On a theorem concerning the uniqeness of formal proofs

ANDOU, Yuuki*
Hosei University, Tokyo 102-8160, Japan.

October 20, 1999
(AC99 at Tokyo Metropolitan University)

Abstract

Confluency of a system of formal proof is showed. The system corresponds with classical logic and formalized in an extended typed lambda terms which contains full logical connectives in the construction of types.

1 Introduction

In this paper, we investigate about the uniqueness of proofs, not about the uniqueness proof. The notions used in a formal system, such as formulas and proofs, are strictly defined as sequences of symbols. Therefore, we can recognize whether given two proofs are identical in a formal system. The main theorem of this paper is the confluency, that is Church-Rosser property, of a formal system. This therem leads the fact that if a proof is given and if it has a normal proof in the sence of some defined reductions, we have the uniq normal proof for the given one. In the words of calculation, this fact means that the calculation corresponds with our reduction has uniq value. Historically, confluency is proved for many formal systems. The system we treat here is an extended typed-lambda calculus which corresponds with first order classical natural deduction, with one conclusion as same as Gentzen's original system, including the logical connectives for disjunction and existential quantifier. If the system contains these two logical connectives, we meet a difficulty to have the confluency even in the intuitionistic logic. This difficulty is caused by structural aspects of the reductions.

Now we mention about the construction of this paper. First we define our typed term system and our reductions. Next we define some notions used in the proof of our theorem. Then we state our theorem and show the outline of the proof of the theorem. The details of the proof are omitted. These can be seen in our paper([4]).

2 λ^C-calculus

In this section, we define the system of λ^C-calculus which is an extension of typed λ-calculus in Church-style.

*E-mail: `norakuro@i.hosei.ac.jp`

2.1 Typed-terms

Types are formulas of a first order language $\mathcal{L}$ which contains $\supset$, $\wedge$, $\vee$, $\forall$, and $\exists$ as logical symbols, and $\bot$ as a propositional constant. We use $\neg A$ as an abbreviation of the formula $A \supset \bot$. Two formulas are identical if they vary only by bound variables. We use the following letters, possibly with sub- or superscripts, as metavariables.

- a, b, c : for individual variables of $\mathcal{L}$.
- p, q : for $\mathcal{L}$-terms.
- A, B, C, D : for $\mathcal{L}$-formulas.
- x, y, z : for λ^C-variables.
- r, s, t, u, v, w : for λ^C-terms.

There are assumed to be uncountably many λ^C-variables. *Typed λ^C-terms* and their *free λ^C-variables* are defined as follows, where the set of all free λ^C-variables of a λ^C-term t is denoted by $FV(t)$:

- If x is a λ^C-variable and A a $\mathcal{L}$-formula, then (x^A) is a λ^C-term of type A, and $FV((x^A)) = \{x^A\}$. Moreover, we call the term (x^A) an *axiom-term*.
- By the following schemata, we explain other constructions of terms. A schema of the form

$$\frac{\begin{array}{ccc}[x_1^{A_1}] & & [x_n^{A_n}] \\ t_1 : B_1 & \cdots & t_n : B_n\end{array}}{u : C}\ (*)$$

  means that: if t_i is a λ^C-term of type B_i for each i, then u is a λ^C-term of type C, and $FV(u) = \bigcup_{1\le i\le n}(FV(t_i) \setminus \{x_i^{A_i}\})$. Moreover, we call u a *(*)-term.*

$$\frac{\begin{array}{c}[x^A] \\ t : B\end{array}}{(\lambda x^A.t) : A \supset B}\ (\supset I)_, \quad \frac{t : A \supset B \quad u : A}{t(u) : B}\ (\supset E)_,$$

$$\frac{t : A \quad u : B}{\langle t, u\rangle : A \wedge B}\ (\wedge I)_, \quad \frac{t : A \wedge B}{t(\pi 0) : A}\ (\wedge E_0)_, \quad \frac{t : A \wedge B}{t(\pi 1) : B}\ (\wedge E_1)_,$$

$$\frac{t : A}{\{t, B\} : A \vee B}\ (\vee I_0)_, \quad \frac{t : B}{\{A, t\} : A \vee B}\ (\vee I_1)_, \quad \frac{t : A \vee B \quad \begin{array}{c}[x^A] \\ u : C\end{array} \quad \begin{array}{c}[y^B] \\ v : C\end{array}}{t(\lambda^\vee x^A.u, y^B.v) : C}\ (\vee E)_,$$

$$\frac{t : A}{(\lambda^\forall a.t) : \forall a A}\ (\forall I)_, \quad \frac{t : \forall a A}{t(p) : A[p/a]}\ (\forall E)_,$$

$$\frac{t : A[p/a]}{(\sigma_A^{p,a} t) : \exists a A}\ (\exists I)_, \quad \frac{t : \exists a A \quad \begin{array}{c}[x^A] \\ u : C\end{array}}{t(\lambda^\exists x^A.u) : C}\ (\exists E)_,$$

$$\frac{\begin{array}{c}[x^{\neg C}]\\ t:\bot\end{array}}{(\lambda^{\bot}x^{\neg C}.t):C}\ (\bot_c).$$

In the schemata above, $A[p/a]$ represents the type obtained from A by substituting p for all free occurrences of a in A. The constructions $(\forall I)$ and $(\exists E)$ are subject to the usual eigenvariable conditions for $\mathcal{L}$-variables. That is, in $(\forall I)$, the $\mathcal{L}$-variable a does not occur in the types of any free λ^C-variables in t, and in $(\exists E)$, the λ^C-variable a does not occur in the types of any free λ^C-variables in u except x^A, nor in C.

Note that any λ^C-term ends with a right bracket), $\rangle$, or } as a sequence of letters.

Eliminator. In each of the schemata $(\supset E)$, $(\wedge E)$, $(\vee E)$, $(\forall E)$, and $(\exists E)$, the newly constructed term is represented in the form $t\varepsilon$. We call this expression ε an *eliminator* from D_0 to D_1 where D_0 and D_1 are the type of t and $t\varepsilon$ respectively.

Note that an eliminator is a sequence of letters which starts with a left round bracket and ends with a right round bracket. Moreover, an eliminator itself is not a λ^C-term.

2.2 $\lambda^{\bot}$-regularity

An axiom term $(x^{\neg C})$ is $\lambda^{\bot}$-*nice* in a term t iff for every occurrence of $(x^{\neg C})$ in t there exists a term u of type C such that the occurrence of $(x^{\neg C})$ is in the form $(x^{\neg C})(u)$ as a subterm of t and also the occurrence of $(x^{\neg C})$ is not bound in t. A term w is $\lambda^{\bot}$-*regular* iff for every subterm $(\lambda^{\bot}x^{\neg C}.t)$ in w, $(x^{\neg C})$ is $\lambda^{\bot}$-nice in t.

To simplify the representation of the reduction, we assume that every λ^C-term is $\lambda^{\bot}$-regular. This causes no loss of generality, because a non-regular $\lambda^{\bot}$-abstraction can be transformed to a regular one as follows. If an occurrence $(x^{\neg C})$ violates the regularity of the $\lambda^{\bot}$-abstraction $(\lambda^{\bot}x^{\neg C}.t)$, then replace the occurrence $(x^{\neg C})$ by $(\lambda y^C.(x^{\neg C})(y^C))$ using a new variable y.

The reduction in the λ^C-calculus will be defined in the next subsection and will have the property that the contractum of any $\lambda^{\bot}$-regular λ^C-term is also $\lambda^{\bot}$-regular.

For a term $(\lambda^{\bot}x^{\neg C}.t)$, we use a notation for brackets as follows: for every occurrence of subterms of the form $(x^{\neg C})(u)$ in t, the right-most bracket of the subterm is indexed by x, that is, the subterm is denoted by $(x^{\neg C})(u)_x$.

2.3 Reductions

Now we define two kinds of contractions of λ^C-terms as follows. One is called *essential* and the other *structural*. They are denoted by $\rhd_e$ and $\rhd_s$ respectively.

- $(\lambda x^A.t)(u) \rhd_e t[u/x^A]$
- $\langle t,u\rangle(\pi 0) \rhd_e t,\quad \langle t,u\rangle(\pi 1) \rhd_e u$
- $\{t,B\}(\lambda^{\vee}x^A.u, y^B.v) \rhd_e u[t/x^A],\quad \{A,t\}(\lambda^{\vee}x^A.u, y^B.v) \rhd_e v[t/y^B]$
- $(\lambda^{\forall}a.t)(p) \rhd_e t[p/a]$
- $(\sigma_A^{p,a}t)(\lambda^{\exists}x^A.u) \rhd_e u[p/a;t/x^A]$

- $t(\lambda^{\vee} x^A.u, y^B.v)\varepsilon \rhd_s t(\lambda^{\vee} x^A.u\varepsilon, y^B.v\varepsilon)$
- $t(\lambda^{\exists} x^A.u)\varepsilon \rhd_s t(\lambda^{\exists} x^A.u\varepsilon)$
- $(\lambda^{\perp} x^{\neg C}.t)\varepsilon \rhd_s (\lambda^{\perp} x^{\neg D}.t[\ \varepsilon)_x/\ )_x, x^{\neg D}/x^{\neg C}])$

In the schemata above, ε represents an eliminator from C to D. The expression $t[u/x^A]$ represents the term obtained from t by substituting u for all occurrences (x^A) in t. The expression $t[\ \varepsilon)_x/\ )_x, x^{\neg D}/x^{\neg C}]$ represents the term obtained from t by substituting $\ \varepsilon)_x$ for all occurrences $\ )_x$ and $x^{\neg D}$ for all occurrences $x^{\neg C}$ in t simultaneously. The expression $t[p/a]$ represents the term obtained from t by substituting p for all occurrences a in t. Moreover, $u[p/a; t/x^A]$ is the abbreviation of $u[p/a][t/x^{A[p/a]}]$.

We denote $t \succ s$ iff s is obtained from t by replacing an occurrence of a $\rhd_e$- or $\rhd_s$-redex by its contractum, also the relation $\gg$ is defined as the reflexive and transitive closure of $\succ$.

3 Parallel reduction and theorems.

3.1 Extended structural reduction.

We prepare some notions for the explanation of the parallel reduction.

Segment-successor. For a term w, a binary relation denoted by $[\mathrm{ss}]_w$ on occurrences of subterms in w is defined as follows:

- $t(\lambda^{\vee} x^A.u, y^B.v)\ [\mathrm{ss}]_w\ u$
- $t(\lambda^{\vee} x^A.u, y^B.v)\ [\mathrm{ss}]_w\ v$
- $t(\lambda^{\exists} x^A.u)\ [\mathrm{ss}]_w\ u$
- $(\lambda^{\perp} x^{\neg C}.t)\ [\mathrm{ss}]_w\ u$, where u occurs in t in the form $(x^{\neg C})(u)_x$.

If $t\ [\mathrm{ss}]_w\ s$ holds, we call s a *segment-successor* of t, and t the *segment-predecessor* of s. Note that if $t\ [\mathrm{ss}]_w\ s$ holds, then the types of t and s are identical.

We denote the reflexive and transitive closure of $[\mathrm{ss}]_w$ by $\sqsupseteq_w$.

Segment-tree. Let r be an occurrence of a subterm in a term w, and suppose r is a $(\vee E)$-, $(\exists E)$-, or $(\perp_c)$-term and has no segment-predecessor in w. A *segment-tree* from r in w is a set, say $\mathcal{T}$, of occurrences of subterms in w such that, for every $t \in \mathcal{T}$,

- t is a $(\vee E)$-, $(\exists E)$-, or $(\perp_c)$-term,
- $r \sqsupseteq_w t$,
- For every occurrence s of a subterm in w, $r \sqsupseteq_w s \sqsupseteq_w t$ implies $s \in \mathcal{T}$.

Therefore, a partial ordered set $(\mathcal{T}, \sqsupseteq_w)$ is a tree with root r in the ordinary sense. Note that all elements of a segment-tree have the same type. We call such type the type of the segment-tree, and denote the fact that $\mathcal{T}$ has the type C by $\mathcal{T}^C$.

Let $\mathcal{T}$ be a segment-tree from r in w. An occurrence of a subterm v in w is called an *accepter* of $\mathcal{T}$ iff v is a segment-successor of an element of $\mathcal{T}$ and v is not in $\mathcal{T}$. We denote accepters of $\mathcal{T}$ by indexing the letter $\mathcal{T}$ with subscripts to its the right-most bracket. If $(\lambda^{\perp} x^{\neg C}.t) \in \mathcal{T}$, then the represented occurrence of the symbol $\lambda^{\perp}$ is called a *carrier* of $\mathcal{T}$. We denote carriers of $\mathcal{T}$ by indexing the letter $\mathcal{T}$ with subscripts to them.

If a set $\mathcal{A}$ of occurrences of subterms in a term w and a set $\mathcal{C}$ of occurrences of the symbol $\lambda^{\perp}$ in w are given, it is possible to decide whether there exists a segment-tree $\mathcal{T}$ in w such that the set of all accepters of $\mathcal{T}$ equals to $\mathcal{A}$ and the set of all carriers of $\mathcal{T}$ equals to $\mathcal{C}$. Moreover, if such a $\mathcal{T}$ exists, it is uniquely determined. So, we describe a segment-tree in a term by the set of subscripts to some occurrences of right brackets and the symbol $\lambda^{\perp}$.

A segment-tree $\mathcal{T}$ from r in w is called the *maximal segment-tree* from r in w iff no accepter of $\mathcal{T}$ has a segment-successor in w.

Corresponding segment-tree. Let $\mathcal{T}$ be a segment-tree in w, therefore, let w have indexes $\mathcal{T}$ at some occurrences of brackets and the symbol $\lambda^{\perp}$. Suppose the root of $\mathcal{T}$ is w itself and $w \succ w'$. Then we can define a segment-tree from w' in w' or empty-set which canonically corresponds with $\mathcal{T}$ in w according to the transformation of occurrences of brackets and the symbol $\lambda^{\perp}$ from the redex to contractum. Precisely, at the contraction schemata in the previous section: (1) indexes in t, u, v, and ε are transformed into the contractum with the symbol they are attached to, and (2) if the the right-most brackets of the essential redex are indexed (in the case of structural redex, being subject to the case (1)), then the right-most brackets of contractum will be newly indexed.

Note that the corresponding segment-tree may be streched or shortened as its length and may be increased or decreased as the numbers of the branches at its nodes. Moreover, a segment-tree may correspond with empty-set, in that case only index for an accepter remained without its predecessor.

The segment-tree in w' which corresponds with a segment-tree $\mathcal{T}$ in w provided that $w \succ w'$, may be represented also by $\mathcal{T}$ if it does not cause any confusion. This denotation is matched with the representation of segment-tree by the indexes and their transformation with the symbols.

Extended structural reduction. Let $\mathcal{T}$ be a segment-tree from r in w, and suppose r occurs in the form $r\varepsilon$ with an eliminator ε from C to D. The *extended structural reduction* of w along $\mathcal{T}$ is defined as the transformation to the term, say *w'*, which obtained from w, by replacing all indexed brackets $\rfloor_{\mathcal{T}}$ by $\rfloor\varepsilon$ where $\rfloor$ is one of the brackets $)$, $\rangle$, or $\}$, and by replacing all occurrences of $\neg C$ which are superscripts of λ^{C}-variables bounded by carriers of $\mathcal{T}$ by $\neg D$. We denote this reduction by $w \succ_{\mathcal{T}} w'$.

The following facts hold by definition.

Fact. *Every structural reduction is also an extended structural reduction. Actually, it is the special case that the length of the segment-tree is exactly one, i.e. the segment-tree consists only of its root.*

Fact. *If $w \succ_{\mathcal{T}} w'$, then there exists an eliminator ε and a reduction sequence $w_0 \succ \cdots \succ w_n$ such that, $w \equiv w_0$, $w' \equiv w_n$, and for every i, the reduction $w_{i-1} \succ w_i$ is caused by a structural contraction whose redex is of the form $v\varepsilon$ for some v.*

3.2 Parallel reduction.

In this sub-section we prove the Church-Rosser theorem for the reduction $\succ$. We shall use the extended method of parallel reduction (Takahashi [15]).

Definition (Parallel reduction). A binary relation on λ^C-terms called parallel reduction and denoted by $\succ_p$ is defined inductively by the schemata below.

- $(x^A) \succ_p (x^A)$.
- If $t \succ_p t'$, $u \succ_p u'$, and $v \succ_p v'$, then

  $(\lambda x^A.t) \succ_p (\lambda x^A.t')$, $\quad t(u) \succ_p t'(u')$,
  $\langle t, u\rangle \succ_p \langle t', u'\rangle$, $\quad t(\pi 0) \succ_p t'(\pi 0)$, $\quad t(\pi 1) \succ_p t'(\pi 1)$,
  $\{t, B\} \succ_p \{t', B\}$, $\quad \{A, t\} \succ_p \{A, t'\}$, $\quad t(\lambda^\vee x^A.u, y^B.v) \succ_p t'(\lambda^\vee x^A.u', y^B.v')$,
  $(\lambda^\forall a.t) \succ_p (\lambda^\forall a.t')$, $\quad t(p) \succ_p t'(p)$,
  $(\sigma_A^{p,a} t) \succ_p (\sigma_A^{p,a} t')$, $\quad t(\lambda^\exists x^A.u) \succ_p t'(\lambda^\exists x^A.u')$, and
  $(\lambda^\perp x^{\neg C}.t) \succ_p (\lambda^\perp x^{\neg C}.t')$.

- If $t \succ_p t'$, $u \succ_p u'$, and $v \succ_p v'$, then

  $(\lambda x^A.t)(u) \succ_p t'[u'/x^A]$,
  $\langle t, u\rangle(\pi 0) \succ_p t'$, $\quad \langle t, u\rangle(\pi 1) \succ_p u'$,
  $\{t, B\}(\lambda^\vee x^A.u, y^B.v) \succ_p u'[t'/x^A]$, $\quad \{A, t\}(\lambda^\vee x^A.u, y^B.v) \succ_p v'[t'/y^B]$,
  $(\lambda^\forall a.t)(p) \succ_p t'[p/a]$, and
  $(\sigma_A^{p,a} t)(\lambda^\exists x^A.u) \succ_p u'[p/a; t'/x^A]$.

- If $t \succ_p t'$ and $\varepsilon \mathrel{\widetilde{\succ_p}} \varepsilon'$, and $\mathcal{T}$ is a segment-tree from t in $t\varepsilon$, and $t'\varepsilon' \succ_{\mathcal{T}} w$, then $t\varepsilon \succ_p w$,

  where $\varepsilon \mathrel{\widetilde{\succ_p}} \varepsilon'$ means that:

  (a) $\varepsilon \equiv \varepsilon' \equiv (\pi 0)$, $(\pi 1)$, or (p), or
  (b) $[\varepsilon \equiv (u)$ and $\varepsilon' \equiv (u')]$, $[\varepsilon \equiv (\lambda^\vee x^A.u, y^B.v)$ and $\varepsilon' \equiv (\lambda^\vee x^A.u', y^B.v')]$, or $[\varepsilon \equiv (\lambda^\exists x^A.u)$ and $\varepsilon' \equiv (\lambda^\exists x^A.u')]$,

  provided that $u \succ_p u'$, and $v \succ_p v'$.

Note that the definition of $\succ_p$ above is well-defined because at each step it keeps the property that if $t \succ_p t'$ then $t \gg t'$. Under this condition, in the case that $t \succ_p t'$ and $\mathcal{T}$ is a segment-tree from t in t, we can construct a corresponding object of $\mathcal{T}$ in t'.

We will use the notation $\widetilde{\succ_p}$ above also in the remaining part of this paper.

To prove the diamond-property of $\succ_p$, first we define a transformation between terms, denoted by complete development.

Definition (Complete development). For a term t, we define the term denoted by t^* inductively as follows.

- $(x^A)^* \equiv (x^A)$.
- $(\lambda x^A.t)^* \equiv (\lambda x^A.t^*)$, $\quad t(u)^* \equiv t^*(u^*)$ if $t(u)$ is not a redex.
- $\langle t, u\rangle^* \equiv \langle t^*, u^*\rangle$, $\quad t(\pi 0)^* \equiv t^*(\pi 0)$ if $t(\pi 0)$ is not a redex, $\quad t(\pi 1)^* \equiv t^*(\pi 1)$ if $t(\pi 1)$ is not a redex.
- $\{t, B\}^* \equiv \{t^*, B\}$, $\quad \{A, t\}^* \equiv \{A, t^*\}$, $\quad t(\lambda^\vee x^A.u, y^B.v)^* \equiv t^*(\lambda^\vee x^A.u^*, y^B.v^*)$ if $t(\lambda^\vee x^A.u, y^B.v)$ is not a redex.
- $(\lambda^\forall a.t)^* \equiv (\lambda^\forall a.t^*)$, $\quad t(p)^* \equiv t^*(p)$ if $t(p)$ is not a redex.
- $(\sigma_A^{p,a} t)^* \equiv (\sigma_A^{p,a} t^*)$, $\quad t(\lambda^\exists x^A.u)^* \equiv t^*(\lambda^\exists x^A.u^*)$ if $t(\lambda^\exists x^A.u)$ is not a redex.
- $(\lambda^\perp x^{\neg C}.t)^* \equiv (\lambda^\perp x^{\neg C}.t^*)$.
- $(\lambda x^A.t)(u)^* \equiv t^*[u^*/x^A]$.
- $\langle t, u\rangle(\pi 0)^* \equiv t^*$, $\quad \langle t, u\rangle(\pi 1)^* \equiv u^*$.
- $\{t, B\}(\lambda^\vee x^A.u, y^B.v)^* \equiv u^*[t^*/x^A]$, $\quad \{A, t\}(\lambda^\vee x^A.u, y^B.v)^* \equiv v^*[t^*/y^B]$.
- $(\lambda^\forall a.t)(p)^* \equiv t^*[p/a]$.
- $(\sigma_A^{p,a} t)(\lambda^\exists x^A.u)^* \equiv u^*[p/a; t^*/x^A]$.
- Let $\mathcal{T}_m$ be the maximal segment-tree from t in $t\varepsilon$, and if $t^*\widetilde{\varepsilon^*} \succ_{\mathcal{T}_m} w$, then $t\varepsilon^* \equiv w$,

  where $\widetilde{\varepsilon^*}$ denotes:

  (a) ε if $\varepsilon \equiv (\pi 0)$, $(\pi 1)$, or (p).
  (b) (u^*) if $\varepsilon \equiv (u)$.
  (c) $(\lambda^\vee x^A.u^*, y^B.v^*)$ if $\varepsilon \equiv (\lambda^\vee x^A.u, y^B.v)$.
  (d) $(\lambda^\exists x^A.u^*)$ if $\varepsilon \equiv (\lambda^\exists x^A.u)$.

We will use the notation $\widetilde{*}$ above also in the remaining part of this paper.

Main lemma. *If $s \succ_p s'$, then $s' \succ_p s^*$.*

A proof of this lemma will be given in the next section. From this lemma, we have the next theorem immediately.

Theorem (diamond property of $\succ_p$). *If $s \succ_p s'$ and $s \succ_p s''$, then there exists s''' such that $s' \succ_p s'''$ and $s'' \succ_p s'''$.*

Then, by the facts that if $s \succ t$ then $s \succ_p t$, and that if $s \succ_p t$ then $s \gg t$, we have our main theorem below.

Theorem (Church-Rosser property of $\succ$). *If $s \gg s'$ and $s \gg s''$, then there exists s''' such that $s' \gg s'''$ and $s'' \gg s'''$.*

References

[1] Y., Andou, A normalization-procedure for the first order classical natural deduction with full logical symbols, Tsukuba J. Math. 19 (1995) 153-162.

[2] Y., Andou, On a reduction-procedure for full first order classical natural deduction, Doctoral thesis, University of Tsukuba, 1995.

[3] Y., Andou, A Canonical extension of Curry-Howard isomorphism to classical logic, in: G. Baum M. Frias editors, Anales WAIT'99 5-10, (SADIO, Buenos Aires, 1999)

[4] Y., Andou, Church-Rosser property of a simple reduction for full first order classical natural deduction, Submitted to a journal.

[5] K. Fujita, An interpretation of classical predicate logic in $\lambda\mu$-calculus, manuscript.

[6] G. Gentzen, Untersuchungen über das logische Schliessen, Math. Zeit. 39 (1935) 176-210, 405-431.

[7] P. de Groote On the strong normalization of natural deduction with premutation-conversions, in: P. Narendran and M. Rusinowitch editors, Rewriting Techniques and Applications, 45-59, (Springer, Berlin, 1999), LNCS 1631.

[8] J. R. Hindley, J. P. Seldin, Introduction to combinators and λ-calculus, (Cambridge University Press, 1986)

[9] M. Parigot, Classical proofs as programs, in: G. Gottlob, A. Leitsch, and D. Mundici editors, Computational Logic and Proof Theory, 263-277, (Springer, Berlin, 1993), LNCS 713.

[10] M. Parigot, $\lambda\mu$-Calculus: an algorithmic interpretation of classical natural deduction, in: A. Voronkov, editor, Logic Programming and Automated Reasoning, 190-201, (Springer, Berlin, 1992),LNAI 624.

[11] M. Parigot, Strong normalization for second order classical natural deduction, in: Logic in Computer Science, 39-46, (IEEE Computer Society Press, 1993), LICS 1993.

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

[13] D. Prawitz, Natural deduction - A proof theoretical study, (Almqvist & Wiksell, Stockholm, 1965)

[14] D. Prawitz, Ideas and results in proof theory, in: Proceedings of the Second Scandinavian Logic Symposium, 235-307, (North-Holland, Amsterdam, 1971)

[15] M. Takahashi, Parallel reduction in λ-calculus, Information and Computation 118 (1995) 120-127.

[16] A. S. Troelstra, H. Schwichtenberg, Basic Proof Theory, (Cambridge University Press, Cambridge, 1996)