BULLETIN OF THE
GREEK MATHEMATICAL SOCIETY
Vol. 55, (2008) 65–73

Properties of developments via simple types

George Koletsos and George Stavrinos

Abstract

By using an infinity of extra constants every λ-term with indexed redexes is interpreted into a term in the simply typed λ-calculus à la Curry. A development becomes a usual β-reduction in the simply typed lambda calculus and the corresponding properties of developments come out from the corresponding properties (strong normalization and Church-Rosser) holding in this system. In this way we obtain a complete simulation of the notion of development into the system of simply typed lambda calculus.

Keywords: developments, strong normalization, Church-Rosser property, simple types

1. Introduction

A *development* is a reduction sequence in which only redexes in a pre-elected set, the indexed redexes (and their residuals through the reduction), are allowed to be contracted. The notion is defined in [1] where the basic properties —strong normalization and uniqueness of normal form— are proved (the finiteness of developments). The notion of developments and its properties play an important role in the theory of λ-calculus. As an application, for example, we can use this in order to prove the Church-Rosser theorem for the usual untyped λ-calculus, see [1].

In this paper we show that the whole idea of working with developments can be reduced to working in the plain (without indexed redexes) simply typed λ-calculus $\lambda_{\rightarrow}$. We define an embedding $[\_]$ that associates to every λ-term M with indexed redexes a set of representatives $[M]$ in the calculus $\lambda_{\rightarrow}$. A non-indexed redex in M becomes a "blocked" redex in any of its representatives in $[M]$ and applications are "neutralized" in an appropriate way so that no new redexes can be created through the reduction process. In this way we can fully simulate, in $\lambda_{\rightarrow}$, the notion of development by using only the usual notion of β-reduction. The properties of developments —strong normalization and Church-Rosser— are obtained immediately via the corresponding properties of simply typed lambda calculus. There are well known proofs for these properties, for example, in the case of strong normalization, proofs by Tait [12] and Gandy [3] and in the case of Church-Rosser property, proofs by Koletsos [6] and Statman [11].

This paper is a continuation of previous work by Parigot and Krivine in [9], by Koletsos and Stavrinos in [7, 8] and by Ghilezan and Kunčak in [5] and [4]. The authors in [9] and [7, 8] simulate the process of developments in the system $\mathcal{D}$ of intersection types by using one extra constant to "block" redexes and applications while in [5] and [4] two extra constants are used to type, with ground type 0, the

corresponding "blocked" term in the simply typed λ-calculus. In [9] and [7, 8] the unique constant is put only in front of redexes and applications while in [5] and [4] one of the constants is put, as before, in front of redexes and applications and the other in front of the λ-abstractions (in order to give always the ground type) thus using more "positions" to put constants. In this paper we use the same "positions" for putting constants as, for example, in [9] but we use many constants (instead of one). This permits to type the corresponding term in the simply typed lambda calculus and to give a simulation of the process of developments in this calculus. The properties of developments —strong normalization and Church-Rosser— follow from the corresponding properties in the simply typed lambda calculus.

2. Developments

We present briefly the notion of development. We rely mainly on [1] where the reader can find all the details.

Definition 2.1 The set Λ' of indexed λ-terms is inductively defined as follows:
($\mathcal{V} = \{x, y, z, \ldots\}$ is an infinite set of variables)

(1) $x \in \mathcal{V} \Rightarrow x \in \Lambda'$

(2) $x \in \mathcal{V},\ M \in \Lambda' \Rightarrow (\lambda x.M) \in \Lambda'$

(3) $M, N \in \Lambda' \Rightarrow (MN) \in \Lambda'$

(4) $x \in \mathcal{V},\ M, N \in \Lambda' \Rightarrow ((\lambda_0 x.M)N) \in \Lambda'$

Remark 2.1 We use only one index, the index 0. If we omit clause (4) in the definition 2.1 we obtain the set Λ ($\Lambda \subseteq \Lambda'$) of the usual λ-terms. We write $MN_1 \cdots N_k$ for $(\cdots(MN_1)\cdots N_k)$. We omit parentheses if there is no confusion.

The redexes of the form $(\lambda_0 x.M)N$ are called *indexed* redexes and the redexes of the form $(\lambda x.M)N$ are called *usual* redexes. In Λ, the notion of reduction is the β-reduction defined as usual by $\beta : (\lambda x.M)N \rightarrow_\beta M[x := N]$.

Note that $M[x := N]$ denotes the term that results after substituting every free occurrence of x in M by N. We adopt the Barendregt's variable convention according to which no bound and free variable in any context has the same name.

In Λ' we are allowed to contract only indexed redexes so the only notion of reduction in Λ' is β_0 defined by

$$\beta_0 : (\lambda_0 x.M)N \rightarrow_{\beta_0} M[x := N]$$

If $M' \in \Lambda'$ then $||M'||$ is the λ-term in Λ resulting from erasing all the indices from the indexed redexes in M', i.e. $||x|| = x$, $||(\lambda x.M)|| = (\lambda x.||M||)$, $||(MN)|| = (||M||\,||N||)$ and $||((\lambda_0 x.M)N)|| = ((\lambda x.||M||)||N||)$.

Notation: If R is a notion of reduction, i.e. a binary relation on the set of all terms, then $\rightarrow_R$ is the contextual closure of R and $\twoheadrightarrow_R$ is the reflexive and transitive closure of $\rightarrow_R$ (see [1]).

It is well known that if $\mathcal{F}$ is a set of redex occurrences in a term M in Λ we can associate to M a term $(M, \mathcal{F})$ in Λ' by indexing all the redex occurrences in $\mathcal{F}$ (i.e. replacing every $(\lambda x.M)N$ in $\mathcal{F}$ by $(\lambda_0 x.M)N$). If we start a β_0-reduction from $(M, \mathcal{F})$ then for every term M' in this reduction sequence, say σ', we associate in

the obvious way a term $||M'||$ and to each indexed redex in M' a redex occurrence in $||M'||$. Through this correspondence we can define the notion of *residuals* of $\mathcal{F}$ relative to the reduction sequence σ (starting from M) corresponding to σ', see [1].

So the notion of development is completely simulated (or defined) in terms of the β_0-reduction in Λ'. In that way the finiteness of developments and the Church-Rosser property of developments are properties respectively equivalent to the strong normalization property and the Church-Rosser property of the β_0-reduction in Λ'.

3. Simply Typed Lambda Calculus

We present briefly the system of simply typed lambda calculus (à la Curry), the system $\lambda_{\rightarrow}$, and some of its main properties, see also [2].

Definition 3.1 Types are constructed from the variables $\alpha_1, \alpha_2, \ldots$ (a denumerable set of variables) inductively as follows:

(1) $\alpha_1, \alpha_2, \ldots$ are types.

(2) If σ and τ are types then $(\sigma \rightarrow \tau)$ is a type.

We use $\sigma, \tau, \rho, \ldots$ to denote types. Usually we omit outer parentheses.

Definition 3.2 A *context* Γ is a set $\{x_1 : \sigma_1, \cdots, x_k : \sigma_k\}$ of variable assignments ($x_i \in \Lambda$ is a term variable and σ_i is a type). All the $x_1, \cdots, x_k$ in a context are distinct and if we write $\Gamma, x : \sigma$ we presuppose that x does not occur in Γ. If $x_i : \sigma_i \in \Gamma$ we say that x_i is *declared* in Γ or x_i has a *declaration* in Γ.

Definition 3.3 We define the notion $\Gamma \vdash M : \sigma$ (the subject $M \in \Lambda$ is typed with type σ in the context Γ) by the following rules:

(1) $\Gamma, x : \sigma \vdash x : \sigma$

(2) $\Gamma, x : \sigma \vdash M : \tau \;\Rightarrow\; \Gamma \vdash \lambda x.M : \sigma \rightarrow \tau$

(3) $\Gamma \vdash M : \sigma \rightarrow \tau,\ \Gamma \vdash N : \sigma \;\Rightarrow\; \Gamma \vdash MN : \tau$

The following is immediate.

Lemma 3.1 *If $\Gamma \vdash M : \sigma$, then $\Gamma, x : \tau \vdash M : \sigma$ (weakening).*

The next proposition analyzes how a term of a certain form can get typed.

Proposition 3.1 (Generation lemma) *(1) If $\Gamma \vdash x : \sigma$, then $x : \sigma \in \Gamma$.*

(2) If $\Gamma \vdash MN : \tau$, then $\Gamma \vdash M : \sigma \rightarrow \tau$ and $\Gamma \vdash N : \sigma$, for some σ.

(3) If $\Gamma \vdash \lambda x.M : \rho$, then $\Gamma, x : \sigma \vdash M : \tau$ and $\rho = \sigma \rightarrow \tau$, for some σ, τ.

Proof. Easy induction on the length of derivation. □

Proposition 3.2 (Subject reduction property) *Suppose $M \twoheadrightarrow_\beta M'$. Then*

$$\Gamma \vdash M : \sigma \;\Rightarrow\; \Gamma \vdash M' : \sigma$$

Proof. Induction on the generation of $\twoheadrightarrow_\beta$ using the generation lemma and the following lemma. □

Lemma 3.2 *Suppose* $\Gamma, x : \sigma \vdash M : \tau$ *and* $\Gamma \vdash N : \sigma$. *Then* $\Gamma \vdash M[x := N] : \tau$.

Proof. By induction on the generation of $\Gamma, x : \sigma \vdash M : \tau$. □

Simply typed lambda calculus satisfies the following important properties.

Theorem 3.1 *Let* $M \in \Lambda$ *and let* $\Gamma \vdash M : \sigma$, *i.e.* M *is typable. Then*

(1) M *is strongly normalizable, i.e. every* β*-reduction starting from* M *terminates.*

(2) M *satisfies the Church-Rosser property, i.e. if*

$$M_1 \ {}_\beta\!\!\twoheadleftarrow M \twoheadrightarrow_\beta M_2$$

for any M_1 *and* M_2, *then*

$$M_1 \twoheadrightarrow_\beta M_3 \ {}_\beta\!\!\twoheadleftarrow M_2$$

for some lambda term M_3.

Proof. (1) Proof by Gandy [3] using the weak normalization property. Proof by Tait [12] using the notion of reducibility.

(2) Direct proofs of this property using reducibility arguments and logical relations can be found in Koletsos [6], Statman [11] and Mitchell [10]. □

Remark 3.1 All the proofs above have been given directly for the system of simply typed lambda calculus. In the case of Church-Rosser property, the proofs above do not use in any sense the well-known Church-Rosser property for the usual full λ-calculus with β-reduction. So any proof of the Church-Rosser property for the full system of λ-calculus which makes use of 3.1(2) must be considered as a new proof of the Church-Rosser property.

4. Embedding Λ' in the system $\Lambda^{\mathcal{C}}$

In what follows we suppose that we have added to the variables $x, y, z, \ldots$ of Λ an infinite set of new variables $\mathcal{C} = \{c_1, c_2, \ldots, c_n, \ldots\}$. These variables will be used in the definition of $\Lambda^{\mathcal{C}}$ and will be considered as *constants* in the sense that only free occurrences of these variables will be allowed (no lambda abstraction on these variables is allowed). So now the set of variables of Λ is $\mathcal{V} = \{x, y, z, \ldots\} \cup \{c_1, c_2, \ldots, c_n, \ldots\}$.

The set of terms $\Lambda^{\mathcal{C}}$ is defined inductively as follows:

Definition 4.1 (The set $\Lambda^{\mathcal{C}}$) (1) If x is a variable not in $\mathcal{C}$ ($x \in \mathcal{V} \setminus \mathcal{C}$) then $x \in \Lambda^{\mathcal{C}}$.

(2) If x is a variable not in $\mathcal{C}$ and $M \in \Lambda^{\mathcal{C}}$ then $(\lambda x.M) \in \Lambda^{\mathcal{C}}$.

(3) If $M, N \in \Lambda^{\mathcal{C}}$ and $c_i \in \mathcal{C}$ then $c_i MN \in \Lambda^{\mathcal{C}}$.

(4) If $\lambda x.M \in \Lambda^{\mathcal{C}}$ and $N \in \Lambda^{\mathcal{C}}$ then $(\lambda x.M)N \in \Lambda^{\mathcal{C}}$.

Remark 4.1 The set $\Lambda^{\mathcal{C}}$ can be considered as part of Λ in the sense that we could have singled out some variables in $\mathcal{V}$ (the set $\mathcal{C}$) and used only variables in $\mathcal{V} \setminus \mathcal{C}$ to construct the bindings in λ-abstractions. So any term in $\Lambda^{\mathcal{C}}$ can be considered as a term in Λ where all the bound variables belong to $\mathcal{V} \setminus \mathcal{C}$. This of course does not mean that any such term in Λ belongs to $\Lambda^{\mathcal{C}}$ because in $\Lambda^{\mathcal{C}}$ any non-redex application must be preceded by a constant c_i, for example $\lambda x.yx \notin \Lambda^{\mathcal{C}}$ but $\lambda x.c_i yx \in \Lambda^{\mathcal{C}}$.

Definition 4.2 $\Lambda^{\mathcal{C}!}$ is the subset of $\Lambda^{\mathcal{C}}$ consisting of the terms having at most one occurrence of any of the variables in $\mathcal{C}$. That is if $M \in \Lambda^{\mathcal{C}!}$ and $c_i \in \mathcal{C}$ then c_i has at most one occurrence in M.

Lemma 4.1 *Let $M \in \Lambda^{\mathcal{C}}$ and let $M \to_\beta N$. Then $N \in \Lambda^{\mathcal{C}}$.*

Proof. Easy induction on M using the following lemma. □

Lemma 4.2 *If $M, N \in \Lambda^{\mathcal{C}}$ and $x \in \mathcal{V} \setminus \mathcal{C}$ then $M[x := N] \in \Lambda^{\mathcal{C}}$.*

Proof. Induction on M. The only non trivial case is when $M = (\lambda y.P)Q$. Then $M[x := N] = (\lambda y.P)[x := N]Q[x := N] = (\lambda y.P[x := N])Q[x := N]$ and the result follows by the inductive hypothesis. □

Now, for each term M in $\Lambda^{\mathcal{C}}$ we define the term $|M| \in \Lambda'$ obtained from M by "forgetting" all the constants in M. At the same time we transform the usual redexes in M into indexed redexes in $|M|$.

Definition 4.3 We give the definition of $|M|$, M in $\Lambda^{\mathcal{C}}$, by induction on M.

(1) $|x| = x$

(2) $|\lambda x.N| = \lambda x.|N|$

(3) $c_i \in \mathcal{C}$, $|c_i PQ| = |P|\,|Q|$

(4) $|(\lambda x.P)Q| = (\lambda_0 x.|P|)|Q|$

It is clear, use simple induction, that for every $M' \in \Lambda'$ there exists an $M \in \Lambda^{\mathcal{C}}$ s.t. $M' = |M|$, i.e. the correspondence $\Lambda^{\mathcal{C}} \xrightarrow{|\_|} \Lambda'$ is surjective.

We denote the set of all such M by $[M']$, i.e. $[M'] = \{M \in \Lambda^{\mathcal{C}} \mid |M| = M'\}$ is the inverse image of M' under $|\_|$. $[M']$ is the set of *representatives* in $\Lambda^{\mathcal{C}}$ of the indexed term M' in Λ'. We can obtain a representative in $[M']$ by replacing every application (PQ) in M' which is not an indexed redex by $c_i PQ$, where c_i is any constant in $\mathcal{C}$ and by replacing every λ_0 by λ. It is clear that we can arrange so that this representative in $[M']$ is chosen to be a member of $\Lambda^{\mathcal{C}!}$, i.e. to have an occurrence of a constant c_i at most once. It is clear also that two representatives of the same term differ only in the names of the constants used, that is they are identical as strings of symbols except on the positions we have occurrences of constants, the constants occurring may be different. For example, $(\lambda x.c_1 xx)(\lambda x.c_1 xx)$ and $(\lambda x.c_2 xx)(\lambda x.c_3 xx)$ are representatives of the same term $(\lambda_0 x.xx)(\lambda x.xx)$, the second belonging to $\Lambda^{\mathcal{C}!}$.

Lemma 4.3 *If $P, Q \in \Lambda^{\mathcal{C}}$ then $|P[x := Q]| = |P|[x := |Q|]$.*

Proof. Induction on P. □

We can see that β-reduction in $\Lambda^{\mathcal{C}}$ simulates the β_0-reduction in Λ' in the following sense.

Proposition 4.1 *Let $M \in [M']$ and $M \to_\beta N$. Then $M' \to_{\beta_0} |N|$.*

Proof. By induction on M'.

If $M' = x$ then M must be x so there is nothing to prove.

If $M' = \lambda x.P'$. Then the only possibility for M is to have the form $\lambda x.P$ with $|P| = P'$. But then $M \to_\beta N$ means that $N = \lambda x.P^*$ with $P \to_\beta P^*$. By I.H. $|P| \to_{\beta_0} |P^*|$ which gives $M' = \lambda x.P' = \lambda x.|P| \to_{\beta_0} \lambda x|P^*| = |N|$.

If $M' = P'Q'$ and $P'Q'$ is not a indexed redex. Then M cannot be a whole redex because in that case $M' = |M|$ would be an indexed redex. So $M = c_i PQ$ for some $c_i \in \mathcal{C}$ with $P' = |P|$ and $Q' = |Q|$. Because $M \to_\beta N$ gives either $N = c_i P^* Q$ or $N = c_i P Q^*$ with either $P \to_\beta P^*$ or $Q \to_\beta Q^*$, the result follows by I.H.

If $M' = (\lambda_0 x.P')Q'$ then $M = (\lambda x.P)Q$ with $P' = |P|$ and $Q' = |Q|$. We have two possibilities. In the first, either $N = (\lambda x.P^*)Q$ or $N = (\lambda x.P)Q^*$ with either $P \to_\beta P^*$ or $Q \to_\beta Q^*$. In this case, by I.H. $P' \to_{\beta_0} |P^*|$ or $Q' \to_{\beta_0} |Q^*|$ so we get the result. In the second case $N = P[x := Q]$. But then $M' \to_{\beta_0} |N|$ because, by lemma 4.3, $|N| = |P|[x := |Q|] = P'[x := Q']$. □

Proposition 4.2 *Let $M, N \in \Lambda^{\mathcal{C}}$ and $M \to_\beta N$. Then $|M| \to_{\beta_0} |N|$.*

Proof. The proposition is a restatement of proposition 4.1 because $M \in [\,|M|\,]$. □

Proposition 4.3 *Let $M', N' \in \Lambda'$ and $M' \to_{\beta_0} N'$. Then, if $M \in [M']$ there exists $N \in [N']$ s.t. $M \to_\beta N$.*

Proof. By induction on M'.

If $M' = x$ then there is nothing to prove.

If $M' = \lambda x.P'$. Then the only possibility for M is to have the form $\lambda x.P$ with $|P| = P'$. But then $M' \to_{\beta_0} N'$ means that $N' = \lambda x.P^*$ with $P' \to_{\beta_0} P^*$. By I.H. there exists P_1 s.t. $P \to_\beta P_1$ which gives $M = \lambda x.P \to_\beta \lambda x.P_1 = N$ and $|N| = |\lambda x.P_1| = \lambda x.|P_1| = \lambda x.P^* = N'$.

If $M' = P'Q'$ and $P'Q'$ is not a indexed redex. Then M cannot be a whole redex because in that case $M' = |M|$ would be an indexed redex. So $M = c_i PQ$ for some $c_i \in \mathcal{C}$ with $P' = |P|$ and $Q' = |Q|$. Because $M' \to_{\beta_0} N'$ that gives $N' = P^*Q'$ or $P'Q^*$ with either $P' \to_{\beta_0} P^*$ or $Q' \to_{\beta_o} Q^*$. By I.H. there exists P_1 or Q_1 s.t. $|P_1| = P^*$, $|Q_1| = Q^*$ and $P \to_\beta P_1$ or $Q \to_\beta Q_1$. This gives $M = c_i PQ \to_\beta N = c_i P_1 Q$ or $c_i PQ_1$. But then $|N| = |P_1||Q|$ or $|P||Q_1|$ which is equal either to P^*Q' or $P'Q^*$, that is to N'.

If $M' = (\lambda_0 x.P')Q'$ then $M = (\lambda x.P)Q$ with $P' = |P|$ and $Q' = |Q|$. We have two possibilities. In the first, $N' = (\lambda_0 x.P^*)Q'$ or $(\lambda_0 x.P')Q^*$ and either $P' \to_{\beta_0} P^*$ or $Q' \to_{\beta_0} Q^*$. In this case, by I.H. there exist $P \to_\beta P_1$ or $Q \to_\beta Q_1$ and $|P_1| = P^*$,

$|Q_1| = Q^*$ so that $M \to_\beta N$ with either $N = (\lambda x.P_1)Q$ or $N = (\lambda x.P)Q_1$, in either case $|N| = N'$. In the second case, $N' = P'[x := Q']$. But then $M = (\lambda x.P)Q \to_\beta P[x := Q] = N$ and by lemma 4.3, $|N| = |P|[x := |Q|] = P'[x := Q'] = N'$. □

From the above propositions we get the following corollaries.

Corollary 4.1 (Lifting) *If $M', N' \in \Lambda'$ and $M' \twoheadrightarrow_{\beta_0} N'$ then for any $M \in [M']$ there exists $N \in [N']$ s.t. $M, N \in \Lambda^{\mathcal{C}}$ and $M \twoheadrightarrow_\beta N$. It is obvious that M can be chosen to be in $\Lambda^{\mathcal{C}!}$.*

Corollary 4.2 (Projecting) *If $M, N \in \Lambda^{\mathcal{C}}$ and $M \twoheadrightarrow_\beta N$ then $|M| \twoheadrightarrow_{\beta_0} |N|$.*

5. Embedding the developments into the simply typed lambda calculus

In the previous section we saw how we can embed (and simulate) the β_0-reduction in Λ' (system equivalent to the developments in Λ) into the β-reduction in the system $\Lambda^{\mathcal{C}}$, considered as a subsystem of the full λ-calculus Λ. In this section we show that this embedding can in fact be restricted to the simply typed λ-calculus. To this end we prove that the λ-terms in $\Lambda^{\mathcal{C}!}$ can be typed in the system $\lambda_\to$.

Proposition 5.1 *Let M be in $\Lambda^{\mathcal{C}!}$ and let $c_1, \ldots, c_n$ be the constants occurring in M ($c_i \in \mathcal{C}$). Let Γ be an arbitrary context which contains only declarations of variables in $\mathcal{V} \setminus \mathcal{C}$ [1] and in which every free variable x in M with $x \notin \mathcal{C}$ has a declaration. Then there exist types $\sigma_1, \ldots, \sigma_n, \sigma$ such that*

$$\Gamma, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash M : \sigma$$

Proof. By induction on M.

M is a variable. So $M = x$ and $x \notin \mathcal{C}$. By hypothesis, for some type σ, $x : \sigma \in \Gamma$. So $\Gamma \vdash x : \sigma$.

$M = \lambda x.N$. Because x is bound in M we can consider that x is not declared in Γ. Also $x \notin \mathcal{C}$. By I.H. for any type τ there exist types $\sigma_1, \ldots, \sigma_n, \rho$ s.t. $\Gamma, x : \tau, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash N : \rho$ from which $\Gamma, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash \lambda x.N : \tau \to \rho$.

$M = c_i PQ$. Since $M \in \Lambda^{\mathcal{C}!}$, c_i does not belong to either P or Q. For the same reason the constants occurring in P and Q form two disjoint sets. Let $c_{k_1}, \ldots, c_{k_l}$ be the distinct constants belonging to P and let $c_{m_1}, \ldots, c_{m_p}$ be those belonging to Q, i.e. $\{c_i, c_{k_1}, \ldots, c_{k_l}, c_{m_1}, \ldots, c_{m_p}\} = \{c_1, \ldots, c_n\}$. By induction hypothesis

$$\Gamma, c_{k_1} : \sigma_{k_1}, \ldots, c_{k_l} : \sigma_{k_l} \vdash P : \rho \quad \text{and} \quad \Gamma, c_{m_1} : \sigma_{m_1}, \ldots, c_{m_p} : \sigma_{m_p} \vdash Q : \tau$$

and if $\Gamma' = \{c_{k_1} : \sigma_{k_1}, \ldots, c_{k_l} : \sigma_{k_l}\}$ and $\Gamma'' = \{c_{m_1} : \sigma_{m_1}, \ldots, c_{m_p} : \sigma_{m_p}\}$ then by weakening[2] $\Gamma, c_i : \rho \to (\tau \to \sigma), \Gamma', \Gamma'' \vdash P : \rho$ and $\Gamma, c_i : \rho \to (\tau \to \sigma), \Gamma', \Gamma'' \vdash Q : \tau$ and $\Gamma, \Gamma', \Gamma'', c_i : \rho \to (\tau \to \sigma) \vdash c_i : \rho \to (\tau \to \sigma)$. By applying the typing rules $\Gamma, \Gamma', \Gamma'', c_i : \rho \to (\tau \to \sigma) \vdash c_i PQ : \sigma$, i.e.

$$\Gamma, c_i : \rho \to (\tau \to \sigma), c_{k_1} : \sigma_{k_1}, \ldots, c_{k_l} : \sigma_{k_l}, c_{m_1} : \sigma_{m_1}, \ldots, c_{m_p} : \sigma_{m_p} \vdash M : \sigma$$

[1] i.e. if $x : \sigma \in \Gamma$ then $x \notin \mathcal{C}$

[2] It is permitted to apply the weakening lemma 3.1 because c_i and every constant in Γ' and Γ'' are all different.

$M = (\lambda x.P)Q$. We may suppose that x is not declared in Γ. By I.H. (and possible weakenings as above because $M \in \Lambda^{\mathcal{C}!}$)

$$\Gamma, x : \rho, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash P : \sigma \text{ and } \Gamma, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash Q : \tau$$

But the first typing holds for arbitrary ρ so holds for $\rho = \tau$, that is $\Gamma, x : \tau, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash P : \sigma$ from which $\Gamma, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash \lambda x.P : \tau \to \sigma$ and finally $\Gamma, c_1 : \sigma_1, \ldots, c_n : \sigma_n \vdash (\lambda x.P)Q : \sigma$. □

Corollary 5.1 *Every term in $\Lambda^{\mathcal{C}!}$ is strongly normalizable and satisfies the Church-Rosser property.*

Proof. Use theorem 3.1. □

So now it is easy to show that the developments satisfy the properties of strong normalization and Church-Rosser.

Proposition 5.2 *(1) Every term in Λ' is strongly normalizable (with respect to β_0-reduction).*

(2) Every term in Λ' satisfies the Church-Rosser property.

Proof. (1) Let $M' \in \Lambda'$. We can arrange to find $M \in [M']$ with $M \in \Lambda^{\mathcal{C}!}$. By proposition 4.3 every infinite β_0-reduction $M' \to_{\beta_0} M'_1 \to_{\beta_0} \cdots$ would give an infinite reduction $M \to_\beta M_1 \to_\beta \cdots$ starting from M, a contradiction.

(2) Let $M' \in \Lambda'$ and let $M'_2 \ {}_{\beta_0}\!\!\twoheadleftarrow M' \twoheadrightarrow_{\beta_0} M'_1$. By corollary 4.1 we can choose $M \in \Lambda^{\mathcal{C}!}$ s.t. $|M| = M'$ and $M_2 \ {}_{\beta}\!\!\twoheadleftarrow M \twoheadrightarrow_\beta M_1$ ($|M_1| = M'_1$ and $|M_2| = M'_2$). Because M satisfies the Church-Rosser property there exists M_3 s.t. $M_2 \twoheadrightarrow_\beta M_3 \ {}_{\beta}\!\!\twoheadleftarrow M_1$. By corollary 4.2, $M'_2 = |M_2| \twoheadrightarrow_{\beta_0} |M_3| \ {}_{\beta_0}\!\!\twoheadleftarrow |M_1| = M'_1$. □

Remark 5.1 It is crucial that M in the proof of proposition 5.2 is chosen to be in $\Lambda^{\mathcal{C}!}$. In that way it acquires the properties of the terms in $\Lambda^{\mathcal{C}!}$, i.e. strong normalization and Church-Rosser.

Note that when we have a process of development, i.e. $M' \twoheadrightarrow_{\beta_0} N'$, then in order to simulate this process in $\Lambda^{\mathcal{C}}$ we choose a representative $M \in [M']$, so that $M \in \Lambda^{\mathcal{C}!}$. In that way a reduction $M \to_\beta M_1 \to_\beta \cdots \to_\beta M_n = N$ (with $N \in [N']$) simulates the development but M_i may not belong to $\Lambda^{\mathcal{C}!}$ anymore, as some of its constants may be multiplied through the reduction process. However even if M_i is not in $\Lambda^{\mathcal{C}!}$ it is still typed in $\lambda_\to$ because of the subject reduction property, so that the whole reduction is inside the simply typed lambda calculus.

References

1. Barendregt H. P., *The lambda calculus, its syntax and semantics*, North-Holland (1984).
2. Barendregt H. P., Lambda calculi with types, *Handbook of Logic in Computer Science*, vol. 2, Oxford University Press (1992).
3. Gandy R. O., Proofs of strong normalization, in: Hindley J. R. and Seldin J. P. (eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda calculus and Formalism*, Academic Press (1980), 457–477.
4. Ghilezan S., Generalized finiteness of developments in typed lambda calculi, *Journal of Automata, Languages and Combinatorics* 1 (1996), 247–257.
5. Ghilezan S. and Kunčak V., Confluence of Untyped Lambda Calculus via Simple Types, in: Restivo A., Ronchi Della Rocca S., and Roversi L. (eds.), *ICTCS 2001*, LNCS 2202 (2001), 38–49.
6. Koletsos G., Church-Rosser theorem for typed functional systems, *Journal of Symbolic Logic* 50 (1985), 782–790.
7. Koletsos G. and Stavrinos G., Church-Rosser theorem for conjunctive type systems, in: Kakas A. and Sinachopoulos A. (eds.), *Proceedings of the First Panhellenic Symposium on Logic*, University of Cyprus (1997), 25–37.
8. Koletsos G. and Stavrinos G., Church-Rosser property and intersection types, The Australasian Journal of Logic, Vol. 6 (2008), http://www.philosophy.unimelb.edu.au/ajl/.
9. Krivine J.-L., *Lambda-calcul, types et modèles*, Masson (1990).
10. Mitchell J., Type Systems for Programming Languages, in: Leeuven J. (ed.), *Handbook of Theoretical Computer Science*, Elsevier Science Publishers B. V. (1990), 365–458.
11. Statman R., Logical relations and the simply typed lambda calculus, *Information and Control* 65 (1985), 85–97.
12. Tait W., Intensional interpretations of functionals of finite type, *Journal of Symbolic Logic* 32 (1967), 198–212.

◇ George Koletsos
Department of Computer Science
National Technical University of Athens
Zografou, 15780 Athens, Greece
koletsos@math.ntua.gr

◇ George Stavrinos
M.P.L.A., Department of Mathematics
University of Athens
Zografou, 15784 Athens, Greece
g.stavrinos@math.ntua.gr