ECOLE NORMALE SUPERIEURE

Modularity of Strong Normalization in the Algebraic-λ-cube

Franco BARBANERA
Maribel FERNÁNDEZ
Herman GEUVERS

LIENS - 96 - 8

Département de Mathématiques et Informatique

CNRS URA 1327

Modularity of Strong Normalization in the Algebraic-λ-cube

Franco BARBANERA[*]
Maribel FERNÁNDEZ
Herman GEUVERS[**]

LIENS - 96 - 8

May 1996

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00
Adresse électronique : maribel@dmi.ens.fr

[*]Universitá di Torino, Italy

Adresse électronique : barba@di.unito.it

[**]Catholic University of Nijmegen
The Netherlands

Adresse électronique : herman@cs.kun.nl

Modularity of Strong Normalization in the algebraic-λ-cube Preliminary Version

Franco Barbanera
Dipartimento di Informatica
Universitá di Torino
Corso Svizzera 185, 10149 Torino, Italy
`barba@di.unito.it`

Maribel Fernández[*]
DMI - LIENS (CNRS URA 1327)
École Normale Supérieure
45, rue d'Ulm, 75005 Paris, France
`maribel@dmi.ens.fr`

Herman Geuvers
Faculty of Mathematics and Informatics
Catholic University of Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
`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 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. This result is proven for the algebraic extension of the Calculus of Constructions, which contains all the systems of the algebraic-λ-cube.

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.

1 Introduction

Many different computational models have been studied by theoretical computer scientists. One of the main motivations for the development of such 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 computational models cannot exploit all its possibilities to help the development of actual computing tools unless also their interactions and possible (in)compatibilities are investigated. In this framework, many research efforts have been devoted in the last years to the study of the interactions between two closely related models of computation: the one based on β-reduction on λ-terms and the one 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 types specification. The combination of these two models has also provided an alternative in the design of new programming languages: the *algebraic functional languages* [JO91]. These languages allow algebraic definitions of data types and operators (as in equational languages like OBJ) and definition of higher-order functions (as in functional languages like ML), in a unified framework.

[*]Part of this work was done while the author was at LRI, Université de Paris Sud.

The study of systems based on λ-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 [Klo87, Dou91]. For typed languages things work out nicely. In [BTG90] and [Oka89] it is shown that the system obtained by combining a terminating first-order many-sorted term rewrite system with the second order typed λ-calculus is again terminating with respect to β-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 [BTG92]. In [JO91] both results are extended to combinations of first- and higher-order rewriting systems with second order λ-calculus, under certain conditions on the form of the rewrite rules.

The question that naturally arises is whether such a nice interaction between typed λ-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 [CH88]. Indeed, considering only first-order algebraic rewriting, this problem has already been addressed in [Barb90]. A strong restriction was however imposed on the definition of the combined system: in the type conversion rule ($conv$) only β-conversion ($=_\beta$) was considered as equality. So, even if there was a rewrite rule $x + 0 \rightarrow x$ 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 the modularity of strong normalization: confluence does not hold in general for $R\beta$-equality, where R is the given set of first-order algebraic rules.

In this paper we extend the Calculus of Constructions, adding not only first- but also higher-order algebraic rewriting, and considering in the type conversion rule the $R\beta$-equality generated by the algebraic reductions together with β-reduction. Considering $R\beta$-equality, a proof of strong normalization can no longer rely on the confluence property. Actually, 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, using $R\beta$-equality in ($conv$) even the definition of the system is more involved. Indeed the rule ($conv$) is part of the definition of the terms of the system and one cannot define a notion of $R\beta$-equality 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 terms of the pure calculus enriched with algebraic constants, and using this relation on pseudoterms in the rule ($conv$). The second solution, to be better discussed 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 [JO91]) algebraic rewriting of all the systems of the so-called λ-cube [Ber88], [Bar91]. This extension will be called *algebraic-λ-cube* (λR-cube for short).

The main result we prove for the systems of the λR-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 said 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 [Geu92] (see also [Geu93]). We prove strong normalization in three steps. By means of a reduction preserving translation we prove strong normalization of the extended Calculus of Constructions to be implied by the same property of system $\boldsymbol{\lambda}_{R\omega}$ (the extended typed λ-calculus of order ω).

The strong normalization property of this last system was proven in [BF93b] by a reduction preserving translation, showing it to be implied by the same property of system $\lambda_{\wedge R}$ (a type assignment system for λ-calculus with intersection types and algebraic rewriting), which in turn was proven strongly normalizing in [BF93a].

Finally, we also prove that local confluence is a modular property of the systems of the algebraic-λ-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 [BFG94]. It is organized as follows: In Section 2 we recall the definition of the λ-cube whose algebraic extension, the λR-cube, will be defined in Section 3. Section 4 will be devoted to the metatheory of the λR-cube. Among others, the Subject Reduction property will be proved. We will then outline the skeleton of the strong normalization proof in Section 5. The main points of such proof will be the topic of Sections 6 and 7 (we recall system $\lambda_{\wedge R}$ in Appendix A). In Section 8 we prove the modularity of confluence. Section 9 contains the conclusions.

2 The Cube of λ-calculi

The λ-cube is a coherent collection of eight type systems. Each system (generically denoted by $\lambda\_$) is placed on a vertex of the cube in a way that geometrically exploits the possible dependencies between types and terms. From another point of view the λ-cube forms a natural fine-structure of the Calculus of Constructions of Coquand and Huet [CH88]. We will present the cube in the style of the Pure Type Systems [Ber88, Geu93, Bar91].

The systems of the cube are based on a set T of *pseudoterms* defined by the following abstract syntax:

$$T ::= \mathbf{x} \mid \star \mid \Box \mid (TT) \mid \lambda \mathbf{x}{:}T.T \mid \Pi \mathbf{x}{:}T.T$$

where $\mathbf{x}$ 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*.

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 φ. 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.

$\vec{M}$ 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 $A \rightarrow B$ if $x \notin FV(B)$.

Just as in the untyped λ-calculus, terms that only differ from each other in their bound variables will be identified: we work *modulo α-conversion*.

On pseudoterms the relation of β-reduction ($\twoheadrightarrow_\beta$) is defined as the transitive, reflexive and compatible closure of the reduction rule

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

The relation of β-conversion ($=_\beta$) is the least equivalence relation generated by $\twoheadrightarrow_\beta$.

Lemma 2.1 (Church-Rosser for pseudoterms [Bar86]) *The relation $\twoheadrightarrow_\beta$ on pseudoterms satisfies the Church-Rosser property, i.e., for M and N pseudoterms:*

$$M =_\beta N \Rightarrow \exists P[M \twoheadrightarrow_\beta P, N \twoheadrightarrow_\beta P]$$

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

The following notations will be used:

If $\Gamma = \langle x_1 : A_1, \ldots, x_n : A_n\rangle$ then $\Gamma, x : B$ will denote the pseudocontext $\langle x_1 : A_1, \ldots, x_n : A_n, x : B\rangle$.

$\Gamma' \subseteq \Gamma$ (Γ' *is subcontext* of Γ) iff $x : A$ in $\Gamma' \Rightarrow x : A$ in Γ.

$\Gamma \twoheadrightarrow_\beta \Gamma'$ ($\Gamma =_\beta \Gamma'$) 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 $1 \leq i \leq n$.

A *statement* in a system $\lambda\_$ of the cube is an expression of the form

$$\Gamma \vdash_{\lambda\_} M : A$$

where Γ is a pseudocontext and M and A pseudoterms.

We show now, for each system of the 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 $\Gamma \vdash_{\lambda\_} M : A$ is legal", from now on we shall just write $\Gamma \vdash_{\lambda\_} M : A$.

The legal statements of a system $\lambda\_$ 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

$(\mathcal{P})$ $\quad \vdash \star : \Box$

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

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

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

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

(conv) $$\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : p}{\Gamma \vdash M : B} \qquad \text{if } p \in \{\star, \Box\} \text{ and } A =_\beta B$$

Specific rules

(Π) $$\frac{\Gamma \vdash A : p_1 \quad \Gamma, x{:}A \vdash B : p_2}{\Gamma \vdash \Pi x{:}A.B : p_2}$$

The specific rules, which characterize the different systems of the cube, are all introduction rules, obtained by instantiating with $\star$ and $\Box$ the parametric rule (Π).

In the following we will denote by $\mathcal{P}$ the set $\{\star, \Box\}$[1]. Given particular $p_1, p_2 \in \mathcal{P}$, the corresponding (Π)-rule will be called (p_1, p_2).

It is not difficult to check that 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.

We can now define the systems of the cube.

Definition 2.2 (The λ-cube) *The* cube of typed λ-calculi *(λ-cube) is the set of the type systems $\boldsymbol{\lambda}_{\to}$, $\boldsymbol{\lambda}_2$, $\boldsymbol{\lambda}_P$, $\boldsymbol{\lambda}_{\overline{\omega}}$, $\boldsymbol{\lambda}_{P2}$, $\boldsymbol{\lambda}_{\omega}$, $\boldsymbol{\lambda}_{P\overline{\omega}}$ and $\boldsymbol{\lambda}_{P\omega}$ defined by the General Axioms and Rules above and, respectively, by the following specific rules:*

$$\begin{array}{lcllll}
\boldsymbol{\lambda}_{\to} & = & \{\ (\star\ ,\star)\ \} & & & \\
\boldsymbol{\lambda}_{2} & = & \{\ (\star\ ,\star), & (\Box,\star)\ \} & & \\
\boldsymbol{\lambda}_{P} & = & \{\ (\star\ ,\star), & & (\star\ ,\Box)\} & \\
\boldsymbol{\lambda}_{\overline{\omega}} & = & \{\ (\star\ ,\star), & & & (\Box,\Box)\ \} \\
\boldsymbol{\lambda}_{P2} & = & \{\ (\star\ ,\star), & (\Box,\star), & (\star\ ,\Box)\} & \\
\boldsymbol{\lambda}_{\omega} & = & \{\ (\star\ ,\star), & (\Box,\star), & (\Box,\Box)\ \} & \\
\boldsymbol{\lambda}_{P\overline{\omega}} & = & \{\ (\star\ ,\star), & & (\star\ ,\Box), & (\Box,\Box)\ \} \\
\boldsymbol{\lambda}_{P\omega} & = & \{\ (\star\ ,\star), & (\Box,\star), & (\star\ ,\Box), & (\Box,\Box)\ \}
\end{array}$$

We can now draw a picture of the λ-cube as follows.

$$\begin{tikzcd}
& \boldsymbol{\lambda}_{\omega} \arrow[rr] & & \boldsymbol{\lambda}_{P\omega} \\
\boldsymbol{\lambda}_{2} \arrow[rr] \arrow[ur] & & \boldsymbol{\lambda}_{P2} \arrow[ur] & \\
& \boldsymbol{\lambda}_{\overline{\omega}} \arrow[rr] \arrow[uu] & & \boldsymbol{\lambda}_{P\overline{\omega}} \arrow[uu] \\
\boldsymbol{\lambda}_{\to} \arrow[rr] \arrow[uu] \arrow[ur] & & \boldsymbol{\lambda}_{P} \arrow[uu] \arrow[ur] &
\end{tikzcd}$$

System $\boldsymbol{\lambda}_{\to}$ is a variant of Church's simply typed λ-calculus [Church40], while $\boldsymbol{\lambda}_P$ and $\boldsymbol{\lambda}_{P2}$ are variants of the AUTOMATH system AUT-QE, defined by de Bruijn [deB80]. $\boldsymbol{\lambda}_P$ is also known under the name LF, studied by Harper, Honsell and Plotkin [HHP87]. $\boldsymbol{\lambda}_2$ and $\boldsymbol{\lambda}_\omega$ are the systems F and F_ω, defined by Girard [Gir72], and $\boldsymbol{\lambda}_{P\omega}$ is the Calculus of Constructions of Coquand and Huet [CH88]. From now on we will refer to $\boldsymbol{\lambda}_{P\omega}$ also as $\boldsymbol{\lambda}_C$.

Besides their computational aspects, an important feature of the eight type systems of the λ-cube is the relation with (intuitionistic) logics (see[Bar91, Geu93] for discussions about such

[1] In the literature $\star$ and $\Box$ are usually called *sorts*. This name is not used here since later we will add algebraic rewriting to the λ-cube and then the name "sort" will have a different meaning.

a relationship.)

We introduce now some relevant sets, to be used later in the analysis of the λ-cube and its algebraic extension.

Definition 2.3 *Given a system $\lambda\_$, we define*

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

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.

Lemma 2.4 (Subject Reduction [GN91]) *Let $\Gamma \in$ $\mathsf{Context}$($\lambda\_$), $B, B', C \in$ Term($\lambda\_$) and $\Gamma \vdash_{\lambda\_} B : C$.*

$$B \twoheadrightarrow_\beta B' \ \Rightarrow \ \Gamma \vdash_{\lambda\_} B' : C.$$

Notice that the reduction relation $\twoheadrightarrow_\beta$ has not been defined on terms, but on pseudoterms. This is motivated by the use of the reduction relation in the (conv) rule, which in turn is used to define what a (legal) term is. One could now argue that the side condition $A =_\beta B$ in the conversion rule, can be proved by means of a chain of reductions-expansions where also pseudoterms that are not legal terms are present. This fact however causes no trouble at all since, by Lemmas 2.1 and 2.4, we can always get $A =_\beta B$ by means of a chain containing only legal terms.

We will see that we will not be able to profit of such a nice feature of the calculus when algebraic rewriting will be added to the cube. Much more effort will then be necessary to provide sound definitions and proofs.

Lemma 2.5 (Stripping for the λ-cube [GN91]) *Let Γ be a context, M, N and R terms.*

$$\begin{array}{lll}
(i) & \Gamma \vdash p : R, \ \textit{with } p \in \mathcal{P} & \Rightarrow \ p \equiv \star, \ R \equiv \Box \\
(ii) & \Gamma \vdash x : R, \ \textit{with } x \in \mathsf{Var} & \Rightarrow \ R =_\beta A, \ \textit{ and } x : A \in \Gamma \textit{ for some term } A \\
(iii) & \Gamma \vdash \Pi x{:}M.N : R & \Rightarrow \ \Gamma \vdash M : p_1, \Gamma, x{:}M \vdash N : p_2 \textit{ and } R =_\beta p_2, \\
 & & \quad \textit{for some } p_1, p_2 \in \mathcal{P} \\
(iv) & \Gamma \vdash \lambda x{:}M.N : R & \Rightarrow \ \Gamma, x{:}M \vdash N : B, \Gamma \vdash \Pi x{:}M.B : p \\
 & & \quad \textit{and } R =_\beta \Pi x{:}M.B, \\
 & & \quad \textit{for some term } B \textit{ and } p \in \mathcal{P} \\
(v) & \Gamma \vdash MN : R & \Rightarrow \ \Gamma \vdash M : \Pi x{:}A.B, \Gamma \vdash N : A \\
 & & \quad \textit{and } R =_\beta B[N/x], \\
 & & \quad \textit{for some terms } A \textit{ and } B.
\end{array}$$

3 Adding Algebraic Rewriting to the λ-cube

The aim of the present section is to extend the λ-cube in order to have in it also algebraic features and rewriting. To do so, the first step is to consider a denumerable set $\mathcal{S}$ of *sorts*:

$$\mathcal{S} = \{s_1, s_2, \ldots\}$$

The elements of $\mathcal{S}$ denote *algebraic base types*. To look at these algebraic types as types of the systems of the λ-cube we add, for each $s_i \in \mathcal{S}$, the following axiom to the general rules of the cube:

$$(alg1) \quad \vdash s_i : \star.$$

Out of the algebraic base types we define, by induction, the set of algebraic types.

Definition 3.1 (Algebraic types) *Let $\mathcal{S}$ be a denumerable set of sorts. The set $\mathsf{T}_\mathcal{S}$ of* algebraic types *on $\mathcal{S}$ is inductively defined as follows:*

- *If $s \in \mathcal{S}$ then $s \in \mathsf{T}_\mathcal{S}$*
- *If $\sigma, \tau \in \mathsf{T}_\mathcal{S}$ then $\Pi x{:}\sigma.\tau \in \mathsf{T}_\mathcal{S}$*

It is straightforward to check that, for any $\sigma \in \mathsf{T}_\mathcal{S}$, we can infer $\vdash_{\lambda\_} \sigma : \star$, and that all the algebraic types have the form $\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \sigma$ ($n \geq 0$, $\sigma \in \mathcal{S}$).

We will call *first-order algebraic types* the elements $\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \sigma \in \mathsf{T}_\mathcal{S}$ such that $\sigma, \sigma_i \in \mathcal{S}$ $(1 \leq i \leq n)$. A context $\Gamma = \langle x_1{:}A_1, \ldots, x_n{:}A_n \rangle$ is called *algebraic* if $A_i \in \mathsf{T}_\mathcal{S}$ $(1 \leq i \leq n)$.

Lemma 3.2 *Let $\tau_1, \tau_2 \in \mathsf{T}_\mathcal{S}$. $\tau_1 =_\beta \tau_2 \;\; \Rightarrow \;\; \tau_1 \equiv \tau_2$.*

Proof By definition of $\mathsf{T}_\mathcal{S}$ and the Church-Rosser property of the λ-cube. □

The next step is now to consider functions on algebraic types, i.e. a set $\mathcal{F}$ of function symbols (*signature*), where

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

and $\mathcal{F}_\tau$ denotes the set of function symbols whose functionality (type) is τ. We assume $\mathcal{F}_\tau \cap \mathcal{F}_{\tau'} = \emptyset$ if $\tau \not\equiv \tau'$. Each function symbol f in $\mathcal{F}$ is assumed to have an arity which, when necessary, will be denoted by superscripts (f^n)[2].

The introduction of the signature is naturally expressed in the framework of the cube by the following axioms:

$$(alg2) \quad \vdash f : \sigma$$

for each $f \in \mathcal{F}_\sigma$.

A function symbol f^m is called *first-order* if it has a first-order algebraic type $s_1 \rightarrow \ldots \rightarrow s_n \rightarrow s$ and $n \equiv m$. Function symbols which are not first-order will be called *higher-order*.

We denote by Σ the set of all first-order function symbols in $\mathcal{F}$ and by $f, g, \ldots$ its generic elements. Capital letters $F, G, \ldots$ denote instead generic higher-order function symbols. When

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

it is clear from the context, we use f to denote a generic (first- or higher-order) element of $\mathcal{F}$.

Sorts and function symbols can now be used to build pseudoterms, i.e. pseudoterms are now defined by

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

where $\mathbf{f}$ and $\mathbf{s}$ range over $\mathcal{F}$ and $\mathcal{S}$, respectively.

A function symbol f^n is said to be *saturated* in a pseudoterm M if any occurrence of its appears in subterms of the form $fP_1 \dots P_m$ with $m \equiv n$.

Next thing to do now is to define in our setting the notion of algebraic term, i.e. the natural translation of the notion of term of an algebraic term rewriting system.

Definition 3.3 (Algebraic Terms) *(i) An* algebraic pseudoterm *is a pseudoterm formed only by variables and function symbols of the signature.*

(ii) An algebraic term in Γ *(in system $\lambda\_$), for $\Gamma \in \mathsf{Context}(\lambda\_)$[3], is an algebraic pseudoterm t such that there exists a term A such that*

1. $\Gamma \vdash_{\lambda\_} t : A$
2. *any $f \in \mathcal{F}$ is saturated in t*
3. $x : B \in \Gamma$ *with* $x \in FV(t) \Rightarrow \exists\sigma \in \mathsf{T}_{\mathcal{S}}.B =_\beta \sigma$.

(iii) A first-order algebraic term t *(in Γ) is an algebraic term (in Γ) such that*

1. *any $f \in \mathcal{F}$ occurring in t is first-order*
2. *there is no subterm of t of the form xP.*

Notice that, by Lemma 3.2, the σ of (ii).3 of the above definition is unique.

The following lemma provides evidence of the fact that algebraic terms (in a given context Γ) are well defined.

Lemma 3.4 *Let t be an algebraic term in Γ such that $\Gamma \vdash_{\lambda\_} t : A$.
Then there exists a unique $\tau \in \mathsf{T}_{\mathcal{S}}$ such that $\tau =_\beta A$.*

Proof We check existence first; by induction on the structure of t.
It is straightforward to check that the cases listed below are the only ones that can occur if t is algebraic.

- $t \equiv x$.
  Immediate by definition and the Stripping lemma.
- $t \equiv f$ with $f \in \mathcal{F}$.
  Immediate by the Stripping lemma (f can be seen as a variable in such case).
- $t \equiv MN$.
  By Stripping we get that $\Gamma \vdash M : \Pi x{:}C.B$, $\Gamma \vdash N : C$ and $A =_\beta B[N/x]$. We can now apply the induction hypothesis to get $\tau' =_\beta \Pi x{:}C.B$ with $\tau' \in \mathsf{T}_{\mathcal{S}}$. Therefore, by Church-Rosser and definition of $\mathsf{T}_{\mathcal{S}}$, it follows that $\tau' \equiv \tau_1 \to \tau_2$, $C =_\beta \tau_1$ and $B =_\beta \tau_2$, with $\tau_1, \tau_2 \in \mathsf{T}_{\mathcal{S}}$. It is easy to see that $\tau_2 =_\beta A$ since $\tau_2 =_\beta B$, $A =_\beta B[N/x]$ and $\tau_2[N/x] \equiv \tau_2$.

[3] Notice that we are still considering derivability in the λ-cube, since its algebraic extension is being defined.

The uniqueness of τ follows from Lemma 3.2. $\Box$

Notice that it is not possible to speak of algebraic terms independently of contexts, since it would have no meaning at all to consider the following as an algebraic term

$$X : \star \to s_1, \alpha : \star \vdash_{\lambda\_} f(X\alpha)a : s$$

where $f : s_1 \to s_2 \to s$ and $a : s_2$. However, if we restrict to first-order algebraic terms, we can avoid to take into account contexts, as the following lemma shows.

Lemma 3.5 *In any system $\lambda\_$, let t be a non-variable first-order algebraic term in Γ. Then, for any Γ' such that $\Gamma' \vdash t : A$, t is algebraic in Γ'.*

Proof By definition of first-order algebraic term: in t all the variables appear as arguments of a function symbol, and function symbols have types in $\mathsf{T}_{\mathcal{S}}$. $\Box$

We define now the notion of rewrite rule and from that the notion of rewrite relation. A rewrite rule will be a pair of algebraic terms (since algebraic terms depend on contexts, so do rewrite rules). It is however easy to see that we cannot, given a rewrite rule, define a rewrite relation on the *terms* of the algebraic extension of the λ-cube: the rewrite relation will be used to define legal terms (in the type conversion rule), so a circularity would arise[4]. As in the case of the reduction relation $\twoheadrightarrow_\beta$ for the λ-cube, rewriting will be defined on pseudoterms and not on terms. In this way we have no problem of circularity since the pseudoterms of the algebraic extension have already been defined[5]. The rewriting relation on terms will then be the restriction to terms of the rewriting relation on pseudoterms.

However, some extra care is needed, as the following example shows: Consider the rewrite rule

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

where $\Gamma = \langle X{:}s_1 \to s_2, x{:}s_1\rangle$, $f \in \mathcal{F}_{s_2 \to s_3 \to s}$, $a \in \mathcal{F}_{s_3}$ and $h \in \mathcal{F}_{s_3 \to s}$. r_Γ induces the following reduction relation on pseudoterms:

$$P(f(Xx)a) \to_r P(ha). \tag{2}$$

If we now restrict to terms the obtained relation it could happen that the following terms can be obtained in the extended system: $\Gamma' \vdash_R M{:}P(f(Xx)a)$ and $\Gamma' \vdash_R P(ha) : \star$, where $\vdash_R$ denote derivability in the algebraic extension of λ-cube and $X{:}\star \to s_2, x{:}\star \in \Gamma'$.

Now, since by the reduction (2) we have $P(f(Xx)a) =_r P(ha)$, we should be allowed to infer also $\Gamma' \vdash_R M : P(ha)$. This however would have no sense, since in the context Γ' the term $f(Xx)a$ has no meaning as algebraic term because the variable X is not algebraic in Γ'.

To overcome this problem we will modify the naive definition of rewrite rule presented before. In the definition we are giving below, the term t of a rule $r : t \to t'$ has to be "rewritable", a condition that, as we shall see, implies that, for any context Γ, t is algebraic in Γ whenever it is typable in it. Rewrite rules become then, in a sense, independent from contexts. This is not a real restriction, since all first-order terms and many useful higher-order terms can be easily shown to be rewritable.

[4] This difficulty is caused by the presence of dependent types: algebraic reductions can occur inside types.

[5] Another solution could be to stratify the definition of the system, defining it by levels starting from the λ-cube and using in the (conv) rule for the level i, the rewrite relation induced on the terms of the system defined at level $i-1$. The final system would then be the limit of such a chain of systems. Our choice is however motivated, with respect to this one, by its simplicity.

Definition 3.6 *A term t is* rewritable *if*

1. *it is algebraic in some context Γ*
2. *for any $x \in FV(t)$ there exists a subterm $fP_1 \dots P_k$ of t such that $f \in \mathcal{F}$ and $P_j \equiv x$ for some $1 \leq j \leq k$.*

It is straightforward to check that a variable cannot be a rewritable algebraic term.

Lemma 3.7 *Any non-variable first-order algebraic term is rewritable.*

Proof By definition of first-order algebraic term. $\Box$

Lemma 3.8 *Let t be a rewritable algebraic term. Then $\exists \Gamma', \sigma$, unique and algebraic, such that $\forall \Gamma, A$:*

$$\Gamma \vdash t : A \ \Rightarrow\ (\Gamma_{|FV(t)} =_\beta \Gamma') \ \&\ (A =_\beta \sigma)$$

and such that $\Gamma' \vdash_{\boldsymbol{\lambda}_\rightarrow} t : \sigma$.

Proof To prove the first part, let $x \in FV(t)$. By definition of rewritable term there exists a subterm $fP_1 \dots P_k$ of t with $P_j \equiv x$ for some $1 \leq j \leq k$. Then, by Stripping, $x{:}B \in \Gamma$, and $B =_\beta \tau \in \mathsf{T}_\mathcal{S}$ (τ is unique since the type of f in the signature is unique). We take Γ' such that $x{:}\tau \in \Gamma'$. Now, by Lemma 3.4, there exists a unique σ such that $\Gamma \vdash t{:}\sigma$ and $\sigma =_\beta A$. Now, from $\Gamma \vdash t{:}\sigma$ we get $\Gamma' \vdash_{\boldsymbol{\lambda}_\rightarrow} t{:}\sigma$. $\Box$

The lemma above shows that rewritable algebraic terms are "independent" from contexts and systems (recall that $\boldsymbol{\lambda}_\rightarrow$ is a subsystem of all the systems of the λ-cube), so justifying point 1. of Definition 3.6.

We can now use the notion of rewritable term to define rewrite rules.

We introduce first the notion of λ-rewrite rule and then that of (proper) rewrite rule. The former notion, more general of the usual one of algebraic reduction, is given since our results will hold also for a particular class of λ-rewrite rules.

Definition 3.9 *A* λ-rewrite rule *is a pair of terms $\langle t, t' \rangle$, where t is algebraic (for some context), while t' can also contain abstractions, and such that*

1. *t is rewritable*[6]
2. *$FV(t') \subseteq FV(t)$*
3. *For any context Γ and pseudoterm A:*

$$\Gamma \vdash t : A \ \Rightarrow\ \Gamma \vdash t' : A.$$

A (proper) rewrite rule will now be a λ-rewrite rule involving only algebraic terms.

Definition 3.10 (Rewrite rules)

(i) *A* rewrite rule *r is a λ-rewrite rule $\langle t, t' \rangle$ such that also t' is algebraic.*

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

[6] As stated before, this condition subsumes the usual condition "t not a variable" for rewrite rules.

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

A (λ-)rewrite rule will be denoted in the following by

$$r : t \to t'.$$

Given a set R of rewrite rules, we denote by FOR and HOR the subsets of first-order and higher-order rules of R, respectively. In the following, when λ-rewrite rules are present, we assume them to be in the set HOR.

A rewrite rule induces a rewriting relation on pseudoterms as follows.

Definition 3.11 *Let M and N be pseudoterms.*
$M \to_r N$ iff there exists a rewrite rule $r : t \to t'$, a context $C[\,]$ and a substitution φ such that $M \equiv C[t\varphi]$ and $N \equiv C[t'\varphi]$.

If R is a set of rewrite rules we define

$$M \to_R N \;\Leftrightarrow\; \exists r \in R\,.M \to_r N$$

and

$$M \to_{R\beta} N \;\Leftrightarrow\; M \to_R N \vee M \to_\beta N.$$

$\twoheadrightarrow_r$, $\twoheadrightarrow_R$ and $\twoheadrightarrow_{R\beta}$ denote the reflexive and transitive closure of $\to_r$, $\to_R$ and $\to_{R\beta}$, respectively.

Many definitions of *higher-order rewriting* appear in the literature, none of them has been universally accepted, and ours is one among the many possible ones. Recently, a general definition of higher-order rewriting has been proposed by van Oostrom and van Raamsdonk [OR93, OR94], which subsumes systems like Klop's CRSs [KOR93] and Nipkow's HRSs [Nip91]. Even if our higher-order rewriting rules can be looked at from vanOostrom and vanRaamsdonk's point of view, it is not clear whether it is so also for the whole systems of the λR-cube being defined in the next subsection.

3.1 The λR-cube

Once one specifies a set $\mathcal{S}$ of sorts, a signature $\mathcal{F}$ and a set R of rewrite rules, it would seem that to complete the definition of the algebraic extension of the λ-cube, it suffices, besides the additional axioms for sorts and function symbols given before, to replace the rule (*conv*) by the following one:

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : p \quad B =_{R\beta} B'}{\Gamma \vdash A : B'}$$

where $=_{R\beta}$ is the least congruence containing $\to_{R\beta}$.

This however would not work. As pointed out before, when we have $A =_\beta B$ in the pure cube, the Church-Rosser property of $=_\beta$, together with the Subject Reduction property, ensure that A and B are always equal via β-reductions and β-expansions that remain inside the set of well-typed terms. Here instead we cannot rely, in general, on the Church-Rosser property for $=_{R\beta}$ since we wish to prove the modularity of strong normalization for every set of strongly normalizing reduction rules (even for those which are non confluent[7]).

Therefore we cannot consider $=_{R\beta}$ in the (*conv*) rule, but instead we have to consider only the $R\beta$-reduction relation.

[7]Moreover, even if we restricted ourselves to strongly normalizing and confluent rewrite rules, we could not rely on the confluence property, since there exists no proof, at the time being, of the modularity of confluence for the λR-cube.

The absence, in general, of the Church-Rosser property for the algebraic extension of the λ-cube makes also difficult to prove some properties easily provable in the pure cube, like Subject Reduction for β-reduction alone.

We can now present the complete definition of the algebraic extension of the λ-cube.

Definition 3.12 (The λR-cube) *Let $\mathcal{S} = \{s_1, s_2 \ldots\}$ be a set of sorts, $\mathcal{F} = \{f_1, f_2, \ldots\}$ a signature on $\mathcal{S}$ and R a set of rewrite rules*[8].
*The $\lambda\langle\mathcal{S}, \mathcal{F}, R\rangle$-*cube *($\lambda R$-cube for short) is defined by adding the following axioms to the axioms of the λ-cube*

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

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

and by replacing the following rule for the (conv) rule

$$(red_{R\beta}) \quad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : p}{\Gamma \vdash M : B} \quad A \rightarrow_{R\beta} B \textit{ or } B \rightarrow_{R\beta} A$$

The eight systems of the λR-cube will be called $\boldsymbol{\lambda}_{R\rightarrow}$, $\boldsymbol{\lambda}_{R2}$, $\boldsymbol{\lambda}_{RP}$, $\boldsymbol{\lambda}_{R\overline{\omega}}$, $\boldsymbol{\lambda}_{RP2}$, $\boldsymbol{\lambda}_{R\omega}$, $\boldsymbol{\lambda}_{RP\overline{\omega}}$ and $\boldsymbol{\lambda}_{RP\omega}$. λ_{R-} will denote a generic system of the λR-cube. $\boldsymbol{\lambda}_{RP\omega}$ will also be called $\boldsymbol{\lambda}_{RC}$.

The λR-cube can be drawn as follows

$$\begin{tikzcd}
 & \boldsymbol{\lambda}_{R\omega} \arrow[rr] & & \boldsymbol{\lambda}_{RC} \\
\boldsymbol{\lambda}_{R2} \arrow[ur] \arrow[rr] & & \boldsymbol{\lambda}_{RP2} \arrow[ur] & \\
 & \boldsymbol{\lambda}_{R\overline{\omega}} \arrow[uu] \arrow[rr] & & \boldsymbol{\lambda}_{RP\overline{\omega}} \arrow[uu] \\
\boldsymbol{\lambda}_{R\rightarrow} \arrow[uu] \arrow[ur] \arrow[rr] & & \boldsymbol{\lambda}_{RP} \arrow[uu] \arrow[ur] &
\end{tikzcd}$$

Obviously, all the notions not affected by the introduction of the algebraic features, like that of kind, object and so on, remain unchanged w.r.t. their definitions in the λ-cube.

Note that when we write $M \rightarrow_r N$, for M, N terms of the λR-cube, the relation $\rightarrow_r$ is the one defined in 3.11 now restricted to terms.

Some systems of the λR-cube are already present in the literature. In particular, when R is a first-order system, $\boldsymbol{\lambda}_{R\rightarrow}$ is the system studied in [BT88, Oka89], while $\boldsymbol{\lambda}_{R2}$ is equivalent to the system defined by Breazu-Tannen and Gallier [BTG90]. The systems of [JO91] correspond to $\boldsymbol{\lambda}_{R\rightarrow}$ and $\boldsymbol{\lambda}_{R2}$. We have already mentioned in the introduction which results were proved for these systems.

The style of presentation we have used for the algebraic component of the λR-cube is somewhat different from the usual one for many-sorted Term Rewriting Systems (TRS's, see [DJ88] for an overview) where, given a set $\mathcal{S}$ of sorts, a signature is formed by function symbols having an *arity*, which is a string $s_1 \ldots s_n$ ($n \geq 0$), and a *codomain* s. In a TRS terms in s are expressions of the form $f(v_1, \ldots, v_n)$ where f has arity $s_1 \ldots s_n$, codomain s, and v_i ($1 \leq i \leq n$)

[8] Recall that, by Lemma 3.8, the notion of rewrite rule is independent from contexts and systems.

is a term in s_i. As pointed out in [BTG90], "it is easy to see that currying establishes the expected relation between many-sorted algebraic rewriting in a TRS and our definition of algebraic rewriting." So extending the λ-cube with the "first-order computation features" of a given TRS corresponds to define the $\lambda\langle\mathcal{S},\mathcal{F},R\rangle$-cube where $\mathcal{S}$ is the set of sorts of the TRS and any $f{:}s_1 \to \ldots \to s_n \to s$ corresponds to an f in the TRS with arity $s_1 \ldots s_n$ and codomain s. A rewrite rule $r : q \to q'$ in a TRS becomes $c(r) : curry(q) \to curry(q')$ in R. It has to be remarked, however, that currying has the side-effect of producing terms in the λR-cube which have no meaning in the context of the TRS, like, for instance, $f curry(v_1) curry(v_2)$ if $n > 2$. This fact, far from being a drawback, provides flexibility and expressive power to the λR-cube. Moreover, it is easy to check that a terminating TRS induces a strongly normalizing rewrite relation on the algebraic terms of the corresponding λR-cube.

Notation 3.13 *We use $A \overset{c}{=}_\beta B$ to denote that terms A and B are equal via β-reductions and β-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 $E_1, \ldots, E_n$ for which*

$$A \twoheadrightarrow_{R\beta} E_1 \twoheadleftarrow_{R\beta} E_2 \twoheadrightarrow_{R\beta} E_3 \cdots E_n \twoheadrightarrow_{R\beta} B.$$

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.)

The replacement of rule $(red_{R\beta})$ for the rule $(conv)$ of λ-cube is only needed for systems with dependent types: the following lemma (which will be proved at the end of Section 4), shows that such a replacement is useless for systems without $(\star,\Box)$-rules, i.e. without dependent types.

Lemma 3.14 *For $A, M \in \mathsf{Term}(\lambda_{R-})$, and λ_{R-} any system of the λR-cube without $(\star,\Box)$-rules: $A \in \mathsf{Constr}(\lambda_{R-})$ & M subexpression of $A \Rightarrow M \in \mathsf{Kind}(\lambda_{R-}) \vee M \in \mathsf{Constr}(\lambda_{R-})$*

Lemma 3.15 *Let λ_{R-} be a system of the λR-cube without $(\star,\Box)$-rules. Then in λ_{R-} the rule $(conv)$ is equivalent to the rule $(red_{R\beta})$.*

Proof Easy by Lemma 3.14 and the Church-Rosser property for the λ-cube. $\Box$

Now that we have defined the λR-cube, we can define what is an algebraic term (in Γ) in the λR-cube (the notion of algebraic term used to define it was in the λ-cube).

Definition 3.16 *An* algebraic term *(in Γ) in the λR-cube is an algebraic pseudoterm such that $\Gamma \vdash_{R\lambda\_} t : A$ and*

$$\forall x \in FV(t).[x : B \in \Gamma \Rightarrow B \overset{c}{=}_{R\beta} \tau \in \mathsf{T}_{\mathcal{S}}].$$

Moreover any occurrence of a function symbol has to be saturated in t.

Being in the λR-cube does not modify essentially the properties of algebraic terms. It is not difficult to check that Lemma 3.8 can be proved in the context of the λR-cube, using $\overset{c}{=}_{R\beta}$ instead of $=_\beta$, essentially in the same way, once some metatheory for the λR-cube has been developed (see Sect.4).

Lemma 3.17 *Let t be a rewritable algebraic term. Then*

$$\Gamma \vdash t : A \Rightarrow (\Gamma_{|FV(t)} \overset{c}{=}_{R\beta} \Gamma') \;\&\; (A \overset{c}{=}_{R\beta} \sigma)$$

where Γ' and σ are algebraic, unique and such that $\Gamma' \vdash_{\boldsymbol{\lambda}_{R\to}} t : \sigma$.

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

$$\{r : f(X\,x)xX \rightarrow f(X\,x)(X\,x)X\}.$$

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

$$f((\lambda y{:}s.y)x)x(\lambda y{:}s.y) \rightarrow_r f((\lambda y{:}s.y)x)((\lambda y{:}s.y)x)(\lambda y{:}s.y) \rightarrow_\beta$$

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

Then one has necessarily to restrict the notion of higher-order rule in order to get strongly normalizing systems. Following [JO91] 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, the notation only makes appear explicitly some of the subterms of t.

Definition 3.18 (The general schema [JO91]) (i) *A λ-higher-order rewrite rule $r : t \rightarrow t'$* satisfies the general schema w.r.t. *R if it is of the form*

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

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

1. $\vec{X} \subseteq \vec{Y}$[9]
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 a term in $\boldsymbol{\lambda}_{R\rightarrow}$*[10]
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 $HOR = \{l_i \rightarrow r_i\}_{i \in I}$ of higher-order rewrite rules* satisfies the general schema *(w.r.t. FOR) if each rule $l_i \rightarrow r_i \in HOR$ satisfies the general schema w.r.t. $FOR \cup \{l_j \rightarrow r_j\}_{j<i}$. This implies that there is no mutual recursion in HOR.*

[9] Note that this condition ensures that $F\,\vec{l}[\vec{X},\vec{x}]\,\vec{Y}$ is rewritable.

[10] In fact we can consider v in λ_{R2} or $\lambda_{R\omega}$ as well.

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 version of F, while condition 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 3.19 *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 [JO91]:*

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

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

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

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 3.20 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*

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

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:*

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

Now, using foldr, *and assuming that $+$, $\times$, 0 and 1 are already defined, we can define the functions:*

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

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*

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

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 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 in case we consider reductions on algebraic terms only. It is in fact possible to code Toyama's example of non-termination [Toy87], as it can be seen below.

Example 3.21 *Let us 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:

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

$$\to_{r_1} f(FX)(FX)(FX) \to_{r_3} \dots$$

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 [Toy87]).

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 λR-cube.

Definition 3.22 (*i*) *A rewrite rule $r : t \to t'$ is* conservative *if for any variable x 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 conservative if each of them is so.*

Example 3.23 *The first-order system defining the function append in Example 3.19 is conservative.*

We are not the first to impose conservativity on first-order rules in order to get strong normalization. In [Rus87], for instance, it was shown that strong normalization is a modular property of disjoint unions of first-order term rewriting systems that are conservative. In practice, however, conservativity is not a relevant restriction, since most implementations of rewrite systems use sharing, and shared-reductions are always conservative.

We can now state our main result.

Theorem 3.24 (Main Theorem) *Let R be a set of rewrite rules such that*

1. *FOR is conservative and strongly normalizing on first-order algebraic terms*[11]

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

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

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

[11] By Lemma 3.5 we can avoid referring to a context.

4 Metatheory of the λR-cube

In this section we present the syntactical properties of $\boldsymbol{\lambda}_{RC}$ 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 λ-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 4.1 (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 that this also implies that, if $\Gamma, x{:}A \vdash M, N : C$ with $M \overset{c}{=}_{R\beta} N$ and $\Gamma \vdash Q : A$, then $M[Q/x] \overset{c}{=}_{R\beta} N[Q/x]$.

Lemma 4.2 (Stripping for the λR-cube) *For Γ a context, M, N and R terms, we have the following.*

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

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 4.3 (Correctness of Types) *For Γ a context, M and A terms,*

$$\Gamma \vdash M : A \Rightarrow \exists p \in \mathcal{P}[A \equiv p \vee \Gamma \vdash A : p].$$

Proof The proof can be given by analysing the derivation tree of $\Gamma \vdash M : A$, like in the proof of 4.2, 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 4.2), $\Gamma, x{:}A \vdash B : p'$ for some $p' \in \mathcal{P}$. Now by Substitution (Proposition 4.1), we conclude that $\Gamma \vdash B[N/x] : p'$.$\Box$

Lemma 4.4 *(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 one $N' \equiv (\lambda x{:}A.M)Q$, 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.$\Box$

Lemma 4.5

$$\Gamma \vdash \Pi x{:}A.B : C \Rightarrow C \in \mathcal{P}.$$

Proof $C \stackrel{c}{=}_{R\beta} \star$ or $C \stackrel{c}{=}_{R\beta} \Box$ by Stripping *(v)*. But then, by Lemma 4.4 *(ii)*, $C \equiv \star$ or $C \equiv \Box$.

Lemma 4.6

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

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 $N \stackrel{c}{=}_{R\beta} M$. Recall that $N \stackrel{c}{=}_{R\beta} M$ means that M and N are equal via expansions and reductions that remain inside the set of well-typed terms.

- $M \equiv \Pi\vec{x}{:}\vec{A}.\star$ and $M \twoheadrightarrow_\beta N$. Then $N \equiv \Pi\vec{x}{:}\vec{A'}.\star$ with $\vec{A} \twoheadrightarrow_\beta \vec{A'}$, so $\Gamma \vdash N : \Box$. (Use Lemma 4.4.*(iii)* ($\Rightarrow$).) If $M \twoheadrightarrow_R N$, then we are done similarly.
- $M \equiv \Pi\vec{x}{:}\vec{A}.\star$ and $N \twoheadrightarrow_\beta M$. 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 definition of rewrite rule. $\Box$

Corollary 4.7 (βR-preservation of $\star$ and $\Box$) *Let* $p, p' \in \{\star, \Box\}$.

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

Proof If $p \equiv p' \equiv \star$ we are done, otherwise, in case $p \equiv \Box$ or $p' \equiv \Box$, one has $\Gamma \vdash M : \Box$, $\Gamma' \vdash N : \Box$ by Lemma 4.6, i.e. $p \equiv p'$. $\Box$

The following lemma is similar to Lemma 3.2. Its proof however is much harder since now we cannot rely on the Church-Rosser property.

Lemma 4.8 *Let* $\sigma_1, \sigma_2 \in \mathsf{T}_{\mathcal{S}}$.

$$\sigma_1 \overset{c}{=}_{R\beta} \sigma_2 \;\Rightarrow\; \sigma_1 \equiv \sigma_2.$$

Proof In Sect. 6 a translation $\tau : \{\Box\} \cup \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \cup \mathsf{Constr}(\boldsymbol{\lambda}_{RC}) \to \mathsf{Term}(\boldsymbol{\lambda}_{R\omega})$ will be defined, such that $\tau(\sigma) \equiv \sigma$ for $\sigma \in \mathsf{T}_{\mathcal{S}}$. Moreover, for such a translation Lemma 6.6 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 σ_1 and σ_2.
It is not difficult to check that referring to Lemma 6.6 as we have done above creates no circularity. We refer to it only in order not to duplicate its proof here. $\Box$

Lemma 4.9 (Classification)

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

Proof For the first 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 4.7. For the second 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 βR-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 βR-preservation of $\star$ and $\Box$ (Corollary 4.7), $p \equiv p_0$.

$s \in \mathcal{S}$ 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 βR-preservation of $\star$ and $\Box$, $p \equiv p'$.

$f \in \mathcal{F}$ 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 $\Box$, $p \equiv p'$.

Π 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 $p_0, p'_0 \in \mathcal{P}$. 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 $p \equiv p'$.

λ 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 (vi), 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 4.4(ii)) Similarly $B' \overset{c}{=} \Pi x{:}A.C'$ with $\Gamma' \vdash A : p'_1$, $\Gamma', x{:}A \vdash M : C' : p'$ and $\Gamma' \vdash \Pi x{:}A.C' : p'$, for a rule (p'_1, p'). Now, by induction $p \equiv p'$.

app Let $\Gamma \vdash MN : D : p$ and $\Gamma' \vdash MN : D' : p'$. Then, by Stripping, $\Gamma \vdash M : \Pi x{:}A.B : p_2$, $\Gamma \vdash N : A : p_1$, $\Gamma \vdash B[N/x] : p_2$ and $B[N/x] \overset{c}{=} D$, for a rule (p_1, p_2). 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 $B'[N/x] \overset{c}{=} D'$ for a rule (p_1', p_2'). Now, by induction $p_2 \equiv p_2'$. Also, by βR-preservation of $\star$ and $\Box$, $p \equiv p_2$ and $p' \equiv p_2'$ and so $p \equiv p'$. $\Box$

Lemma 4.10 (Uniqueness of formation) *Let Γ and Γ' be contexts and B a term.*

$$B \textit{ formed by } (p_1, p_2) \textit{ in } \Gamma \ \& \ B \textit{ formed by } (p_1', p_2') \textit{ in } \Gamma' \ \Rightarrow \ p_1 \equiv p_1', p_2 \equiv p_2'.$$

Proof Direct consequence of Corollary 4.7. $\Box$

The lemma above allows us to use the terminology "formed by" without mentioning the context Γ.

It still remains to prove Lemma 3.14, for which we need the following technical lemma.

Lemma 4.11 *Let K be the set inductively defined by*

1. $\star \in K$

2. $k_1, k_2 \in K \ \Rightarrow \ k_1 \rightarrow k_2 \in K$

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

Proof By induction on the structure of k using Lemma 4.4.(iii). $\Box$

Lemma 3.14 *For $A, M \in \mathsf{Term}(\lambda_{R-})$ and λ_- any system of λR-cube without $(\star, \Box)$-rules:* $A \in \mathsf{Constr}(\lambda_{R-})$ & M *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 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 4.11, so are all their subexpressions. $\Box$

4.1 The Subject Reduction Property

That of subject reduction is a property that, as stated before, requires more effort to be proved in $\boldsymbol{\lambda}_{RC}$ than in $\boldsymbol{\lambda}_C$, because we cannot rely on the Church-Rosser property for pseudoterms.

In the following we will prove separatedly subject reduction for $\twoheadrightarrow_R$ and for $\twoheadrightarrow_\beta$.

Lemma 4.12 *Let $r : t \rightarrow t'$ 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 $\vec{B}$ be the types of $\vec{N}$ in the first statements $\vec{\Gamma_1} \vdash \vec{N} : \vec{B}$ 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: $\vec{\Gamma_1}, \vec{x}{:}\vec{B}, \vec{\Gamma_2} \vdash t : A$. By definition of reduction rule we can infer $\vec{\Gamma_1}, \vec{x}{:}\vec{B}, \vec{\Gamma_2} \vdash t' : A$. Hence, since we have also that $\vec{\Gamma_1} \vdash \vec{N} : \vec{B}$, by Substitution (Proposition 4.1), it follows $\Gamma \vdash t'[\vec{N}/\vec{x}] : A$. $\Box$

Proposition 4.13 (Subject Reduction Lemma for rewriting, SR$_R$) *For Γ a context, P, P' and D terms and $r \in R$,*

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

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 4.12. □

It turns out that Subject Reduction for β 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

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

but we can not 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 4.14 *The λ-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* ω-*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* ω-abstraction *if M is a constructor, A a kind.*

We can decorate the λs correspondingly, so we can speak of the λ_0s of a term, etc. We now also define the notions of β^0-reduction, β^2-reduction, β^P-reduction and β^ω-reduction by just restricting reduction to the redexes with the appropriate subscript attached to the symbol λ. We use an arrow with a superscript above it to denote these restricted reductions, so $\overset{P\omega}{\to}_\beta$ etcetera.

We also have the usual notion of *weak-head-reduction*, a term M weak-head-reducing to M', notation $M \to_{wh} M'$, if M is itself a redex, say $M \equiv (\lambda x{:}A.P)Q$ and M' is obtained by contracting $(\lambda x{:}A.P)Q$ (the *head-redex*). Similarly we have $\twoheadrightarrow_{wh}$ as 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 4.15 (SR$_{\beta P\omega}$) *For Γ and Γ' contexts, P, P' and D terms,*

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

Proof The proof is by simultaneous induction on the derivation. Just as in the usual proof for β 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

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

where the λ is a λ_P or a λ_ω. Then by applying Stripping (4.2) to the first premise, we find

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

So, again by Stripping

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

By the fact that we have a P- or ω-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 4.4(*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, $\Pi\vec{x}{:}\vec{A}.\star \overset{c}{=} \Pi\vec{x}{:}\vec{B}.\star$, 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 4.6 and 4.4. Hence

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

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 4.1) to (5) and (1) we get

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

By applying Correctness of Types (Lemma 4.3) to the first premise, we find $\Gamma \vdash \Pi x{:}B.C : p'$ for some $p' \in \mathcal{P}$ and hence by Stripping

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

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. □

Corollary 4.16 (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 ω-redex in M, then $(\lambda x{:}A'.N')Q'$ is a P- or ω-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 4.17 (Confluence of $\beta^{P\omega}$) *For $M, P, N \in$ 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

$$\begin{tikzcd} M \arrow[r, two heads, "P\omega", "\beta"'] \arrow[d, two heads, "P\omega"', "\beta"] & P \arrow[d, dotted, two heads, "P\omega"', "\beta"] \\ N \arrow[r, dotted, two heads, "P\omega", "\beta"'] & Q \end{tikzcd}$$

Proof By completeness of developments and the fact that $\beta^{P\omega}$-redexes are stable under $\overset{P\omega}{\twoheadrightarrow}_\beta$. (Corollary 4.16)

Lemma 4.18 *If $\Gamma \vdash (\lambda x{:}C.P)Q : p \in \mathcal{P}$, then this is a P- or ω-redex.*

Proof First we know by Lemma 4.4 that p must be $\star$. By Stripping we find A, B and D such that

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

Hence $D[Q/x] \equiv \star$ and so $D \equiv \star$, because Q can't be $\star$. (This follows from Stripping and Lemma 4.4.) Hence $\Pi x{:}A.D : \Box$ and so $\Pi x{:}C.B : \Box$ by βR-preservation of elements of $\mathcal{P}$. Hence $B \equiv \Pi \vec{y}{:}\ldots.\star$ and so $B : \Box$ and P is a constructor. We conclude that the λ is decorated with P or ω and we are done. $\Box$

As a consequence of this we find that Subject Reduction holds for weak-head-reduction on types. That is

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

Lemma 4.19 (Commutativity of weak-head-reduction and β-reduction) *For $M, P, N \in$ Term,*

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

In a diagram

$$\begin{tikzcd} M \arrow[r, two heads, "wh"'] \arrow[d, two heads, "\beta"] & P \arrow[d, dotted, two heads, "\beta"] \\ N \arrow[r, dotted, two heads, "wh"'] & Q \end{tikzcd}$$

Proof We prove it for the case of a one step weak-head-reduction, so say $M \rightarrow_{wh} P$, 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 4.20 (Commutativity of weak-head-reduction and R-reduction) *For* $M, P, N \in \mathsf{Term}$,

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

In a diagram

$$\begin{tikzcd}
M \arrow[r, two heads, "wh"'] \arrow[d, two heads, "R"] & P \arrow[d, two heads, dotted, "R"] \\
N \arrow[r, two heads, dotted, "wh"'] & Q
\end{tikzcd}$$

Proof The proof is exactly the same as the previous one. □

We also have postponement of R-reduction with respect to weak-head-reduction on types and kinds. That is, if M is a type or kind, then

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

Or in a diagram

$$\begin{tikzcd}
M \arrow[r, two heads, dotted, "wh"'] \arrow[d, two heads, "R"] & P \arrow[d, two heads, dotted, "R"] \\
N \arrow[r, two heads, "wh"'] & Q
\end{tikzcd}$$

Lemma 4.21 *If* $\Pi x{:}A.C \overset{c}{=} \Pi x{:}B.D$ *and all the terms on the reduction-expansion-path from* $\Pi x{:}A.C$ *to* $\Pi x{:}B.D$ *are types or kinds, then* $\Pi x{:}A.C \overset{c}{=} \Pi x{:}B.D$ *via a path that only uses* Π-*terms.*

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

$$\Pi x{:}A.C \downarrow_{\rho_1} E_1 \downarrow_{\rho_2} E_2 \downarrow_{\rho_3} \ldots E_{n-1} \downarrow_{\rho_n} \Pi x{:}B.D$$

where $\downarrow_{\rho_i}$ means the sharing of a common well-typed reduct via ρ_i-reduction. (So ρ_i ranges over $\{\beta, R\}$.) We are going to prove the Lemma by showing that there are Π-terms $F_1, \ldots, F_{n-1}$ such that

$$\Pi x{:}A.C \downarrow_{\rho_1} F_1 \downarrow_{\rho_2} F_2 \downarrow_{\rho_3} \ldots F_{n-1} \downarrow_{\rho_n} \Pi x{:}B.D$$

This is done by taking $F_i \equiv \mathrm{whnf}(E_i)$, the weak-head-normal-form of E_i. That this works is shown by going through the sequence

$$\Pi x{:}A.C, E_1, E_2, \ldots E_{n-1}, \Pi x{:}B.D$$

from right to left (or from left to right if one prefers).

First note that if $E_{n-1} \downarrow_R \Pi x{:}B.D$, then E_{n-1} R-reduces to a Π-term and hence (because E_{n-1} is of type an element of $\mathcal{P}$), it must be a Π-term itself. So in that case E_{n-1} is already in whnf and we are done.

Now, if ρ_n is β then $E_{n-1} \twoheadrightarrow_\beta \Pi x{:}B'.D'$ for some B' and D'. Hence

$$E_{n-1} \twoheadrightarrow_{wh} \Pi x{:}B_{n-1}.D_{n-1} \twoheadrightarrow_\beta \Pi x{:}B'.D'$$

where $\Pi x{:}B_{n-1}.D_{n-1}$ is the whnf of E_{n-1}. Taking whnf of a type is done by $\beta^{P\omega}$-reduction, so $\Pi x{:}B_{n-1}.D_{n-1}$ is still well-typed. We can now apply the commutativity of weak-head-reduction with the reductions β and R, and the postponement of β and R with respect to weak-head-reduction to obtain the following diagram.

$$\begin{tikzcd}
 & E_{n-2} \arrow[dl, "\rho_{n-2}"'] \arrow[d, dotted, two heads, "wh"] \arrow[dr, "\rho_{n-1}"] & & E_{n-1} \arrow[dl, "\rho_{n-1}"'] \arrow[d, dotted, two heads, "wh"] \arrow[dr, "\beta"] & & \Pi x{:}B.D \arrow[dl, "\beta"'] \\
- \arrow[d, dotted, two heads, "wh"] & \mathrm{whnf}(E_{n-2}) \arrow[dl, "\rho_{n-2}"] \arrow[dr, "\rho_{n-1}"'] & - \arrow[d, dotted, two heads, "wh"] & \Pi x{:}B_{n-1}.D_{n-1} \arrow[dl, "\rho_{n-1}"] \arrow[r, dotted, two heads, "\beta"'] & \Pi x{:}B'.D' & \\
- & & - & & &
\end{tikzcd}$$

Proceeding in this way until $\Pi x{:}A.C$, we find well-typed Π-terms $F_1, \ldots, F_{n-1}$ such that

$$\Pi x{:}A.C \downarrow_{\rho_1} F_1 \downarrow_{\rho_2} F_2 \downarrow_{\rho_3} \ldots F_{n-1} \downarrow_{\rho_n} \Pi x{:}B.D \; \Box$$

Corollary 4.22 *If* $\Pi x{:}A.C \overset{c}{=} \Pi x{:}B.D$, *then* $A \overset{c}{=} B$ *and* $C \overset{c}{=} D$.

Proposition 4.23 (Subject Reduction for β) *For* Γ *and* Γ' *contexts,* P, P' *and* D *terms,*

$$\begin{aligned} \Gamma \vdash P : D \;\&\; P \twoheadrightarrow_\beta P' &\;\Rightarrow\; \Gamma \vdash P' : D \\ \Gamma \vdash P : D \;\&\; \Gamma \twoheadrightarrow_\beta \Gamma' &\;\Rightarrow\; \Gamma' \vdash P : D. \end{aligned}$$

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 $\Pi x{:}A.C \overset{c}{=} \Pi x{:}B.D$ implies $A \overset{c}{=} B$ and $C \overset{c}{=} D$, which was proved in the previous Corollary. $\Box$

Lemma 4.24 (Subject Reduction) *For* Γ *a context,* P, P' *and* D *terms,*

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

Proof Immediate from Propositions 4.13 and 4.23.

5 The proof of the Main Theorem

In this section we present the skeleton of the proof of the Main Theorem (3.24). 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 χ is strongly normalizing.

The proof consists in three main steps

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

where system $\lambda_{\wedge R}$ is a type assignment system consisting in the extension of the intersection system of Coppo and Dezani [CD80] (see also [BCD83]) with the algebraic rewriting defined by a set R of rewrite rules.

The definition of $\lambda_{\wedge R}$ is recalled in Appendix A, while for the proof of $\lambda_{\wedge R} \models \mathsf{SN}$ we refer to [BF93a]. The proof in [BF93a] 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 [JO91].

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 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.

This method has been used by Harper, Honsell and Plotkin [HHP87] to obtain SN of their system LF (roughly corresponding to $\boldsymbol{\lambda}_P$) using SN of simply typed lambda calculus (corresponding to $\boldsymbol{\lambda}_{\rightarrow}$).

The translation for proving $\lambda_{\wedge R} \models \mathsf{SN} \Rightarrow \boldsymbol{\lambda}_{R\omega} \models \mathsf{SN}$ is nothing but a *type-erasing* function. Its definition and the proof of reduction-preservation will be the argument of Sect. 7.

The translation and the reduction preservation proof for $\boldsymbol{\lambda}_{R\omega} \models \mathsf{SN} \Rightarrow \boldsymbol{\lambda}_{RC} \models \mathsf{SN}$ will be instead the argument of Sect. 6.

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

As announced in Sect.5 we will define a translation from terms of $\boldsymbol{\lambda}_{RC}$ to terms of $\boldsymbol{\lambda}_{R\omega}$ 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 in [GN91] to prove the strong normalization for the pure λ_C . Geuvers and Nederhof's translation can be seen as a generalization to higher order of the map defined by Harper, Honsell and Plotkin, to prove the strong normalization property of the LF system [HHP87].

As in [HHP87] and [GN91], it is not possible to define a reduction-preserving map $[-]$ such that

$$\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} M{:}A \Rightarrow [\Gamma] \vdash_{\boldsymbol{\lambda}_{R\omega}} [M]{:}[A]$$

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

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

The definitions of $[-]$ and $\tau(-)$ do not speak for themselves. We refer to [GN91] for some intuitions about their definition as generalizations of the maps defined in [HHP87].

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

Definition 6.1 *The map* $\rho : \{\Box\} \cup \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \to \mathsf{Kind}(\boldsymbol{\lambda}_{R\omega})$ *is inductively defined by:*

1. $\rho(\star) = \rho(\Box) = \star$
2. $\rho(\Pi\alpha{:}\, M.N) = \rho(M) \to \rho(N)$ *if* $\Pi\alpha{:}\, M.N$ *is formed by* $(\Box, \Box)$
3. $\rho(\Pi x{:}\, M.N) = \rho(N)$ *if* $\Pi x{:}\, M.N$ *is formed by* $(\star, \Box)$.

The case distinction in the definition is correct by Lemma 4.10, and by Lemma 4.4(iii) there are no more cases.

The following properties of ρ will be used.

Property 6.2 (i) *If* $k_1, k_2 \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$ *and* $k_1 \overset{c}{=}_{R\beta} k_2$ *then* $\rho(k_1) \equiv \rho(k_2)$.

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

Proof (i) By Lemma 4.4$(iii)(\Leftarrow)$ we have that $k_1 \equiv \Pi\vec{x}{:}\vec{A}.\star \overset{c}{=}_{R\beta} \Pi\vec{x}{:}\vec{B}.\star \equiv k_2$. From this fact it immediately follows from Corollary 4.22 that, if $|\vec{A}| = n$ and $|\vec{B}| = m$, $n = m$ and

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

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 ρ to k_1 and k_2 it follows that, by Corollary 4.7 and (5) above, only one of the two following cases can occur.

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

In both the above cases the thesis follows from the induction hypothesis.

(ii) By Lemma 4.4.(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, $C \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \Rightarrow C[A/u] \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$. $\Box$

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 6.3 *The map* $\tau : \{\Box\} \cup \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \cup \mathsf{Constr}(\boldsymbol{\lambda}_{RC}) \to \mathsf{Term}(\boldsymbol{\lambda}_{R\omega})$ *is inductively defined by:*

1. $\tau(\star) = \tau(\Box) = 0 : \star$
2. $\tau(\alpha) = \alpha$ *if* α *is a variable.*
   $\tau(s) = s$ *if* $s \in \mathcal{S}$.
3. $\tau(\Pi\alpha{:}\, M.N) = \Pi\alpha{:}\, \rho(M).\tau(M) \to \tau(N) : \star$ *if* $\Pi\alpha{:}\, M.N$ *is formed by* $(\Box, \Box)$ *or* $(\Box, \star)$.
   $\tau(\Pi x{:}\, M.N) = \Pi x{:}\, \tau(M).\tau(N)$ *if* $\Pi x{:}\, M.N$ *is formed by* $(\star, \Box)$ *or* $(\star, \star)$.
4. $\tau(\lambda\alpha{:}\, M.N) = \lambda\alpha{:}\, \rho(M).\tau(N)$ *if* $\lambda\alpha{:}\, M.N$ *is formed by* $(\Box, \Box)$.
   $\tau(\lambda x{:}\, M.N) = \tau(N)$ *if* $\lambda x{:}\, M.N$ *is formed by* $(\star, \Box)$.

5. *$\tau(MN) = \tau(M)\tau(N)$ if MN is formed by $(\Box, \Box)$.*

   $\tau(MN) = \tau(M)$ if MN is formed by $(\star, \Box)$.

The definition by cases is correct by Lemma 4.10. Lemma 6.4 below guarantees that the range of τ is really the set of $\mathsf{Term}(\boldsymbol{\lambda}_{R\omega})$.

In order to map $\mathsf{Context}(\boldsymbol{\lambda}_{RC})$ into $\mathsf{Context}(\boldsymbol{\lambda}_{R\omega})$ 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 extend now the map τ in such a way that it acts also on $\mathsf{Context}(\boldsymbol{\lambda}_{RC})$ yielding elements of $\mathsf{Context}(\boldsymbol{\lambda}_{R\omega})$:

1. Let $A \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \cup \mathsf{Type}(\boldsymbol{\lambda}_{RC})$.

   $\tau(x : A) = x : \tau(A)$ if $x \in Var^{\star}$.

   $\tau(\alpha : A) = \alpha : \rho(A), x^{\alpha} : \tau(A)$ if $\alpha \in Var^{\Box}$.

2. Let $\Gamma = \langle u_1 : A_1, u_2 : A_2, \ldots, u_n : A_n\rangle \in \mathsf{Context}(\boldsymbol{\lambda}_{RC})$.

   $\tau(\Gamma) = \langle 0 : \star, d : \perp, \tau(u_1 : A_1), \tau(u_2 : A_2), \ldots, \tau(u_n : A_n)\rangle$.

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

Now, if $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} B : \star$ we shall put $c^B \equiv dB$ and if $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} B : \Box$, 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 $B \in \mathsf{Kind}(\boldsymbol{\lambda}_{R\omega}) \cup \mathsf{Type}(\boldsymbol{\lambda}_{R\omega})$, $N \in \mathsf{Term}(\boldsymbol{\lambda}_{R\omega})$ and variables u.

The following lemma states that ρ and τ are well-defined.

Lemma 6.4 *Let $\Gamma \in \mathsf{Context}(\boldsymbol{\lambda}_{RC})$, $M, N \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$.*

$$\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} M : N : \Box \quad \textit{or} \quad \Gamma \vdash_{\boldsymbol{\lambda}_{RC}} M : N \equiv \Box \quad \Rightarrow \quad \tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(M) : \rho(N)$$

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

1. If $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} M : N$ 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 τ and ρ.

2. If the last rule is (var) then the conclusion is $\Gamma', a : N \vdash_{\boldsymbol{\lambda}_{RC}} a : N$ and so, by induction and definition of τ and ρ, $\tau(\Gamma', a : N) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(a) : \rho(N)$.

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 6.2.

5. If $M \equiv \Pi u{:}\, B_1.B_2$, $N \equiv p \in \mathcal{P}$ and the last applied rule is (Π) then by induction $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1) : \star$ and $\tau(\Gamma, u : B_1) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_2) : \star$.

   If $B_1 \in \mathsf{Type}(\boldsymbol{\lambda}_{RC})$ then $\tau(\Gamma, u : B_1) = \tau(\Gamma), u : \tau(B_1)$ and so $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1) \to \tau(B_2) : \star$ by rule $(\star, \star)$.

   If $B_1 \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$ then $\tau(\Gamma, u : B_1) = \tau(\Gamma), u : \rho(B_1), x^u : \tau(B_1)$ and so, by rules $(\star, \star)$ and $(\Box, \star)$, we have $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \Pi u{:}\, \rho(B_1).\tau(B_1) \to \tau(B_2) : \star$.

6. If $M \equiv \lambda u{:}\, B_1.B_2$, $N \equiv \Pi u{:}\, B_1.C_2 \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$ and the last rule is (λ) then by induction $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1) : \star$ and $\tau(\Gamma, u : B_1) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_2) : \rho(C_2)$.

   If $B_1 \in \mathsf{Type}(\boldsymbol{\lambda}_{RC})$ then $\tau(\lambda u{:}\, B_1.B_2) \equiv \tau(B_2)$, $\rho(\Pi u{:}\, B_1.C_2) \equiv \rho(C_2)$ and $\tau(u : B_1) = u : \tau(B_1)$. By definition of τ and ρ and by Property 6.2, 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 $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_2) : \rho(C_2)$.

   If $B_1 \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$ 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 τ and ρ and by Property 6.2, 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 $\tau(\Gamma), u : \rho(B_1) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_2) : \rho(C_2)$ and by one application of (λ) (rule $(\Box, \Box)$), $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \lambda u{:}\, \rho(B_1).\tau(B_2) : \rho(B_1) \to \rho(C_2)$.

7. If $M \equiv B_1 B_2$, $N \equiv C_2[B_2/x] \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$ and the last rule is the Application rule, let $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} B_1 : \Pi x{:}\, C_1.C_2$ and then by the Stripping Lemma and induction we obtain:

   If $B_2 \in \mathsf{Object}(\boldsymbol{\lambda}_{RC})$ then
   $\tau(B_1 B_2) \equiv \tau(B_1)$, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1) : \rho(\Pi x{:}\, C_1.C_2)$
   and $\rho(\Pi x{:}\, C_1.C_2) \equiv \rho(C_2) \equiv \rho(C_2[B_2/x])$.

   If $B_2 \in \mathsf{Constr}(\boldsymbol{\lambda}_{RC})$ then $\tau(B_1 B_2) \equiv \tau(B_1)\tau(B_2)$, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1) : \rho(\Pi x{:}\, C_1.C_2)$ and $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_2) : \rho(C_1)$. Besides, $\rho(\Pi x{:}\, C_1.C_2) \equiv \rho(C1) \to \rho(C_2)$, so $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1 B_2) : \rho(C_2) \equiv \rho(C_2[B_2/x])$. $\Box$

Lemma 6.5 $A \in \mathsf{T}_{\mathcal{S}} \Rightarrow \tau(A) \equiv A$.

Proof Immediate by definition of τ. $\Box$

Lemma 6.6 *Let* $B, B' \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \cup \mathsf{Constr}(\boldsymbol{\lambda}_{RC})$.

(i) *If* $x \in Var^{\star}$ *and* $A \in \mathsf{Object}(\boldsymbol{\lambda}_{RC})$ *then* $\tau(B[A/x]) \equiv \tau(B) \equiv \tau(B)[A/x]$.

(ii) *If* $\alpha \in Var^{\Box}$ *and* $A \in \mathsf{Constr}(\boldsymbol{\lambda}_{RC})$ *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 6.2 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 $M \in \mathsf{Type}(\boldsymbol{\lambda}_{RC})$ then $\tau((\lambda u{:}\, M.N)P) \equiv \tau(N) \equiv \tau(N[P/u])$ by definition of τ and by (i). If $M \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$ 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 τ and by (ii).

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

Definition 6.7 *The map* $[-] : \mathsf{Kind}(\boldsymbol{\lambda}_{RC}) \cup \mathsf{Constr}(\boldsymbol{\lambda}_{RC}) \cup \mathsf{Object}(\boldsymbol{\lambda}_{RC}) \rightarrow \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ *is inductively defined by*

1. $[\star] = c^0$
2. $[x] = x$ *if* $x \in Var^\star$
3. $[\alpha] = x^\alpha$ *if* $\alpha \in Var^\square$
4. $[s] = c^0$ *if* $s \in \mathcal{S}$
5. $[f] = f$ *if* $f \in \mathcal{F}$
6. $[\Pi x{:}\, M.N] = c^{0\rightarrow 0\rightarrow 0}[M][N][c^{\tau(M)}/x]$ *if* $\Pi x{:}\, M.N$ *is formed by* $(\star,\star)$ *or* $(\star,\square)$
   $[\Pi \alpha{:}\, M.N] = c^{0\rightarrow 0\rightarrow 0}[M][N][c^{\tau(M)}/x^\alpha][c^{\rho(M)}/\alpha]$ *if* $\Pi \alpha{:}\, M.N$ *is formed by* $(\square,\star)$ *or* $(\square,\square)$
7. $[\lambda x{:}\, M.N] = (\lambda z{:}\, 0.\lambda x{:}\, \tau(M).[N])[M]$ *where* z *is a fresh variable, if* $\lambda x{:}\, M.N$ *is formed by* $(\star,\star)$ *or* $(\star,\square)$
   $[\lambda \alpha{:}\, M.N] = (\lambda z{:}\, 0.\lambda \alpha{:}\, \rho(M).\lambda x^\alpha{:}\, \tau(M).[N])[M]$ *where* z *is a fresh variable, if* $\lambda \alpha{:}\, M.N$ *is formed by* $(\square,\star)$ *or* $(\square,\square)$
8. $[MN] = [M][N]$ *if* MN *is formed by* $(\star,\star)$ *or* $(\star,\square)$
   $[MN] = [M]\tau(N)[N]$ *if* MN *is formed by* $(\square,\star)$ *or* $(\square,\square)$.

This definition by cases is correct by the Unicity of formation lemma. The following theorems state that $[-]$ satisfies the required conditions: Theorem 6.8 shows that the range of $[-]$ is really $\mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$, and Theorem 6.12 shows that the mapping preserves all possible reduction sequences.

Theorem 6.8 *Let* $\Gamma \in \mathsf{Context}(\boldsymbol{\lambda}_{RC})$, $M, N \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$.
If $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} M : N$ *then* $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [M] : \tau(N)$.

Proof By induction on the structure of M. By Lemma 6.4 we know that $\tau(\Gamma)$ is a legal context in $\boldsymbol{\lambda}_{R\omega}$.

1. $M \equiv \star$, then $N \equiv \square$ by Stripping (Lemma 4.2(i) and Lemma 4.4(ii)). It follows that $[\star] \equiv c^0 : 0 \equiv \tau(\square)$ in $\tau(\Gamma)$.
2. $M \equiv s \in S$, then $N \equiv \star$ by Stripping (ii) and Lemma 4.4(ii). It follows that $[s] \equiv c^0 : 0 \equiv \tau(\star)$ in $\tau(\Gamma)$.
3. $M \equiv u \in Var^\star \cup Var^\square$, then $u : A \in \Gamma$, with $A \overset{c}{=}_{R\beta} N$. If $u \equiv \alpha \in Var^\square$ 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 $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [u] : \tau(A)$, and by Lemma 6.6, using some applications of (red_β), $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [u] : \tau(N)$.

4. $M \equiv f \in \mathcal{F}$, then, by Stripping (iv), $N \stackrel{c}{=}_{R\beta} \sigma$ where σ is the type of f in the signature $\mathcal{F}$. We have that $[f] \equiv f : \sigma$ in $\tau(\Gamma)$. By Lemma 6.5, $\tau(\sigma) \equiv \sigma$ and then, by Lemma 6.6 and some applications of (red_β), we find $[f] : \tau(N)$ in $\tau(\Gamma)$.

5. $M \equiv \Pi u{:}\, B_1.B_2$, then $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} B_1 : p_1$ and $\Gamma, u : B_1 \vdash_{\boldsymbol{\lambda}_{RC}} B_2 : p_2$ for suitable $p_1, p_2 \in \mathcal{P}$. By induction $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1] : 0$ and $\tau(\Gamma, u : B_1) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_2] : 0$. By Lemma 6.4, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}}$, $\tau(B_1) : \star$ and so $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} c^{\tau(B_1)} : \tau(B_1)$.

   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 and the Substitution Lemma, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_2][c^{\tau(B_1)}/x] : 0$. By using the Application rule twice we get $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} c^{0 \to 0 \to 0}[B_1][B_2][c^{\tau(B_1)}/x] : 0$.

   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 $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_2][c^{\tau(B_1)}/x^\alpha, c^{\rho(B_1)}/\alpha] : 0$. By using the Application rule twice we get $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} c^{0 \to 0 \to 0}[B_1][B_2][c^{\tau(B_1)}/x^\alpha, c^{\rho(B_1)}/\alpha] : 0$.

   In both cases $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [\Pi u{:}\, B_1.B_2] : 0$.

6. $M \equiv \lambda u{:}\, B_1.B_2$, then $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} B_1 : p_1$ and $\Gamma, u : B_1 \vdash_{\boldsymbol{\lambda}_{RC}} B_2 : C_2 : p_2$ for suitable $C_2 \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$, $p_1, p_2 \in \mathcal{P}$, with $N \stackrel{c}{=}_{R\beta} \Pi u{:}\, B_1.C_2$.

   By induction $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1] : 0$ and $\tau(\Gamma, u : B_1) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_2] : \tau(C_2)$. By Lemma 6.4, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(B_1) : \star$ and $\tau(\Gamma, u : B_1) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(C_2) : \star$.

   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 (λ) and one of the Application rule we get $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} (\lambda z{:}\, 0.\lambda x{:}\, \tau(B_1)[B_2])[B_1] : \Pi x{:}\, \tau(B_1).\tau(C_2)$.

   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 (λ) and one of the Application rule we get

   $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} (\lambda z{:}\, 0.\lambda \alpha{:}\, \rho(B_1).\lambda x^\alpha{:}\, \tau(B_1).[B_2])[B_1] : \Pi \alpha{:}\, \rho(B_1).\tau(B_1) \to \tau(C_2)$.

   In both cases $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [\lambda u{:}\, B_1.B_2] : \tau(\Pi u{:}\, B_1.C_2)$ and so, by Lemma 6.6(iv), some applications of rule (red_β) enable us to get $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [\lambda u{:}\, B_1.B_2] : \tau(N)$.

7. $M \equiv B_1 B_2$ then $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} B_1 : \Pi u{:}\, C_1.C_2$ and $\Gamma \vdash_{\boldsymbol{\lambda}_{RC}} B_2 : C_1$ for some $C_1, C_2 \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$ with $N \stackrel{c}{=}_{R\beta} C_2[B_2/u]$.

   By induction $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1] : \tau(\Pi u{:}\, C_1.C_2)$ and $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_2] : \tau(C_1)$. By Lemma 6.4, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} \tau(C_1) : \star$.

   If $C_1 \in \mathsf{Type}(\boldsymbol{\lambda}_{RC})$, $u \equiv x \in Var^\star$, then $\tau(\Pi x{:}\, C_1.C_2) \equiv \Pi x{:}\, \tau(C_1).\tau(C_2)$, so $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1][B_2] : \tau(C_2)[[B_2]/x] \equiv \tau(C_2[[B_2]/x]) \equiv \tau(C_2) \equiv \tau(C_2[B_2/x])$ (by Lemma 6.6).

   If $C_1 \in \mathsf{Kind}(\boldsymbol{\lambda}_{RC})$, $u \equiv \alpha \in Var^\Box$, then
   $\tau(\Pi \alpha{:}\, C_1.C_2) \equiv \Pi \alpha{:}\, \rho(C_1).\tau(C_1) \to \tau(C_2)$,
   so $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1]\tau(B_2)[B_2] : \tau(C_2)[[B_2]/x^\alpha, \tau(B_2)/\alpha] \equiv \tau(C_2[B_2/\alpha])$ (by Lemma 6.6).

   In both cases $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1 B_2] : \tau(C_2[B_2/u])$ and then, by Lemma 6.6.(iv) and some applications of rule red_β, $\tau(\Gamma) \vdash_{\boldsymbol{\lambda}_{R\omega}} [B_1 B_2] : \tau(N)$. $\Box$

In what follows we will prove that if $M \to_{R\beta} N$ in $\boldsymbol{\lambda}_{RC}$ then $[M] \to_{R\beta} [N]$ in $\boldsymbol{\lambda}_{R\omega}$, i.e., $[-]$ preserves all reduction sequences. For the proof we need some lemmas.

Lemma 6.9 *Let $M, N \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$ such that $M[N/x], M[N/\alpha] \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$.*
If $x \in Var^\star$ and $N \in \mathsf{Object}(\boldsymbol{\lambda}_{RC})$ then $[M[N/x]] \equiv [M][[N]/x]$.
If $\alpha \in Var^\Box$, $N \in \mathsf{Constr}(\boldsymbol{\lambda}_{RC})$ then $[M[N/\alpha]] \equiv [M][\tau(N)/\alpha, [N]/x^\alpha]$.

Proof By induction on the structure of M.

If $M \in (Var^\star \cup Var^\Box \cup \mathcal{F} \cup \mathcal{S} \cup \{\star\})$ 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 6.6
2. $\rho(Q[N/u]) \equiv \rho(Q)[[N]/u]$, by Property 6.2
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 6.6 and because x^α 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 6.2 and because x^α 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. $\Box$

Now, we extend the definition of $[-]$ to substitutions, in the usual way: Let φ be a substitution in $\boldsymbol{\lambda}_{RC}$, $[\varphi]$ 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 6.8.

Now we can prove:

Lemma 6.10 *(i) Let $M \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$ and let φ be a substitution in $\boldsymbol{\lambda}_{RC}$ such that $Dom(\varphi) \subset Var^\star$ and $M\varphi \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$. Then $[M\varphi] \equiv [M][\varphi]$.*

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

Proof (i) By Lemma 6.9.
(ii) It is not difficult to check that the proof of the first part of Lemma 6.9 can be modified in order to consider replacement instead of substitution. $\Box$

Lemma 6.11 *Let t be an algebraic term. Then $[t] \equiv t$.*

Proof By definition of algebraic term and map $[-]$. $\Box$

Now we can show that all the reduction sequences are preserved.

Theorem 6.12 *Let $M, M' \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$. If $M \rightarrow_{R\beta} M'$ then $[M] \rightarrow^+_{R\beta} [M']$.*

Proof By induction on the structure of M.

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

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

- If $M \equiv (\lambda u{:}\, B_1.B_2)C$ and $M' \equiv B_2[C/u]$, we distinguish again two cases:

– If $(\lambda u\colon B_1.B_2)C$ is formed by $(\star,\star)$ or $(\star,\Box)$ then
$[M] \equiv [(\lambda u\colon B_1.B_2)][C] \equiv (\lambda z\colon 0.\lambda u : \tau(B_1).[B_2])[B_1][C] \to_\beta [B_2]\ [[C]/u]$,
because z is a fresh variable. Moreover, $[B_2][[C]/u] \equiv [B_2[C/u]]$, by Lemma 6.9.

– If $(\lambda u\colon B_1.B_2)C$ is formed by $(\Box,\star)$ or $(\Box,\Box)$ then
$[M] \equiv [(\lambda u\colon B_1.B_2)]\tau(C)[C] \equiv (\lambda z\colon 0.\lambda u\colon \rho(B_1).\lambda x^u\colon \tau(B_1).[B_2])[B_1]\tau(C)[C] \to_\beta$
$\to_\beta [B_2][\tau(C)/u,[C]/x^u] \equiv [B_2[C/u]]$ by Lemma 6.9.

- 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 $Var^\Box$. By Lemma 6.10, $[M] \equiv [t][\varphi]$ and $[M'] \equiv [t'][\varphi]$. Moreover, by Lemma 6.11, $[t] \equiv t$ and $[t'] \equiv t'$. Hence $M \equiv [t][\varphi] \to^r [t'][\varphi] \equiv [M']$. $\Box$

Using the previous theorem we can now easily prove the main result of this section.

Theorem 6.13 *$\to_{R\beta}$-strong normalization of $\boldsymbol{\lambda}_{R\omega}$ implies $\to_{R\beta}$-strong normalization of $\boldsymbol{\lambda}_{RC}$.*

Proof By contradiction. Assume $\to_{R\beta}$ is strongly normalizing in $\boldsymbol{\lambda}_{R\omega}$ and there is an infinite sequence of $\to_{R\beta}$-reductions in $\boldsymbol{\lambda}_{RC}$ starting from a term $M_1 \in \mathsf{Term}(\boldsymbol{\lambda}_{RC})$:

$$M_1 \to_{R\beta} M_2 \to_{R\beta} M_3 \to_{R\beta} \dots$$

By Theorem 6.12,

$$[M_1] \to_{R\beta} [M_2] \to_{R\beta} [M_3] \to_{R\beta} \dots$$

which contradicts the hypothesis. Then all sequences of $\to_{R\beta}$-reductions are finite in $\boldsymbol{\lambda}_{RC}$. $\Box$

As an application of this result, we obtain the strong normalization of the generalization to λ_C of the language described in [JO91].

7 $\lambda_{\wedge R} \models \mathsf{SN} \Rightarrow \lambda_{\omega R} \models \mathsf{SN}$

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

As done for the proof of

$$\boldsymbol{\lambda}_{R\omega} \models SN \Rightarrow \boldsymbol{\lambda}_{RC} \models SN$$

we shall use a translation from terms of $\boldsymbol{\lambda}_{R\omega}$ 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 in [Gir72] for $\boldsymbol{\lambda}_\omega$ (i.e., without algebraic features), but it holds for $\boldsymbol{\lambda}_{R\omega}$ because the function symbols in $\mathcal{F}$ can be looked at as free variables in β-reductions.

Lemma 7.1 ([Gir72]) *$\to_\beta$ is strongly normalizing on terms of $\boldsymbol{\lambda}_{R\omega}$.*

Let us define, by induction on the structure of terms, a "type erasing" function from $\mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ to Λ_F (see Definition A.2).

Definition 7.2 (The type erasing function $|-|$) .
The map $|-| : \mathsf{Object}(\boldsymbol{\lambda}_{R\omega}) \to \Lambda_F$ is inductively defined by

1. *$|x| = x$ if $x \in Var^{\star}$.*

2. *$|f| = f$ if $f \in \mathcal{F}$.*

3. *$|\lambda x{:}\, A.q| = \lambda x.|q|$ if $\lambda x{:}\, A.q$ is formed by $(\star, \star)$.*

4. *$|\lambda x{:}\, A.q| = |q|$ if $\lambda x{:}\, A.q$ is formed by $(\Box, \star)$.*

5. *$|pq| = |p||q|$ if pq is formed by $(\star, \star)$*

6. *$|pQ| = |p|$ if pQ is formed by $(\Box, \star)$.*

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 3.14.

Property 7.3 *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 $| - |$ is well-defined, i.e. $|M| \in \Lambda_{\wedge R}$ in case $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ (Theorem 7.9 below). To do this we need some lemmas. In the following a type A will be called *arrow-ground* if its β-normal form $A{\downarrow_\beta}$ belongs to $\mathsf{T}_{\mathcal{S}}$.

Recall that, since in $\boldsymbol{\lambda}_{R\omega}$ we do not have rules $(\star, \Box)$ it is not possible to have algebraic reductions inside types (see Lemma 3.14); so $\overset{c}{=}_{R\beta}$ between types is indeed $=_\beta$.

Lemma 7.4 *Let $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$. Then there exists a context $C[\,]$ and a term N such that: $M \equiv C[N]$, N is a variable or an object formed by a $(\star, \star)$-rule and $|\, C[z] \,| \equiv z$ (for z fresh variable). Besides:*

1. *if $|N| \equiv cN_1 \ldots N_m$ (c variable or function symbol) and c has arrow-ground type δ with $\delta{\downarrow_\beta} \equiv \sigma_1 \to \ldots \sigma_m \to \sigma$ then there exist $N'_i (1 \leq i \leq m)$ 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 $n \geq 0$ 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 each $\Pi u_i : D_i \ldots$ is formed by $(\Box, \star)$, A' is the type of N and A is the type of M.

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

Lemma 7.5 *Let $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ such that $\Gamma \vdash M : A$ and $|M|$ is in β-normal form. Then there exist B and σ such that $B \vdash |M| : \sigma$. Besides,*

1. *if A is an arrow-ground type then $\sigma \equiv A{\downarrow_\beta}$*

2. *if $x : \delta \in \Gamma$ with δ arrow-ground type, then $x : \delta{\downarrow_\beta} \in B$.*

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

1. If $|M|$ is a constant we get that $|M|$ is typable and the second condition is trivially satisfied. The first condition is a consequence of Lemma 7.4.

2. If $|M| \equiv x$ is a variable we distinguish the four possible cases:

   if A is not arrow-ground and δ is so, $x : \delta\!\downarrow_\beta \vdash x : \delta\!\downarrow_\beta$ satisfies the conditions;

   if δ is not arrow-ground and A is so, $x : A\!\downarrow_\beta \vdash x : A\!\downarrow_\beta$ satisfies the conditions;

   if both A and δ are arrow-ground, by Stripping (iii) and the fact we are in $\boldsymbol{\lambda}_{R\omega}$, we have that $A\!\downarrow_\beta \equiv \delta\!\downarrow_\beta$. Hence $x : A\!\downarrow_\beta \vdash x : A\!\downarrow_\beta$ satisfies the conditions;

   if neither A nor δ is arrow-ground, any application of (var), say $x : s \vdash x : s$ with $s \in \mathcal{S}$, 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 $g \in \mathcal{F}$ 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 Γ' 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, σ_i such that $B_i \vdash N_i : \sigma_i$, and besides, if A_i is arrow-ground then $\sigma_i \equiv A_i\!\downarrow_\beta$, and if $x : \delta \in \Gamma'$ with δ arrow ground then $x : \delta\!\downarrow_\beta \in B_i$. It is now straightforward to get a derivation for $B_1 \cup^\wedge \ldots \cup^\wedge B_m \cup^\wedge \{g : \sigma_1 \to \ldots \to \sigma_m \to \sigma\} \vdash |M| : \sigma$, where the symbol $\cup^\wedge$ denotes the following operation between bases:

   $B \cup^\wedge B' = \{x : \sigma \wedge \tau \mid x : \sigma \in B \textit{ and } x : \tau \in B'\} \cup \{x : \sigma \mid x : \sigma \in B \textit{ and } x \notin B'\} \cup \{x : \tau \mid x \notin B \textit{ and } x : \tau \in B'\}$ (it is straightforward to check that if $B \vdash M : \beta$ then we have also $B \cup^\wedge B' \vdash M : \beta$ for all B'). Notice that if $x : \delta \in \Gamma'$ with δ arrow ground then the type of x in $B_1 \cup^\wedge \ldots \cup^\wedge B_m \cup^\wedge$ it 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 $B_i \vdash N_i : \sigma_i$'s, and by the fact noticed above, we get that, for all $x \neq g$, $x : \delta \in \Gamma$ with δ arrow ground implies $x : \delta\!\downarrow_\beta \in B$. For what concerns g we know that if $g : \delta \in \Gamma$ with δ arrow ground then $\delta\!\downarrow_\beta \equiv \sigma_1 \to \ldots \to \sigma_m \to \sigma$ by Lemma 7.4. So the second condition is satisfied as well.

   (b) $n = 0$, $m > 0$ and $g \in \mathcal{F}_{\sigma_1 \to \ldots \to \sigma_p \to \sigma}$.

   By the form of the term and Lemma 7.4, there exist Γ' and N'_i, A'_i $(1 \leq i \leq m)$ 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 $1 \leq i \leq m$, and $\tau =_\beta \sigma_{m+1} \to \sigma_p \to \sigma$). By induction, for $1 \leq i \leq m$ there exists B_i such that $B_i \vdash N_i : \sigma_i$ 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 7.4). Then, $B_1 \cup^\wedge \ldots \cup^\wedge B_m \vdash g N_1 \ldots N_m : \sigma_{m+1} \to \ldots \to \sigma_p \to \sigma$ satisfies the conditions.

   (c) $n \neq 0$, $|M| \equiv \lambda y.N$.

   By the form of the term, Lemma 7.4 and Stripping, there exist Γ', 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 $\Pi\alpha_i{:}\,\beta_i \ldots$. are formed by $(\square, \star)$ and $\Pi y{:}\,\tau \ldots$. is formed by $(\star, \star)$. Now, if A is arrow ground then $k = 0$ and A', τ are arrow-ground, then $A\!\downarrow_\beta = \Pi y{:}\,\tau\!\downarrow_\beta .A'\!\downarrow_\beta$. By induction, $B \vdash N : \sigma$ with $\sigma \equiv A'\!\downarrow_\beta$ and $y : \tau\!\downarrow_\beta \in B$. Hence $B - \{y : \tau\!\downarrow_\beta\} \vdash \lambda y.N : \tau\!\downarrow_\beta \to \sigma \equiv A\!\downarrow_\beta$ satisfies both conditions. $\square$

Lemma 7.6 *Let $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ and $|M| \equiv Q[P/x]$ where P is a term such that if $x \notin FV(Q)$ then P is in β-normal form, and $\exists P' \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ s.t. $|P'| \equiv P$. Then*

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

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 $B \vdash Q[P/x] : \sigma$ 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 $\{B_i \vdash P : \delta_i \mid i \in I\}$ 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 $B_1 \cup^\wedge \ldots \cup^\wedge B_n \vdash P : \bigwedge_{i\in I} \delta_i$. (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 $B, x : \bigwedge_{i\in I} \delta_i \vdash Q : \sigma$ by extending the basis B and replacing in the proof of $B \vdash Q[P/x] : \sigma$ the deductions of $B, x : \bigwedge_{i\in I} \delta_i \vdash x : \delta_j$ (obtained using rules (Ax) and $(\wedge E)$ in $\lambda_{\wedge R}$) for the subdeductions of $B_j \vdash P : \delta_j$. So by rule $(\rightarrow I)$ in $\lambda_{\wedge R}$ we have that $B \vdash \lambda x.Q : (\bigwedge_{i\in I} \delta_i) \rightarrow \sigma$ and since we have $B_1 \cup^\wedge \ldots \cup^\wedge B_n \vdash P : \bigwedge_{i\in I} \delta_i$ we can obtain $B' \vdash (\lambda x.Q)P : \sigma$ 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 7.5 we have that $\exists B_1, \delta$ such that $B_1 \vdash P : \delta$. Besides, we know that $\exists B, \sigma$ such that $B \vdash Q[P/x] : \sigma$, i.e., $B \vdash Q : \sigma$ since x is not free in Q. Then $B, x : \delta \vdash Q : \sigma$ and from this $B \vdash \lambda x.Q : \delta \rightarrow \sigma$. Now it is easy to construct a proof of $B' \vdash (\lambda x.Q)P : \sigma$ where $B' = B \cup^\wedge B_1$. $\Box$

Lemma 7.7 *Let $M[M'/x] \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ where $x \in Var^\star$.*

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

Proof By induction on the structure of M.

1. If $M \equiv f \in \mathcal{F}$ 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 M is formed by $(\star, \star)$ 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 M is formed by $(\Box, \star)$ then $|M[M'/x]| \equiv |N'[M'/x]|$, which by induction is equal to $|N'|[|M'|/x] \equiv |M|[|M'|/x]$.
3. If $M \equiv M_1 M_2$ then $|M[M'/x]| \equiv |M_1[M'/x] M_2[M'/x]|$. Now there are again two subcases:
   (a) If M is formed by $(\star, \star)$ 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 is formed by $(\Box, \star)$ then $|M[M'/x]| \equiv |M_1[M'/x]|$ which by induction is equal to $|M_1|[|M'|/x] \equiv |M|[|M'|/x]$. $\Box$

Lemma 7.8 *Let $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$. If $|M| \rightarrow_\beta N$ then there exists $M' \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ such that $N \equiv |M'|$ and $M \rightarrow_\beta M'$.*

Proof In order to have $|M| \rightarrow_\beta N$ it has to be $M \equiv C'[(\lambda x{:}\, A.q)q']$ where $C'[\,]$ is a context and $(\lambda x{:}\, A.q)q'$ is formed by $(\star, \star)$. Then $|M| = C[(\lambda x.|q|)|q'|] \rightarrow_\beta C[|q|[|q'|/x]] = N$ 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 $M \rightarrow_\beta M'$ and $|M'| = N$ by Lemma 7.7. $\Box$

Theorem 7.9 *If $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ then there exist B, σ such that $B \vdash |M| : \sigma$.*

Proof $|M|$ is $\rightarrow_\beta$-strongly normalizable by Lemma 7.1 and Lemma 7.8. Then all the reduction strategies allow us to get the β-normal form of $|M|$, in particular the reduction strategy according to which a contraction $(\lambda x.Q)P \rightarrow_\beta Q[P/x]$ is performed only if P is in β-normal form (i.e. the rightmost-innermost evaluation). Hence we can find B and σ by applying Lemma 7.5 to the β-normal form of $|M|$ and iterating Lemma 7.6 backwards along the chain of reduction steps which leads from $|M|$ to its β-normal form. $\Box$

So, we have proved that if $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ then $|M| \in \Lambda_{\wedge R}$. In fact we can also prove that for algebraic terms we can obtain the same type.

Lemma 7.10 *If $\Gamma \vdash M : \tau$ is an algebraic term in $\boldsymbol{\lambda}_{R\omega}$ then $|M| \in \Lambda_{\wedge RH}$ and $|M|$ is typable with $\tau\downarrow_\beta$ from a basis B such that if $x : \tau' \in \Gamma$ then $x : \tau'\downarrow_\beta \in B$.*

Proof By induction on the structure of M.

If M is a variable the result is trivial.

If $M \equiv fM_1 \ldots M_n$ where $f \in \mathcal{F}_{\sigma_1 \ldots \sigma_n \rightarrow \sigma}$ then $M_1 \ldots M_n$ are algebraic terms of type $\sigma_1 \ldots \sigma_n$ (in β-normal form) in Γ, and $\Gamma \vdash M : \tau$ implies $\tau\downarrow_\beta \equiv \sigma$. By induction there exist $B_1, \ldots, B_n$ such that if $x : \sigma' \in \Gamma$ then $x : \sigma'\downarrow_\beta \in B_i$ and $B_i \vdash |M_i| : \sigma_i$. Then $B_1 \cup \ldots \cup B_n \vdash |M| : \sigma$.

If $M \equiv XM_1 \ldots M_n$ where $X : \tau_1 \rightarrow \ldots \tau_n \rightarrow \tau \in \Gamma$ and $M_1 \ldots M_n$ are algebraic terms of type $\tau_1 \ldots \tau_n$ in Γ then by induction there exist $B_1, \ldots, B_n$ such that if $x : \tau' \in \Gamma$ then $x : \tau'\downarrow_\beta \in B_i$ and $B_i \vdash |M_i| : \tau_i\downarrow_\beta$. Then $B_1 \cup \ldots \cup B_n \cup X : \tau_1\downarrow_\beta \rightarrow \ldots \tau_n\downarrow_\beta \rightarrow \tau\downarrow_\beta \vdash |M| : \tau\downarrow_\beta$. $\Box$

The following theorem will be fundamental for proving that $\boldsymbol{\lambda}_{R\omega}$ terms are $\rightarrow_{R\beta}$-strongly normalizing.

Theorem 7.11 *[BF93a] $\rightarrow_{R\beta}$ is strongly normalizing in $\Lambda_{\wedge R}$ if $\rightarrow_{FOR}$ is conservative and strongly normalizing on first-order algebraic terms and if HOR satisfies the general schema.*

In fact, in the following we shall prove that if $M \rightarrow_{R\beta} M'$ then $|M| \rightarrow^*_{R\beta} |M'|$.

A β-reduction on a $\boldsymbol{\lambda}_{R\omega}$ term such that the β-redex is formed by (p_1, p_2) will be called in the following a $\beta(p_1, p_2)$-reduction.

Let φ be a substitution for $\boldsymbol{\lambda}_{R\omega}$, 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 7.12 *Let t be an algebraic term of $\boldsymbol{\lambda}_{R\omega}$. Then $|t| \equiv t$.*

Proof By induction on the structure of t. $\Box$

Lemma 7.13 *Let $M, N \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$. If $M \rightarrow_{R\beta} N$ then $|M| \rightarrow_{R\beta} |N|$ or $|M| \equiv |N|$ if the reduction is actually a $\beta(\Box, \Box)$- or a $\beta(\Box, \star)$-reduction.*

Proof Let $M = C[P]$ where P is a β-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 β-reduction which rule the redex has been formed by.

Let $P = (\lambda a{:}\, A.Q)Q'$ where $\lambda a{:}\, A.Q$ is formed by $(\Box, \Box)$. It is easy to check that, by definition of $|\ |$, if $P : S : \Box$ then $|C[P]| = |C[U]|$ for each $U : S$. We have therefore $|M| \equiv |N|$.

Let $P = (\lambda a{:}\, A.Q)Q'$ where $\lambda a{:}\, A.Q$ is formed by $(\Box, \star)$. We have that $|(\lambda a{:}\, A.Q)Q'| = |Q| = |Q[Q'/a]|$ by definition of $|\ |$, and then $|M| \equiv |N|$.

Let $P = (\lambda x{:}\, A.Q)Q'$ where $\lambda x{:}\, A.Q$ is formed by $(\star, \star)$. We have that $|(\lambda x{:}\, A.Q)Q'| = (\lambda x.|Q|)|Q'| \rightarrow_\beta |Q|[|Q'|/x]$ and $|Q|[|Q'|/x] = |Q[Q'/x]|$ by Lemma 7.7. It follows then $|M| \rightarrow_\beta |N|$.

Let $P = A\varphi$ and $A\varphi \rightarrow_R A'\varphi$ using a rule $r : s \rightarrow t$ such that $A \equiv \rho(s)$ and $A' \equiv \rho(t)$ where ρ is a renaming of variables. By Lemma 7.7, $|A\varphi| = |A||\varphi|$. By Theorem 7.9, $|A||\varphi| \in \Lambda_{\wedge R}$. Moreover, since A is algebraic, $A \equiv |A|$ (by Lemma 7.12), then $|A||\varphi| \rightarrow_R |A'||\varphi| = |A'\varphi|$. It follows then $|M| \rightarrow_R |N|$. $\Box$

We can prove now the main theorem of this section.

Theorem 7.14 *Assume R is conservative and strongly normalizing on first-order algebraic terms and HOR satisfies the general schema. If $M \in \mathsf{Term}(\boldsymbol{\lambda}_{R\omega})$ then M is $\rightarrow_{R\beta}$-strongly normalizable.*

Proof By Lemma 4.11 only the cases $M \in \mathsf{Constr}(\boldsymbol{\lambda}_{R\omega})$ and $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ have to be considered.

If $M \in \mathsf{Constr}(\boldsymbol{\lambda}_{R\omega})$ then, since $\boldsymbol{\lambda}_{R\omega}$ objects cannot occur in constructors (Lemma 3.14), it follows that each reduction in M is actually a β-reduction. Hence, strong normalization of M follows from Lemma 7.1.

If $M \in \mathsf{Object}(\boldsymbol{\lambda}_{R\omega})$ then we assign to M the pair $\mathcal{I}(M) = \langle |M|, M\rangle$. By Theorem 7.9, $|M| \in \Lambda_{\wedge R}$ and then by Theorem 7.11 $|M|$ is $\rightarrow_{R\beta}$-strongly normalizing. By Lemma 7.1, M is $\rightarrow_\beta$-strongly normalizable. Then the ordering $>\equiv (\rightarrow_{R\beta}, \rightarrow_{\beta(\Box,\Box)} \cup \rightarrow_{\beta(\star,\Box)})_{lex}$ is well founded on these pairs. Moreover, $M \rightarrow_{R\beta} M'$ implies $\mathcal{I}(M) > \mathcal{I}(M')$, so there is no infinite reduction sequence. $\Box$

8 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 [New42]). So we shall prove that $\rightarrow_{\beta R}$ is locally confluent on $\boldsymbol{\lambda}_{RC}$. 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 8.1 *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 μ 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 in 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 8.2 *Consider the β-rule: $(\lambda x.M)N \rightarrow M\{x \mapsto N\}$*[12]*. 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 β: $(\lambda x.M, F(M\{x \mapsto 0\}))$.*

The following lemmas show that the confluence of FOR for algebraic terms transfers to $\boldsymbol{\lambda}_{RC}$-terms, and that for the class of higher order rewrite systems which we consider, the absence of critical pairs implies confluence (note that this is not true for arbitrary higher-order rewrite systems, as shown in [Nip91]).

Lemma 8.3 *If FOR is confluent on the set of algebraic terms of $\boldsymbol{\lambda}_{RC}$ then $\rightarrow_{FOR}$ is locally confluent on $\boldsymbol{\lambda}_{RC}$.*

Proof Let us first prove local confluence, by structural induction. Let M be a term in $\boldsymbol{\lambda}_{RC}$. 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, $N_1 \rightarrow^*_{FOR} N'\{z_i \mapsto A_i\}$ and $N_2 \rightarrow^*_{FOR} N'\{z_i \mapsto A_i\}$.

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

Lemma 8.4 *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 $\boldsymbol{\lambda}_{RC}$.*

[12]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.

Proof In order to prove local confluence it is sufficient to show the commutation of $\rightarrow_{HOR}$ reductions on overlapped redexes. Let t be a term in $\boldsymbol{\lambda}_{RC}$ 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. $\Box$

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

Theorem 8.5 (Local Confluence of $\rightarrow_{R\beta}$ in $\boldsymbol{\lambda}_{RC}$) *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 β and HOR[13]) then $\rightarrow_{R\beta}$ is locally confluent in $\boldsymbol{\lambda}_{RC}$.*

Proof It suffices to show the commutation of β-, $\rightarrow_{FOR}$- and $\rightarrow_{HOR}$-reductions on overlapped redexes. But since $\rightarrow_{\beta}$ is confluent, $\rightarrow_{HOR}$ is confluent (by Lemma 8.4) and $\rightarrow_{FOR}$ is confluent (by Lemma 8.3), 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. $\Box$

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

9 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 Th. Coquand and C. Paulin-Mohring [Coq90], 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 recently proved by B. Werner [Wer94]. The problem of extending the CCI with pattern-matching definitions was studied by Th. Coquand [Coq92]. In particular, in [Coq92] 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

[13] See example 8.2.

is important to study under which conditions these proofs are modular. Our results show that in order to prove strong normalization of any of the systems in the λR-cube it is sufficient to prove termination of the first order rewrite rules in R on algebraic 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 restriction on first order rules is not important in practice, since most implementations of rewriting use sharing, and shared reductions are always conservative. The general schema, however, limits the power of the higher-order rules. The generalization of the proof of strong normalization to wider classes of higher-order rules will be the subject of future work.

References

[vB93] S. van Bakel. *Intersection type disciplines in lambda calculus and applicative term rewriting systems.* PhD thesis, University of Nijmegen, 1993.

[Barb90] F. Barbanera. Adding algebraic rewriting to the calculus of constructions: Strong normalization preserved. In *Proc. of the 2nd Int. Workshop on Conditional and Typed Rewriting*, 1990.

[BF93a] F. Barbanera and M. Fernández. Combining first and higher order rewrite systems with type assignment systems. *Proc. of the Int. Conference on Typed Lambda Calculi and Applications*, Utrecht, LNCS 664, Springer Verlag, 1993.

[BF93b] F. Barbanera and M. Fernández. Modularity of termination and confluence in combinations of rewrite systems with λ_ω. *Proc. of the 20th Int. Colloquium on Automata, Languages, and Programming*, Lund, LNCS 700, Springer Verlag, 1993.

[BFG94] F. Barbanera, M. Fernández, and H. Geuvers. Modularity of Strong Normalization in the algebraic-λ-cube. *Proc. of the 9th IEEE Symposium on Logic In Computer Science*, Paris, 1994.

[Bar86] H. Barendregt. *Lambda Calculus : its syntax and demantics*, second edition. North Holland, 1986.

[Bar91] H. Barendregt. Introduction to generalised type systems. *Journal of Functional Programming*, 1991.

[BCD83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter λ-model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[Ber88] S. Berardi. *Type dependence and constructive mathematics.* PhD thesis, Mathematical Institute, University of Torino, 1990.

[BT88] V. Breazu-Tannen. Combining algebra and higher-order types. In *Proc. 3rd IEEE Symposium on Logic In Computer Science*, Edinburgh, 1988.

[BTG90] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization. *Theoretical Computer Science*, 83(1), 1991.

[BTG92] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic confluence. *Information and Computation*, 82:3–28, 1992.

[deB80] N.G. deBruijn. A survey of the project Automath. In *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism.*, eds. J.P. Seldin, J.R. Hindley, Academic Press, New York, 1980.

[CC90] F. Cardone and M. Coppo. Two extensions of Curry's type inference system. In P. Odifreddi, editor, *Logic and Computer Science.* Academic Press, 1990.

[Coq90] Th. Coquand and C. Paulin-Mohring. Inductively defined types. In *Proc. of Colog'88*, LNCS 417, Springer-Verlag, 1990.

[Coq92] Th. Coquand. Pattern matching with dependent types. In *Proc. of the Workshop on Logical Frameworks*, 1992.

[CD80] M. Coppo, and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4), 1980.

[CDHL84] M. Coppo, M. Dezani, F. Honsell, and G. Longo. Extended type structures and filter lambda models. In *Logic Colloquium 82, Amsterdam*, 1984.

[CDV80] M. Coppo, M. Dezani-Ciancaglini and B. Venneri. Principal type schemes and λ calculus semantics. In *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism.*, eds. J.P. Seldin, J.R. Hindley, Academic Press, New York, 1980.

[CH88] Th. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[Church40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5, 1940.

[DJ88] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[Dou91] D. J. Dougherty. Adding algebraic rewriting to the untyped lambda calculus. In *Proc. 4th Rewriting Techniques and Applications*, Como, LNCS 488, Springer-Verlag, 1991.

[Geu92] H. Geuvers. The Church-Rosser property for $\beta\eta$-reduction in typed λ-calculi. *Proc. 7th IEEE Symposium on Logic In Computer Science*, Santa Cruz, 1992.

[Geu93] H. Geuvers. *Logics and type systems.* PhD thesis, Dept. of Computer Science, University of Nijmegen, 1993.

[GN91] H. Geuvers and M.J. Nederhof. A simple modular proof of the strong normalization for the calculus of constructions. *Journal of Functional Programming*, vol.1, 1991.

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur.* Thèse d'Etat, Univ. Paris VII, France, 1972.

[HHP87] R. Harper, F. Honsell and G. Plotkin. A framework for defining logics. *Proc. of the 2nd IEEE Symposium on Logic In Computer Science.*, Washington DC, 1987.

[Hin86] R. Hindley and J. Seldin. *Introduction to Combinators and λ-calculus.* Cambridge University Press, 1986.

[Hin90] R. Hindley. Types with intersection, an introduction. *Formal aspects of Computing*, 1990.

[JO91] J.-P. Jouannaud and M. Okada. Executable higher-order algebraic specification languages. In *Proc. of the 6th IEEE Symposium Logic In Computer Science*, Amsterdam, 1991.

[Klo87] J. W. Klop. Term rewriting systems: a tutorial. *EATCS Bulletin*, 32:143–182, 1987.

[KOR93] J. W. Klop, V. van Oostrom and F. van Raamsdonk. Combinatory Reduction Systems, introduction and survey. *Theoretical Computer Science* 121(1-2), 1993.

[New42] M. H. A. Newman. On theories with a combinatorial definition of 'equivalence'. *Ann. Math.*, 43(2):223–243, 1942.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proc. of the 6th IEEE Symposium Logic In Computer Science*, Amsterdam, 1991.

[Oka89] M. Okada. Strong normalizability for the combined system of the types lambda calculus and an arbitrary convergent term rewrite system. In *Proc. ISSAC 89*, Portland, 1989.

[OR93] V. van Oostrom and F. van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. IR 333, Vrije Universiteit, Amsterdam, August 1993.

[OR94] V. van Oostrom and F. van Raamsdonk. Weak ortogonality implies confluence: the higher order case. To appear in *Logical Foundation of Computer Science*.

[Pott80] G. Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism.*, eds. J.P. Seldin, J.R. Hindley, Academic Press, New York, 1980.

[Rus87] M. Rusinowitch. On termination of the direct sum of term rewriting systems. *Information Processing Letters*, 26:65–70, 1987.

[Toy87] Y. Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, April 1987.

[Wer94] B. Werner. *Méta-théorie du Calcul des Constructions Inductives.* Thèse Univ. Paris VII, France.

A System $\lambda_{\wedge R}$

System $\lambda_{\wedge R}$ is a type assignment system with intersection types and algebraic features which was introduced in [BF93a], 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, in order to keep track of the premises statements depend on. We shall refer to [Hin86] 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 were originally devised in [CD80, CDV80] and deeply investigated afterwards in several papers, among which we recall [BCD83, Pott80, CDHL84, vB93]. We refer to the above mentioned papers and to the surveys [CC90] and [Hin90] for motivations and applications of intersection types. System $\lambda_{\wedge R}$ can be considered an extension of a fundamental system with intersection types (called $\vdash_{-\omega}$ in [vB93]).

We begin the description of system $\lambda_{\wedge R}$ by considering a set $\mathcal{S}$ 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.

As said before, to define a type assignment system we have to specify a set of types, a set of (untyped) terms and a set of inference rules to assign types to terms.

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

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

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 Sect. 3, we denote by $\mathsf{T}_{\mathcal{S}}$ 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 $\mathcal{X}$ of (untyped) variables and $\mathbf{f}$ over F. Terms are then untyped λ-terms with constants and on them the usual notion of β-reduction is defined.

Definition A.3 (*i*) *A* statement *is an expression of the form $M : \sigma$ where $\sigma \in \mathsf{T}_{\mathcal{S}\wedge}$ 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 $\sigma_i \in \mathsf{T}_{\mathcal{S}}$ $(1 \leq i \leq n)$ and is of the form $\tau_1 \rightarrow \ldots \rightarrow \tau_m \rightarrow \tau$ for m arity of f_i.*

Definition A.4 (Inference rules) *Let* Ax *be a set of axiom statements.*

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

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

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

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

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

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

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

A term M will be called *typable* if there exists a basis B and a type σ such that $B \vdash^{\wedge} M : \sigma$. We shall denote by $\Lambda_{\wedge R}$ the set of typable terms.

To completely specify system $\lambda_{\wedge R}$ we have to give a set of algebraic rewrite rules. To define what algebraic rewriting is in our type assignment system 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. The definitions of all these notions would be however quite similar to those given in Sect. 3 in a typed context, so, instead of doing that we can equivalently define algebraic rewriting for system $\lambda_{\wedge R}$ as induced by a set $R = FOR \cup HOR$ of *typed* rewrite rules as presented in Definition 3.10. Before doing that, in order to be precise, we have to give a small technical definition.

Definition A.5 *Given a set $\mathcal{S}$ of sorts, a set F of constants, and a set* Ax *of axiom statements, let t be an algebraic term for $\mathcal{S}$ and $\mathcal{F}$ as defined in 3.3, where $\mathcal{F}$ is the signature naturally induced by F and* Ax.
$\mathbf{t}$ *is the untyped term obtained by replacing any occurrence of a function symbol of $\mathcal{F}$ 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 rewrite rules for a signature $\mathcal{F}$ as in 3.10. Let $M, M' \in \Lambda_{\wedge R}$, $r : t \rightarrow t' \in R$. The reduction relation $\rightarrow_r$ on $\Lambda_{\wedge R}$ is defined as follows.*
$M \rightarrow^r M'$ *if* $M \equiv C[\mathbf{t}[N_1/x_1, \ldots, N_n/x_n]]$ *(where $N_1, \ldots, N_n \in \Lambda_{\wedge R}$ 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 λ-terms.)*
$M \rightarrow_R M'$ *if* $M \rightarrow^r M'$ *for some* $r : t \rightarrow t' \in R$.

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

Note that because of the following lemma, which follows easily by definition of 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 rewritable typed term t.

Lemma A.7 *Let t be a* typed *rewritable term in a rule $r : t \rightarrow t' \in R$. Moreover, let σ and Γ'' be the algebraic type and context of Lemma 3.17 relative to 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 γ_1 is the type of the variable x in t.

System $\lambda_{\wedge R}$ is then completely specified by a quadruple $\langle \mathcal{S}, F, \mathsf{Ax}, R \rangle$, where $\mathcal{S}$ is a set of sorts, F a set of constants, Ax a set of axiom statements and R a set of rewrite rules (in the sense of Sect. 3 and for the signature $\mathcal{F}$ induced by F and Ax).

The main property of $\lambda_{\wedge R}$, and the most useful for us, is strong normalization.

Theorem A.8 ($\lambda_{\wedge R} \models \mathsf{SN}$ [BF93a]) *Let $\lambda_{\wedge R}$ be the system defined by*

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

where R is such that

1. *FOR is conservative and first-order algebraic terms are strongly normalizable w.r.t. $\rightarrow_{FOR}$*
2. *HOR satisfies the general schema (w.r.t. FOR).*

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