Electronic Notes in Theoretical Computer Science 42 (2001)

URL: <http://www.elsevier.nl/locate/entcs/volume42.html> 15 pages

Parallel Reduction in Type Free *λµ*-Calculus

Kensuke Baba 1

*Graduate School of Information Science and Electrical Engineering Kyushu University*

*Fukuoka 812-8581 Japan*

Sachio Hirokawa 2

*Computing and Communications Center Kyushu University*

*Fukuoka 812-8581 Japan*

Ken-Etsu Fujita 3

*Department of Mathematics and Computer Science Shimane University*

*Matsue 690-8504 Japan*

**Abstract**

The typed *λµ*-calculus is known to be strongly normalizing and weakly Church- Rosser, and hence becomes confluent. In fact, Parigot formulated a parallel re- duction to prove confluence of the typed *λµ*-calculus by “Tait-and-Martin-Lo¨f” method. However, the diamond property does not hold for his parallel reduction. The confluence for type-free *λµ*-calculus cannot be derived from that of the typed *λµ*-calculus and is not confirmed yet as far as we know. We analyze granularity of the reduction rules, and then introduce a new parallel reduction such that both renaming reduction and consecutive structural reductions are considered as one step parallel reduction. It is shown that the new formulation of parallel reduction has the diamond property, which yields a correct proof of the confluence for type free *λµ*-calculus. The diamond property of the new parallel reduction is also applicable to a call-by-value version of the *λµ*-calculus containing the symmetric structural reduction rule.

1 Email: [baba@i.kyushu-u.ac.jp](mailto:baba@i.kyushu-u.ac.jp)

2 Email: [hirokawa@cc.kyushu-u.ac.jp](mailto:hirokawa@cc.kyushu-u.ac.jp)

3 Email: [fujiken@cis.shimane-u.ac.jp](mailto:fujiken@cis.shimane-u.ac.jp)

*◯*c *2001 Published by Elsevier Science B. V. Open access under* [*CC BY-NC-ND license*](http://creativecommons.org/licenses/by-nc-nd/3.0/)*.*

# Introduction

Parigot’s *λµ*-calculus [15] is an extension ofthe typed *λ*-calculus with two new term constructors [*α*]*M* and *µα.M* . The term [*α*]*M* indexes *α* to a type of *M* and the term *µα.M* has a type indexed by *α* in *M* . These constructors provide classical proofs and can at the same time be considered as control operators for functional programming language. Approximately, we can think of [*a*]*M* as (throw a M) and *µa.M* as (catch a M) in terms ofCommon Lisp.

The *λµ*-terms *M* are defined as follows:

*M* ::= *x | λx.M | MM | µα.M |* [*α*]*M* The calculus has the following basic reduction rules.

*β*-reduction: (*λx.M* )*N → M* [*x* := *N* ]

Structural reduction: (*µα.M* )*N → µα.M* [[*α*]*w* := [*α*](*wN* )] Renaming: [*β*](*µα.M* ) *→ M* [*α* := *β*]

We assume some familiarity to *λ*-calculus [2,11,12]. In the structural reduc- tion, the substitution is defined as follows:

1. *x*[[*α*]*w* := [*α*](*wN* )] = *x*
2. (*λx.M* )[[*α*]*w* := [*α*](*wN* )] = *λx.M* [[*α*]*w* := [*α*](*wN* )]
3. (*MM* )[[*α*]*w* := [*α*](*wN* )] = *M* [[*α*]*w* := [*α*](*wN* )]*M* [[*α*]*w* := [*α*](*wN* )]
4. (*µβ.M* )[[*α*]*w* := [*α*](*wN* )] = *µβ.M* [[*α*]*w* := [*α*](*wN* )]

**5-1.** ([*β*]*M* )[[*α*]*w* := [*α*](*wN* )] = [*β*](*M* [[*α*]*w* := [*α*](*wN* )]*N* ) if *α* = *β*

**5-2.** ([*β*]*M* )[[*α*]*w* := [*α*](*wN* )] = [*β*]*M* [[*α*]*w* := [*α*](*wN* )] if *α /*= *β*

In [15], Parigot outlined the proof for confluence of the *λµ*-calculus. He formulated the parallel reduction and claimed the diamond property for the parallel reduction:

If *M ⇒ N* then *N ⇒ M∗*.

Here *M∗* is a term obtained by reducing all the redexes in *M* . *M∗* is usually referred as the “complete development” [2] of *M* . The formulation of the par- allel reduction is based on “Tait-and-Martin-Lo¨f” method, which is explained clearly in [20]. The method is applicable to prove confluence of many reduc- tion systems. However, the direct application of the method does not work for the *λµ*-calculus. In fact, the diamond property does not hold for the for- mulation ofparallel reduction in [15], see also observations in [5]. It is not so trivial to prove the confluence as it seems to be.

The *λµ*-calculus is known to be strongly normalizing [16,17] and weak Church-Rosser. For notions of deduction, these two properties yield the con- fluence [2] for the typed terms [6,7]. But type free *λµ*-calculus is not strongly normalizing. For instance, the term (*λx.xx*)(*λx.xx*) does not have a normal

form. It is quite embarrassing that a correct proof of the confluence for type free *λµ*-calculus has never been published as far as we know.

We think that the reason why the diamond property does not hold for the parallel reduction comes from the sequential nature of the structural re- duction rule. Consider a term *M* = (*µα.*[*α*](*µβ.*[*α*]*x*))*y* which has a renam- ing redex and a structural redex. We have the terms *N*1 = (*µα.*[*α*]*x*)*y* and *N*2 = *µα.*[*α*]((*µβ.*[*α*](*xy*))*y*) by a renaming and a structural reductions respec- tively. Then we have *M ⇒ N*1 and *M ⇒ N*2. Ifthe diamond property would hold, then *N*1 and *N*2 were reducible to the same term *M∗* in one step re- duction. However, this is impossible here. After the structural reduction, the “residual” of renaming redex in *M* is no longer a renaming redex in *N*2. To make the residual back to a renaming redex, we need another step ofstructural reduction. We consider such a successive sequence ofstructural reductions as a one step parallel reduction. With such a formulation, we can prove the strong diamond property for the parallel reduction.

*M* = (*µα.*[*α*](*µβ.*[*α*]*x*))*y*

R / ❅ S

/ ❅

/s ❅❘

*N*1 = (*µα.*[*α*]*x*)*y N*2 = *µα.*[*α*]((*µβ.*[*α*](*xy*))*y*)

❅ S S/

❅

❅❘

*µα.*[*α*](*xy*)

R /s

/s

/

We consider the *λµ*-calculus as a programming language and reduction as computation. The reduction rules of the *λµ*-calculus capture the mechanism of functional programming languages with control [9,4,8]. However we can not apply an arbitrary reduction for implementation of programming language. Usually we fix a reduction strategy. The call-by-value *λµ*-calculus *λµv* was first considered by Ong and Stewart [14]. The *λµv*-calculus contains another reduction rule called “symmetric structural reduction” such that:

*N* (*µα.M* ) *→ µα.M* [[*α*]*w* := [*α*](*Nw*)]*.*

Note that a subsystem is not always confluent even if the whole system is confluent. Therefore, the confluence of *λµv* does not yield the confluence of *λµ*, even if we ignore the symmetric structural reduction rule. We shall also formulate an appropriate parallel reduction for a call-by-value version of the *λµ*-calculus and prove the strong diamond property.

In this paper, we deal with type free *λµ*-calculi, and the definition of the *λµ*-terms is distinct from that of the original ones. For example, the well- formed term ([*α*]*M* )*N* is not a term in the original *λµ*-calculus, since [*α*]*M*

is not an unnamed but a named term. Does the confluence proof for type free *λµ*-calculus still work for that of the original one? The parallel reduction relation we define in the following sections is included in the transitive and reflexive closure of reduction rules, and therefore the complete development of an original term is also an original term. Hence our proof method is still and all sound for the case of the original *λµ*-calculus.

# Parallel Reduction in *λµ*-Calculus

We define the parallel reduction in the following. The rules 1–8 are obtained by a straightforward application of Tait-and-Martin-Lo¨f method to *β*-reduction, structural reduction and renaming. The last inference rule 9 is introduced in the present paper. It combines a renaming and a consecutive sequence of structural reductions. It is easy to see that the transitive and reflexive closure of“*→*” is identical to the transitive closure of“*⇒*”.

## Definition 2.1

* 1. *x ⇒ x*

*M ⇒ M'*

* 1. *λx.M ⇒ λx.M'*

*M ⇒ M' N ⇒ N'*

* 1. *MN ⇒ M'N' M ⇒ M'*
  2. *µα.M ⇒ µα.M'*

*M ⇒ M'*

* 1. [*α*]*M ⇒* [*α*]*M'*

*M ⇒ M' N ⇒ N'*

* 1. (*λx.M* )*N ⇒ M'*[*x* := *N'*]

*M ⇒ M' N ⇒ N'*

* 1. (*µα.M* )*N ⇒ µα.M'*[[*α*]*w* := [*α*](*wN'*)]

*M ⇒ M'*

* 1. [*β*](*µα.M* ) *⇒ M'*[*α* := *β*]

*M ⇒ M' N*1 *⇒ N' ··· Nn ⇒ N'*

1

*n*

* 1. [*β*]((*µα.M* )*N*1 *··· Nn*) *⇒ M'*[[*α*]*w* := [*β*](*wN' ··· N'* )]

1 *n*

We define the complete development *M∗* ofa term *M* as follows:

## Definition 2.2

1. *M* = *x*. Then *M∗* = *x*.
2. *M* = *λx.M*1. Then *M∗* = *λx.M∗*.

1

1. *M* = *M*1*M*2.
   1. *M*1 = *λx.M*3. Then *M∗* = *M∗*[*x* := *M∗*].

3 2

* 1. *M*1 = *µα.M*3. Then *M∗* = *µα.M∗*[[*α*]*w* := [*α*](*wM∗*)].

3 2

* 1. *M∗* = *M∗M∗* o.w.

1 2

1. *M* = *µα.M*1. Then *M∗* = *µα.M∗*.

1

1. *M* = [*α*]*M*1.
   1. *M*1 = *µβ.M*2. Then *M∗* = *M∗*[*β* := *α*].

2

* 1. *M*1 = (*µβ.M*2)*N*1 *··· Nn*. Then *M∗* = *M∗*[[*β*]*w* := [*α*](*wN∗ ··· N∗*)].
  2. *M∗* = [*α*]*M∗* o.w.

1

2 1 *n*

# Diamond Property of Parallel Reduction

A gap in the proofofconfluence in [15] can be supplied by (2) ofthe following lemma. Without the rule 9, (2) does not hold true.

**Lemma 3.1** (1) *If M ⇒ M' and N ⇒ N', then M* [*x* := *N* ] *⇒ M'*[*x* := *N'*]*.*

1. *If M ⇒ M' and N ⇒ N', then M* [[*α*]*w* := [*α*](*wN* )] *⇒ M'*[[*α*]*w* := [*α*](*wN'*)]*.*
2. *If M ⇒ M', then M* [*β* := *α*] *⇒ M'*[*β* := *α*]*.*

**Proof.** (1) can be easily shown by induction on the structure of *M ⇒ M'*.

(3) is trivial. (2) can also be proved by induction on the structure of *M ⇒ M'*. Most cases are routine. Non-trivial cases are when the last inference of *M ⇒ M'* is either 8 or 9. To save the space ofthe paper, we explain only the case of8.

**Case 8.** The last inference rule is 8.

By definition of *M ⇒ M'*, *M* and *M'* have the form *M* = [*β*](*µγ.M*1) and

*M'* = *M'* [*γ* := *β*], and then *M ⇒ M'* has the following form:

1

*M*1 *⇒ M'*

1

8

*M* = [*β*](*µγ.M*1) *⇒ M'* [*γ* := *β*]= *M'*

1

Since *γ* is a bound variable, we can assume *γ /*= *α*.

**Case 8.1.** *α* = *β*.

Then we have the following results:

*M* [[*α*]*w* := [*α*](*wN* )] = ([*α*](*µγ.M*1))[[*α*]*w* := [*α*](*wN* )]

= [*α*]((*µγ.M*1[[*α*]*w* := [*α*](*wN* )])*N* )*, M'*[[*α*]*w* := [*α*](*wN'*)] = *M'* [*γ* := *α*][[*α*]*w* := [*α*](*wN'*)]

1

= *M'* [[*α*]*w* := [*α*](*wN'*)][[*γ*]*w* := [*γ*](*wN'*)][*γ* := *α*].

1

By induction hypothesis for *M*1 *⇒ M'* , we have *M*1[[*α*]*w* := [*α*](*wN* )] *⇒*

1

*M'* [[*α*]*w* := [*α*](*wN'*)]. Thus we have [*α*]((*µγ.M*1[[*α*]*w* := [*α*](*wN* )])*N* ) *⇒*

1

*M'* [[*α*]*w* := [*α*](*wN'*)][[*γ*]*w* := [*α*](*wN'*)][*γ* := *α*] by the rule 9. Hence Lemma holds.

1

**Case 8.2.** *α /*= *β*.

Then the substitutions give the following results:

|  |  |  |
| --- | --- | --- |
| *M* [[*α*]*w* := [*α*](*wN* )] | = | ([*β*](*µγ.M*1))[[*α*]*w* := [*α*](*wN* )] |
| *M'*[[*α*]*w* := [*α*](*wN'*)] | =  = | [*β*](*µγ.M*1[[*α*]*w* := [*α*](*wN* )]),  *M'* [*γ* := *β*][[*α*]*w* := [*α*](*wN* )]  1 |
|  | = | *M'* [[*α*]*w* := [*α*](*wN* )][*γ* := *β*].  1 |

By induction hypothesis for *M*1, we have *M*1[[*α*]*w* := [*α*](*wN* )] *⇒ M'* [[*α*]*w* := [*α*](*wN'*)]. Therefore we have [*β*](*µγ.M*1[[*α*]*w* := [*α*](*wN* )]) *⇒ M'* [[*α*]*w* :=

1

1

[*α*](*wN'*)][*γ* := *β*] by the rule 8. Thus Lemma holds.

*✷*

**Theorem 3.2** *For any λµ-term M and M', if M ⇒ M' then M' ⇒ M∗.*

The proof is by induction on the structure of *M ⇒ M'* and is shown in Appendix.

**Theorem 3.3** *If M ⇒ M*1 *and M ⇒ M*2*, then there exists some M*3 *such that M*1 *⇒ M*3 *and M*2 *⇒ M*3*.*

**Proof.** Put *M*3 = *M∗*. Then Theorem 3.3 holds by Theorem 3.2.

*✷*

Since the transitive and reflexive closure of“*→*” is identical to the transi- tive closure of“*⇒*”, we have the confluence ofthe type free *λµ*-calculus.

**Theorem 3.4** *The type free λµ-calculus is confluent.*

# Parallel Computation in Call-by-Value *λµ*-Calculus

A call-by-value version ofthe *λµ*-calculus was first provided by Ong and Stew- art [14]. As compared with the call-by-name system [15,16,17], one can adopt some reduction rule more in the call-by-value system; so-called symmetric structural reduction [15] such that *N* (*µα.M* ) *→ µα.M* [[*α*]*w* := [*α*](*Nw*)]. It is known that adding such a reduction rule breaks down the confluence unless the above term *N* is in the form of a value. In this section, the notion of values as an extended form is introduced based on observations in [5,6,7].

*V* ::= *x | λx.M |* [*α*]*M*

This notion is closed under both value-substitutions and substitutions induced by structural reduction or symmetric structural reduction defined below.

A context *E* [ ] with a hole [ ] is defined as usual, such that

*E* ::= [ ] *| EM | V E.*

Let *n ≥* 0 and *M* be a term. We will write *En*[*En−*1[*·· E*1[*N* ] *· ·*]] for *E* [*N* ], where each *Ei /≡* [ ] is either in the form of *V* [ ] or [ ]*M* . For simplicity, such *Ei* also denotes the value *V* or the term *M* .

The call-by-value *λµ*-calculus consists ofthe following reduction rules:

*βv*-reduction (*λx.M* )*V →v M* [*x* := *V* ]

Structural reduction (*µα.M*1)*M*2 *→v µα.M*1[[*α*]*w* := [*α*](*wM*2)] Symmetric structural reduction *V* (*µα.M* ) *→v µα.M* [[*α*]*w* := [*α*](*V w*)] Renaming reduction [*β*](*µα.V* ) *→v V* [*α* := *β*]

This renaming rule is different from that in [14]. The distinction is essential under the extended form of values, and this form of renaming would also be natural from the viewpoint of CPS-translation such as in [5,6,7].

We will show that the new parallel reduction can also be applicable to proving confluence for the call-by-value system of the *λµ*-calculus, contrary to the straightforward use of parallel reduction in [14]. To prove this, we define parallel reductionas follows:

## Definition 4.1

1. *x x*

*M M'*

1. *λx.M λx.M'*

*M M' N N'*

1. *MN M'N' M M'*
2. *µα.M µα.M'*

*M M'*

1. [*α*]*M* [*α*]*M'*

*M M' V N'*

1. (*λx.M* )*V M'*[*x* := *N'*]

*M M' N N'*

1. (*µα.M* )*N µα.M'*[[*α*]*w* := [*α*](*wN'*)]

*M M' V N'*

1. *V* (*µα.M* ) *µα.M'*[[*α*]*w* := [*α*](*N'w*)]

*V M' E*1 *E' ··· En E' E* = *En*[*·· E*1[ ] *· ·*] *E'* = *E'* [*·· E'* [ ] *· ·*]

1 *n n* 1

1. [*α*](*E* [*µβ.V* ]) *M'*[[*β*]*w* := [*α*](*E'*[*w*])]

Now it can be seen that the transitive and reflexive closure of *→v* is equiv-

alent to the transitive closure of.

**Lemma 4.2** (1) *If V M, then M is also in the form of a value.*

1. *If M N and V N', then M* [*x* := *V* ] *N* [*x* := *N'*]*.*
2. *If M M' and N N', then M* [[*α*]*w* := [*α*](*wN* )] *M'*[[*α*]*w* := [*α*](*wN'*)]*.*
3. *If M M' and V N', then M* [[*α*]*w* := [*α*](*V w*)] *M'*[[*α*]*w* := [*α*](*N'w*)]*.*
4. *If M M', then M* [*β* := *α*] *M'*[*β* := *α*]*.*
5. *Let n ≥* 0*. Let E* [] = *En*[*·· E*1[ ] *· ·*] *and E'*[] = *E'* [*··E'* [ ] *· ·*]*. If M M' and*

*n* 1

*Ei E'* (1 *≤ i ≤ n*)*, then M* [[*α*]*w* := [*α*](*E* [*w*])] *M'*[[*α*]*w* := [*α*](*E'*[*w*])]*.*

*i*

**Proposition 4.3** *For any λµ-term M, there exists M∗ such that for any N, N M∗ whenever M N.*

**Proof.** By induction on the derivation of. Here, the complete development

*M∗* can be given inductively as follows:

## Definition 4.4

1. *M* = *x*. Then *M∗* = *x*.
2. *M* = *λx.M* . Then *M∗* = *λx.M∗*.
3. *M* = *M*1*M*2.
   1. *M*1 = *λx.M*3 and *M*2 = *V*2. Then *M∗* = *M∗*[*x* := *V ∗*].

3 2

* 1. *M*1 = *µα.M*3. Then *M∗* = *µα.M∗*[[*α*]*w* := [*α*](*wM∗*)].

3 2

* 1. *M*1 = *V*1 and *M*2 = *µα.M*4. Then *M∗* = *µα.M∗*[[*α*]*w* := [*α*](*V ∗w*)].

4 1

* 1. *M∗* = *M∗M∗* o.w.

1 2

1. *M* = *µα.M*1. Then *M∗* = *µα.M∗*.

1

1. *M* = [*α*]*M*1.
   1. *M*1 = *E* [*µβ.V*2]. Then *M∗* = *V ∗*[[*β*]*w* := [*α*](*E∗*[*w*])], where *E∗*[ ] is defined as *E∗*[*·· E∗*[ ] *· ·*]for *E* [] = *En*[*·· E*1[ ] *· ·*] and *n ≥* 0.

2

*n* 1

* 1. *M∗* = [*α*]*M∗* o.w.

1

We show only the case *M* of [*α*]*M*1. The remaining cases can also be justified following a similar pattern.

1. Case [*α*]*M*1 of[*α*](*E* [*µβ.V* ]):

1-1. *M N* = [*α*]*N*1 is derived from *E* [*µβ.V* ] *N*1 by 5: 1-1-1. *E* [*µβ.V* ] *≡ µβ.V* :

In this case, *µβ.V N*1 = *µβ.N*2 is derived from *V N*2 by 4, where *N*2 is also a value. From the induction hypothesis, we have *N*2 *V ∗*, and hence *N* = [*α*](*µβ.N*2) *V ∗*[*β* := *α*]= *M∗* is obtained by 9.

1-1-2. *E* [*µβ.V* ] *≡ En*[*·· E*1[*µβ.V* ] *· ·*] (*n ≥* 1):

Since *E*1[*µβ.V* ] is not a value, *En*[*·· E*1[*µβ.V* ] *· ·*] *N*1 must be derived from

*E*1[*µβ.V* ] *N'*

2

*j*

and *Ej E'*

(2 *≤ j ≤ n*) by the successive use of 3, where

*N*1 = *E'* [*·· E'* [*N'* ] *· ·*]. Here, we have two cases for *E*1 and two derivations for

*n* 2 2

each ofthose.

1-1-2-1. *E*1[*µβ.V* ] *≡ V*1(*µβ.V* ):

1-1-2-1-1. *V*1(*µβ.V* ) *N'* = *N' N'*

is derived from *µβ.V N'*

and *V*1 *N'*

by 3:

5

5

2 3 4 4 3

Since *µβ.V N'*

4

= *µβ.N'*

must be derived from *V N'*

by 4, we have

*' V ∗* by the induction hypothesis, where *N'* is also a value. Let *E'* be

*N*

5

1

5

*N'* [ ], where *N'* is a value. Then the induction hypothesis gives *N' V ∗*

3 3 3 1

abbreviated as *E' E∗*. From the induction hypotheses for *Ej* (2 *≤ j ≤ n*),

1

1

we also have *E' E∗*, and then [*α*](*E'* [*··E'* [*µβ.N'* ] *· ·*]) *V ∗*[[*β*]*w* := [*α*](*E∗*[*··*

*j j n* 1 5 *n*

*E∗*[*w*] *· ·*])] is obtained by 9.

1

1-1-2-1-2. *V*1(*µβ.V* ) *N'* = *N'* [[*α*]*w* := [*α*](*N' w*)] is derived from *V*1 *N'*

and *V N'*

4

2 3 4 3

by 8:

The induction hypotheses give *N' V ∗* and *N' V ∗*. From the substi-

3 1 4

tution lemma, we have *N'* [[*β*]*w* := [*β*](*N' w*)] *V ∗*[[*β*]*w* := [*β*](*V ∗w*)], where

4 3 1

*N'* is also a value and values are closed under the substitutions. The induction

4

hypotheses for *Ej* (2 *≤ j ≤ n*) also give *E' E∗*. Hence, the use of9 derives

*j j*

[*α*](*E'* [*·· E'* [*µβ.N'* [[*β*]*w* := [*β*](*N' w*)]] *· ·*])

*n* 2 4 3

(*V ∗*[[*β*]*w* := [*β*](*V ∗w*)])[[*β*]*w* := [*α*](*E∗*[*·· E∗*[*w*] *· ·*])],

1 *n* 2

whose right-hand side is equivalent to *V ∗*[[*β*]*w* := [*α*](*E∗*[*·· E∗*[*w*] *· ·*])], where

*n* 1

*E∗* is *V ∗*[ ].

1 1

1-1-2-2. *E*1[*µβ.V* ] *≡* (*µβ.V* )*M*2:

In this case, we have two derivations for (*µβ.V* )*M*2 *N'*

2

by the use of 3

or 7. Each case can be verified following a similar pattern to the above two

cases.

1-2. *M N* = *N'*[[*α*]*w* := [*β*](*E'*[*w*])] is derived from *V N'* and *Ei E'*

*i*

(1 *≤ i ≤ n*), where *E* [] = *En*[*·· E*1[ ] *· ·*] and *E'*[] = *E'* [*·· E'* [ ] *· ·*]:

*n* 1

The successive application ofthe substitution lemma to the induction hy-

potheses.

1. Otherwise:

The straightforward use of the induction hypothesis.

*✷*

Finally, the confluence for the call-by-value *λµ*-calculus can be confirmed, sincehas the diamond property.

**Theorem 4.5** *The type free λµ-calculus of call-by-value has the confluence property.*

# Related Work and Further Problems

## Sequentiality of Reduction

Parallel reduction is a very clear and intuitive idea which means to reduce a number ofredexes (existing in the term) simultaneously. It is often applied to prove confluence ofreduction system. However, a naive formulation ofpar- allel reduction does not always work. The *λµ*-calculus is one ofsuch reduction systems. We have reasoned that the difficulty comes from the sequentiality of the structural reductions. So we proposed that a consecutive sequence of the structural reductions should be considered as one step of parallel reduction. As pointed out in Takahashi [20], the idea does not work for *λη−*1 either, i.e., *λ*-calculus with *η*-expansion: *M → λx.Mx*. The confluence of *λη−*1 is proved in [1,13]. Jay and Ghani [13] proved the confluence by introducing “parallel ex- pansion” which includes, roughly speaking, a consecutive application of *η−*1:

*M → λx*1*.Mx*1 *→ λx*1*x*2*.Mx*1*x*2 *→ λx*1*x*2*x*3*.Mx*1*x*2*x*3 *→ ··· .* Van Raams- donk [18] introduced the notion of “superdevelopment” to prove confluence ofthe orthogonal combinatory reduction systems. The superdevelopment is a reduction sequence in which some redexes that are created during reduction may be contracted, besides redexes that descend from the initial term. A key idea ofthese works is to overcome some sequentiality ofreductions. We cannot show, at the current stage, what kind of reduction contains such sequential- ity in general. To find some criteria ofsuch sequentiality is one of further work.

## One More Reduction Rule

Parigot’s *λµ*-calculus [15] has one more reduction rule *η∗*:

*µα.*[*α*]*M → M*

if *α* has no free occurence in *M* . Consider a term *M* = *µα.*[*α*]((*µβ.*[*γ*]*x*)*yz*). Then *M* has *η∗*-redex and the redex with respect to the rule 9 ofDefinition2.1. The reduction ofeach redex is represented by “*η∗*” and “RS” in the following figure:

*µα.*[*α*]((*µβ.*[*γ*]*x*)*yz*)

❅ *η∗*

❅

❅❘

RS

❄

*µα.*[*γ*]*x*

S✑

✑✰✑

(*µβ.*[*γ*]*x*)*yz* S✑

✑✰✑

(*µβ.*[*γ*]*x*)*z*

By “RS”, we reach *µα.*[*γ*]*x* with one step parallel reduction. If we apply *η∗* first, we have (*µβ.*[*γ*]*x*)*yz* from which we cannot reach *µα.*[*γ*]*x* with one step parallel reduction. However, we can overcome this situation by counting a series of structural reductions as one step. On the basis of this idea, we have a formulation of parallel reduction with *η∗* as well, although the formal de- scription is skipped here for simplicity.

## Practical Application of Type Free *λµ*-Calculi

In order to make the *λµ*-calculi in practical application, we need to realize some machines. In fact, Bierman [3], de Groote [10], and Selinger [19] pro- posed such abstract machines and analyzed their behavior. However, all those machines are sequential in nature. We expect that our parallel reduction could yield natural extensions ofthose machines.

# Acknowledgement

The authors would like to thank the referees very much for useful comments.

# References

1. Akama, Y.: *On Mints’s Reduction for CCC-Calculus*, Lecture Notes in Computer Science **664** (1993), 1–15.
2. Barendregt, H. P.: “The Lambda Calculus,” 2nd ed., North-Holland, Amsterdam, 1984.
3. Bierman, G. M.: *A Computational Interpretation of the λµ-Calculus*, University of Cambridge Computer Laboratory Technical Report **448** (1998).
4. Felleisen, M., D. P. Friedman, E. Kohlbecker, and B. Duba: *A Syntactic Theory of Sequential Control*, Theoretical Computer Science **52** (1987), 205–237.
5. Fujita, K.: *Calculus of Classical Proofs I*, Lecture Notes in Computer Science

**1345** (1997), 321–335.

1. Fujita, K.: *Explicitly Typed λµ-Calculus for Polymorphism and Call-by-Value*, Lecture Notes in Computer Science **1581** (1999), 162–176.
2. Fujita, K.: *Domain-Free λµ-Calculus*, Informatique Th´eorique et Applications, forthcoming.
3. Griffin, T.: *A Formulae-as-Types Notion of Control*, Conference Record of 17th ACM Symposium on Principles of Programming Languages (1990), 47–58.
4. de Groote, Ph.: *On the Relation between the λµ-Calculus and the Syntactic Theory of Sequential Control*, Lecture Notes in Computer Science **822** (1994), 31–43.
5. de Groote, Ph.: *An environment machine for the λµ-Calculus*, Mathematical Structures in Computer Science, to appear.
6. Hindley, J. R.: “Basic Simple Type Theory”, Cambridge University Press, 1997.
7. Hindley, J. R. and J. P. Seldin: “Introduction to Combinators and *λ*-Calculus”, Cambridge University Press, 1986.
8. Jay, C. B. and N. Ghani: *The Virtues of Eta-expansion*, LFCS report ECS- LFCS-92-243 (1992).
9. Ong, C. -H. L. and C. A. Stewart: *A Curry-Howard Foundation for Functional Computation with Control*, Proc. 24th Annual ACM SIGPLAN-SIGACT Symposium of Principles of Programming Languages (1997).
10. Parigot, M.: *λµ-calculus: An Algorithmic Interpretation of Classical Natural Deduction*, Lecture Notes in Artificial Intelligence **624** (1992), 190–201.
11. Parigot, M.: *Strong Normalization for Second Order Classical Natural Deduction*, Proc. 8th Annual IEEE Symposium on Logic in Computer Science (1993), 39–46.
12. Parigot, M.: *Proofs of Strong Normalization for Second Order Classical Natural Deduction*, J. Symbolic Logic **62 (4)** (1997), 1461–1479.
13. van Raamsdonk, F.: *Confluence and Superdevelopment*, Lecture Notes in Computer Science **690** (1993), 168–182.
14. Selinger, P.: *Implementation of the Call-By-Name λµν-Calculus*, manuscript.
15. Takahashi, M.: *Parallel Reductions in λ-Calculus*, Information and Computation **118** (1995), 120–127.

# Appendix: Proof of Theorem 3.2

**Theorem 3.2** *For any λµ-term M and M', if M ⇒ M' then M' ⇒ M∗.*

**Proof.** By induction on the structure of *M ⇒ M'*.

1. *M* = *x*.

Then we have *M∗* = *M'* = *x*. Thus we have *M' ⇒ M∗*.

1. *M* = *λx.M* .

Then *M ⇒ M'* has the following form.

*M*1 *⇒ M'*

1

2

*M* = *λx.M*1 *⇒ λx.M'* = *M'*

1

By induction hypothesis we have *M' ⇒ M∗*. Thus we have *M'* = *λx.M' ⇒*

1 1 1

*λx.M∗* = *M∗*.

1

1. *M* = *M*1*M*2.
   1. *M* = (*λx.M*3)*M*2.

Then we have *M∗* = *M∗*[*x* := *M∗*] and the last inference rule of *M ⇒ M'*

3 2

is either 3 or 6.

* + 1. The last inference rule is 3.

Then *M ⇒ M'* has the following form.

*M*3 *⇒ M' λx.M*3 *⇒ λx.M'*

3

2

*M*2 *⇒ M'*

3 2 3

*M* = (*λx.M*1)*M*2 *⇒* (*λx.M'* )*M'* = *M'*

3 2

By induction hypothesis, we have *M' ⇒ M∗* and *M' ⇒ M∗*. Applying the

2 2 3 3

rule 6, we have (*λx.M'* )*M' ⇒ M∗*[*x* := *M∗*]. Thus *M' ⇒ M∗* holds.

3 2 3 2

* + 1. The last inference rule is 6.

*M ⇒ M'* has the following form.

*M*3 *⇒ M'*

*M*2 *⇒ M'*

3 2 6

*M* = (*λx.M*3)*M*2 *⇒ M'* [*x* := *M'* ]

3 2

By induction hypothesis, we have *M' ⇒ M∗* and *M' ⇒ M∗*. By Lemma 3.1

2 2 3 3

(1), it follows *M'* [*x* := *M'* ] *⇒ M∗*[*x* := *M∗*]. Thus *M' ⇒ M∗* and Theorem

holds.

3 2 3 2

* 1. *M* = (*µα.M*3)*M*2.

Then the last inference of *M ⇒ M'* is either 3 or 7.

* + 1. The last inference rule of *M ⇒ M'* is 3. Then *M ⇒ M'* has the following form.

*M*3 *⇒ M'*

3

4

*µα.M*3 *⇒ µα.M' M*2 *⇒ M'*

3 2 3

*M* = (*µα.M*3)*M*2 *⇒* (*µα.M'* )*M'*

3 2

By induction hypothesis we have *M' ⇒ M∗* and *M' ⇒ M∗*. Applying the

2 2 3 3

rule 7, we have (*µαM'* )*M' ⇒ µα.M∗*[[*α*]*w* := [*α*](*wM∗*)]. Hence *M' ⇒ M∗*

3 2 3 2

and Theorem holds.

* + 1. The last inference rule of *M ⇒ M'* is 7. Then *M ⇒ M'* has the following form.

*M*3 *⇒ M' M*2 *⇒ M'*

3 2 7

(*µα.M*3)*M*2 *⇒ µα.M'* [[*α*]*w* := [*α*](*wM'* )]

3 2

By induction hypothesis we have *M' ⇒ M∗* and *M' ⇒ M∗*. Apply-

2

2

3

3

ing Lemma 3.1 (2), we have *µα.M'* [[*α*]*w* := [*α*](*wM'* )] *⇒ µα.M∗*[[*α*]*w* :=

3 2 3

[*α*](*wM∗*)]. Hence *M' ⇒ M∗* and Theorem holds.

2

* 1. *M* = *M*1*M*2 and *M*1 is not a *λ*-abstraction or a *µ*-abstraction. Then *M∗* = *M∗M∗* and *M ⇒ M'* has the following form.

1 2

*M*1 *⇒ M' M*2 *⇒ M'*

1 2 3

*M* = *M*1*M*2 *⇒ M' M'* = *M'*

1 2

By induction hypothesis, we have *M' ⇒ M∗* and *M' ⇒ M∗*. Thus we have

1 1 2 2

*M' M' ⇒ M∗M∗*. Therefore Theorem holds.

1 2 1 2

1. *M* = *µα.M*1.

Then we have *M∗* = *µα.M∗* and *M ⇒ M'* has the following form.

1

*M*1 *⇒ M'*

1

4

*µα.M*1 *⇒ µα.M'*

1

By induction hypothesis we have *M' ⇒ M∗*. Applying the rule 4, we have

1 1

*µα.M' ⇒ µα.M∗*. Hence Theorem holds.

1 1

1. *M* = [*α*]*M*1.
   1. *M* = [*α*](*µβ.M*2).

Then we have *M∗* = *M∗*[*β* := *α*]. The last inference rule of *M ⇒ M'* is

2

either 5 or 8.

* + 1. The last inference rule of *M ⇒ M'* is 5. Then *M ⇒ M'* has the following form.

*M*2 *⇒ M'*

2

4

*µβ.M*2 *⇒ µβ.M'*

2 5

*M* = [*α*](*µβ.M*2) *⇒* [*α*](*µβ.M'* )= *M'*

2

By induction hypothesis we have *M' ⇒ M∗*. Apply the rule 8. Then we

2

2

have [*α*](*µβ.M'* ) *⇒ M∗*[*β* := *α*]. Hence *M ⇒ M'* and Theorem holds.

2 2

* + 1. The last inference rule of *M ⇒ M'* is 8.

*M*2 *⇒ M'*

2

8

*M* = [*α*](*µβ.M*2) *⇒ M'* [*β* := *α*]= *M'*

2

By induction hypothesis, we have *M' ⇒ M∗*. Apply Lemma 3.1 (3). Then

2

2

we have *M'* [*β* := *α*] *⇒ M∗*[*β* := *α*], hence *M' ⇒ M∗*.

2 2

* 1. *M* = [*α*]((*µβ.M*2)*N*1 *··· Nn*).

Then we have *M∗* = *M∗*[[*β*]*w* := [*α*](*wN∗ ··· N∗*)]. The last inference rule

2 1 *n*

of *M ⇒ M'* is either 5 or 9.

* + 1. The last inference rule of *M ⇒ M'* is 5. Then *M ⇒ M'* has the following form.

(*µβ.M*2)*N*1 *⇒ Q' N*2 *⇒ N' ··· Nn ⇒ N'*

2 *n*

(*µβ.M*2)*N*1 *··· Nn ⇒ Q'N' ··· N'*

3*, ··· ,* 3

1 *n* 5

*M* = [*α*]((*µβ.M*2)*N*1 *··· Nn*) *⇒* [*α*](*Q'N' ··· N'* )= *M'*

1 *n*

Then the last inference rule of (*µβ.M*2)*N*1 *⇒ Q'* is either 3 or 7.

* + - 1. The last inference rule of (*µβ.M*2)*N*1 *⇒ Q'* is 3.

Then *Q'* = *µβ.M'* for some *M'* , and *M ⇒ M'* has the following form.

2

2

*M*2 *⇒ M'*

2 4

*µβ.M*2 *⇒ µβ.M'*

*N*1 *⇒ N'*

*N*2 *⇒ N'*

*··· Nn ⇒ N'*

2 1 2 *n*

(*µβ.M*2)*N*1 *··· Nn ⇒* (*µβ.M'* )*N' ··· N'*

3*, ··· ,* 3

2 1 *n* 5

*M* = [*α*]((*µβ.M*2)*N*1 *··· Nn*) *⇒* [*α*]((*µβ.M'* )*N' ··· N'* )= *M'*

2 1 *n*

By induction hypothesis, we have *M' ⇒ M∗,N' ⇒ N∗ ··· ,N' ⇒ N∗*.

2 2 1 1 *n n*

Apply the rule 9. Then we have [*α*]((*µβ.M'* )*N' ··· N'* ) *⇒ M∗*[[*β*]*w* :=

2 1 *n* 2

[*α*](*wN∗ ··· N∗*)]. Hence *M' ⇒ M∗* holds.

1 *n*

* + - 1. The last inference rule of (*µβ.M*2)*N*1 *⇒ Q'* is 7.

Then *M ⇒ M'* has the following form.

*M*2 *⇒ M' N*1 *⇒ N'*

2 1 7

(*µβ.M*2)*N*1 *⇒ µβ.M'* [[*β*]*w* := [*β*](*wN'* )] *N*2 *⇒ N'*

*··· Nn ⇒ N'*

2 1 2

*n* 3*, ··· ,* 3

(*µβ.M*2)*N*1 *··· Nn ⇒* (*µβ.M'* [[*β*]*w* := [*β*](*wN'* )])*N' ··· N'*

2 1 1 *n* 5

*M* = [*α*]((*µβ.M*2)*N*1 *··· Nn*) *⇒* [*α*]((*µβ.M'* [[*β*]*w* := [*β*](*wN'* )])*N' ··· N'* ) = *M'*

2 1 1 *n*

By induction hypothesis, we have *M' ⇒ M∗* and *N' ⇒ N∗*. By Lemma 3.1

2 2 1 1

(2), we have *M'* [[*β*]*w* := [*β*](*wN'* )] *⇒ M∗*[[*β*]*w* := [*β*](*wN∗*)]. On the other

2 1 2 1

hand, we have *N' ⇒ N∗, ··· N' ⇒ N∗* by induction hypothesis. Apply the

2 2 *n n*

rule 9. Then we have

[*α*]((*µβ.M'* [[*β*]*w* := [*β*](*wN'* )])*N' ··· N'* )

2 1 2 *n*

*⇒ M∗*[[*β*]*w* := [*β*](*wN∗*)][[*β*]*w* := [*α*](*wN∗ ··· N∗*)]

2 1 2 *n*

= *M*2[[*β*]*w* := [*α*](*wN∗N∗ ··· N∗*)].

1. 2 *n*

Hence *M ⇒ M'* and Theorem holds.

* + 1. The last inference rule of *M ⇒ M'* is 9. Then *M ⇒ M'* has the following form.

*M*2 *⇒ M' N*1 *⇒ N' ··· Nn ⇒ N'*

2 1 *n* 9

[*α*]((*µβ.M*2)*N*1 *··· Nn*) *⇒ M'* [[*β*]*w* := [*α*](*wN' ··· N'* )]

1. 1 *n*

By induction hypothesis we have *M' ⇒ M∗,N' ⇒ N∗, ··· ,N' ⇒ N∗*.

2 2 1 1 *n n*

Apply Lemma 3.1 (2) and (3), we have *M∗*[[*β*]*w* := [*α*](*wN' ··· N'* )] *⇒*

2 1 *n*

*M*2 *∗* [[*β*]*w* := [*α*](*wN∗ ··· N∗*)]. Hence *M' ⇒ M∗* and Theorem holds.

1 *n*

* 1. *M* = [*α*]*M*1 and *M*1 is not of the form *M*1 = *µβ.M*2 or *M*1 = (*µβ.M*2)*N*1 *··· Nn*.

Then we have *M∗* = [*α*]*M∗* and *M ⇒ M'* has the following form.

1

*M*1 *⇒ M'*

1

5

[*α*]*M*1 *⇒* [*α*]*M'*

1

By induction hypothesis, we have *M' ⇒ M∗*. Applying the rule 5, we have

1 1

[*α*]*M' ⇒* [*α*]*M∗*. Hence Theorem holds.

1 1

*✷*