Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 140 (2005) 101–111
www.elsevier.com/locate/entcs

Why the Usual Candidates of Reducibility Do Not Work for the Symmetric $\lambda\mu$-calculus

René David

Equipe de Logique, Université de Savoie
73376 Le Bourget du Lac, France

Karim Nour

Equipe de Logique, Université de Savoie
73376 Le Bourget du Lac, France

Abstract

The symmetric $\lambda\mu$-calculus is the $\lambda\mu$-calculus introduced by Parigot in which the reduction rule μ', which is the symmetric of μ, is added. We give examples explaining why the technique using the usual candidates of reducibility does not work. We also prove a standardization theorem for this calculus.

Keywords: $\lambda\mu$-calculus, reducibility.

1 Introduction

Since it has been understood that the Curry-Howard isomorphism relating proofs and programs can be extended to classical logic, various systems have been introduced: the λ_c-calculus (Krivine [11]), the λ_{exn}-calculus (de Groote [6]), the $\lambda\mu$-calculus (Parigot [17]), the λ^{Sym}-calculus (Barbanera & Berardi [1]), the λ_Δ-calculus (Rehof & Sorensen [23]), the $\overline{\lambda}\mu\tilde{\mu}$-calculus (Curien & Herbelin [3]), ...

[1] Email: david@univ-savoie.fr
[2] Email: nour@univ-savoie.fr

doi:10.1016/j.entcs.2005.06.020

The first calculus which respects the intrinsic symmetry of classical logic is λ^{Sym}. It is somehow different from the previous calculi since the main connector is not the arrow as usual but the connectors *or* and *and*. The symmetry of the calculus comes from the de Morgan laws.

The second calculus respecting this symmetry has been $\overline{\lambda}\mu\tilde{\mu}$. The logical part is the (classical) sequent calculus instead of natural deduction.

Natural deduction is not, intrinsically, symmetric but Parigot has introduced the so called *Free deduction* [16] which is completely symmetric. The $\lambda\mu$-calculus comes from there. To get a confluent calculus he had, in his terminology, to fix the inputs on the left. To keep the symmetry, it is enough to keep the same terms and to add a new reduction rule (called the μ'-reduction) which is the symmetric rule of the μ-reduction and also corresponds to the elimination of a cut. We get then a symmetric calculus that is called the *symmetric $\lambda\mu$-calculus*.

The μ'-reduction has been considered by Parigot for the following reasons. The $\lambda\mu$-calculus (with the β-reduction and the μ-reduction) has good properties : confluence in the un-typed version, subject reduction and strong normalization in the typed calculus. But this system has, from a computer science point of view, a drawback: the unicity of the representation of data is lost. It is known that, in the λ-calculus, any term of type N (the usual type for the integers) is β-equivalent to a Church integer. This no more true in the $\lambda\mu$-calculus and we can find normal terms of type N that are not Church integers. Parigot has remarked that by adding the μ'-reduction and some simplification rules the unicity of the representation of data is recovered and subject reduction is preserved, at least for the simply typed system, even though the confluence is lost.

Barbanera & Berardi proved the strong normalization of the λ^{Sym}-calculus by using candidates of reducibility but, unlike the usual construction (for example for Girard's system F), the definition of the interpretation of a type needs a rather complex fix-point operation. Yamagata [24] has used the same technique to prove the strong normalization of the $\beta\mu\mu'$-reduction where the types are those of system F and Parigot, again using the same ideas, has extended Barbanera & Berardi's result to a logic with second order quantification.

The following property trivially holds in the $\lambda\mu$-calculus:
If $(\lambda xM\ N\ P_1...P_n) \triangleright^* (\lambda xM'\ N'\ P'_1...P'_n) \triangleright (M'[x := N']\ P'_1...P'_n)$, then we may start the reduction by reducing the β redex, i.e $(\lambda xM\ N\ P_1...P_n) \triangleright (M[x := N]\ P_1...P_n) \triangleright^* (M'[x := N']\ P'_1...P'_n)$. This point is the key in the proof of two results for this calculus:

(1) If N and $(M[x := N]\ P_1...P_n)$ are in SN, then so is $(\lambda xM\ N\ P_1...P_n)$. Similarly, if N and $(M[\alpha =_r N]\ P_1 ... P_n)$ are in SN, then so is $(\mu\alpha M\ N\ P_1 ... P_n)$. They are at the base of the proof of the strong normalization of the typed calculus.

(2) The standardization theorem.

Even though this result remains (trivially) true in the symmetric $\lambda\mu$-calculus and the standardization theorem still holds in this calculus, point (1) above is no more true. This simply comes from the fact that an infinite reduction of $(\lambda x M\ N)$ does not necessarily reduce the β redex (and similarly for $(\mu\alpha M\ N)$) since it can also reduce the μ' redex.

The other key point in the proof of the strong normalization of typed calculus is the following property which remains true in the symmetric $\lambda\mu$-calculus.

(3) If $M_1, ..., M_n$ are in SN, then so is $(x\ M_1 ...\ M_n)$.

This paper is organized as follows. Section 2 defines the symmetric $\lambda\mu$-calculus and its reduction rules. We give the proof of (3) in section 3. Section 4 gives the counter-examples for (1). Finally we prove the standardization theorem in section 5.

2 The symmetric $\lambda\mu$-calculus

The set (denoted as $\mathcal{T}$) of $\lambda\mu$-terms or simply terms is defined by the following grammar where $x, y, ...$ are λ-variables and $\alpha, \beta, ...$ are μ-variables:

$$\mathcal{T} ::= x \mid \lambda x\mathcal{T} \mid (\mathcal{T}\ \mathcal{T}) \mid \mu\alpha\mathcal{T} \mid (\alpha\ \mathcal{T})$$

Note that we adopt here a more liberal syntax (also called de Groote's calculus) than in the original calculus since we do not ask that a $\mu\alpha$ is immediately followed by a $(\beta\ M)$ (denoted $[\beta]M$ in Parigot's notation).

Even though this paper is only concerned with the un-typed calculus, the $\lambda\mu$-calculus comes from a Logic and, in particular, the μ-constructor comes from a logical rule. To help the reader un-familiar with it, we give below the typing and the reduction rules.

The types are those of the simply typed $\lambda\mu$-calculus i.e. are built from atomic formulas and the constant symbol $\perp$ with the connector $\rightarrow$. As usual $\neg A$ is an abbreviation for $A \rightarrow \perp$.

The typing rules are given by figure 1 below where Γ is a context, i.e. a set of declarations of the form $x : A$ and $\alpha : \neg A$ where x is a λ (or intuitionistic) variable, α is a μ (or classical) variable and A is a formula.

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

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x M : A \to B} \to_i \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B} \to_e$$

$$\frac{\Gamma, \alpha : \neg A \vdash M : \perp}{\Gamma \vdash \mu\alpha M : A} \perp_e \qquad \frac{\Gamma, \alpha : \neg A \vdash M : A}{\Gamma, \alpha : \neg A \vdash (\alpha\ M) : \perp} \perp_i$$

Figure 1.

Note that, here, we also have changed Parigot's notation but these typing rules are those of his classical natural deduction. Instead of writing

$$M : (A_1^{x_1}, ..., A_n^{x_n} \vdash B, C_1^{\alpha_1}, ..., C_m^{\alpha_m})$$

we have written

$$x_1 : A_1, ..., x_n : A_n, \alpha_1 : \neg C_1, ..., \alpha_m : \neg C_m \vdash M : B$$

The cut-elimination procedure corresponds to the reduction rules given below. There are three kinds of cuts.

- A *logical cut* occurs when the introduction of the connective $\to$ is immediately followed by its elimination. The corresponding reduction rule (denoted by β) is:

$$(\lambda x M\ N) \triangleright M[x := N]$$

- A *classical cut* occurs when $\perp_e$ appears as the left premiss of a $\to_e$. The corresponding reduction rule (denoted by μ) is:

$$(\mu\alpha M\ N) \triangleright \mu\alpha M[\alpha =_r N]$$

  where $M[\alpha =_r N]$ is obtained by replacing each sub-term of M of the form $(\alpha\ U)$ by $(\alpha\ (U\ N))$.

- A *symmetric classical cut* occurs when $\perp_e$ appears as the right premiss of a $\to_e$. The corresponding reduction rule (denoted by μ') is:

$$(M\ \mu\alpha N) \triangleright \mu\alpha N[\alpha =_l M]$$

  where $N[\alpha =_l M]$ is obtained by replacing each sub-term of N of the form $(\alpha\ U)$ by $(\alpha\ (M\ U))$.

Remark

It is shown in [17] that the $\beta\mu$-reduction is confluent but neither $\mu\mu'$ nor $\beta\mu'$ is. For example $(\mu\alpha x\, \mu\beta y)$ reduces both to $\mu\alpha x$ and to $\mu\beta y$. Similarly $(\lambda z x\ \mu\beta y)$ reduces both to x and to $\mu\beta y$.

The following property is straightforward.

Theorem 2.1 *If* $\Gamma \vdash M : A$ *and* $M \triangleright M'$ *then* $\Gamma \vdash M' : A$.

3 If $M_1, ..., M_n$ are in SN, then so is $(x\ M_1 ... \ M_n)$

The proofs are only sketched. More details can be found in [10] where an arithmetical proof of the strong normalization of the $\beta\mu\mu'$-reduction for the simply typed calculus is given.

Definition 3.1 • $cxty(M)$ is the number of symbols occurring in M.

- We denote by $N \leq M$ (resp. $N < M$) the fact that N is a sub-term (resp. a strict sub-term) of M.
- The reflexive and transitive closure of $\triangleright$ is denoted by $\triangleright^*$.
- If M is in SN i.e. M has no infinite reduction, $\eta(M)$ will denote the length of the longest reduction starting from M.
- We denote by $N \prec M$ the fact that $N \leq M'$ for some M' such that $M \triangleright^* M'$ and either $M \triangleright^+ M'$ or $N < M'$. We denote by $\preceq$ the reflexive closure of $\prec$.

Lemma 3.2 (i) *If* $(M\ N) \triangleright^* \lambda x P$, *then* $M \triangleright^* \lambda y M_1$ *and* $M_1[y := N] \triangleright^* \lambda x P$.

(ii) *If* $(M\ N) \triangleright^* \mu\alpha P$, *then either* ($M \triangleright^* \lambda y M_1$ *and* $M_1[y := N] \triangleright^* \mu\alpha P$) *or* ($M \triangleright^* \mu\alpha M_1$ *and* $M_1[\alpha =_r N] \triangleright^* P$) *or* ($N \triangleright^* \mu\alpha N_1$ *and* $N_1[\alpha =_l M] \triangleright^* P$).

Proof Easy. □

Lemma 3.3 *Assume* $M, N \in SN$ *and* $(M\ N) \notin SN$. *Then, either* ($M \triangleright^* \lambda y P$ *and* $P[y := N] \notin SN$) *or* ($M \triangleright^* \mu\alpha P$ *and* $P[\alpha =_r N] \notin SN$) *or* ($N \triangleright^* \mu\alpha P$ *and* $P[\alpha =_l M] \notin SN$).

Proof By induction on $\eta(M) + \eta(N)$. □

Lemma 3.4 *The term* $(x\ M_1\ ...\ M_n)$ *never reduces to a term of the form* $\lambda y M$.

Proof By induction on n. Use lemma 3.2. □

Definition 3.5 • Let $M_1, ..., M_n$ be terms and $1 \leq i \leq n$. We will denote by $M[\alpha =_i (M_1\ ...\ M_n)]$ the term M in which every sub-term of the form $(\alpha\ U)$ is replaced by $(\alpha\ (x\ M_1\ ...\ M_{i-1}\ U\ M_{i+1}\ ...\ M_n))$.

- We will denote by Σ_x the set of simultaneous substitutions of the form $[\alpha_1 =_{i_1} (M_1^1\ ...\ M_n^1), ..., \alpha_k =_{i_k} (M_1^k\ ...\ M_n^k)]$.

Lemma 3.6 *Assume* $(x\ M_1\ ...\ M_n) \triangleright^* \mu\alpha M$. *Then, there is an* i *such that* $M_i \triangleright^* \mu\alpha P$ *and* $P[\alpha =_i (M_1\ ...\ M_n)] \triangleright^* M$.

Proof By induction on n. Use lemmas 3.2 and 3.4. □

Lemma 3.7 *Assume $M_1, ..., M_n \in SN$ and $(x\ M_1\ ...\ M_n) \notin SN$. Then, there is an $1 \leq i \leq n$ such that $M_i \triangleright^* \mu\alpha\, U$ and $U[\alpha =_i (M_1\ ...\ M_n)] \notin SN$.*

Proof Let k be the least such that $(x\ M_1\ ...\ M_{k-1}) \in SN$ and $(x\ M_1\ ...\ M_k) \notin SN$. Use lemmas 3.3, 3.4 and 3.6. □

Lemma 3.8 *Let M be a term and $\sigma \in \Sigma_x$. If $M[\sigma] \triangleright^* \mu\alpha P$ (resp. $M[\sigma] \triangleright^* \lambda x P$) , then $M \triangleright^* \mu\alpha Q$ (resp. $M \triangleright^* \lambda x Q$) for some Q such that $Q[\sigma] \triangleright^* P$.*

Proof By induction on M. □

The next lemma is the key of the proof of theorem 3.10. Though intuitively clear (if the cause of non SN is the substitution $\delta =_i (P_1 ... P_n)$, this must come from some $(\delta\ M') \prec M$) its proof is rather technical.

Lemma 3.9 *Let M be a term and $\sigma \in \Sigma_x$. Assume δ is free in M but not free in $Im(\sigma)$. If $M[\sigma] \in SN$ but $M[\sigma][\delta =_i (P_1 ... P_n)] \notin SN$, there is $M' \prec M$ and σ' such that $M'[\sigma'] \in SN$ and $(x\ P_1 ... P_{i-1}\ M'[\sigma']\ P_{i+1} ... P_n) \notin SN$.*

Proof See [10] for more detail. □

Theorem 3.10 *Assume $M_1, ..., M_n$ are in SN. Then $(x\ M_1\ ...\ M_n) \in SN$.*

Proof We prove a more general result. Let $M_1, ..., M_n$ be terms and $\sigma_1, ..., \sigma_n$ be in Σ_x. If $M_1[\sigma_1]$, ..., $M_n[\sigma_n] \in SN$, then $(x\ M_1[\sigma_1]\ ...\ M_n[\sigma_n]) \in SN$. This is done by induction on $(\Sigma\eta(M_i), \Sigma cxty(M_i))$. Assume $(x\ M_1[\sigma_1]\ ...\ M_n[\sigma_n]) \notin SN$. By lemma 3.7, there is an i such that $M_i[\sigma_i] \triangleright^* \mu\alpha\, U$ and $U[\alpha =_i (M_1[\sigma_1]\ ...\ M_n[\sigma_n])] \notin SN$. By lemma 3.8, $M_i \triangleright^* \mu\alpha Q$ for some Q such that $Q[\sigma_i] \triangleright^* U$. Thus $Q[\sigma_i][\alpha =_i (M_1[\sigma_1]\ ...\ M_n[\sigma_n])] \notin SN$. By lemma 3.9, let $M' \prec Q \preceq M_i$ and σ' be such that $M'[\sigma'] \in SN$ and $(x\ M_1[\sigma_1] ... M_{i-1}[\sigma_{i-1}]\ M'[\sigma']\ M_{i+1}[\sigma_{i+1}] ... M_n[\sigma_n]) \notin SN$. This contradicts the induction hypothesis since $(\eta(M'), cxty(M')) < (\eta(M_i), cxty(M_i))$. □

4 The counter-examples

Definition 4.1 Let U and V be terms.

- $U \hookrightarrow V$ means that each reduction of U which is long enough must go through V, i.e. there is some n_0 such that, for all $n > n_0$, if $U = U_0 \triangleright U_1 \triangleright ... \triangleright U_n$ then $U_p = V$ for some p.
- $U \curvearrowright V$ means that U has only one redex and $U \triangleright V$.

Remark

It is easy to check that if $U \hookrightarrow V$ (resp. $U \curvearrowright V$) and $V \in SN$, then $U \in SN$.

Definition 4.2 • Let $M_0 = \lambda x(x\ P\ \underline{0})$ and $M_1 = \lambda x(x\ P\ \underline{1})$ where $\underline{0} = \lambda x\lambda yy$, $\underline{1} = \lambda x\lambda yx$, $P = \lambda x\lambda y\lambda z\ (y\ (z\ \underline{1}\ \underline{0})\ (z\ \underline{0}\ \underline{1})\ \lambda d\underline{1}\ \Delta\ \Delta)$ and $\Delta = \lambda x(x\ x)$.

- Let $M = \langle (x\ M_1), (x\ M_0)\rangle$, $M' = \langle (\beta\ \lambda x(x\ M_1)), (\beta\ \lambda x(x\ M_0))\rangle$ where $\langle T_1, T_0\rangle$ denotes the pair of terms, i.e. the term $\lambda f(f\ T_1\ T_0)$ where f is a fresh variable.
- Let $N = (\alpha\ \lambda z(\alpha\ z))$.

Lemma 4.3 (i) $(M_1\ M_0), (M_0\ M_1) \notin SN$.

(ii) $(M_0\ M_0), (M_1\ M_1) \in SN$.

Proof

(i) Assume $i \neq j$, then

$$\begin{aligned}(M_i\ M_j) &\triangleright^* (P\ P\ \underline{j}\ \underline{i})\\ &\triangleright^* (\underline{j}\ (\underline{i}\ \underline{1}\ \underline{0})\ (\underline{i}\ \underline{0}\ \underline{1})\ \lambda d\underline{1}\ \Delta\ \Delta)\\ &\triangleright^* (\underline{0}\ \lambda d\underline{1}\ \Delta\ \Delta)\\ &\triangleright^* (\Delta\ \Delta)\end{aligned}$$

and thus $(M_i\ M_j) \notin SN$.

(ii) It is easy to check that $(M_i\ M_i) \hookrightarrow (\underline{1}\ \lambda d\underline{1}\ \Delta\ \Delta) \curvearrowright (\lambda y\lambda d\underline{1}\ \Delta\ \Delta) \curvearrowright (\lambda d\underline{1}\ \Delta) \curvearrowright \underline{1}$. □

Proposition 4.4 *$M[x := \mu\alpha N] \in SN$ but $(\lambda xM\ \mu\alpha N) \notin SN$.*

Proof (a) Since $M[x := \mu\alpha N] = \langle (\mu\alpha N\ M_1), (\mu\alpha N\ M_0)\rangle$, by theorem 3.10, to show that $M[x := \mu\alpha N] \in SN$, it is enough to show that $(\mu\alpha N\ M_i) \in SN$.

$$\begin{aligned}(\mu\alpha N\ M_i) &\curvearrowright \mu\alpha(\alpha(\lambda z(\alpha\ (z\ M_i))M_i))\\ &\curvearrowright \mu\alpha(\alpha\ (\alpha\ (M_i\ M_i)))\\ &\hookrightarrow \mu\alpha(\alpha\ (\alpha\ \underline{1}))\end{aligned}$$

(b)

$$\begin{aligned}(\lambda xM\ \mu\alpha N) &\triangleright^* \mu\alpha(\alpha\ (\lambda xM\ \lambda z(\alpha\ (\lambda xM\ z))))\\ &\triangleright^* \mu\alpha(\alpha\ (\lambda xM\ \lambda z(\alpha\ \langle (z\ M_1), (z\ M_0)\rangle)))\\ &\triangleright^* \mu\alpha(\alpha\ \langle (\alpha\ \langle (M_1\ M_1), (M_1\ M_0)\rangle), (\alpha\ \langle (M_0\ M_1), (M_0\ M_0)\rangle)\rangle)\\ &\triangleright^* \mu\alpha(\alpha\ \langle (\alpha\ \langle \underline{1}, (\Delta\ \Delta)\rangle), (\alpha\ \langle \underline{1}, (\Delta\ \Delta)\rangle)\rangle)\end{aligned}$$

and thus $(\lambda xM\ \mu\alpha N) \notin SN$.

□

Proposition 4.5 *$M'[\beta =_r \mu\alpha N] \in SN$ but $(\mu\beta M' \ \mu\alpha N) \notin SN$.*

Proof (a) $(\lambda x(x \ M_i) \ \mu\alpha N)$ has two redexes thus either

$$\begin{aligned}(\lambda x(x \ M_i) \ \mu\alpha N) &\triangleright (\mu\alpha N \ M_i)\\ &\curvearrowright \mu\alpha(\alpha(\lambda z(\alpha \ (z \ M_i)) \ M_i))\\ &\curvearrowright \mu\alpha(\alpha \ (\alpha \ (M_i \ M_i)))\\ &\hookrightarrow \mu\alpha(\alpha \ (\alpha \ \underline{1}))\end{aligned}$$

or

$$\begin{aligned}(\lambda x(x \ M_i) \ \mu\alpha N) &\triangleright \mu\alpha(\alpha(\lambda x(x \ M_i) \ \lambda z(\alpha \ (\lambda x(x \ M_i) \ z))))\\ &\hookrightarrow \mu\alpha(\alpha \ (\alpha \ (M_i \ M_i)))\\ &\hookrightarrow \mu\alpha(\alpha \ (\alpha \ \underline{1}))\end{aligned}$$

Thus $(\lambda x(x \ M_i) \ \mu\alpha N) \hookrightarrow \mu\alpha(\alpha \ (\alpha \ \underline{1}))$ and, by theorem 3.10, it follows that $M'[x := \mu\alpha N] = \langle(\beta \ (\lambda x(x \ M_1) \ \mu\alpha N)), (\beta \ (\lambda x(x \ M_0) \ \mu\alpha N))\rangle \in SN$.

(b)

$$\begin{aligned}(\mu\beta M' \ \mu\alpha N) &\triangleright^* \mu\alpha(\alpha \ (\mu\beta M' \ \lambda z(\alpha \ (\mu\beta M' \ z))))\\ &\triangleright^* \mu\alpha(\alpha \ (\mu\beta M' \ \lambda z(\alpha \ \mu\beta\langle(\beta \ (z \ M_1)), (\beta \ (z \ M_0))\rangle)))\\ &\triangleright^* \mu\alpha(\alpha \ \mu\beta\langle(\beta \ (\alpha \ \mu\beta\langle(\beta \ \underline{1}), (\beta \ (\Delta \ \Delta))\rangle)),\\ &\qquad\qquad (\beta \ (\alpha \ \mu\beta\langle(\beta \ (\Delta \ \Delta)), (\beta \ \underline{1})\rangle))\rangle)\end{aligned}$$

and thus $(\mu\beta M' \ \mu\alpha N) \notin SN$.

□

5 Standardization

In this section we give a standardization theorem for the $\beta\mu\mu'$-reduction. It also holds for the $\mu\mu'$-reduction and its proof simply is a restriction of the other one.

Definition 5.1 (i) The sequence $(M_i)_{1 \le i \le n}$ is standard iff one of the following cases hold:

(a) For all i, $M_i = \lambda x N_i$ (resp. $M_i = \mu\alpha N_i$, $M_i = (x \ N_i)$, $M_i = (\alpha \ N_i)$) and the sequence $(N_i)_{1 \le i \le n}$ is standard

(b) There are standard sequences $(N_i)_{1 \le i \le k}$ and $(P_i)_{k \le i \le n}$ such that, for $1 \le i \le k$, $M_i = (N_i \ P_k)$ and, for $k \le i \le n$, $M_i = (N_k \ P_i)$.

(c) There is a standard sequence $(N_i)_{1 \le i \le k}$ and Q such that, either, for $1 \le i \le k$, $M_i = (N_i \ Q)$ and $N_k = \lambda x P$ and N_{k-1} does not begin with λ and $M_{k+1} = P[x := Q]$ and the sequence $(M_i)_{k+1 \le i \le n}$ is standard.

or, for $1 \leq i \leq k$, $M_i = (N_i\ Q)$ and $N_k = \mu\alpha P$ and N_{k-1} does not begin with μ and $M_{k+1} = P[\alpha =_r Q]$ and the sequence $(M_i)_{k+1 \leq i \leq n}$ is standard.

or, for $1 \leq i \leq k$, $M_i = (Q\ N_i)$ and $N_k = \mu\beta P$ and N_{k-1} does not begin with μ and $M_{k+1} = P[\beta =_l Q]$ and the sequence $(M_i)_{k+1 \leq i \leq n}$ is standard.

(ii) $M \triangleright_{st} M'$ iff there is a standard sequence $(M_i)_{1 \leq i \leq n}$ such that $M = M_1$ and $M' = M_n$.

Remarks and notation

- The clauses in 1 above correspond to a definition by induction on the ordered pair $(n, cxty(M_1))$.
- It is easy to check that, restricted to the λ-calculus, this definition is equivalent to the usual definition of a standard reduction.
- Clearly, if $M \triangleright_{st} M'$ then $M \triangleright^* M'$. In this case, we will denote the length of the reduction by $lg(M \triangleright_{st} M')$.

Lemma 5.2 *Assume $M \triangleright_{st} P$ and $N \triangleright_{st} Q$. Then : (a) $\mu\alpha M \triangleright_{st} \mu\alpha P$, (b) $\lambda x M \triangleright_{st} \lambda x P$, (c) $(M\ N) \triangleright_{st} (P\ Q)$, (d) $M[x := N] \triangleright_{st} P[x := Q]$ and (e) for $j \in \{l, r\}$, $M[\alpha =_j N] \triangleright_{st} P[\alpha =_j Q]$.*

Proof (a), (b) and (c) are immediate. (d) and (e) are proved by induction on $(lg(M \triangleright_{st} P), cxty(M))$ and a straightforward case analysis on the definition of a standard sequence bringing from M to P. □

Lemma 5.3 *Assume $M \triangleright_{st} P$ and $P \triangleright Q$. Then $M \triangleright_{st} Q$.*

Proof This is proved by induction on $(lg(M \triangleright_{st} P), cxty(M))$ and by case analysis on the reduction $M \triangleright_{st} P$. The only case which is not immediate is the following: $M = (M_1\ M_2) \triangleright^* (N_1\ M_2) \triangleright^* (N_1\ N_2) = P$ where $M_1 \triangleright_{st} N_1$ and $M_2 \triangleright_{st} N_2$. If the redex reduced in $P \triangleright Q$ is in N_1 or N_2 the result follows immediately from the induction hypothesis. Otherwise, assume, for example that $N_1 = \mu\alpha R$ and $Q = R[\alpha =_r N_2]$. Let the reduction $M_1 \triangleright_{st} N_1$ be as follows: $M_1 \triangleright_{st} \mu\alpha R_1 \triangleright_{st} \mu\alpha R$ where $\mu\alpha R_1$ is the first term in the reduction that begins with μ. It follows then from lemma 5.2 that the following reduction is standard. $M = (M_1\ M_2) \triangleright_{st} (\mu\alpha R_1\ M_2) \triangleright \mu\alpha R_1[\alpha =_r M_2] \triangleright_{st} \mu\alpha R[\alpha =_r N_2]$. □

Theorem 5.4 *Assume $M \triangleright^* P$. Then $M \triangleright_{st} P$.*

Proof By induction on the length of the reduction $M \triangleright^* M_1$. The result follows immediately from lemma 5.3. □

References

[1] F. Barbanera and S. Berardi. *A symmetric lambda-calculus for classical program extraction.* In M. Hagiya and J.C. Mitchell, editors, Proceedings of theoretical aspects of computer software, TACS'94. LNCS (789), pp. 495-515. Springer Verlag, 1994.

[2] R. Constable and C. Murthy. *Finding computational content in classical proofs.* In G. Huet and G. Plotkin, editors, Logical Frameworks, pp. 341-362, Cambridge University Press, 1991.

[3] P.L. Curien and H. Herbelin. *The duality of computation.* Proc. International Conference on Functional Programming, September 2000, Montral, IEEE, 2000.

[4] J.-Y. Girard. *A new constructive logic: classical logic.* MSCS (1), pp. 255-296, 1991.

[5] P. de Groote. *A CPS-translation of the lambda-mu-calculus.* In S. Tison, editor, 19th International Colloquium on Trees in Algebra and Programming, CAAP'94, volume 787 of Lecture Notes in Computer Science, pp. 85-99. Springer, 1994.

[6] P. de Groote. *A simple calculus of exception handling.* In M. Dezani and G. Plotkin, editors, Second International Conference on Typed Lambda Calculi and Applications, TLCA'95, volume 902 of Lecture Notes in Computer Science, pp. 201-215. Springer, 1995.

[7] R. David. *Normalization without reducibility.* Annals of Pure and Applied Logic (107), pp. 121-130, 2001.

[8] R. David and K. Nour. *A short proof of the strong normalization of the simply typed lambda mu calculus.* Schedae Informaticae n12, pp. 27-34, 2003.

[9] R. David and K. Nour. *A short proof of the strong normalization of classical natural deduction with disjunction.* The Journal of Symbolic Logic n 68.4, pp. 1277-1288, 2003.

[10] R. David and K. Nour. *Arithmetical proofs of strong normalization results for the symmetric $\lambda\mu$-calculus.* To appear in TLCA'05.

[11] J.-L. Krivine. *Classical logic, storage operators and 2nd order lambda-calculus.* Annals of Pure and Applied Logic (68), pp. 53-78, 1994.

[12] C.R. Murthy. *An evaluation semantics for classical proofs.* In Proceedings of the sixth annual IEEE symposium on logic in computer science, pp. 96-107, 1991.

[13] K. Nour. *La valeur d'un entier classique en $\lambda\mu$-calcul.* Archive for Mathematical Logic (36), pp. 461-471, 1997.

[14] K. Nour. *A non-deterministic classical logic (the $\lambda\mu^{++}$-calculus).* Mathematical Logic Quarterly (48), pp. 357-366, 2002.

[15] K. Nour and K. Saber. *A semantical proof of the strong normalization theorem of full propositionnal classical natural deduction.* Manuscript 2004.

[16] M. Parigot. *Free Deduction: An analysis of "computations" in classical logic.* Proceedings. Lecture Notes in Computer Science, Vol. 592, Springer, pp. 361-380, 1992.

[17] M. Parigot. *$\lambda\mu$-calculus: An algorithm interpretation of classical natural deduction.* Lecture Notes in Artificial Intelligence (624), pp. 190-201. Springer Verlag, 1992.

[18] M. Parigot. *Strong normalization for second order classical natural deduction.* In Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science, pp. 39-46, Montreal, Canada, 19–23 June 1993. IEEE Computer Society Press.

[19] M. Parigot. *Classical proofs as programs.* In G. Gottlob, A. Leitsch, and D. Mundici, eds., Proc. of 3rd Kurt Godel Colloquium, KGC'93, vol. 713 of Lecture Notes in Computer Science, pp. 263-276. Springer-Verlag, 1993.

[20] M. Parigot. *Proofs of strong normalization for second order classical natural deduction.* Journal of Symbolic Logic, 62 (4), pp. 1461-1479, 1997.

[21] E. Polonovsky. *Substitutions explicites, logique et normalisation.* PhD thesis, Paris 7, 2004.

[22] W. Py. *Confluence en $\lambda\mu$-calcul.* PhD thesis, University of Chambéry, 1998.

[23] N.J. Rehof and M.H. Sorensen. *The λ_Δ-calculus.* In M. Hagiya and J.C. Mitchell, editors, Proceedings of the international symposium on theoretical aspects of computer software, TACS'94, LNCS (789), pp. 516-542. Springer Verlag, 1994.

[24] Y. Yamagata. *Strong normalization of second order symmetric lambda-mu calculus.* TACS 2001, Lecture Notes in Computer Science 2215, pp. 459-467, 2001.