \\ N \overset{c}{=}_{R\beta} M\end{array}\right\} \Rightarrow \Gamma \vdash N : \Box.$$

*Proof*
We do the proof for $M \twoheadrightarrow_{R\beta} N$ and $N \twoheadrightarrow_{R\beta} M$, the general case follows by induction on the definition of $N \overset{c}{=}_{R\beta} M$. Recall that $N \overset{c}{=}_{R\beta} M$ means that $M$ and $N$ are equal via expansions and reductions that remain inside the set of well-typed terms. By Lemma 3.5.($iii$)($\Leftarrow$) we have that $M \equiv \Pi\vec{x}{:}\vec{A}.\star$.

- $M \twoheadrightarrow_\beta N$. Then $N \equiv \Pi\vec{x}{:}\vec{A}'.\star$ with $\vec{A} \twoheadrightarrow_\beta \vec{A}'$, so $\Gamma \vdash N : \Box$. (Use Lemma 3.5.($iii$) ($\Rightarrow$).) If $M \twoheadrightarrow_R N$, then we are done similarly.
- $N \twoheadrightarrow_\beta M$. We prove that $N$ must be of the form $\Pi\vec{x}{:}\vec{D}.\star$ and we are done. Suppose $N \equiv (\lambda\vec{y}{:}\vec{B}.P)\vec{Q}$. Then either $P$ or $Q$ contains a subterm of the form $\Pi\vec{z}{:}\vec{C}.\star$. We even know that this subterm occurs as $\lambda q{:}R.\Pi\vec{z}{:}\vec{C}.\star$ or as $R(\Pi\vec{z}{:}\vec{C}.\star)$. Both are impossible for typing reasons. If $N \twoheadrightarrow_R M$ the thesis is a consequence of the cube-embeddability of the rewrite rules.

□

*Corollary 3.8* (*βR-preservation of* $\star$ *and* $\Box$)
Let $p, p' \in \{\star, \Box\}$.

$$\left.\begin{array}{c}\Gamma \vdash M : p \\ \Gamma' \vdash N : p' \\ N \overset{c}{=}_{R\beta} M\end{array}\right\} \Rightarrow p \equiv p'.$$

*Proof*
Suppose $p \equiv \Box$, i.e. $\Gamma \vdash M : \Box$. Then $\Gamma' \vdash N : \Box$ by Lemma 3.7 and so $p \equiv p'$. Similarly if $p' \equiv \Box$. Hence $p \equiv \Box$ iff $p' \equiv \Box$. □

The following lemma is hard to prove, because we cannot rely on the Church–Rosser property of $\rightarrow_{R\beta}$.

*Lemma 3.9*
Let $\sigma_1, \sigma_2 \in \mathsf{T}_{\mathscr{S}}$. $\sigma_1 \overset{c}{=}_{R\beta} \sigma_2 \Rightarrow \sigma_1 \equiv \sigma_2$.

*Proof*
In section 5 a translation $\tau : \{\Box\} \cup \mathsf{Kind}(\lambda_{RC}) \cup \mathsf{Constr}(\lambda_{RC}) \rightarrow \mathsf{Term}(\lambda_{R\omega})$ will be defined, such that $\tau(\sigma) \equiv \sigma$ for $\sigma \in \mathsf{T}_{\mathscr{S}}$. Moreover, for such a translation Lemma 5.6 will ensure that $\tau(A) =_\beta \tau(B)$ in case $A \overset{c}{=}_{R\beta} B$. Hence, from $\sigma_1 \overset{c}{=}_{R\beta} \sigma_2$ we get $\sigma_1 \equiv \tau(\sigma_1) =_\beta \tau(\sigma_2) \equiv \sigma_2$. By Church–Rosser for $=_\beta$ it follows $\sigma_1 \equiv \sigma_2$, since no reduction is applicable to $\sigma_1$ and $\sigma_2$. It is not difficult to check that referring to Lemma 5.6 as we have done above creates no circularity. We refer to it only in order not to duplicate its proof here. □

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Lemma 3.10* (*Classification*)

$$\begin{aligned} \mathsf{Type} \cap \mathsf{Kind} &= \emptyset, \\ \mathsf{Obj} \cap \mathsf{Constr} &= \emptyset. \end{aligned}$$

*Proof*
For the former, it suffices to prove the following:

$$\Gamma \vdash M : p,\ \Gamma' \vdash M : p' \Rightarrow p \equiv p'. \tag{3}$$

This is an immediate consequence of Corollary 3.8. For the latter, it suffices to prove the following property:

$$\Gamma \vdash M : B : p,\ \Gamma' \vdash M : B' : p' \Rightarrow p \equiv p'. \tag{4}$$

We prove this statement by induction on the structure of $M$, using $\beta R$-preservation of $\star$ and $\Box$. The proof is not really difficult but still a bit tricky, and we therefore give it in quite some detail.

var It suffices to show that, if $\Gamma \vdash x : B : p$ with $x \in \mathsf{Var}^{p_0}$, then $p \equiv p_0$. Now, if $\Gamma \vdash x : B$, then $x : A \in \Gamma$ with $\Gamma \vdash A : p_0$ and $A \overset{c}{=}_{R\beta} B$. Hence by $\beta R$-preservation of $\star$ and $\Box$ (Corollary 3.8), $p \equiv p_0$.

$s \in \mathscr{S}$ If $\Gamma \vdash s : B$ and $\Gamma' \vdash s : B'$ then, by Stripping, we get $B' \overset{c}{=}_{R\beta} \star \overset{c}{=}_{R\beta} B$. Hence, by $\beta R$-preservation of $\star$ and $\Box$, $p \equiv p'$.

$f \in \mathscr{F}$ If $\Gamma \vdash f : B$ and $\Gamma' \vdash f : B'$, then, by Stripping($iv$), we get $B \overset{c}{=}_{R\beta} \tau \overset{c}{=}_{R\beta} B'$. Hence, by $\beta R$-preservation of $\star$ and $\Box$, $p \equiv p'$.

$\Pi$ If $\Gamma \vdash \Pi x{:}A.B : C : p$ and $\Gamma' \vdash \Pi x{:}A.B : C' : p'$, then, by Stripping ($v$), we get $C \overset{c}{=}_{R\beta} p_0$ and $C' \overset{c}{=}_{R\beta} p_0'$ with $p_0, p_0' \in \{\star, \Box\}$. It follows that $\Gamma \vdash \Pi x{:}A.B : p_0$ and $\Gamma' \vdash \Pi x{:}A.B : p_0'$. Hence, by property (3), $p_0 \equiv p_0'$, from which $C \overset{c}{=}_{R\beta} C'$. $\beta R$-preservation of $\star$ and $\Box$ allows now to infer $p \equiv p'$.

$\lambda$ Suppose that $\Gamma \vdash \lambda x{:}A.M : B : p$ and $\Gamma' \vdash \lambda x{:}A.M : B' : p'$. Then, by Stripping ($vi$), $B \overset{c}{=} \Pi x{:}A.C$ and $\Gamma \vdash \Pi x{:}A.C : \overline{p}$ with $\overline{p} \equiv p$ by $\beta R$—preservation of $\star$ and $\Box$. By Stripping again, case ($v$), we have also $\Gamma \vdash A : p_1$ and $\Gamma, x{:}A \vdash M : C : p$ (we have the same $p$ in $\Gamma, x{:}A \vdash M : C : p$ and $\Gamma \vdash \Pi x{:}A.C : p$ by Lemma 3.5(ii)) Similarly $B' \overset{c}{=} \Pi x{:}A.C'$ with $\Gamma' \vdash A : p_1'$, $\Gamma', x{:}A \vdash M : C' : p'$ and $\Gamma' \vdash \Pi x{:}A.C' : p'$. Now, by induction $p \equiv p'$.

app Let $\Gamma \vdash MN : D : p$ and $\Gamma' \vdash MN : D' : p'$. Then, by Stripping, Correctness of Types, $\beta R$-preservation of $\star$ and $\Box$ and Substitution, we have that $\Gamma \vdash M : \Pi x{:}A.B : p_2$, $\Gamma \vdash N : A : p_1$, $\Gamma \vdash B[N/x] : p_2$ and $B[N/x] \overset{c}{=}_{R\beta} D$. At the same time $\Gamma' \vdash M : \Pi x{:}A'.B' : p_2'$, $\Gamma' \vdash N : A' : p_1'$, $\Gamma' \vdash B'[N/x] : p_2'$ and $B'[N/x] \overset{c}{=}_{R\beta} D'$. Now, by induction, $p_2 \equiv p_2'$. Also, by $\beta R$-preservation of $\star$ and $\Box$, $p \equiv p_2$ and $p' \equiv p_2'$ and so $p \equiv p'$.

□

It still remains to prove Lemma 2.14, for which we need the following technical lemma:

*Lemma 3.11*
Let $K$ be the set inductively defined by

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

1. $\star \in K$
2. $k_1, k_2 \in K \Rightarrow k_1 \to k_2 \in K$

Then $\Gamma \vdash_{\lambda_{R-}} k : \Box \Rightarrow k \in K$, where $\lambda_{R-}$ is any system of the $\lambda R$-cube without $(\star, \Box)$-rules.

*Proof*
By induction on the structure of $k$ using Lemma 3.5(iii). □

*Lemma 2.14*
For $A, M \in \mathsf{Term}(\lambda_{R-})$ and $\lambda_{R-}$ any system of the $\lambda R$-cube without $(\star, \Box)$-rules:

$$A \in \mathsf{Constr}(\lambda_{R-}) \;\&\; M \text{ subexpression of } A \Rightarrow M \in \mathsf{Kind}(\lambda_{R-}) \vee M \in \mathsf{Constr}(\lambda_{R-})$$

*Proof*
By induction on the structure of $A$. For the $A$ variable the thesis follows immediately. In case $A$ is of the form $\Pi x{:}P.Q$, $\lambda x{:}P.Q$ or $PQ$, since we do not have $(\star, \Box)$-rules, $P$ and $Q$ have necessarily to be constructors or kinds, then by the induction hypothesis or Lemma 3.11, so are all their subexpressions. □

### *3.1 The subject reduction property*

That of subject reduction is a property that, as stated before, requires more effort to be proved in $\lambda_{RC}$ than in $\lambda_C$, because we cannot rely on the Church–Rosser property for pseudoterms. In the following we will prove separately subject reduction for $\twoheadrightarrow_R$ and for $\twoheadrightarrow_\beta$.

*Lemma 3.12*
Let $r : t \to t'$ be a rewriting rule in $R$. $\Gamma \vdash t[\vec{N}/\vec{x}] : A \Rightarrow \Gamma \vdash t'[\vec{N}/\vec{x}] : A$.

*Proof*
Let $B_i$ $(1 \le i \le n,\ n = |N|)$ be the type of $N_i$ in the first statement $\Gamma_i' \vdash N_i : B_i$ that we meet going up in the derivation of $\Gamma \vdash t[\vec{N}/\vec{x}] : A$ (since $t$ is algebraic, these statements will be in applications of rule (app)). It is easy to check that it is possible to modify the derivation in order to get a derivation for: $\vec{\Gamma}', \vec{x}{:}\vec{B}, \vec{\Gamma}'' \vdash t : A$. By definition of reduction rule we can infer $\vec{\Gamma}', \vec{x}{:}\vec{B}, \vec{\Gamma}'' \vdash t' : A$. Hence, since we have also that $\Gamma_i' \vdash N_i : B_i$ $(1 \le i \le n)$, by Substitution (Proposition 3.1), it follows $\Gamma \vdash t'[\vec{N}/\vec{x}] : A$. □

*Proposition 3.13* (*Subject Reduction Lemma for Rewriting, $SR_R$*)
For $\Gamma$ a context, $P, P'$ and $D$ terms,

$$\Gamma \vdash P{:}D \;\&\; P \twoheadrightarrow_R P' \Rightarrow \Gamma \vdash P'{:}D.$$

*Proof*
By induction on the derivation of $\Gamma \vdash P{:}D$. The only case which is not trivial or does not follow from the induction hypothesis is when, by the application of rule (app), one gets an $R$-redex. The thesis in such a case is a consequence of Lemma 3.12. □

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

It turns out that Subject Reduction for $\beta$ is a much harder nut to crack. The standard proof is by induction on the derivation. Here we run into a problem with the case:

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

where $(\lambda x{:}C.M)N \to_\beta M[N/x]$ and we want to show that $\Gamma \vdash M[N/x] : B[N/x]$. By Stripping we conclude that

$$\Gamma \vdash \lambda x{:}C.M : \Pi x{:}C.D$$

with

$$\Pi x{:}C.D \overset{c}{=}_{R\beta} \Pi x{:}A.B,$$

and we know that in the reduction-expansion chain from $\Pi x{:}C.D$ to $\Pi x{:}A.B$ we only have types and kinds, but we cannot conclude from this that $C =_{\beta R} A$ and $D =_{\beta R} B$, because we don't have Church–Rosser. Notice that the problem is even more difficult: we have to show that $C \overset{c}{=}_{R\beta} A$ and $D \overset{c}{=}_{R\beta} B$.

*Definition 3.14*
The $\lambda$-abstractions in a well-typed term of our system (but the definition immediately extends to pseudoterms) are split into four classes, the 0-, 2-, $P$- and $\omega$-abstractions, as follows.

1. $\lambda x{:}A.M$ is a 0-*abstraction* if $M$ is an object, $A$ a type,
2. $\lambda \alpha{:}A.M$ is a 2-*abstraction* if $M$ is an object, $A$ a kind,
3. $\lambda x{:}A.M$ is a $P$-*abstraction* if $M$ is a constructor, $A$ a type,
4. $\lambda \alpha{:}A.M$ is a $\omega$-*abstraction* if $M$ is a constructor, $A$ a kind.

We can decorate the $\lambda$s correspondingly, so we can speak of the $\lambda_0$s of a term, etc. We now also define the notions of $\beta^0$-reduction, $\beta^2$-reduction, $\beta^P$-reduction and $\beta^\omega$-reduction by just restricting reduction to the redexes with the appropriate subscript attached to the symbol $\lambda$. We use an arrow with a superscript above it to denote these restricted reductions (or the union of some of them), so $\overset{P\omega}{\to}_\beta$, etc.

We also have the usual notion of $\beta$-*weak-head-reduction*: a term $M$ $\beta$-*weak-head-reduces* to $M'$, notation $M \to_{wh} M'$, if $M$ is itself a $\beta$-redex, say $M \equiv (\lambda x{:}A.P)Q$ and $M'$ is obtained by contracting $(\lambda x{:}A.P)Q$ (the $\beta$-*head-redex*). In the following, we shall not consider (weak)-head-reduction, where the head-redex is an $R$-redex. Therefore, we omit explicit reference to $\beta$ and refer to $\beta$-weak-head-reduction as *weak-head-reduction*. As usual, we define $\twoheadrightarrow_{wh}$ as the transitive reflexive closure of $\to_{wh}$. Note that a term of the form $\Pi x{:}A.B$ is in *weak-head-normal-form* (whnf).

*Lemma 3.15* ($SR_{\beta^{P\omega}}$)
For $\Gamma$ and $\Gamma'$ contexts, $P, P'$ and $D$ terms,

$$\Gamma \vdash P : D \;\&\; P \overset{P\omega}{\to}_\beta P' \quad \Rightarrow \quad \Gamma \vdash P' : D$$
$$\Gamma \vdash P : D \;\&\; \Gamma \overset{P\omega}{\to}_\beta \Gamma' \quad \Rightarrow \quad \Gamma' \vdash P : D.$$

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Proof*

The proof is by simultaneous induction on the derivation. Just as in the usual proof for $\beta$, the only interesting case is when the last rule is (app) with $P \equiv (\lambda x{:}A.M)N$ and $P' \equiv M[N/x]$. We then have

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

where the $\lambda$ is a $\lambda_P$ or a $\lambda_\omega$. Then by applying Stripping (Lemma 3.3) to the first premise, we find

$$\begin{gathered} \Gamma, x{:}A \vdash M : C' \quad (1) \\ \Gamma \vdash \Pi x{:}A.C' : p\,(\in \{\star, \Box\}) \\ \Pi x{:}A.C' \overset{c}{=}_{R\beta} \Pi x{:}B.C. \end{gathered}$$

So, again by Stripping

$$\begin{gathered} \Gamma \vdash A : p_1 \quad (2) \\ \Gamma, x{:}A \vdash C' : p \\ \text{for some } p_1, p \in \{\star, \Box\}. \end{gathered}$$

By the fact that we have a $P$- or $\omega$-redex, we know that $\lambda x{:}A.M$ is a constructor and hence that $\Pi x{:}A.C'$ and $\Pi x{:}B.C$ are kinds. So by Lemma 3.5(iii), we know that $\Pi x{:}A.C' \equiv \Pi \vec{x}{:}\vec{A}.\star$ and $\Pi x{:}B.C \equiv \Pi \vec{x}{:}\vec{B}.\star$. Now, by (1), $\Pi \vec{x}{:}\vec{A}.\star \overset{c}{=}_{R\beta} \Pi \vec{x}{:}\vec{B}.\star$, so all the well-typed terms on the reduction-expansion path from $\Pi \vec{x}{:}\vec{A}.\star$ to $\Pi \vec{x}{:}\vec{B}.\star$ are of the form $\Pi \vec{x}{:}\vec{C}.\star$, because of Lemmas 3.7 and 3.5. Hence

$$\begin{gathered} A \overset{c}{=}_{R\beta} B \quad (3) \\ C' \overset{c}{=}_{R\beta} C. \quad (4) \end{gathered}$$

So, applying the conversion rules to (2) and $\Gamma \vdash N : B$, using (3), we get

$$\Gamma \vdash N : A. \quad (5)$$

Applying Substitution (Proposition 3.1) to (5) and (1) we get

$$\Gamma \vdash M[N/x] : C'[N/x]. \quad (6)$$

By applying Correctness of Types (Lemma 3.4) to the first premise, we find $\Gamma \vdash \Pi x{:}B.C : p'$ for some $p' \in \{\star, \Box\}$ and hence by Stripping

$$\Gamma, x{:}B \vdash C : p'(\in \{\star, \Box\}). \quad (7)$$

Now apply Substitution to $\Gamma \vdash N : B$ and (7) to get

$$\Gamma \vdash C[N/x] : p'. \quad (8)$$

Apply the conversion rules to (6) and (8) (using (4)) to conclude

$$\Gamma \vdash M[N/x] : C[N/x]$$

and we are done. □

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Corollary 3.16* (*Stability of $\beta^{P\omega}$-redexes under $\overset{P\omega}{\twoheadrightarrow_\beta}$*)
For $M$ a term, if

$$M \equiv C[(\lambda x{:}A.N)Q] \overset{P\omega}{\twoheadrightarrow_\beta} C'[(\lambda x{:}A'.N')Q'] \equiv M'$$

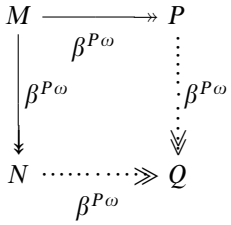
and $(\lambda x{:}A.N)Q$ is a $P$- or $\omega$-redex in $M$, then $(\lambda x{:}A'.N')Q'$ is a $P$- or $\omega$-redex in $M'$.

*Proof*
If $N$ is a constructor in $M$, then $N'$ is a constructor in $M'$ by subject reduction for $\beta^{P\omega}$. □

*Lemma 3.17* (*Confluence of $\beta^{P\omega}$*)
For $M, P, N \in \mathsf{Term}$,

$$M \overset{P\omega}{\twoheadrightarrow_\beta} P \;\&\; M \overset{P\omega}{\twoheadrightarrow_\beta} N \Rightarrow \exists Q[P \overset{P\omega}{\twoheadrightarrow_\beta} Q \;\&\; N \overset{P\omega}{\twoheadrightarrow_\beta} Q].$$

In a diagram




*Proof*
By completeness of developments and the fact that $\beta^{P\omega}$-redexes are stable under $\overset{P\omega}{\twoheadrightarrow_\beta}$. (Corollary 3.16) □

*Lemma 3.18*
If $\Gamma \vdash (\lambda x{:}C.P)Q : p \in \{\star, \Box\}$, then this is a $P$- or $\omega$-redex.

*Proof*
First we know by Lemma 3.5 that $p$ must be $\star$. By Stripping we find $A, B$ and $D$ such that

$$\begin{array}{rcl}
\Gamma, x{:}C & \vdash & P : B, \\
\Gamma & \vdash & Q : A, \\
\Gamma & \vdash & \lambda x{:}C.P : \Pi x{:}C.B, \\
\Pi x{:}C.B & \overset{c}{=}_{R\beta} & \Pi x{:}A.D, \\
D[Q/x] & \overset{c}{=}_{R\beta} & \star
\end{array}$$

Hence $D[Q/x] \equiv \star$ and so $D \equiv \star$, because $Q$ can't be $\star$. (This follows from Stripping and Lemma 3.5.) Hence $\Pi x{:}A.D : \Box$ and so $\Pi x{:}C.B : \Box$ by $\beta R$-preservation of elements of $\{\star, \Box\}$. Hence $B \equiv \Pi \vec{y}{:}\ldots.\star$ and so $B : \Box$ and $P$ is a constructor. We conclude that the $\lambda$ is decorated with $P$ or $\omega$ and we are done. □

As a consequence, we find that subject reduction holds for weak-head-reduction on types. That is

$$\Gamma \vdash M : \star, M \twoheadrightarrow_{wh} N \Rightarrow \Gamma \vdash N : \star.$$

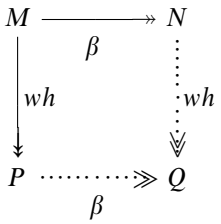
https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

We now formulate two useful lemmas on the interaction between $\twoheadrightarrow_R$, $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_{wh}$. Note that the properties hold for *pseudoterms*.

*Lemma 3.19* (*Commutativity of weak-head-reduction and β-reduction*)
For $M, P, N \in T$,

$$M \twoheadrightarrow_{wh} P \;\&\; M \twoheadrightarrow_\beta N \Rightarrow \exists Q[P \twoheadrightarrow_\beta Q \;\&\; N \twoheadrightarrow_{wh} Q].$$

In a diagram

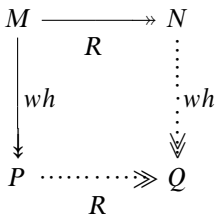



*Proof*
Recall that *weak-head-reduction* refers only to $\beta$-weak-head-reduction. We prove the lemma for the case of a one step weak-head-reduction: if $M \to_{wh} P \;\&\; M \twoheadrightarrow_\beta N$, then $P \twoheadrightarrow_\beta Q \;\&\; N \to_{wh} Q$ for some $Q$. This immediately implies the general case, as can be seen by filling in the diagram. So, let $M \to_{wh} P$, say $M \equiv (\lambda x{:}A.B)C$. If $N \equiv (\lambda x{:}A'.B')C'$ with $A \twoheadrightarrow_\beta A'$, $B \twoheadrightarrow_\beta B'$ and $C \twoheadrightarrow_\beta C'$ we are done by taking $Q \equiv B'[C'/x]$. If in the reduction from $M$ to $N$ the head-redex is contracted we are also done, by taking $Q \equiv N$. □

*Lemma 3.20* (*Commutativity of weak-head-reduction and R-reduction*)
For $M, P, N \in T$,

$$M \twoheadrightarrow_{wh} P \;\&\; M \twoheadrightarrow_R N \Rightarrow \exists Q[P \twoheadrightarrow_R Q \;\&\; N \twoheadrightarrow_{wh} Q].$$

In a diagram




*Proof*
The proof is exactly the same as the proof for Lemma 3.19. □

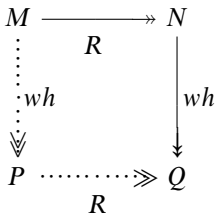
We also have postponement of $R$-reduction with respect to weak-head-reduction, but only for types and kinds. (Postponement of $R$-reduction w.r.t. weak-head-reduction does not hold in general: an $R$-reduction can create a weak-head-redex.)

*Lemma 3.21* (*Postponement of* R*-reduction wrt weak-head-reduction for types and kinds*)
If $M$ is a type or kind, then

$$M \twoheadrightarrow_R N \twoheadrightarrow_{wh} Q \Rightarrow \exists P\,[M \twoheadrightarrow_{wh} P \twoheadrightarrow_R Q].$$

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

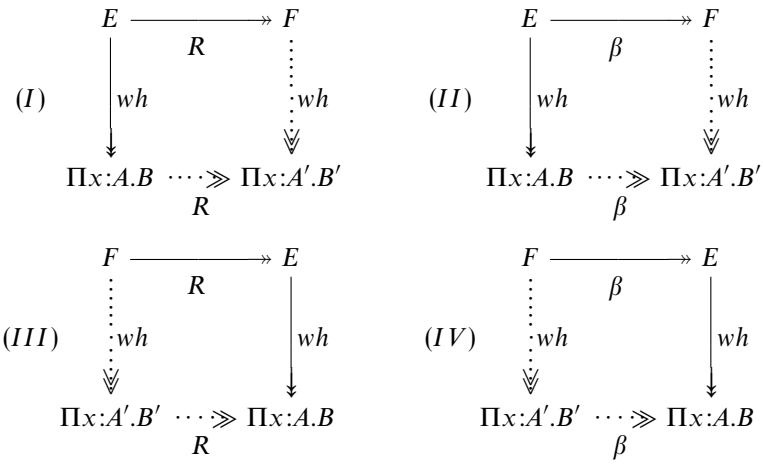
Or in a diagram




*Proof*
We prove the lemma for the case of a one step weak-head-reduction: If $M \twoheadrightarrow_R N \to_{wh} Q$, then $M \to_{wh} P \twoheadrightarrow_R Q$ for some $P$. This implies the general case, as can be verified by filling in the diagram. So, let $M$ be a type or a kind with $M \twoheadrightarrow_R N \to_{wh} Q$. From $N \to_{wh} Q$ we conclude that $N \equiv (\lambda x{:}A.B)C$ and $Q \equiv B[C/x]$ for some $A, B$ and $C$. As $M$ is a type or kind, we know that $M \not\equiv t\varphi$ for any reduction rule $t \to t'$ and substitution $\varphi$. Hence, $M \equiv (\lambda x{:}A'.B')C'$ with $A' \twoheadrightarrow_R A$, $B' \twoheadrightarrow_R B$ and $C' \twoheadrightarrow_R C$. Now, take $P \equiv B'[C'/x]$. Then $M \to_{wh} P$ and $P \twoheadrightarrow_R Q$. □

We now list the four main properties that we need to prove the main lemma.

*Lemma 3.22*
Let $\Pi x{:}A.B$, $E$ and $F$ be types or kinds. We have the following properties.




*Proof*
The validity of diagram (I), respectively diagram (II), follows from Lemma 3.20, respectively Lemma 3.19. Let's focus on diagram (I): following Lemma 3.20, there is a $Q$ such that $F \twoheadrightarrow_{wh} Q$ and $\Pi x{:}A.B \twoheadrightarrow_R Q$. But then $Q$ must be of the form $\Pi x{:}A'.B'$ with $A \twoheadrightarrow_R A'$ and $B \twoheadrightarrow_R B'$. So we are done.

For diagram (III) we use Lemma 3.21. This gives us a term $Q$ such that $F \twoheadrightarrow_{wh} Q$ and $Q \twoheadrightarrow_R \Pi x{:}A.B$. Due to Lemma 3.18, $SR_R$ (Proposition 3.13) and $SR_\beta^{P\omega}$ (Lemma 3.15), we know that $Q \twoheadrightarrow_R \Pi x{:}A.B$ is a well-typed reduction path, with only types or kinds on it. But then it cannot be the case that there is a term of the form $t\varphi$ for any reduction rule $t \to t'$ and substitution $\varphi$ on this reduction path. Hence all terms on the path must be $\Pi$-terms and $Q \equiv \Pi x{:}A'.B'$ for some $A'$ and $B'$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

For diagram (IV) we use the well-known Standardization Theorem. It states that if $M \twoheadrightarrow_\beta N$, then there is a 'standard' reduction path from $M$ to $N$, i.e. a path that reduces outside-in. If one considers such a reduction path from $F$ to $\Pi x{:}A.B$, either $F \equiv \Pi x{:}A'.B'$ or there must be a reduction step $(\lambda y{:}C.D)Q \to_\beta \Pi x{:}A'.B'$ on the path (for some $C, D, Q, A'$ and $B'$). But then, the reduction path from $F$ to $(\lambda y{:}C.D)Q$ must consist of weak-head-reductions only, because any other reduction would block the possibility to contract $(\lambda y{:}C.D)Q$. So we are done. □

*Lemma 3.23* (*Main Lemma for Subject Reduction*)
If $\Pi x{:}A.B \overset{c}{=}_{R\beta} \Pi x{:}A'.B'$ and all the terms on the reduction-expansion-path from $\Pi x{:}A.B$ to $\Pi x{:}A'.B'$ are types or kinds, then $\Pi x{:}A.B \overset{c}{=}_{R\beta} \Pi x{:}A'.B'$ via a path that only uses Π-terms.

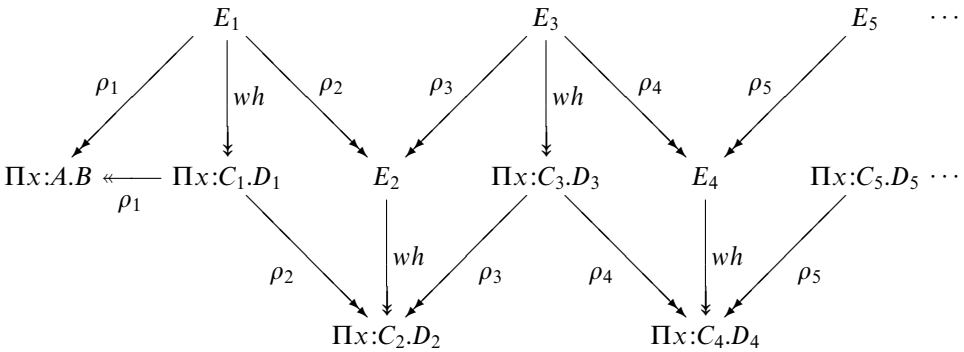
*Proof*
We take a look at the well-typed terms that are on the path from $\Pi x{:}A.B$ to $\Pi x{:}A'.B'$. We can depict the situation as follows:

$$\Pi x{:}A.B \;{}_{\rho_1}\!\!\twoheadleftarrow E_1 \twoheadrightarrow_{\rho_2} E_2 \;{}_{\rho_3}\!\!\twoheadleftarrow \ldots E_{n-1} \twoheadrightarrow_{\rho_n} \Pi x{:}A'.B'$$

where $\rho_i$ ranges over $\{\beta, R\}$. We prove the lemma by showing that there are Π-terms $F_1, \ldots, F_{n-1}$ such that

$$\Pi x{:}A.B \;{}_{\rho_1}\!\!\twoheadleftarrow F_1 \twoheadrightarrow_{\rho_2} F_2 \;{}_{\rho_3}\!\!\twoheadleftarrow \ldots F_{n-1} \twoheadrightarrow_{\rho_n} \Pi x{:}A'.B'$$

Consider the following diagram, and note that all arrows exist due to Lemma 3.22 (diagrams (III) and (IV) for the first tile, (I), (II), (III) and (IV) for the rest).




Proceeding in this way until $\Pi x{:}A'.B'$, we find well-typed Π-terms $F_1 \equiv \Pi x{:}C_1.D_1$, …, $F_{n-1} \equiv \Pi x{:}C_{n-1}.D_{n-1}$ such that

$$\Pi x{:}A.B \;{}_{\rho_1}\!\!\twoheadleftarrow F_1 \twoheadrightarrow_{\rho_2} F_2 \;{}_{\rho_3}\!\!\twoheadleftarrow \ldots F_{n-1} \twoheadrightarrow_{\rho_n} \Pi x{:}A'.B'$$

□

*Corollary 3.24*
If $\Pi x{:}A.C \overset{c}{=}_{R\beta} \Pi x{:}B.D$ and the reduction-expansion-path contains only types and kinds, then $A \overset{c}{=}_{R\beta} B$ and $C \overset{c}{=}_{R\beta} D$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Proposition 3.25* (*Subject Reduction for* $\beta$)
For $\Gamma$ and $\Gamma'$ contexts, $P, P'$ and $D$ terms,

$$\Gamma \vdash P : D \;\&\; P \twoheadrightarrow_\beta P' \;\Rightarrow\; \Gamma \vdash P' : D$$
$$\Gamma \vdash P : D \;\&\; \Gamma \twoheadrightarrow_\beta \Gamma' \;\Rightarrow\; \Gamma' \vdash P : D.$$

*Proof*
By induction on the derivation one proves the statement for a one-step reduction. The only interesting case is when the last rule is (app) and $P \equiv (\lambda x{:}A.B)C$, $P' \equiv B[C/x]$. We then use the fact that when $\Pi x{:}A.C \overset{c}{=}_{R\beta} \Pi x{:}B.D$ and the reduction-expansion-path contains only types and kinds, then $A \overset{c}{=}_{R\beta} B$ and $C \overset{c}{=}_{R\beta} D$, which was proved in the previous corollary. □

*Lemma 3.26* (*Subject Reduction*)
For $\Gamma$ a context, $P, P'$ and $D$ terms,

$$\Gamma \vdash P{:}D \;\&\; P \to_{R\beta} P' \Rightarrow \Gamma \vdash P'{:}D.$$

*Proof*
Immediate from Propositions 3.13 and 3.25. □

## 4 Proof of the main theorem

In this section we present the skeleton of the proof of the main theorem (2.24). From now on, when dealing with a set $R$ of rewrite rules, we shall implicitly assume that conditions (1) and (2) of the main theorem are satisfied. $\chi \models SN$ will denote the fact that system $\chi$ is strongly normalizing.

The proof consists of three main steps:

- $\lambda_{\wedge R} \models \mathsf{SN}$
- $\lambda_{\wedge R} \models \mathsf{SN} \Rightarrow \lambda_{R\omega} \models \mathsf{SN}$
- $\lambda_{R\omega} \models \mathsf{SN} \Rightarrow \lambda_{RC} \models \mathsf{SN}$

where system $\lambda_{\wedge R}$ is a type assignment system defined as the extension of the intersection system of Coppo and Dezani (1980) (see also Barendregt *et al.*, 1993) with the algebraic features defined by a set $R$ of rewrite rules.

The definition of $\lambda_{\wedge R}$ is recalled in Appendix A, while for the proof of $\lambda_{\wedge R} \models \mathsf{SN}$ (Theorem A.8) we refer to Barbanera and Fernández (1996). The proof in Barbanera and Fernández (1996) is based on the Tait–Girard computability predicate method, and the particular computability predicate results from a generalization to system $\lambda_{\wedge R}$ of the one defined in Jouannaud and Okada (1991).

The proofs of the other two steps are based instead on a method that, together with that of Tait–Girard, is among the most used in proofs of strong normalization, i.e. the method of reduction-preserving translations. It consists in proving that the SN of a system is implied by the same property of another system. Such an implication is proved by a translation from the terms of the former to the terms of the latter, which preserves reductions, i.e. reducible terms are mapped to reducible terms.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

This method has been used by Harper *et al.* (1987) to obtain the SN of their system LF (roughly corresponding to $\lambda_P$ in the $\lambda$-cube) using SN of simply typed lambda calculus ($\lambda_{\rightarrow}$ in the $\lambda$-cube).

The translation to prove $\lambda_{\wedge R} \models \mathsf{SN} \Rightarrow \lambda_{R\omega} \models \mathsf{SN}$ is nothing but a *type-erasing* function. Its definition and the proof of reduction-preservation will be the subject of section 6.

The translation and the reduction-preservation proof for $\lambda_{R\omega} \models \mathsf{SN} \Rightarrow \lambda_{RC} \models \mathsf{SN}$ will be instead the subject of section 5.

## 5 $\lambda_{R\omega} \models \mathsf{SN} \Rightarrow \lambda_{RC} \models \mathsf{SN}$

As mentioned in section 4, we will define a translation from terms of $\lambda_{RC}$ to terms of $\lambda_{R\omega}$ and prove this translation to be reduction-preserving.

The definition of the translation and the argument given to prove its 'reduction-preservation' are similar to those provided by Geuvers and Nederhof (1991) to prove the strong normalization for the pure $\lambda_C$. Geuvers and Nederhof's translation can be seen as a higher-order generalization of the map defined by Harper *et al.* (1987) to prove the strong normalization property of the LF system.

As in Harper *et al.* (1987) and Geuvers and Nederhof (1991), it is not possible to define a reduction-preserving map $[-]$ such that

$$\Gamma \vdash_{\lambda_{RC}} M:A \Rightarrow [\Gamma] \vdash_{\lambda_{R\omega}} [M]:[A]$$

i.e. $[-]$ cannot work uniformly on all the terms of $\lambda_{RC}$. One is then forced to define another map $\tau(-)$ from kinds and constructors to types and to prove that

$$\Gamma \vdash_{\lambda_{RC}} M:A \Rightarrow \tau(\Gamma) \vdash_{\lambda_{R\omega}} [M]:\tau(A).$$

The definitions of $[-]$ and $\tau(-)$ do not speak for themselves. We refer to Geuvers and Nederhof (1991) for some intuitions about their definitions as generalizations of the maps defined in Harper *et al.* (1987).

Also, since $\tau$ cannot work uniformly on constructors and kinds, in its definition we shall use another map, $\rho : \{\Box\} \cup \mathsf{Kind}(\lambda_{RC}) \rightarrow \mathsf{Kind}(\lambda_{R\omega})$ such that if $M$ is a constructor of kind $k$ in $\lambda_{RC}$ then $\tau(M)$ is a constructor of kind $\rho(k)$ in $\lambda_{R\omega}$. $\rho(k)$ is just the $\lambda_{R\omega}$ -kind obtained by erasing from $k$ all type dependencies.

*Definition 5.1*

The map $\rho : \{\Box\} \cup \mathsf{Kind}(\lambda_{RC}) \rightarrow \mathsf{Kind}(\lambda_{R\omega})$ is inductively defined by:

1. $\rho(\star) = \rho(\Box) = \star$
2. $\rho(\Pi\alpha{:}M.N) = \rho(M) \rightarrow \rho(N)$ if $M \in \mathsf{Kind}(\lambda_{RC})$
3. $\rho(\Pi x{:}M.N) = \rho(N)$ if $M \in \mathsf{Type}(\lambda_{RC})$.

The case distinction in the definition is correct by the Classification Lemma (Lemma 3.10), and by Lemma 3.5(iii) there are no more cases.

The following properties of $\rho$ will be used:

*Property 5.2*

(i) If $k_1, k_2 \in \mathsf{Kind}(\lambda_{RC})$ and $k_1 \overset{c}{=}_{R\beta} k_2$ then $\rho(k_1) \equiv \rho(k_2)$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

(ii) If $k \in \mathsf{Kind}(\lambda_{RC})$, $u \in Var^{\star} \cup Var^{\Box}$ and $A \in \mathsf{Term}(\lambda_{RC})$ then $\rho(k) \equiv \rho(k[A/u])$.

*Proof*
(*i*) By Lemma 3.5(*iii*)($\Leftarrow$) we have that $k_1 \equiv \Pi\vec{x}{:}\vec{A}.\star \stackrel{c}{=}_{R\beta} \Pi\vec{x}{:}\vec{B}.\star \equiv k_2$. From this fact, it immediately follows from Corollary 3.24 that, if $|\vec{A}| = n$ and $|\vec{B}| = m$, $n = m$ and

$$A_i \stackrel{c}{=}_{R\beta} B_i \text{ for } 1 \leq i \leq n. \tag{5}$$

We now can prove the thesis by induction on the structure of $k_1$. The base case is immediate. To prove the inductive case let us first notice that if we apply the map $\rho$ to $k_1$ and $k_2$ it follows that, by Corollary 3.8 and (5) above, only one of the two following cases can occur:

1. $\rho(\Pi\vec{x}{:}\vec{A}.\star) \equiv \rho(A_1) \to \rho(\Pi x_2{:}A_2.\ldots\Pi x_n{:}A_n.\star)$ &
   $\rho(\Pi\vec{x}{:}\vec{B}.\star) \equiv \rho(B_1) \to \rho(\Pi x_2{:}B_2.\ldots\Pi x_n{:}B_n.\star)$
2. $\rho(\Pi\vec{x}{:}\vec{A}.\star) \equiv \rho(\Pi x_2{:}A_2.\ldots\Pi x_n{:}A_n.\star)$ & $\rho(\Pi\vec{x}{:}\vec{B}.\star) \equiv \rho(\Pi x_2{:}B_2.\ldots\Pi x_n{:}B_n.\star)$.

In both cases the property follows by induction.

(ii) By Lemma 3.5(iii), we have that $k \equiv \Pi\vec{x}{:}\vec{C}.\star$. Then $k[A/u] \equiv \Pi\vec{x}{:}C[\vec{A}/u].\star$. We can now prove the thesis by induction on the structure of $k$. The base case is immediate. The inductive case follows easily once one realizes that, by the Substitution Lemma, $C \in \mathsf{Kind}(\lambda_{RC}) \Rightarrow C[A/u] \in \mathsf{Kind}(\lambda_{RC})$. □

Now, we choose one of the variables of $Var^{\Box}$ to act as a fixed constant, i.e. it will not be used as a bound variable in an abstraction. This variable will be denoted by 0.

*Definition 5.3*
The map $\tau : \{\Box\} \cup \mathsf{Kind}(\lambda_{RC}) \cup \mathsf{Constr}(\lambda_{RC}) \to \mathsf{Term}(\lambda_{R\omega})$ is inductively defined by:

1. $\tau(\star) = \tau(\Box) = 0 : \star$.
2. $\tau(\alpha) = \alpha$ if $\alpha$ is a variable.
   $\tau(s) = s$ if $s \in \mathscr{S}$.
3. $\tau(\Pi\alpha{:}M.N) = \Pi\alpha{:}\rho(M).\tau(M) \to \tau(N) : \star$ if $M \in \mathsf{Kind}(\lambda_{RC})$.
   $\tau(\Pi x{:}M.N) = \Pi x{:}\tau(M).\tau(N)$ if $M \in \mathsf{Type}(\lambda_{RC})$.
4. $\tau(\lambda\alpha{:}M.N) = \lambda\alpha{:}\rho(M).\tau(N)$ if $M \in \mathsf{Kind}(\lambda_{RC})$.
   $\tau(\lambda x{:}M.N) = \tau(N)$ if $M \in \mathsf{Type}(\lambda_{RC})$.
5. $\tau(MN) = \tau(M)\tau(N)$ if $N \in \mathsf{Constr}(\lambda_{RC})$.
   $\tau(MN) = \tau(M)$ if $N \in \mathsf{Object}(\lambda_{RC})$.

The definition by cases is correct by the Classification Lemma (Lemma 3.10). Lemma 5.4 guarantees that the range of $\tau$ is actually $\mathsf{Term}(\lambda_{R\omega})$.

To map $\mathsf{Context}(\lambda_{RC})$ into $\mathsf{Context}(\lambda_{R\omega})$ we choose, for each variable $\alpha \in Var^{\Box}$, a connected variable $x^{\alpha} \in Var^{\star}$, such that no two variables of $Var^{\Box}$ are connected to the same variable of $Var^{\star}$. We now extend the map $\tau$ in such a way that it also acts on $\mathsf{Context}(\lambda_{RC})$ yielding elements of $\mathsf{Context}(\lambda_{R\omega})$:

1. Let $A \in \mathsf{Kind}(\lambda_{RC}) \cup \mathsf{Type}(\lambda_{RC})$.
   $\tau(x : A) = x : \tau(A)$ if $x \in Var^{\star}$.
   $\tau(\alpha : A) = \alpha : \rho(A), x^{\alpha} : \tau(A)$ if $\alpha \in Var^{\Box}$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

2. Let $\Gamma = \langle u_1 : A_1, u_2 : A_2, \ldots, u_n : A_n\rangle \in \mathsf{Context}(\lambda_{RC})$.
   $\tau(\Gamma) = \langle 0 : \star, d : \bot, \tau(u_1 : A_1), \tau(u_2 : A_2), \ldots, \tau(u_n : A_n)\rangle$.

The reason for putting $0 : \star$ and $d : \bot \equiv \Pi\alpha{:}\star.\alpha$ in the context is that in the following definition of the map $[-]$ on terms of $\lambda_{RC}$ it will be necessary to have a canonical inhabitant for each type and kind. If $\tau(\Gamma) \vdash_{\lambda_{R\omega}} B : \star$ or $\tau(\Gamma) \vdash_{\lambda_{R\omega}} B : \Box$, we want $\tau(\Gamma) \vdash_{\lambda_{R\omega}} c^B : B$ for a $c^B$ which does not depend upon the structure of $\Gamma$.

Now, if $\tau(\Gamma) \vdash_{\lambda_{R\omega}} B : \star$ we shall put $c^B \equiv dB$ and if $\tau(\Gamma) \vdash_{\lambda_{R\omega}} B : \Box$, a canonical inhabitant of $B$ is inductively defined by:

1. if $B \equiv \star$ then $c^\star = 0$;
2. if $B \equiv k_1 \to k_2$ then $c^{k_1 \to k_2} = \lambda\alpha : k_1.c^{k_2}$.

Note that $c^B[N/u] \equiv c^{B[N/u]}$ for all $B \in \mathsf{Kind}(\lambda_{R\omega}) \cup \mathsf{Type}(\lambda_{R\omega})$, $N \in \mathsf{Term}(\lambda_{R\omega})$ and variables $u$.

The following lemma states that $\rho$ and $\tau$ are well-defined:

*Lemma 5.4*
Let $\Gamma \in \mathsf{Context}(\lambda_{RC})$, $M, N \in \mathsf{Term}(\lambda_{RC})$.

$$\Gamma \vdash_{\lambda_{RC}} M : N : \Box \text{ or } \Gamma \vdash_{\lambda_{RC}} M : N \equiv \Box \quad \Rightarrow \quad \tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(M) : \rho(N).$$

*Proof*
By induction on the length of the derivation of $\Gamma \vdash_{\lambda_{RC}} M : N$, distinguishing cases according to the last applied rule.

1. If $\Gamma \vdash_{\lambda_{RC}} M : N$ is an axiom, there are two possibilities: either $M \equiv \star$ and $N \equiv \Box$ and we are done, or $M \equiv s$ and $N \equiv \star$ where $s$ is a sort, and in this case it is easy to check that the lemma holds by definition of $\tau$ and $\rho$.
2. If the last rule is (var) then the conclusion is $\Gamma', \alpha : N \vdash_{\lambda_{RC}} \alpha : N$ and so, by induction and definition of $\tau$ and $\rho$, $\tau(\Gamma', \alpha : N) \vdash_{\lambda_{R\omega}} \tau(\alpha) : \rho(N)$.
3. If the last rule is (weak) the thesis follows from the induction hypothesis, the definition of $\tau(-)$ and $\rho(-)$ and Weakening.
4. If the last rule is $(\mathrm{red}_{R\beta})$ then the thesis follows from the induction hypothesis and Property 5.2.
5. If $M \equiv \Pi u{:}B_1.B_2$, $N \equiv p \in \{\star, \Box\}$ and the last applied rule is $(\Pi)$ then by induction $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1) : \star$ and $\tau(\Gamma, u : B_1) \vdash_{\lambda_{R\omega}} \tau(B_2) : \star$.
   If $B_1 \in \mathsf{Type}(\lambda_{RC})$ then $\tau(\Gamma, u : B_1) \equiv \tau(\Gamma), u : \tau(B_1)$ and so $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1) \to \tau(B_2) : \star$ by rule $(\star, \star)$.
   If $B_1 \in \mathsf{Kind}(\lambda_{RC})$ then $\tau(\Gamma, u : B_1) \equiv \tau(\Gamma), u : \rho(B_1), x^u : \tau(B_1)$ and so, by rules $(\star, \star)$ and $(\Box, \star)$, we have $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \Pi u{:}\rho(B_1).\tau(B_1) \to \tau(B_2) : \star$.
6. If $M \equiv \lambda u{:}B_1.B_2$, $N \equiv \Pi u{:}B_1.C_2 \in \mathsf{Kind}(\lambda_{RC})$ and the last rule is $(\lambda)$ then by induction $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1) : \star$ and $\tau(\Gamma, u : B_1) \vdash_{\lambda_{R\omega}} \tau(B_2) : \rho(C_2)$.
   If $B_1 \in \mathsf{Type}(\lambda_{RC})$ then $\tau(\lambda u{:}B_1.B_2) \equiv \tau(B_2)$, $\rho(\Pi u{:}B_1.C_2) \equiv \rho(C_2)$ and $\tau(u : B_1) \equiv u : \tau(B_1)$. By definition of $\tau$ and $\rho$ and by Property 5.2, $u$ is not a free variable in $\tau(B_2) : \rho(C_2)$, and so by substituting $c^{\tau(B_1)}$ for $u$ we find $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_2) : \rho(C_2)$.
   If $B_1 \in \mathsf{Kind}(\lambda_{RC})$ then $\tau(\lambda u{:}B_1.B_2) \equiv \lambda u{:}\rho(B_1).\tau(B_2)$, $\rho(\Pi u{:}B_1.C_2) \equiv \rho(B_1) \to \rho(C_2)$ and $\tau(\Gamma, u : B_1) = \tau(\Gamma), u : \rho(B_1), x^u : \tau(B_1)$. By definition of $\tau$ and $\rho$ and

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

by Property 5.2, $x^u$ is not a free variable in $\tau(B_2) : \rho(C_2)$, so by substituting $c^{\tau(B_1)}$ for $x^u$ we obtain $\tau(\Gamma), u : \rho(B_1) \vdash_{\lambda_{R\omega}} \tau(B_2) : \rho(C_2)$ and by one application of $(\lambda)$, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \lambda u{:}\rho(B_1).\tau(B_2) : \rho(B_1) \to \rho(C_2)$.

7. If $M \equiv B_1B_2$, $N \equiv C_2[B_2/x] \in \mathsf{Kind}(\lambda_{RC})$ and the last rule is (app), let $\Gamma \vdash_{\lambda_{RC}} B_1 : \Pi x{:}C_1.C_2$ and then by the Stripping Lemma and induction we obtain:
If $B_2 \in \mathsf{Object}(\lambda_{RC})$ then $\tau(B_1B_2) \equiv \tau(B_1)$, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1) : \rho(\Pi x{:}C_1.C_2)$ and $\rho(\Pi x{:}C_1.C_2) \equiv \rho(C_2) \equiv \rho(C_2[B_2/x])$.
If $B_2 \in \mathsf{Constr}(\lambda_{RC})$ then $\tau(B_1B_2) \equiv \tau(B_1)\tau(B_2)$, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1) : \rho(\Pi x{:}C_1.C_2)$ and $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_2) : \rho(C_1)$. Besides, $\rho(\Pi x{:}C_1.C_2) \equiv \rho(C1) \to \rho(C_2)$, so $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1B_2) : \rho(C_2) \equiv \rho(C_2[B_2/x])$. □

□

*Lemma 5.5*
$A \in \mathsf{T}_{\mathscr{S}} \Rightarrow \tau(A) \equiv A$.

*Proof*
Immediate by definition of $\tau$. □

*Lemma 5.6*
Let $B, B' \in \mathsf{Kind}(\lambda_{RC}) \cup \mathsf{Constr}(\lambda_{RC})$.

(i) If $x \in Var^{\star}$ and $A \in \mathsf{Object}(\lambda_{RC})$ then $\tau(B[A/x]) \equiv \tau(B) \equiv \tau(B)[A/x]$.
(ii) If $\alpha \in Var^{\Box}$ and $A \in \mathsf{Constr}(\lambda_{RC})$ then $\tau(B[A/\alpha]) \equiv \tau(B)[\tau(A)/\alpha]$.
(iii) If $B \to_{R\beta} B'$ then $\tau(B) \to_{\beta} \tau(B')$ or $\tau(B) \equiv \tau(B')$.
(iv) $B \overset{c}{=}_{R\beta} B' \Rightarrow \tau(B) =_{\beta} \tau(B')$.

*Proof*
By induction on the structure of $B$.

(i) (ii) Immediate by definition of $\tau(-)$, Property 5.2 and the induction hypothesis.

(iii) We consider $\to_{\beta}$ and $\to_R$ separately.

Assume $B \to_{\beta} B'$. If $B \equiv \Pi u{:}M.N$, $B \equiv \lambda u{:}M.N$ or $B \equiv MN$ with $B' \equiv \Pi u{:}M'.N'$, $B' \equiv \lambda u{:}M'.N'$ or $B' \equiv M'N'$, then $\tau(B) \to_{\beta} \tau(B')$ or $\tau(B) \equiv \tau(B')$ by induction. The interesting situation occurs when $B \equiv (\lambda u{:}M.N)P$ and $B' \equiv N[P/u]$. If $M \in \mathsf{Type}(\lambda_{RC})$ then $\tau((\lambda u{:}M.N)P) \equiv \tau(N) \equiv \tau(N[P/u])$ by definition of $\tau$ and by (i). If $M \in \mathsf{Kind}(\lambda_{RC})$ then $\tau((\lambda\alpha{:}M.N)P) \equiv (\lambda\alpha{:}\rho(M).\tau(N))\tau(P) \to_{\beta} \tau(N)[\tau(P)/\alpha] \equiv \tau(N[P/\alpha])$ by definition of $\tau$ and by (ii).

Assume $B \to_R B'$. Since $B, B' \in \mathsf{Kind}(\lambda_{RC}) \cup \mathsf{Constr}(\lambda_{RC})$ it has to be necessarily that $B \equiv C[A]$ and $B' \equiv C[A']$ for a suitable context for objects $C[\,]$, where $A, A' \in \mathsf{Object}(\lambda_{RC})$ and $A \to_R A'$. Then, $\tau(B) \equiv \tau(B')$ follows by (i). Notice that in (i) the usual notion of substitution is considered, not that of replacement in contexts. However, it is quite straightforward to check that (i) is valid even for the replacement in contexts.

(iv) Immediate from (iii). □

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Definition 5.7*
The map $[-] : \mathsf{Kind}(\lambda_{RC}) \cup \mathsf{Constr}(\lambda_{RC}) \cup \mathsf{Object}(\lambda_{RC}) \to \mathsf{Object}(\lambda_{R\omega})$ is inductively defined by

1. $[\star] = c^0$,
2. $[x] = x$ if $x \in Var^\star$,
3. $[\alpha] = x^\alpha$ if $\alpha \in Var^\Box$,
4. $[s] = c^0$ if $s \in \mathscr{S}$,
5. $[f] = f$ if $f \in \mathscr{F}$,
6. $[\Pi x{:}M.N] = c^{0\to 0\to 0}[M][N][c^{\tau(M)}/x]$ if $M \in \mathsf{Type}(\lambda_{RC})$,
   $[\Pi \alpha{:}M.N] = c^{0\to 0\to 0}[M][N][c^{\tau(M)}/x^\alpha][c^{\rho(M)}/\alpha]$ if $M \in \mathsf{Kind}(\lambda_{RC})$,
7. $[\lambda x{:}M.N] = (\lambda z{:}0.\lambda x{:}\tau(M).[N])[M]$ where $z$ is a fresh variable, if $M \in \mathsf{Type}(\lambda_{RC})$,
   $[\lambda \alpha{:}M.N] = (\lambda z{:}0.\lambda \alpha{:}\rho(M).\lambda x^\alpha{:}\tau(M).[N])[M]$ where $z$ is a fresh variable, if $M \in \mathsf{Kind}(\lambda_{RC})$,
8. $[MN] = [M][N]$ if $N \in \mathsf{Object}(\lambda_{RC})$,
   $[MN] = [M]\tau(N)[N]$ if $N \in \mathsf{Constr}(\lambda_{RC})$.

This definition by cases is correct by the Classification Lemma. The following theorems state that $[-]$ satisfies the required conditions: Theorem 5.8 shows that the range of $[-]$ is actually $\mathsf{Object}(\lambda_{R\omega})$, and Theorem 5.12 shows that the mapping preserves all possible reduction sequences.

*Theorem 5.8*
Let $\Gamma \in \mathsf{Context}(\lambda_{RC})$, $M, N \in \mathsf{Term}(\lambda_{RC})$.
If $\Gamma \vdash_{\lambda_{RC}} M : N$ then $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [M] : \tau(N)$.

*Proof*
By induction on the structure of $M$. By Lemma 5.4 we know that $\tau(\Gamma)$ is a legal context in $\lambda_{R\omega}$.

1. $M \equiv \star$, then $N \equiv \Box$ by Stripping (Lemma 3.3(i)) and Lemma 3.5(ii). It follows that $[\star] \equiv c^0 : 0 \equiv \tau(\Box)$ in $\tau(\Gamma)$.
2. $M \equiv s \in S$, then $N \equiv \star$ by Stripping (ii) and Lemma 3.5(ii). It follows that $[s] \equiv c^0 : 0 \equiv \tau(\star)$ in $\tau(\Gamma)$.
3. $M \equiv u \in Var^\star \cup Var^\Box$, then $u : A \in \Gamma$, with $A \overset{c}{=}_{R\beta} N$. If $u \equiv \alpha \in Var^\Box$ then $\tau(\alpha : A) \equiv \alpha : \rho(A), x^\alpha : \tau(A) \in \tau(\Gamma)$. If $u \equiv x \in Var^\star$ then $\tau(x : A) \equiv x : \tau(A) \in \tau(\Gamma)$. In both cases $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [u] : \tau(A)$, and by Lemma 5.6, and some applications of $(\mathrm{red}_{R\beta})$, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [u] : \tau(N)$.
4. $M \equiv f \in \mathscr{F}$, then, by Stripping $(iv)$, $N \overset{c}{=}_{R\beta} \sigma$ where $\sigma$ is the type of $f$ in the signature $\mathscr{F}$. We have that $[f] \equiv f : \sigma$ in $\tau(\Gamma)$. By Lemma 5.5, $\tau(\sigma) \equiv \sigma$ and then, by Lemma 5.6 and some applications of $(\mathrm{red}_{R\beta})$, we find $[f] : \tau(N)$ in $\tau(\Gamma)$.
5. $M \equiv \Pi u{:}B_1.B_2$, then $\Gamma \vdash_{\lambda_{RC}} B_1 : p_1$ and $\Gamma, u : B_1 \vdash_{\lambda_{RC}} B_2 : p_2$ for suitable $p_1, p_2 \in \{\star, \Box\}$. By induction $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1] : 0$ and $\tau(\Gamma, u : B_1) \vdash_{\lambda_{R\omega}} [B_2] : 0$. By Lemma 5.4, $\tau(\Gamma) \vdash_{\lambda_{R\omega}}, \tau(B_1) : \star$ and so $\tau(\Gamma) \vdash_{\lambda_{R\omega}} c^{\tau(B_1)} : \tau(B_1)$.
   If $p_1 \equiv \star$ and $u \equiv x \in Var^\star$ then $\tau(\Gamma, x : B_1) \equiv \tau(\Gamma), x : \tau(B_1)$, so by induction

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

and the Substitution Lemma, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_2][c^{\tau(B_1)}/x] : 0$. By using the rule (app) twice we get $\tau(\Gamma) \vdash_{\lambda_{R\omega}} c^{0\to 0\to 0}[B_1][B_2][c^{\tau(B_1)}/x] : 0$.
If $p_1 \equiv \Box$ and $u \equiv \alpha \in Var^{\Box}$ then $\tau(\Gamma, \alpha : B_1) \equiv \tau(\Gamma), \alpha : \rho(B_1), x^{\alpha} : \tau(B_1)$, so by induction and the Substitution Lemma $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_2][c^{\tau(B_1)}/x^{\alpha}, c^{\rho(B_1)}/\alpha] : 0$. By using the rule (app) twice we get $\tau(\Gamma) \vdash_{\lambda_{R\omega}} c^{0\to 0\to 0}[B_1][B_2][c^{\tau(B_1)}/x^{\alpha}, c^{\rho(B_1)}/\alpha] : 0$.
In both cases $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [\Pi u{:}B_1.B_2] : 0$.

6. $M \equiv \lambda u{:}B_1.B_2$, then $\Gamma \vdash_{\lambda_{RC}} B_1 : p_1$ and $\Gamma, u : B_1 \vdash_{\lambda_{RC}} B_2 : C_2 : p_2$ for a suitable $C_2 \in \mathsf{Term}(\lambda_{RC})$, $p_1, p_2 \in \{\star, \Box\}$, with $N \overset{c}{=}_{R\beta} \Pi u{:}B_1.C_2$.
By induction $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1] : 0$ and $\tau(\Gamma, u : B_1) \vdash_{\lambda_{R\omega}} [B_2] : \tau(C_2)$. By Lemma 5.4, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(B_1) : \star$ and $\tau(\Gamma, u : B_1) \vdash_{\lambda_{R\omega}} \tau(C_2) : \star$.
If $p_1 \equiv \star$ and $u \equiv x \in Var^{\star}$ then $\tau(\Gamma, x : B_1) \equiv \tau(\Gamma), x : \tau(B_1)$. With two applications of $(\lambda)$ and one of the rule (app) we get $\tau(\Gamma) \vdash_{\lambda_{R\omega}} (\lambda z{:}0.\lambda x{:}\tau(B_1)[B_2])[B_1]{:}\Pi x{:}\tau(B_1).\tau(C_2)$.
If $p_1 \equiv \Box$ and $u \equiv \alpha \in Var^{\Box}$ then $\tau(\Gamma, x : B_1) = \tau(\Gamma), \alpha : \rho(B_1), x^{\alpha} : \tau(B_1)$. With three applications of $(\lambda)$ and one of (app) we get
$\tau(\Gamma) \vdash_{\lambda_{R\omega}} (\lambda z{:}0.\lambda\alpha{:}\rho(B_1).\lambda x^{\alpha}{:}\tau(B_1).[B_2])[B_1] : \Pi\alpha{:}\rho(B_1).\tau(B_1) \to \tau(C_2)$.
In both cases $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [\lambda u{:}B_1.B_2] : \tau(\Pi u{:}B_1.C_2)$ and so, by Lemma 5.6(*iv*), after some applications of rule $(\mathrm{red}_{R\beta})$ we get $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [\lambda u{:}B_1.B_2] : \tau(N)$.

7. $M \equiv B_1B_2$ then $\Gamma \vdash_{\lambda_{RC}} B_1 : \Pi u{:}C_1.C_2$ and $\Gamma \vdash_{\lambda_{RC}} B_2 : C_1$ for some $C_1, C_2 \in \mathsf{Term}(\lambda_{RC})$ with $N \overset{c}{=}_{R\beta} C_2[B_2/u]$.
By induction $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1] : \tau(\Pi u{:}C_1.C_2)$ and $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_2] : \tau(C_1)$. By Lemma 5.4, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} \tau(C_1) : \star$.
If $C_1 \in \mathsf{Type}(\lambda_{RC})$, $u \equiv x \in Var^{\star}$, then $\tau(\Pi x{:}C_1.C_2) \equiv \Pi x{:}\tau(C_1).\tau(C_2)$, so $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1][B_2] : \tau(C_2)[[B_2]/x] \equiv \tau(C_2[[B_2]/x]) \equiv \tau(C_2) \equiv \tau(C_2[B_2/x])$ (by Lemma 5.6).
If $C_1 \in \mathsf{Kind}(\lambda_{RC})$, $u \equiv \alpha \in Var^{\Box}$, then $\tau(\Pi\alpha{:}C_1.C_2) \equiv \Pi\alpha{:}\rho(C_1).\tau(C_1) \to \tau(C_2)$,
so $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1]\tau(B_2)[B_2] : \tau(C_2)[[B_2]/x^{\alpha}, \tau(B_2)/\alpha] \equiv \tau(C_2[B_2/\alpha])$ (by Lemma 5.6).
In both cases $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1B_2] : \tau(C_2[B_2/u])$ and then, by Lemma 5.6.(*iv*) and some applications of rule $(\mathrm{red}_{R\beta})$, $\tau(\Gamma) \vdash_{\lambda_{R\omega}} [B_1B_2] : \tau(N)$.

□

We will now prove that if $M \to_{R\beta} N$ in $\lambda_{RC}$ then $[M] \to^{+}_{R\beta} [N]$ in $\lambda_{R\omega}$, i.e. $[-]$ preserves all reduction sequences. For the proof we need some lemmas.

*Lemma 5.9*
Let $M, N \in \mathsf{Term}(\lambda_{RC})$ such that $M[N/x], M[N/\alpha] \in \mathsf{Term}(\lambda_{RC})$.
If $x \in Var^{\star}$ and $N \in \mathsf{Object}(\lambda_{RC})$ then $[M[N/x]] \equiv [M][[N]/x]$.
If $\alpha \in Var^{\Box}$, $N \in \mathsf{Constr}(\lambda_{RC})$ then $[M[N/\alpha]] \equiv [M][\tau(N)/\alpha, [N]/x^{\alpha}]$.

*Proof*
By induction on the structure of $M$.
If $M \in (Var^{\star} \cup Var^{\Box} \cup \mathscr{F} \cup \mathscr{S} \cup \{\star\})$ the result is trivial. Otherwise the first part of the lemma follows from the induction hypothesis and the fact that:

1. $\tau(Q[N/u]) \equiv \tau(Q)[[N]/u]$, by Lemma 5.6,

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

2. $\rho(Q[N/u]) \equiv \rho(Q)[[N]/u]$, by Property 5.2,
3. $c^{\tau(Q[N/u])} \equiv c^{\tau(Q)}[[N]/u]$, by 1 and the definition of $c^B$,
4. $c^{\rho(Q[N/u])} \equiv c^{\rho(Q)}[[N]/u]$, by 2 and the definition of $c^B$.

The second part of the lemma follows from the induction hypothesis and the fact that:

1. $\tau(Q[N/\alpha]) \equiv \tau(Q)[\tau(N)/\alpha] \equiv \tau(Q)[\tau(N)/\alpha, [N]/x^\alpha]$, by Lemma 5.6 and because $x^\alpha$ is not a free variable in $\tau(Q)$,
2. $\rho(Q[N/\alpha]) \equiv \rho(Q) \equiv \rho(Q)[\tau(N)/\alpha, [N]/x^\alpha]$, by Property 5.2 and because $x^\alpha$ is not a free variable in $\rho(Q)$,
3. $c^{\tau(Q[N/\alpha])} \equiv c^{\tau(Q)}[\tau(N)/\alpha, [N]/x^\alpha]$, by 1 and the definition of $c^B$,
4. $c^{\rho(Q[N/\alpha])} \equiv c^{\rho(Q)}[\tau(N)/\alpha, [N]/x^\alpha]$, by 2 and the definition of $c^B$.

□

We extend the definition of $[-]$ to substitutions, in the usual way: Let $\varphi$ be a substitution in $\lambda_{RC}$, $[\varphi]$ is the substitution such that $Dom([\varphi]) = Dom(\varphi)$ and for each $x \in Dom([\varphi])$, $[\varphi](x) = [\varphi(x)]$. This definition is correct by Theorem 5.8.

*Lemma 5.10*

(i) Let $M \in \mathsf{Term}(\lambda_{RC})$ and let $\varphi$ be a substitution in $\lambda_{RC}$ such that $Dom(\varphi) \subset Var^\star$ and $M\varphi \in \mathsf{Term}(\lambda_{RC})$. Then $[M\varphi] \equiv [M][\varphi]$.

(ii) Let $C[M] \in \mathsf{Term}(\lambda_{RC})$ with $C[\ ]$ context and $M \in \mathsf{Object}(\lambda_{RC})$. Then $[C[M]] \equiv [C][[M]]$.

*Proof*

(i) By Lemma 5.9.

(ii) It is not difficult to check that the proof of the first part of Lemma 5.9 can be modified to consider replacement instead of substitution. □

*Lemma 5.11*

Let $t$ be a cube-embeddable algebraic term. Then $[t] \equiv t$.

*Proof*

By definition of algebraic term, cube-embeddability and map $[-]$. □

Now we can show that all the reduction sequences are preserved, and therefore strong normalization is preserved.

*Theorem 5.12*

Let $M, M' \in \mathsf{Term}(\lambda_{RC})$. If $M \to_{R\beta} M'$ then $[M] \to^+_{R\beta} [M']$.

*Proof*

By induction on the structure of $M$.

For $M \in (Var^\star \cup Var^\Box \cup \mathscr{F} \cup S \cup \{\star\})$ the theorem is trivial.

If $M \equiv \Pi u{:}B_1.B_2$, $M \equiv \lambda u{:}B_1.B_2$, or $M \equiv B_1B_2$ with $M' \equiv B_1B_2'$ or $B_1'B_2$, then $[M] \to_{R\beta} [M']$ follows by induction. Let us consider now the cases where $M$ is a $\beta$-redex or a $R$-redex.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

- If $M \equiv (\lambda u{:}B_1.B_2)C$ and $M' \equiv B_2[C/u]$, we distinguish again two cases:
  - — If $C \in \mathsf{Object}(\lambda_{RC})$ then
    $[M] \equiv [(\lambda u{:}B_1.B_2)][C] \equiv (\lambda z{:}0.\lambda u : \tau(B_1).[B_2])[B_1][C] \to_\beta [B_2]\ [[C]/u]$,
    because $z$ is a fresh variable. Moreover, $[B_2][[C]/u] \equiv [B_2[C/u]]$, by Lemma 5.9.
  - — If $C \in \mathsf{Constr}(\lambda_{RC})$ then
    $[M] \equiv [(\lambda u{:}B_1.B_2)]\tau(C)[C] \equiv (\lambda z{:}0.\lambda u{:}\rho(B_1).\lambda x^u{:}\tau(B_1).[B_2])[B_1]\tau(C)[C] \to_\beta$
    $\to_\beta [B_2][\tau(C)/u, [C]/x^u] \equiv [B_2[C/u]]$ by Lemma 5.9.
- If $M$ is a $R$-redex then it has to be $M \equiv t\varphi$, $M' \equiv t'\varphi$, where $r : t \to t'$ is a rule in $R$ and $Dom(\varphi) \subset Var^\star$ because in $R$ there is no free variable belonging to $Var^\square$. By Lemma 5.10, $[M] \equiv [t][\varphi]$ and $[M'] \equiv [t'][\varphi]$. Moreover, by Lemma 5.11, $[t] \equiv t$ and $[t'] \equiv t'$. Hence $M \equiv [t][\varphi] \to_R [t'][\varphi] \equiv [M']$.

□

*Theorem 5.13*
$\to_{R\beta}$-strong normalization of $\lambda_{R\omega}$ implies $\to_{R\beta}$-strong normalization of $\lambda_{RC}$.

*Proof*
By contradiction. Assume that there exists an infinite sequence of $\to_{R\beta}$-reductions in $\lambda_{RC}$ starting from a term $M_1 \in \mathsf{Term}(\lambda_{RC})$. By Theorem 5.12 there is an infinite sequence of reductions starting from $[M_1]$ in $\lambda_{R\omega}$, which contradicts the hypothesis. Hence all sequences of $\to_{R\beta}$-reductions are finite in $\lambda_{RC}$. □

As an application of this result, we obtain the strong normalization of the generalization to $\lambda_C$ of the language described in Jouannaud and Okada (1991).

## 6 $\lambda_{\wedge R} \models \mathsf{SN} \Rightarrow \lambda_{R\omega} \models \mathsf{SN}$

In this section we prove that SN of $\lambda_{\wedge R}$ implies SN of $\lambda_{R\omega}$. $\lambda_{R\omega}$ belongs to the $\lambda R$-cube, while the intersection type assignment system $\lambda_{\wedge R}$ (see the Appendix) is specified by the quadruple $\langle \mathscr{S}, F, \mathsf{Ax}, R\rangle$ (where $F$ and Ax are naturally induced by the signature $\mathscr{F}$, or equivalently $\mathscr{F}$ is induced by $F$ and Ax).

As for the proof of

$$\lambda_{R\omega} \models \mathsf{SN} \Rightarrow \lambda_{RC} \models \mathsf{SN}$$

we shall use a translation from terms of $\lambda_{R\omega}$ to terms of $\lambda_{\wedge R}$, and prove this translation to be reduction-preserving. Such a translation will be a 'type erasing' function.

The following lemma was proved by Girard (1972) for $\lambda_\omega$ (i.e. without algebraic features), and it holds for $\lambda_{R\omega}$ as well, because the function symbols of $\mathscr{F}$ can be looked at as free variables in $\beta$-reductions.

*Lemma 6.1* (*Girard, 1972*)
$\to_\beta$ is strongly normalizing on terms of $\lambda_{R\omega}$.

Let us define, by induction on the structure of terms, a 'type erasing' function from $\mathsf{Object}(\lambda_{R\omega})$ to $\Lambda_F$ (the set of untyped $\lambda$-terms with constants in $F$, see Definition A.2 in the Appendix).

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Definition 6.2* (*The type erasing function* $|-|$)
The map $|-| : \mathsf{Object}(\lambda_{R\omega}) \to \Lambda_F$ is inductively defined by

1. $|x| = x$ if $x \in Var^\star$.
2. $|f| = f$ if $f \in \mathscr{F}$.
3. $|\lambda x{:}A.q| = \lambda x.|q|$ if $A \in \mathsf{Type}(\lambda_{R\omega})$ and $q \in \mathsf{Object}(\lambda_{R\omega})$
4. $|\lambda x{:}A.q| = |q|$ if $A \in \mathsf{Kind}(\lambda_{R\omega})$ and $q \in \mathsf{Object}(\lambda_{R\omega})$
5. $|pq| = |p||q|$ if $q \in \mathsf{Object}(\lambda_{R\omega})$
6. $|pQ| = |p|$ if $Q \in \mathsf{Constr}(\lambda_{R\omega})$.

The definition by cases is correct by definition of $\lambda_{R\omega}$ and the Classification Lemma.

We shall frequently use, without explicit mention, the following property, stating that it can never happen that an object be a subterm of a constructor. This property can be easily proved using Lemma 2.14.

*Property 6.3*
The erasing function never makes completely disappear subterms that are objects, i.e. given an object $C[q]$ where $C[\,]$ is a context and $q$ is an object itself, $|C[q]| \equiv C'[q']$ where $q' \equiv |q|$ and $C'[\,] \equiv |C[\,]|$.

We shall prove that $|-|$ maps objects of $\lambda_{R\omega}$ to typeable terms in $\lambda_{\wedge R}$, i.e. $|M| \in \Lambda_{\wedge R}$ if $M \in \mathsf{Object}(\lambda_{R\omega})$ (Theorem 6.9 below). To prove that $|M|$ is typable (the symbol $\vdash^\wedge$ denotes typability in $\lambda_{\wedge R}$) we need some lemmas.

In the following, a type $A$ will be called *arrow-ground* if its $\beta$-normal form $A{\downarrow_\beta}$ belongs to $\mathsf{T}_{\mathscr{S}}$. Recall that, since in $\lambda_{R\omega}$ we do not have the rule $(\star, \Box)$, it is not possible to have algebraic reductions inside types (see Lemma 2.14); so $\overset{c}{=}_{R\beta}$ between types is indeed $=_\beta$.

*Lemma 6.4*
Let $M \in \mathsf{Object}(\lambda_{R\omega})$. Then there exists a context $C[\,]$ and an object $N$ such that $M \equiv C[N]$, $|\, C[z]\, | \equiv z$ (where $z$ is a fresh variable), and the following implications hold:
$N \equiv \lambda x{:}A.q \quad \Rightarrow \quad A \in \mathsf{Type}(\lambda_{R\omega})$.
$N \equiv pq \quad \Rightarrow \quad q \in \mathsf{Object}(\lambda_{R\omega})$.
Besides:

1. If $|N| \equiv cN_1 \ldots N_m$ ($c$ variable or function symbol) and $c$ has arrow-ground type $\delta$ with $\delta{\downarrow_\beta} \equiv \sigma_1 \to \ldots \sigma_m \to \sigma$ then there exist $N_i'(1 \le i \le m)$ subterms of $N$ such that $|N_i'| \equiv N_i$ and the type of $N_i'$ is $A_i$ with $A_i =_\beta \sigma_i$.
2. If $|N| \equiv \lambda y.P$ then $N$ is an abstraction.

Furthermore, there exists $n \ge 0$ such that $A{\downarrow_\beta} =_\beta \Pi u_1 : D_1 \ldots \Pi u_n : D_n.A'[N_1/v_1, \ldots, N_m/v_m]$ for some terms $N_1, \ldots, N_m$, where $A'$ is the type of $N$, $A$ is the type of $M$ and each $\Pi u_i : D_i \ldots$ is a type with $D_i \in \mathsf{Kind}(\lambda_{R\omega})$.

*Proof*
The part about $C[\,]$ and its properties easily follows by definition of the map $|-|$. For the rest we use the Stripping Lemma. □

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Lemma 6.5*
Let $M \in \mathsf{Object}(\lambda_{R\omega})$ such that $\Gamma \vdash M : A$ and $|M|$ is in $\beta$-normal form. Then there exist $B$ and $\sigma$ such that $B \vdash^{\wedge} |M| : \sigma$. Besides,

1. if $A$ is an arrow-ground type then $\sigma \equiv A{\downarrow_\beta}$,
2. if $x : \delta \in \Gamma$ with $\delta$ arrow-ground type, then $x : \delta{\downarrow_\beta} \in B$.

*Proof*
By induction on the structure of $|M|$.

1. If $|M|$ is a constant then $|M|$ is typable and the second condition is trivially satisfied. The first condition is a consequence of Lemma 6.4.
2. If $|M| \equiv x$ is a variable we distinguish the four possible cases:
   - if $A$ is not arrow-ground and $\delta$ is so, $x : \delta{\downarrow_\beta} \vdash^{\wedge} x : \delta{\downarrow_\beta}$ satisfies the conditions;
   - if $\delta$ is not arrow-ground and $A$ is so, $x : A{\downarrow_\beta} \vdash^{\wedge} x : A{\downarrow_\beta}$ satisfies the conditions;
   - if both $A$ and $\delta$ are arrow-ground, by Stripping (*iii*) and the fact we are in $\lambda_{R\omega}$, we have that $A{\downarrow_\beta} \equiv \delta{\downarrow_\beta}$. Hence $x : A{\downarrow_\beta} \vdash^{\wedge} x : A{\downarrow_\beta}$ satisfies the conditions;
   - if neither $A$ nor $\delta$ is arrow-ground, any application of (*var*), say $x : s \vdash^{\wedge} x : s$ with $s \in \mathscr{S}$, works.
3. Otherwise, since $|M|$ is in normal form it has necessarily to be of the form $\lambda y_1 \ldots y_n . g N_1 \ldots N_m$ where $g$ is either a variable or $g \in \mathscr{F}$ and the $N_i$'s have the same form as $|M|$. We distinguish three cases:
   (a) $n = 0$, $m > 0$ and $g$ is a variable.
   By the form of the term there exist $\Gamma'$ and $N_i'$, $A_i$ $(1 \leq i \leq m)$, such that $\Gamma \subseteq \Gamma'$, $|N_i'| = N_i$ and $\Gamma' \vdash N_i' : A_i$. Since the $N_i'$'s are objects themselves (they have not been erased by the type erasing function), by induction we get that, for $1 \leq i \leq m$, there exist $B_i, \sigma_i$ such that $B_i \vdash^{\wedge} N_i : \sigma_i$, and besides, if $A_i$ is arrow-ground then $\sigma_i \equiv A_i{\downarrow_\beta}$, and if $x : \delta \in \Gamma'$ with $\delta$ arrow-ground then $x : \delta{\downarrow_\beta} \in B_i$. It is now straightforward to get a derivation for $B_1 \cup^{\wedge} \ldots \cup^{\wedge} B_m \cup^{\wedge} \{g : \sigma_1 \to \ldots \to \sigma_m \to \sigma\} \vdash^{\wedge} |M| : \sigma$, where the symbol $\cup^{\wedge}$ denotes the following operation between bases: $B \cup^{\wedge} B' = \{x : \sigma \wedge \tau \mid x : \sigma \in B \text{ and } x : \tau \in B'\} \cup \{x : \sigma \mid x : \sigma \in B \text{ and } x \notin B'\} \cup \{x : \tau \mid x \notin B \text{ and } x : \tau \in B'\}$ (it is straightforward to check that if $B \vdash^{\wedge} M : \beta$ then we have also $B \cup^{\wedge} B' \vdash^{\wedge} M : \beta$ for any $B'$). Notice that if $x : \delta \in \Gamma'$ with $\delta$ arrow-ground then the type of $x$ in $B_1 \cup^{\wedge} \ldots \cup^{\wedge} B_m \cup^{\wedge} \{g : \sigma_1 \to \ldots \to \sigma_m \to \sigma\}$ is not an intersection.
   Moreover:
   – the first condition is satisfied since, in case $A$ is arrow-ground we can choose $\sigma \equiv A{\downarrow_\beta}$; otherwise we can take a fresh type variable;
   – by the conditions satisfied (by induction) by the $B_i \vdash^{\wedge} N_i : \sigma_i$'s, and by the fact noticed above, we get that, for all $x \neq g$, $x : \delta \in \Gamma$ with $\delta$ arrow-ground implies $x : \delta{\downarrow_\beta} \in B$. For what concerns $g$ we know that if

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

$g : \delta \in \Gamma$ with $\delta$ arrow-ground then $\delta{\downarrow_\beta} \equiv \sigma_1 \to \ldots \to \sigma_m \to \sigma$ by Lemma 6.4. So the second condition is satisfied as well.

(b) $n = 0$, $m > 0$ and $g \in \mathscr{F}_{\sigma_1 \to \ldots \to \sigma_p \to \sigma}$.
By the form of the term and Lemma 6.4, there exist $\Gamma'$ and $N'_i, A'_i$ $(1 \le i \le m)$ such that $|N'_i| = N_i$ and $\Gamma' \vdash N'_i : A'_i$, and $\sigma_1 \to \ldots \to \sigma_p \to \sigma =_\beta A'_1 \to \ldots \to A'_m \to \tau$ (that is, $A'_i =_\beta \sigma_i$ for $1 \le i \le m$, and $\tau =_\beta \sigma_{m+1} \to \sigma_p \to \sigma$). By induction, for $1 \le i \le m$ there exists $B_i$ such that $B_i \vdash^\wedge N_i : \sigma_i$ and if $x : \rho \in \Gamma$ then $x : \rho{\downarrow_\beta} \in B_i$. Moreover, if $A$ is arrow-ground then $\Gamma \vdash M : \sigma_{m+1} \to \ldots \to \sigma_p \to \sigma \equiv A{\downarrow_\beta}$ (by Lemma 6.4). Then, $B_1 \cup^\wedge \ldots \cup^\wedge B_m \vdash^\wedge g N_1 \ldots N_m : \sigma_{m+1} \to \ldots \to \sigma_p \to \sigma$ satisfies the conditions.

(c) $n \neq 0$, $|M| \equiv \lambda y.N$.
By the form of the term, Lemma 6.4 and Stripping, there exist $\Gamma', N', A'$ such that $\Gamma \subseteq \Gamma'$, $\Gamma' \vdash N' : A'$ and $|N'| = N$, and $A =_\beta \Pi\alpha_1 : \beta_1 \ldots \Pi\alpha_k : \beta_k . \Pi y : \tau . A'[N_1/v_1, \ldots, N_n/v_n]$, where $\alpha_1 : \beta_1, \ldots, \alpha_k : \beta_k, y : \tau \subseteq \Gamma'$, the $\beta_i$ are kinds and $\tau$ is a type. Now, if $A$ is arrow-ground then $k = 0$ and $A'$, $\tau$ are arrow-ground, and then $A{\downarrow_\beta} \equiv \Pi y : \tau{\downarrow_\beta} . A'{\downarrow_\beta}$. By induction, $B \vdash^\wedge N : \sigma$ with $\sigma \equiv A'{\downarrow_\beta}$ and $y : \tau{\downarrow_\beta} \in B$. Hence $B - \{y : \tau{\downarrow_\beta}\} \vdash^\wedge \lambda y.N : \tau{\downarrow_\beta} \to \sigma \equiv A{\downarrow_\beta}$ satisfies both conditions.

□

*Lemma 6.6*
Let $M \in \mathsf{Object}(\lambda_{R\omega})$ and $|M| \equiv Q[P/x]$ where $P$ is a term such that if $x \notin FV(Q)$ then $P$ is in $\beta$-normal form, and $\exists P' \in \mathsf{Object}(\lambda_{R\omega})$ s.t. $|P'| \equiv P$. Then

$$\exists B, \sigma \; s.t. \; B \vdash^\wedge Q[P/x] : \sigma \;\Rightarrow\; \exists B' \; s.t. \; B' \vdash^\wedge (\lambda x.Q)P : \sigma$$

*Proof*
We consider two cases according to whether $x$ does occur free in $Q$ or not. We can assume that $x$ does not occur in $B$.

1. $x$ occurs in $Q$.
Let $B \vdash^\wedge Q[P/x] : \sigma$ be a deduction in $\lambda_{\wedge R}$. We shall consider only the occurrences of $P$ in $Q[P/x]$ which replace occurrences of $x$ in $Q$. Let $\{B_i \vdash^\wedge P : \delta_i \mid i \in I\}$ be the set of all the conclusions (in the previous deduction) whose subjects are such occurrences of $P$. Then it is not difficult to obtain a deduction of $B_1 \cup^\wedge \ldots \cup^\wedge B_n \vdash^\wedge P : \bigwedge_{i \in I} \delta_i$. (Note that $\cup$ is not sufficient because the same variable could be bound more than once in a term.) Moreover, we can obtain a deduction of $B, x : \bigwedge_{i \in I} \delta_i \vdash^\wedge Q : \sigma$ by extending the basis $B$ and replacing in the proof of $B \vdash^\wedge Q[P/x] : \sigma$ the deductions of $B, x : \bigwedge_{i \in I} \delta_i \vdash^\wedge x : \delta_j$ (obtained using rules $(Ax)$ and $(\wedge E)$ in $\lambda_{\wedge R}$, see the appendix) for the subdeductions of $B_j \vdash^\wedge P : \delta_j$. So by rule $(\to I)$ in $\lambda_{\wedge R}$ we have that $B \vdash^\wedge \lambda x.Q : (\bigwedge_{i \in I} \delta_i) \to \sigma$ and since we have $B_1 \cup^\wedge \ldots \cup^\wedge B_n \vdash^\wedge P : \bigwedge_{i \in I} \delta_i$ we can obtain $B' \vdash^\wedge (\lambda x.Q)P : \sigma$ where $B' = B_1 \cup^\wedge \ldots \cup^\wedge B_n \cup^\wedge B$.
2. $x$ does not occur in $Q$.
By the hypothesis and by Lemma 6.5 we know that $\exists B_1, \delta$ such that $B_1 \vdash^\wedge P : \delta$. Besides, we know that $\exists B, \sigma$ such that $B \vdash^\wedge Q[P/x] : \sigma$, i.e., $B \vdash^\wedge Q : \sigma$ since $x$ is not free in $Q$. Then $B, x : \delta \vdash^\wedge Q : \sigma$ and from this $B \vdash^\wedge \lambda x.Q : \delta \to \sigma$. Now it is easy to construct a proof of $B' \vdash^\wedge (\lambda x.Q)P : \sigma$ where $B' = B \cup^\wedge B_1$.

□

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Lemma 6.7*
Let $M[M'/x] \in \mathsf{Object}(\lambda_{R\omega})$ where $x \in Var^\star$.

$$|M[M'/x]| \equiv |M|[|M'|/x].$$

*Proof*
By induction on the structure of $M$.

1. If $M \equiv f \in \mathscr{F}$ or $M \in Var^\star$ it is trivial.
2. If $M \equiv \lambda y{:}N.N'$ with $y \equiv x$ it is also trivial. Let us consider the case $y \not\equiv x$. $|M[M'/x]| \equiv |\lambda y{:}N[M'/x].N'[M'/x]|$. Now there are two subcases:
   (a) If $N \in \mathsf{Type}(\lambda_{R\omega})$ then $|M[M'/x]| \equiv \lambda y.|N'[M'/x]|$, which, by induction, is equal to $\lambda y.|N'|[|M'|/x] \equiv |M|[|M'|/x]$.
   (b) If $N \in \mathsf{Kind}(\lambda_{R\omega})$ then $|M[M'/x]| \equiv |N'[M'/x]|$, which by induction, is equal to $|N'|[|M'|/x] \equiv |M|[|M'|/x]$.
3. If $M \equiv M_1M_2$ then $|M[M'/x]| \equiv |M_1[M'/x]M_2[M'/x]|$. Now there are again two subcases:
   (a) If $M_2 \in \mathsf{Object}(\lambda_{R\omega})$ then $|M[M'/x]| \equiv |M_1[M'/x]||M_2[M'/x]|$, which, by induction, is equal to $|M_1|[|M'|/x]|M_2|[|M'|/x] \equiv |M|[|M'|/x]$.
   (b) If $M_2 \in \mathsf{Constr}(\lambda_{R\omega})$ then $|M[M'/x]| \equiv |M_1[M'/x]|$ which, by induction, is equal to $|M_1|[|M'|/x] \equiv |M|[|M'|/x]$.

□

*Lemma 6.8*
Let $M \in \mathsf{Object}(\lambda_{R\omega})$. If $|M| \to_\beta N$ then there exists $M' \in \mathsf{Object}(\lambda_{R\omega})$ such that $N \equiv |M'|$ and $M \to_\beta M'$.

*Proof*
To have $|M| \to_\beta N$ it has to be $M \equiv C'[(\lambda x{:}A.q)q']$ where $C'[\,]$ is a context and $A \in \mathsf{Type}(\lambda_{R\omega})$. Then $|M| \equiv C[(\lambda x.|q|)|q'|] \to_\beta C[|q|[|q'|/x]] \equiv N$ where $C[\,]$ is the context obtained by applying the function $|-|$ to $C'$ in the obvious way. Then we can take $M' = C'[q[q'/x]]$ and we have $M \to_\beta M'$ and $|M'| = N$ by Property 6.3 and Lemma 6.7. □

*Theorem 6.9*
If $M \in \mathsf{Object}(\lambda_{R\omega})$ then there exist $B, \sigma$ such that $B \vdash^\wedge |M| : \sigma$.

*Proof*
$|M|$ is $\to_\beta$-strongly normalizable by Lemma 6.1 and Lemma 6.8. Then all the reduction strategies allow us to get the $\beta$-normal form of $|M|$, in particular the reduction strategy according to which a contraction $(\lambda x.Q)P \to_\beta Q[P/x]$ is performed only if $P$ is in $\beta$-normal form (i.e. the rightmost-innermost evaluation). Hence we can find $B$ and $\sigma$ by applying Lemma 6.5 to the $\beta$-normal form of $|M|$ and iterating Lemma 6.6 backwards along the chain of reduction steps which leads from $|M|$ to its $\beta$-normal form. □

So, we have proved that if $M \in \mathsf{Object}(\lambda_{R\omega})$ then $|M| \in \Lambda_{\wedge R}$.

*Lemma 6.10*
Let $t$ and $t'$ be two terms of $\lambda_{R\omega}$ such that $t \to t' \in R$. Then $|t| \equiv t$ and $|t'| \equiv t'$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Proof*
Easy by Stripping and the definitions of $|-|$ and of cube-embeddable rewrite rule. □

In the following we will prove that if $M$ and $N$ are objects, $M \to_{R\beta} N$ implies $|M| \to^*_{R\beta} |N|$. We shall use the notions of $\beta^0$-, $\beta^2$-, and $\beta^\omega$-reduction as introduced in Definition 3.14.

Let $\varphi$ be a substitution for $\lambda_{R\omega}$, with $Dom(\varphi) \subseteq Var^\star$. We define the substitution $|\varphi|$ for terms in $\Lambda_{\wedge R}$ in the following way: $Dom(|\varphi|) = Dom(\varphi)$ and for each $x \in Dom(|\varphi|)$, $x|\varphi| = |x\varphi|$.

*Lemma 6.11*
Let $M, N \in \mathsf{Object}(\lambda_{R\omega})$. If $M \to_{R\beta} N$ then $|M| \to_{R\beta} |N|$, or $|M| \equiv |N|$ if the reduction is actually a $\beta^2$- or a $\beta^\omega$-reduction.

*Proof*
Let $M \equiv C[P]$ where $P$ is a $\beta$-redex or an $R$-redex and $C[]$ is a suitable context. Let us check the statement of the lemma for each notion of reduction, distinguishing, in the case of a $\beta$-reduction, which sort of $\beta$-reduction it is.

Let $P$ be the $\beta^\omega$-redex $(\lambda a{:}A.Q)Q'$. It is easy to check that, by definition of $|-|$, if $P : S : \Box$ then $|C[P]| = |C[U]|$ for any $U : S$. We have therefore $|M| \equiv |N|$.

Let $P$ be the $\beta^2$-redex $(\lambda a{:}A.Q)Q'$. We have that $|(\lambda a{:}A.Q)Q'| \equiv |Q| \equiv |Q[Q'/a]|$ by definition of $|-|$, and then, by Property 6.3, $|M| \equiv |N|$.

Let $P$ be the $\beta^0$-redex $(\lambda x{:}A.Q)Q'$. We have that $|(\lambda x{:}A.Q)Q'| \equiv (\lambda x.|Q|)|Q'| \to_\beta |Q|[|Q'|/x]$ and $|Q|[|Q'|/x] \equiv |Q[Q'/x]|$ by Lemma 6.7. It follows then, by Property 6.3, $|M| \to_\beta |N|$.

Let $P \equiv t\varphi$ and $t\varphi \to_R t'\varphi$ using a rule $r : t \to t'$. By Lemma 6.7, $|t\varphi| \equiv |t||\varphi|$ and $|t'\varphi| \equiv |t'||\varphi|$. By Theorem 6.9, $|t||\varphi|, |t'||\varphi| \in \Lambda_{\wedge R}$. Moreover, by Lemma 6.10, $t \equiv |t|$ and $t' \equiv |t'|$. Then $|t\varphi| \equiv |t||\varphi| \to_R |t'||\varphi| \equiv |t'\varphi|$ and hence, by Property 6.3, it follows $|M| \to_R |N|$. □

We can prove now the main theorem of this section.

*Theorem 6.12*
$\to_{R\beta}$-strong normalization of $\lambda_{\wedge R}$ implies $\to_{R\beta}$-strong normalization of $\lambda_{R\omega}$.

*Proof*
By Lemma 3.11 only the cases $M \in \mathsf{Constr}(\lambda_{R\omega})$ and $M \in \mathsf{Object}(\lambda_{R\omega})$ have to be considered.

If $M \in \mathsf{Constr}(\lambda_{R\omega})$ then, since $\lambda_{R\omega}$ objects cannot occur in constructors (Lemma 2.14), it follows that any reduction in $M$ is actually a $\beta$-reduction. Hence, strong normalization of $M$ follows from Lemma 6.1.

If $M \in \mathsf{Object}(\lambda_{R\omega})$ then let us define the following interpretation $\mathscr{I}$ on the elements of $\mathsf{Object}(\lambda_{R\omega})$: for $P \in \mathsf{Object}(\lambda_{R\omega})$, $\mathscr{I}(P) = \langle |P|, P\rangle$. We can now define the following ordering on $\{\mathscr{I}(P) \mid P \in \mathsf{Object}(\lambda_{R\omega})\}$:

$$>= (\to_{R\beta}, \to_{\beta^2} \cup \to_{\beta^\omega}),$$

which is well-founded since, for any $P \in \mathsf{Object}(\lambda_{R\omega})$, $|P|$ is typable in $\lambda_{\wedge R}$ (i.e.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

$|P| \in \Lambda_{\wedge R}$) by Theorem 6.9, and hence $\rightarrow_{R\beta}$-strongly normalizable by hypothesis. Besides, $\rightarrow_\beta$ is strongly normalizing on elements of $\lambda_{R\omega}$ by Lemma 6.1.

$\rightarrow_{R\beta}$-strong normalization of $M$ can now be derived by the well-foundedness of $>$ once one realizes that, by Lemma 6.11, for any $P \in \mathsf{Object}(\lambda_{R\omega})$ such that $P \rightarrow_{R\beta} P'$, $\mathscr{I}(P) > \mathscr{I}(P')$. □

## 7 Modularity of confluence

We recall first the definition of confluence. A reduction relation $\rightarrow$ is *confluent* if for any $t$, $v_1$ and $v_2$ such that $t \rightarrow^* v_1$ and $t \rightarrow^* v_2$, there exists $v_3$ such that $v_1 \rightarrow^* v_3$ and $v_2 \rightarrow^* v_3$.

Local confluence is a closely related (weaker) property. $\rightarrow$ is *locally confluent* if for any $t$, $v_1$ and $v_2$ such that $t \rightarrow v_1$ and $t \rightarrow v_2$, there exists $v_3$ such that $v_1 \rightarrow^* v_3$ and $v_2 \rightarrow^* v_3$.

For strongly normalizing relations, local confluence is equivalent to confluence (Newman's Lemma – Newman, 1942). So we shall prove that $\rightarrow_{\beta R}$ is locally confluent on $\lambda_{RC}$. The notion of critical pair is crucial in this proof. Let us recall the definition. Assume terms are represented as trees where the application operator appears explicitly.

*Definition 7.1*
If $l \rightarrow r$ and $s \rightarrow t$ are two rewrite rules (we assume that the variables of $s \rightarrow t$ were renamed so that there is no common variable with $l \rightarrow r$), $p$ is the position of a non-variable subterm of $s$ and $\mu$ is a most general unifier of $s|_p$ and $l$, then $(t\mu, s\mu[r\mu]_p)$ is a *critical pair* formed from those rules. Note that $s \rightarrow t$ may be a renamed version of $l \rightarrow r$. In this case a superposition at the root position is not considered a critical pair.

Thus, a critical pair arises from a most general non-variable overlap between two left-hand sides. Overlaps of higher order variables applied to some arguments do generate critical pairs (these are non-variable overlaps due to the application operator):

*Example 7.2*
Consider the $\beta$-rule: $(\lambda x.M)N \rightarrow M[N/x]$[4]. If a rule $r$ in $HOR$ contains the term $Xt$ (where $X$ is a higher-order variable and $t$ is an arbitrary term) as a subterm of the left-hand side – for instance, consider $r : F(X0) \rightarrow X$ – then there is a critical pair between $r$ and $\beta$: $(\lambda x.M, F(M[0/x]))$.

The following lemmas show that the confluence of $FOR$ for algebraic terms transfers to $\lambda_{RC}$-terms, and that for the class of higher-order rewrite systems which we consider, the absence of critical pairs implies confluence (this is not true for arbitrary higher-order rewrite systems (Nipkow, 1991)).

[4] This is actually a 'meta-rule', or a rule schema. Although one cannot write this rule in $HOR$, it is possible to compute the critical pairs generated by the superpositions of this rule scheme on the left-hand sides of $HOR$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

*Lemma 7.3*
If $FOR$ is confluent on the set of algebraic terms of $\lambda_{RC}$ then $\rightarrow_{FOR}$ is locally confluent on $\lambda_{RC}$.

*Proof*
By induction on the structure of terms. Let $M$ be a term in $\lambda_{RC}$. If $M$ is algebraic the thesis follows trivially from the hypothesis. If $M$ is not algebraic we consider two cases:

1. $M \equiv \star$, $M \equiv \Box$, $M \equiv xP_1 \ldots P_n$ $(n \geq 0)$, $M \equiv (\lambda x : A.P_0)P_1 \ldots P_n$ $(n \geq 0)$, $M \equiv (\Pi x : A.P_0)P_1 \ldots P_n$ $(n \geq 0)$, or $M \equiv FP_1 \ldots P_n$ where $F$ is a higher-order function symbol and $n \geq 0$. In this case the thesis follows from the induction hypothesis, since all the redexes are strictly inside the terms.
2. Otherwise the root of $M$ is a first-order function symbol. In this case we are going to use the notions of *cap* and *aliens*: let $M \equiv ft_1 \ldots t_n$ where $f$ is a first-order function symbol, an *alien subterm* of $M$ is a maximal subterm of $M$ which is not of the same form (that is, a maximal subterm of $M$ which is not rooted by a first-order function symbol). We will denote by $aliens(M)$ the multiset of alien subterms of $M$. The *cap* of $M$ is the first-order algebraic term obtained from $M$ by replacing its aliens by variables (all the occurrences of the same alien subterm are replaced by the same variable).
   Since by assumption the root of $M$ is a first-order function symbol, the cap of $M$ is not a variable, and then $\rightarrow_{FOR}$ is confluent on $aliens(M)$ by the induction hypothesis. Then, we only have to consider the case where $M \rightarrow_{FOR} N_1$ and $M \rightarrow_{FOR} N_2$ in (non-variable) cap positions. In this case $cap(M) \rightarrow_{FOR} cap(N_1)$ and $cap(M) \rightarrow_{FOR} cap(N_2)$ and by hypothesis there exists $N'$ such that $cap(N_1) \rightarrow^*_{FOR} N'$ and $cap(N_2) \rightarrow^*_{FOR} N'$. Each variable $z_i$ of $N'$ appears also in $cap(M)$. Let $A_i$ be the subterm of $M$ replaced by $z_i$ to obtain $cap(M)$. Then, $N_1 \rightarrow^*_{FOR} N'[A_i/z_i]$ and $N_2 \rightarrow^*_{FOR} N'[A_i/z_i]$.

Having proved local confluence, confluence now follows from the strong normalization property, by Newman's Lemma. □

*Lemma 7.4*
Let $HOR$ be a higher-order rewrite system satisfying the general schema. If $HOR$ does not have critical pairs, then $\rightarrow_{HOR}$ is locally confluent on $\lambda_{RC}$.

*Proof*
To prove local confluence, it is sufficient to show the commutation of $\rightarrow_{HOR}$ reductions on overlapped redexes. Let $t$ be a term in $\lambda_{RC}$ such that $t \rightarrow_{HOR} v_1$ at position $p$ and $t \rightarrow_{HOR} v_2$ at position $p.q$. Since there are no critical pairs, the subterm $t|_{p.q}$ of $t$ must be covered by a variable $z$ of the rule applied at position $p$. Let $t'$ be the term obtained out of $t$ by replacing the subterm at position $p.q$ and all other occurrences of $t|_{p.q}$ corresponding to $z$ by a new variable $x$. Then, $t'$ is still reducible at position $p$: $t' \rightarrow_{HOR} v'$. If $x$ appears in $v'$ at positions $m_1, \ldots, m_n$ then $t|_{p.q}$ appears in $v_1$ at the same positions. Let $t''$ be the term obtained after reducing $v_1$ at positions $m_1, \ldots, m_n$. Then $v_2 \rightarrow_{HOR} t''$ at position $p$. Hence $HOR$ is locally confluent, therefore confluent as a consequence of Newman's Lemma. □

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

Now, using a similar argument (and the previous lemmas), we can prove that $\rightarrow_{\beta R}$ is confluent.

*Theorem 7.5* (*Local Confluence of* $\rightarrow_{R\beta}$ *in* $\lambda_{RC}$)
If $FOR$ is confluent on the set of algebraic terms, and $HOR$ does not introduce critical pairs (i.e. there is no critical pair between rules of $HOR$, between $FOR$ and $HOR$, between $\beta$ and $HOR$[5]) then $\rightarrow_{R\beta}$ is locally confluent in $\lambda_{RC}$.

*Proof*
It suffices to show the commutation of $\beta$-, $\rightarrow_{FOR}$- and $\rightarrow_{HOR}$-reductions on overlapped redexes. But since $\rightarrow_{\beta}$ is confluent, and $\rightarrow_{HOR}$ and $\rightarrow_{FOR}$ are locally confluent (by Lemmas 7.4 and 7.3), it is sufficient to prove that for all $t$ such that $t \rightarrow_{R\beta} v_1$ at position $p$ using one of the reduction relations, and $t \rightarrow_{R\beta} v_2$ at position $p.q$ using a different reduction relation, there exists $v_3$ such that $v_1 \rightarrow^*_{R\beta} v_3$ and $v_2 \rightarrow^*_{R\beta} v_3$.

Since there are no critical pairs, the subterm $t|_{p.q}$ of $t$ must be covered by a variable $z$ of the rule applied at position $p$. Let $t'$ be the term obtained out of $t$ by replacing the subterm at position $p.q$ and all other occurrences of $t|_{p.q}$ corresponding to $z$ by a new variable $x$. Then, $t'$ is still reducible at position $p$: $t' \rightarrow_{R\beta} v'$. If $x$ appears in $v'$ at positions $m_1, \ldots, m_n$ then $t|_{p.q}$ appears in $v_1$ at the same positions. Let $t''$ be the term obtained after reducing $v_1$ at positions $m_1, \ldots, m_n$. Then $v_2 \rightarrow_{R\beta} t''$ at position $p$.

Hence $\rightarrow_{R\beta}$ is locally confluent, therefore confluent by Newman's Lemma. □

For example, the class of higher-order rewrite systems defining higher-order functions by primitive recursion (structured recursion) on first-order data structures, satisfy the required hypothesis and then $\rightarrow_{R\beta}$ is confluent in this case.

## 8 Conclusions

We have extended the Calculus of Constructions with algebraic types and rewrite rules. Our system is closely related to the Calculus of Constructions with inductive types (CCI) defined by Coquand and Paulin-Mohring (1990), since CCI can be seen as an extension of the Calculus of Constructions with a particular class of higher-order rewrite rules. The strong normalization of CCI was proved by Werner (1994). The problem of extending the CCI with pattern-matching definitions was studied by Coquand (1992). In particular, in Coquand (1992) there is a notion of recursive schema ensuring strong normalization, and rewrite rules are assumed critical-pair free. In our framework these restrictions apply only to higher-order rules (first-order rules are simply required to be non-duplicating).

Confluence and strong normalization are essential properties of logical systems, since they ensure the consistence of the system. Proving these properties is in general a difficult task, so, it is important to study under which conditions these proofs are modular. Our results show that, to prove strong normalization of any of the systems in the $\lambda R$-cube, it is sufficient to prove termination of the first-order rewrite rules in $R$

[5] See Example 7.2.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

on first-order terms, provided that $R$ satisfies certain syntactical conditions, namely non-duplication for $FOR$ and the general schema for $HOR$. As a consequence, we get the strong normalization of a restriction of CCI (with pattern-matching) where the inductive types are defined by structural induction. The general schema, however, limits the power of the higher-order rules. More powerful versions of the schema are described by Jouannaud and Okada (1996). The generalization of the proof of strong normalization to those systems of higher-order rules will be the subject of future work.

**Acknowledgements**

We would like to thank Mariangiola Dezani and Jean-Pierre Jouannaud for their scientific guidance. The first author is also grateful to Antonietta and Silvia Cocci for many useful discussions.

## A System $\lambda_{\wedge R}$

System $\lambda_{\wedge R}$ is a type assignment system with intersection types and algebraic features which was introduced in Barbanera and Fernández (1996), where a slightly different but equivalent presentation is provided.

Type assignment systems (also called type inference systems) are formal systems for assigning types to untyped terms. These systems are defined by specifying a set of terms, a set of types one assigns to terms, and a set of type inference rules. The rules are usually given in a natural deduction style. Here, we use a slight variation of the standard presentation, to keep track of the premises statements depend on. We shall refer to Hindley and Seldin (1986) for all the notions about type assignment systems that we do not define explicitly.

The particular type assignment system we are going to define contains algebraic features in the style we have used so far, and intersection types. Type assignment systems containing intersection types for the $\lambda$-calculus were originally devised in Coppo and Dezani-Ciancaglini (1980) and Coppo *et al.* (1980), and investigated afterwards in depth in several papers, among which we recall Barendregt *et al.* (1983), Pottinger (1980), Coppo *et al.* (1984) and van Bakel (1993). We refer to the above-mentioned papers and to the surveys by Cardone and Coppo (1990) and Hindley (1990) for motivations and applications of intersection types.

We begin the description of system $\lambda_{\wedge R}$ by considering a set $\mathscr{S}$ of sorts and a set of (untyped) function symbols $F = \{f_1, f_2, \ldots, f_n\}$. Each function symbol is equipped with an *arity*, denoted by superscripts when not clear from the context.

*Definition A.1* (*Types*)
The set $\mathsf{T}_{\mathscr{S}\wedge}$ of types of $\lambda_{\wedge R}$ is defined as follows:

- If $s \in \mathscr{S}$ then $s \in \mathsf{T}_{\mathscr{S}\wedge}$.
- If $\varphi \in \mathscr{V}$ then $\varphi \in \mathsf{T}_{\mathscr{S}\wedge}$, where $\mathscr{V}$ is the set of type variables.
- If $\sigma, \tau \in \mathsf{T}_{\mathscr{S}\wedge}$ then $\sigma \to \tau \in \mathsf{T}_{\mathscr{S}\wedge}$.
- If $\sigma, \tau \in \mathsf{T}_{\mathscr{S}\wedge}$ then $\sigma \wedge \tau \in \mathsf{T}_{\mathscr{S}\wedge}$.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

We will consider types modulo associativity, commutativity and idempotency of the type operator $\wedge$.

A type is *algebraic* if it does not contain $\wedge$ and type variables. As in section 2, we denote by $\mathsf{T}_{\mathscr{S}}$ the set of algebraic types.

*Definition A.2* (*Terms*)
The terms of $\lambda_{\wedge R}$ are defined by the following grammar:

$$\Lambda_F ::= \mathbf{x} \mid \mathbf{f} \mid (\Lambda_F \Lambda_F) \mid \lambda \mathbf{x}.\Lambda_F$$

where $\mathbf{x}$ ranges over a set $\mathscr{X}$ of (untyped) variables and $\mathbf{f}$ over $F$. Terms are then untyped $\lambda$-terms with constants and on them the usual notion of $\beta$-reduction is defined.

*Definition A.3*

(i) A *statement* is an expression of the form $M : \sigma$ where $\sigma \in \mathsf{T}_{\mathscr{S}\wedge}$ and $M \in \Lambda_F$. $M$ is the *subject* of the statement.

(ii) A *basis* (the set of assumptions a statement depends on) is a set of statements with only variables as subjects. Moreover, there are no two statements with the same subject. If $x$ does not occur in the basis $B$ then $B, x : \sigma$ denotes the basis $B \cup \{x : \sigma\}$.

(iii) A *set of axiom statements* (for a set of constants $F = \{f_1, f_2, \ldots, f_n\}$) is a set of statements of the form $\{f_1{:}\sigma_1, f_2{:}\sigma_2, \ldots, f_n{:}\sigma_n\}$ where $\sigma_i \in \mathsf{T}_{\mathscr{S}}$ $(1 \leq i \leq n)$ and is of the form $\tau_1 \to \ldots \to \tau_m \to \tau$ for $m$ arity of $f_i$.

*Definition A.4* (*Inference rules*)
Let $\mathsf{Ax}$ be a set of axiom statements.

$$(\text{var}) \quad B, x{:}\sigma \vdash^{\wedge} x : \sigma$$

$$(\text{Ax}) \quad B \vdash^{\wedge} f : \sigma \qquad \text{for any } f : \sigma \in \mathsf{Ax}$$

$$(\to I) \quad \frac{B, x{:}\sigma \vdash^{\wedge} M : \tau}{B \vdash^{\wedge} \lambda x.M : \sigma \to \tau}$$

$$(\to E) \quad \frac{B \vdash^{\wedge} M : \sigma \to \tau \quad B \vdash^{\wedge} N : \sigma}{B \vdash^{\wedge} (MN) : \tau}$$

$$(\wedge I) \quad \frac{B \vdash^{\wedge} M : \sigma \quad B \vdash^{\wedge} M : \tau}{B \vdash^{\wedge} M : \sigma \wedge \tau}$$

$$(\wedge E) \quad \frac{B \vdash^{\wedge} M : \sigma \wedge \tau}{B \vdash^{\wedge} M : \sigma}$$

Then the set of statements $\mathsf{Ax}$ is a parameter system $\lambda_{\wedge R}$ depends upon.

A term $M$ will be called *typable* if there exists a basis $B$ and a type $\sigma$ such that $B \vdash^{\wedge} M : \sigma$.

We shall denote by $\Lambda_{\wedge R}$ the *set of typable terms*.

To complete the description of system $\lambda_{\wedge R}$ we have to define algebraic rewriting.

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

For this, we could define in the present context the notions of first and higher-order constant, algebraic term (first and higher-order), rewrite rule and so on, as done in Barbanera and Fernández (1996). However, we can equivalently consider a set of cube-embeddable rewrite rules as defined in section 2 and the rewrite relation induced by it on $\Lambda_{\wedge R}$.

Before doing that, to be precise, we have to give a small technical definition.

*Definition A.5*
Given a set $\mathscr{S}$ of sorts, a set $F$ of constants, and a set $\mathsf{Ax}$ of axiom statements, let $t$ be an algebraic term for $\mathscr{S}$ and $\mathscr{F}$ as defined in 2.9, where $\mathscr{F}$ is the signature naturally induced by $F$ and $\mathsf{Ax}$. We denote by $\mathbf{t}$ the untyped term obtained by replacing any occurrence of a function symbol of $\mathscr{F}$ in $t$ by its untyped counter-part in $F$.

In the rest of this section we will implicitly assume a signature to be induced by a set of constants and a set of axiom statements.

*Definition A.6* (*Algebraic rewriting*)
Let $r \in R$ where $R$ is a set of cube-embeddable rewrite rules for a signature $\mathscr{F}$. Let $M, M' \in \Lambda_{\wedge R}$. The reduction relation $\to_R$ on $\Lambda_{\wedge R}$ is defined as follows: $M \to_R M'$ if $M \equiv C[\mathbf{t}[N_1/x_1, \ldots, N_n/x_n]]$ (where $r : t \to t' \in R$, $N_1, \ldots, N_n \in \Lambda_{\wedge R}$ and $C[-]$ is a context) and $M' \equiv C[\mathbf{t}'[N_1/x_1, \ldots, N_n/x_n]]$. Note that since $N_1, \ldots, N_n \in \Lambda_{\wedge R}$ they can contain $\lambda$-terms.

As usual, $\to_{R\beta}$ denotes the union of the reduction relations $\to_R$ and $\to_\beta$ and $\twoheadrightarrow_{R\beta}$ its reflexive and transitive closure.

Note that because of the lemma below, which follows easily by the definition of cube-embeddable rewrite rule, the definition above is sound, i.e. it is not possible to have a term which is possible to rewrite, say $C[\mathbf{t}\varphi] \in \Lambda_{\wedge R}$, such that the type of $\mathbf{t}$ and of its variables are not 'equivalent' to the algebraic types present in the typed term $t$.

*Lemma A.7*
Let $t$ be a *typed* cube-embeddable term in a rule $r : t \to t' \in R$, and let $\sigma$ and $\Gamma'$ be the algebraic type and context of $t$. If $B \vdash^\wedge \mathbf{t} : \tau$ then $\tau \equiv \sigma$ and for all statements $x{:}\gamma \in B$ such that $x$ occurs in $t$: $\gamma \equiv \gamma_1 \wedge \ldots \wedge \gamma_n$ where $\gamma_1$ is the type of the variable $x$ in $t$.

System $\lambda_{\wedge R}$ is then completely specified by a quadruple $\langle \mathscr{S}, F, \mathsf{Ax}, R \rangle$, where $\mathscr{S}$ is a set of sorts, $F$ a set of constants, $\mathsf{Ax}$ a set of axiom statements, and $R$ a set of cube-embeddable rewrite rules for the signature $\mathscr{F}$ induced by $F$ and $\mathsf{Ax}$.

The main property of $\lambda_{\wedge R}$, and the one we need here, is strong normalization. This property was proved in Barbanera and Fernández (1996) in two steps: first considering a rewrite relation induced by a term rewriting system $R = FOR \cup HOR$ such that $FOR$ is non-duplicating and strongly normalizing on the set of first-order algebraic terms *that are typable in* $\lambda_{\wedge R}$, and $HOR$ satisfies the general schema (Theorem 2 in Barbanera and Fernández (1996)). Then it is proved that it is enough to require strong normalization of $FOR$ on the set of *typed (many-sorted)* first-order

https://doi.org/10.1017/S095679689700289X Published online by Cambridge University Press

algebraic terms, since this implies strong normalization on the set of all typeable terms (Theorem 6 in Barbanera and Fernández (1996)). The following theorem is then a direct consequence of the latter:

*Theorem A.8* ($\lambda_{\wedge R} \models \mathsf{SN}$ (*Barbanera and Fernández, 1996*))
Let $\lambda_{\wedge R}$ be the system defined by

$$\langle \mathscr{S}, F, \mathsf{Ax}, R \rangle,$$

where $R$ is a cube-embeddable term rewriting system such that

1. $FOR$ is non-duplicating and strongly normalizing on first-order typed algebraic terms;
2. $HOR$ satisfies the general schema (w.r.t. $FOR$).

Then terms in $\Lambda_{\wedge R}$ are strongly normalizable w.r.t. $\rightarrow_{R\beta}$.

## References

van Bakel, S. (1993) Intersection type disciplines in lambda calculus and applicative term rewriting systems. *PhD thesis*, University of Nijmegen.

Barbanera, F. (1990) Adding algebraic rewriting to the calculus of constructions: Strong normalization preserved. *Proc. 2nd Int. Workshop on Conditional and Typed Rewriting*.

Barbanera, F. and Fernández, M. (1993a) Combining first and higher-order rewrite systems with type assignment systems. *Proc. Int. Conference on Typed Lambda Calculi and Applications*, Utrecht. *Lecture Notes in Computer Science 664*. Springer-Verlag.

Barbanera, F. and Fernández, M. (1993b) Modularity of termination and confluence in combinations of rewrite systems with $\lambda_\omega$. *Proc. 20th Int. Colloquium on Automata, Languages, and Programming*, Lund, Sweden. *Lecture Notes in Computer Science 700*. Springer-Verlag.

Barbanera, F. and Fernández, M. (1996) Intersection Type Assignment Systems with Algebraic Rewriting. *Theoretical Computer Science* **170**, 173–207.

Barbanera, F., Fernández, M. and Geuvers, H. (1994) Modularity of Strong Normalization in the Algebraic-$\lambda$-cube. *Proc. 9th IEEE Symposium on Logic In Computer Science*, Paris, France.

Barbanera, F., Fernández, M. and Geuvers, H. (1996) Modularity of Strong Normalization in the Algebraic-$\lambda$-cube – Preliminary Version. *Rapport de Recherche LIENS96-8*.

Barendregt, H. (1986) *Lambda Calculus: its syntax and semantics, 2nd ed*. North Holland.

Barendregt, H. (1991) Introduction to generalised type systems. *J. Functional Programming*, **1**.

Barendregt, H., Coppo, M. and Dezani-Ciancaglini, M. (1983) A filter $\lambda$-model and the completeness of type assignment. *J. Symbolic Logic*, **48**(4), 931–940.

Berardi, S. (1990) Type dependence and constructive mathematics. *PhD thesis*, Mathematical Institute, University of Torino, Italy.

Breazu-Tannen, V. (1988) Combining algebra and higher-order types. *Proc. 3rd IEEE Symposium on Logic In Computer Science*, Edinburgh, UK.

Breazu-Tannen, V. and Gallier, J. (1991) Polymorphic rewriting conserves algebraic strong normalization. *Theoretical Computer Science*, **83**(1).

Breazu-Tannen, V. and Gallier, J. (1992) Polymorphic rewriting conserves algebraic confluence. *Information and Computation*, **82**, 3–28.

deBruijn, N. G. (1980) A survey of the project Automath. In: J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism.* Academic Press.

Cardone, F. and Coppo, M. (1990) Two extensions of Curry's type inference system. In: P. Odifreddi, editor, *Logic and Computer Science*. Academic Press.

Church, A. (1940) A formulation of the simple theory of types. *J. Symbolic Logic*, **5**.

Coppo, M. and Dezani-Ciancaglini, M. (1980) An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, **21**(4).

Coppo, M., Dezani-Ciancaglini, M. and Venneri, B. (1980) Principal type schemes and λ-calculus semantics. In: J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on combinatory logic, lambda calculus and formalism.* Academic Press.

Coppo, M., Dezani, M., Honsell, F. and Longo, G. (1984) Extended type structures and filter lambda models. *Logic Colloquium 82*, Amsterdam.

Coquand, Th. (1992) Pattern matching with dependent types. *Proc. of the Workshop on Logical Frameworks.*

Coquand, Th. and Paulin-Mohring, C. (1990) Inductively defined types. *Proc. of Colog'88: Lecture Notes in Computer Science 417*. Springer-Verlag.

Coquand, Th. and Huet, G. (1988) The calculus of constructions. *Information and Computation*, **76**, 95–120.

Dershowitz, N. and Jouannaud, J.-P. (1990) Rewrite systems. In: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, vol. B, pp. 243–309. North-Holland.

Dougherty, D. J. (1992) Adding algebraic rewriting to the untyped lambda calculus. *Information and Computation*, **101**, 251–267.

Geuvers, H. (1992) The Church–Rosser property for βη-reduction in typed λ-calculi. *Proc. 7th IEEE Symposium on Logic In Computer Science*, Santa Cruz, CA.

Geuvers, H. (1993) Logics and type systems. *PhD thesis*, Department of Computer Science, University of Nijmegen.

Geuvers, H. and Nederhof, M. J. (1991) A simple modular proof of the strong normalization for the calculus of constructions. *J. Functional Programming*, **1**.

Girard, J.-Y. (1972) Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur. *Thèse d'Etat*, Univ. Paris VII, France.

Harper, R., Honsell, F. and Plotkin, G. (1987) A framework for defining logics. *Proc. 2nd IEEE Symposium on Logic In Computer Science*, Washington, DC.

Hindley, R. (1990) Types with intersection, an introduction. *Formal aspects of Computing.*

Hindley, R. and Seldin, J. (1986) *Introduction to Combinators and λ-calculus.* Cambridge University Press.

Jouannaud, J.-P. and Okada, M. (1991) Executable higher-order algebraic specification languages. *Proc. 6th IEEE Symposium Logic In Computer Science*, Amsterdam.

Jouannaud, J.-P. and Okada, M. (1996) Strong Normalization of Inductive Data Type Systems. Submitted.

Kahrs, S. (1995) Confluence of Curried Term Rewriting Systems. *J. Symbolic Computation*, **19**(6), 601–623.

Kennaway, R., Klop, J. W., Sleep, R. and de Vries, F. J. (1996) Comparing curried and uncurried rewriting. *J. Symbolic Computation*, **21**(1), 15–39.

Klop, J. W. (1987) Term rewriting systems: a tutorial. *EATCS Bulletin*, **32**, 143–182.

Klop, J. W., van Oostrom, V. and van Raamsdonk, F. (1993) Combinatory Reduction Systems, introduction and survey. *Theoretical Computer Science* **121**(1–2).

Newman, M. H. A. (1942) On theories with a combinatorial definition of 'equivalence'. *Ann. Math.*, **43**(2), 223–243.

Nipkow, T. (1991) Higher-order critical pairs. *Proc. 6th IEEE Symposium Logic in Computer Science*, Amsterdam.

Okada, M. (1989) Strong normalizability for the combined system of the types lambda calculus and an arbitrary convergent term rewrite system. *Proc. ISSAC 89*, Portland, OR.

van Oostrom, V. and van Raamsdonk, F. (1993) Comparing combinatory reduction systems and higher-order rewrite systems. *IR 333*, Vrije Universiteit, Amsterdam.

van Oostrom, V. and van Raamsdonk, F. (1994) Weak orthogonality implies confluence: the higher-order case. *Proc. Logical Foundations of Computer Science.*

Pottinger, G. (1980) A type assignment for the strongly normalizable $\lambda$-terms. In: J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism.*. Academic Press.

Rusinowitch, M. (1987) On termination of the direct sum of term rewriting systems. *Information Processing Letters*, **26**, 65–70.

Toyama, Y. (1987) Counterexamples to termination for the direct sum of term rewriting systems. *Infor. Process. Lett.*, **25**, 141–143.

Werner, B. (1994) Méta-théorie du Calcul des Constructions Inductives. *Thèse*, Université Paris VII.