Refining reduction in the lambda calculus

Eindhoven University of Technology

Department of Mathematics and Computing Science

Refining Reduction in the Lambda Calculus

by

F. Kamareddine and R. Nederpelt

94/18

Computing Science Note 94/18
Eindhoven, April 1994

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Refining Reduction in the lambda calculus *

Fairouz Kamareddine †
Department of Computing Science
17 Lilybank Gardens
University of Glasgow
Glasgow G12 8QQ, Scotland
email: fairouz@dcs.glasgow.ac.uk

and

Rob Nederpelt
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O.Box 513
5600 MB Eindhoven, the Netherlands
email: wsinrpn@win.tue.nl

April 14, 1994

*We are grateful for the discussions with Roel Bloo, Tijn Borghuis and Erik Poll, and for the helpful remarks received from them. We are also grateful to Peter Peters for his help concerning Latex and e-mail which enabled us to keep exchanging drafts and corrections of the present paper.

†Kamareddine is grateful to the Department of Mathematics and Computing Science, Eindhoven University of Technology, for their financial support and hospitality from October 1991 to September 1992, and during several short periods in 1993 and 1994.

Capsule Review

In the λ-calculus as we know it, some redexes in a term may not be visible before other redexes have been contracted. For example, in $t \equiv ((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2)x_1$, only $(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3$, and $(\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2$ are visible. Yet when reducing t to a normal form, a third redex must be contracted; namely $(\lambda_{x_5}.x_5x_4)x_1$. This third redex is not immediately visible in t. To solve this problem we switch from the classical notation to what we call *item notation* where the argument occurs before the function and where parenthesis are grouped in a novel way. In our item notation, t will be written as $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$ and we can provide t in this item notation with a bracketing structure [[ ][ ]] where δ and λ correspond to [and] respectively (ignoring $(x_4\delta)x_5$). We extend the notion of a redex from being any opening bracket [next to a closing bracket], to being any pair of matching [and] which are separated by matching brackets. Hence we see immediately that the redexes in t originate from the $\delta\lambda$-couples $(x_2\delta)(\lambda_{x_7})$, $(x_3\delta)(\lambda_{x_6})$ and $(x_1\delta)(\lambda_{x_5})$. This natural matching was not present in the classical notation of t. With item notation, we shall refine reduction in two ways:

1. We shall generalise β-reduction so that any redex can be contracted and hence we can contract the redex based on $(x_1\delta)(\lambda_{x_5})$ before we contract any of the redexes based on $(x_2\delta)(\lambda_{x_7})$ and $(x_3\delta)(\lambda_{x_6})$. That is, the β-rule changes from $(t_1\delta)(\lambda_v)t_2 \rightarrow_\beta t_2[v := t_1]$ to $(t_1\delta)\overline{s}(\lambda_v)t_2 \rightsquigarrow_\beta \overline{s}(t_2[v := t_1])$ for $\overline{s}$ having a matching bracketing structure. I.e. $(t_1\delta)\overline{s}(\lambda_v)$ is a redex in our generalised sense. So

$$\begin{array}{ll} (x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5 & \rightsquigarrow_\beta \\ (x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(((x_4\delta)x_5)[x_5 := x_1]) & \equiv \\ (x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(x_4\delta)x_1 & \end{array}$$

   We show moreover, that the Church Rosser property holds for the generalised β-reduction.

2. An alternative to the generalised notion of β-reduction can be obtained by keeping the old β-reduction and by *reshuffling* the term in hand. So we can reshuffle the term $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$ to $(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(x_1\delta)(\lambda_{x_5})(x_4\delta)x_5$, in order to transform the bracketing structure [[ ][ ]] into [ ][ ][ ], where all the redexes correspond to adjacent [and]. Such a reshuffling is more difficult to describe in classical notation. I.e. it is hard to say what exactly happened when $((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2)x_1$, is reshuffled to $(\lambda_{x_7}.(\lambda_{x_6}.(\lambda_{x_5}.x_5x_4)x_1)x_3)x_2$. This is another attractive feature of our item notation which we shall also describe in this paper (using TS) showing its correctness and well-behavedness. In particular, we show that for any term t, in $TS(t)$ all the δ-items occur next to their matching λ-items. We show moreover, that if $t \rightsquigarrow_\beta t'$ then $(\exists t'')[(TS(t) \rightarrow_\beta t'') \wedge TS(t'') \equiv TS(t')]$.

Abstract

We introduce a λ-calculus notation which enables us to detect in a term, more β-redexes than in the usual notation. On this basis, we define an extended β-reduction which is yet a subrelation of conversion. The Church Rosser property holds for this extended reduction. Moreover, we show that we can transform generalised redexes into usual ones by a process called "term reshuffling".

Keywords: Item notation, Redexes, Church Rosser.

Contents

1 Introduction

In the classical λ-calculus, redexes and β-reduction are defined as follows

Definition 1.1 *(Redexes and β-reduction in classical notation)*
In the classical notation of the λ-calculus, a redex is of the form $(\lambda_v.t)t'$. Moreover, one-step β-reduction $\rightarrow_\beta$, is the compatible relation generated out of the axiom β: $(\lambda_v.t)t' \rightarrow_\beta t[v := t']$. Many step β-reduction $\twoheadrightarrow_\beta$ is the reflexive transitive closure of $\rightarrow_\beta$.

1.1 The item notation and visible redexes

We shall devise a novel notation in this paper where the order in an application is inverted and where the parentheses are grouped differently than those of the classical notation. So that, if $\mathcal{I}$ translates classical terms into our notation, **item notation**, then $\mathcal{I}(t_1 t_2)$ is written as $(\mathcal{I}(t_2)\delta)\mathcal{I}(t_1)$ and $\mathcal{I}(\lambda_v.t)$ is written as $(\lambda_v)\mathcal{I}(t)$. Both $(t\delta)$ and (λ_v) are called **items**.

Example 1.2 $\mathcal{I}((\lambda_{xy}.xy)z) \equiv (z\delta)(\lambda_x)(\lambda_y)(y\delta)x$. The items are $(z\delta)$, (λ_x), (λ_y) and $(y\delta)$.

With our item notation, classical redexes and β-reduction take the following form:

Definition 1.3 *(Classical redexes and β-reduction in item notation)*
*In the item notation of the λ-calculus, a classical redex is of the form $(t'\delta)(\lambda_v)t$. We call the pair $(t'\delta)(\lambda_v)$, a $\delta\lambda$-**pair**, or a $\delta\lambda$-**segment**. Moreover, the classical β-reduction axiom is: $(t'\delta)(\lambda_v)t \rightarrow t[v := t']$. One and many step β-reduction are defined as in Definition 1.1.*

Example 1.4 In the classical term $t \equiv ((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2)x_1$, we have the following redexes (the fact that neither x_6 nor x_7 appear as free variables in their respective scopes does not matter here; this is just to keep the example simple and clear):

1. $(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3$
2. $(\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2$

Written in item notation, t becomes $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$. Here, the two classical redexes correspond to $\delta\lambda$-pairs as follows:

1. $(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3$ corresponds to $(x_3\delta)(\lambda_{x_6})$. We ignore $(\lambda_{x_5})(x_4\delta)x_5$ as it is easily retrievable in item notation. It is the maximal subterm of t to the right of (λ_{x_6}).
2. $(\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2$ corresponds to $(x_2\delta)(\lambda_{x_7})$. Again $(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$ is ignored for the same reason as above.

There is however a third redex which is not visible in the classical term. Namely, $(\lambda_{x_5}.x_5x_4)x_1$. Such a redex will only be visible after we have contracted the above two redexes (we will not discuss the order here). In fact, assume we contract the second redex in the first step, and the first redex in the second step. I.e.

$$\begin{array}{ll} ((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2)x_1 & \rightarrow_\beta \\ ((\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_1 & \rightarrow_\beta \\ (\lambda_{x_5}.x_5x_4)x_1 & \rightarrow_\beta x_1x_4 \end{array}$$

Now, even though all these redexes (i.e. the first, second and third) are *needed* in order to get the normal form of t, only the first two were visible in the classical term at first sight.

The third could only be seen once we have contracted the first two reductions. In item notation, the third redex $(\lambda_{x_5}.x_5x_4)x_1$ corresponds to $(x_1\delta)(\lambda_{x_5})$ but the δ-item and λ-item are separated by the segment $(x_2\delta)(\lambda_{x_7})(x_2\delta)(\lambda_{x_6})$. By extending the notion of a redex and of β-reduction, we can make this redex visible and we can contract it before the other redexes.

The idea is simple; we generalise the notion of a $\delta\lambda$-segment $(t_1\delta)(\lambda_v)$ to a $\delta\lambda$-**couple** being an item $(t_1\delta)$ and an item (λ_v) separated by a segment $\overline{s}$ which is a **well-balanced segment**. A well-balanced segment is a sequence of δ- and λ-items which has the same structure as a matching composite of opening and closing brackets, each δ-item corresponding to an opening bracket and each λ-item corresponding to a closing bracket. We will show that this generalised β-reduction is Church-Rosser. We show moreover that with our item notation, it is possible to shuffle a term so that $\delta\lambda$-couples become $\delta\lambda$-pairs. I.e. so that $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$ becomes $(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(x_1\delta)(\lambda_{x_5})(x_4\delta)x_5$ where the three redexes correspond to the three $\delta\lambda$-pairs $(x_2\delta)(\lambda_{x_7})$, $(x_3\delta)(\lambda_{x_6})$ and $(x_1\delta)(\lambda_{x_5})$. Such a reshuffling is not easy to describe in classical notation. We shall show that term reshuffling is correct and preserves β-reduction.

1.2 The system Λ

We construct the system Λ out of a set of variables $\mathcal{V}$ and a set of operators Ω where a term is either a variable or is of the form $s_1s_2\cdots s_nv$ for variable v and *items* s_i, for $1 \leq i \leq n$. An item is defined either as $(\lambda_{v'})$ for variable v' or as $(t'\delta)$ for t' being a term. Both δ and $\lambda_{v'}$ are called operators; the first is an application operator, the second is an abstraction operator. The following definitions formalise our system Λ.

Definition 1.5 *(Terms in item notation Λ)*

- *We take $\mathcal{V} = \{x_n; n \geq 1\}$ to be the set of variables and assume that if $i \neq j$, then $x_i \neq x_j$ for all $i, j \geq 1$. We let $v, v', v'', v_1, v_2, \ldots$ range over $\mathcal{V}$.*
- *We take $\{\lambda_v; v \in \mathcal{V}\}$ to be the set of the abstraction operators (or* λ**-operators**$)$*. We use $\lambda_v, \lambda_{v'}, \lambda_{v_1}, \ldots$ to range over these operators.*
- *We assume an application operator or* δ**-operator** $\delta \notin \{\lambda_v; v \in \mathcal{V}\}$ *and we use ω, ω', ω'', ω_1, ω_2, ... to range over $\Omega = \{\delta\} \cup \{\lambda_v; v \in \mathcal{V}\}$.*
- *We write terms in item notation as: $\Lambda ::= \mathcal{V} \mid (\Lambda\delta)\Lambda \mid (\lambda\mathcal{V})\Lambda$. We use $t, t_1, t_2, \ldots$ to range over terms in item notation.*

Definition 1.6 *((main) items, (main, $\delta\lambda$) segment, body, endvar, $\delta\lambda$-segment, weight)*

- *If t is a Λ-term and $v \in \mathcal{V}$ then $(t\delta)$ and (λ_v) are* **items**, *the former being a* δ**-item** *and the latter a* λ**-item** *or a* λ_v**-item**. *We use $s, s_1, s_i, \cdots$ as meta-variables for items.*
- *A concatenation of zero or more items is a* **segment**. *In [de Bruijn 93] an item is called a* wagon *and a segment is called a* train. *We use $\overline{s}, \overline{s}_1, \overline{s}, \ldots$ as meta-variables for segments.*
- *Each term t is the concatenation of zero or more items and a variable: $t \equiv s_1s_2\ldots s_nv$. These items $s_1, s_2, \cdots, s_n$ are called the* **main items** *of t. We call the segment $s_1s_2\cdots s_n$,* `body`(t), *and the variable v,* `endvar`(t).

- *Analogously, a segment $\overline{s}$ is a concatenation of zero or more items $\overline{s} \equiv s_1 s_2 \cdots s_n$. These items $s_1, s_2, \cdots, s_n$ are called the* **main items**, *this time of $\overline{s}$.*
- *A concatenation of main items is a* **main segment**.
- *An important case of a segment is that of a $\delta\lambda$-***segment**, *being a δ-item immediately followed by a λ-item. Sometimes, we call the $\delta\lambda$-segment, a $\delta\lambda$-pair.*
- *The* **weight** *of a Λ-segment $\overline{s}$, $\mathtt{weight}(\overline{s})$, is the number of main items that compose the segment. The weight of a Λ-term t is the weight of $\mathtt{body}(t)$.*

Example 1.7 Let the Λ-term t be defined as $(\lambda_{x_2})((x_2\delta)(\lambda_{x_3})x_3\delta)(\lambda_{x_4})x_4$ and let the segment $\overline{s}$ be $(\lambda_{x_2})((x_2\delta)(\lambda_{x_3})x_3\delta)(\lambda_{x_4})$. Then the main items of both t and $\overline{s}$ are (λ_{x_2}), $((x_2\delta)(\lambda_{x_3})x_3\delta)$ and (λ_{x_4}), being a λ_{x_2}-item, a δ-item, and a λ_{x_4}-item.

We identify terms that differ only in the name of bound variables (i.e. we take terms modulo α-conversion). For example $(\lambda_{x_1})x_1 \equiv (\lambda_{x_2})x_2$. Furthermore, we assume the Barendregt variable convention which is formally stated as follows:

Definition 1.8 *(BC Barendregt's Convention for Λ)*
Names of bound variables will always be chosen such that they differ from the free ones in a term. Hence, we will not have $(v\delta)(\lambda_v)v$, but $(v\delta)(\lambda_{v'})v'$ instead. We extend BC to meta-terms *like $((\lambda_{v'})t_2)[v := t']$. This avoids the danger of clash of variables, since we assume that no free variable in t' occurs bound in $(\lambda_{v'})t_2$. Moreover, the extended BC implies that $v \not\equiv v'$.*

With our *BC*, we define substitution as follows:

Definition 1.9 *(Substitution in Λ with BC)*
If t, t' are Λ-terms and $v \in \mathcal{V}$ and if BC is assumed, then we define the result of substituting t' for all the free occurrences of v in t as follows:

$$t[v := t'] =_{df} \begin{cases} t' & \text{if } t \equiv v \\ v' & \text{if } t \equiv v' \not\equiv v \\ (t_1[v := t']\delta)t_2[v := t'] & \text{if } t \equiv (t_1\delta)t_2 \\ (\lambda_{v'})t_2[v := t'] & \text{if } t \equiv (\lambda_{v'})t_2 \end{cases}$$

With this implicit substitution, β-reduction is defined as follows:

Definition 1.10 *(β-reduction $\rightarrow_\beta$ in Λ)*
In Λ, β-reduction $\rightarrow_\beta$, is the least compatible relation generated out of the following axiom:

$$(\beta) \qquad (t''\delta)(\lambda_v)t' \rightarrow_\beta t'[v := t'']$$

2 Generalising redexes and β-reduction

As we have argued above, a term can contain a δ-item $(t_1\delta)$ and a λ_v-item (λ_v) such that eventually the reduction based on the $\delta\lambda$-segment $(t_1\delta)(\lambda_v)$ should take place. This reduction however, can only so far take place when $(t_1\delta)$ and (λ_v) are not separated by other items or segments. This makes it difficult to use the λ-calculus as a basis for many applications which depend heavily on manipulating the order in which reduction and substitution take place in

a term. Based on this observation, we shall in this section introduce a general β-reduction which enables the manipulation of the order of reduction and substitution. This general β-reduction will be an extension of the known β-reduction in that not only the $\delta\lambda$-segments result in firing reductions, but a more general notion of $\delta\lambda$-segments which we call $\delta\lambda$-couples.

2.1 Extending redexes to $\delta\lambda$-couples

Why should we in the term $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$ not allow that the reduction based on $(x_1\delta)(\lambda_{x_5})$ gets fired? There is no reason why we should not carry out some reductions before other ones. We ask the reader to convince himself of the fact that priority for the mentioned $\delta\lambda$-firing does not affect the final result. If we look moreover at this term, we find that what separates $(x_1\delta)$ and (λ_{x_5}) is a segment with a particular structure. The same holds for the segment $(t_2\delta)(\lambda_v)$ separating $(t_1\delta)$ and $(\lambda_{v'})$ in $(t_1\delta)(t_2\delta)(\lambda_v)(\lambda_{v'})$. These are the "well-balanced" structures as we discussed before and will define below. Basically, the idea is that when one desires to start a β-reduction on the basis of a certain δ-item and a λ-item occurring in one segment, the *matching* of the δ and the λ in question is the important thing, even when they are separated by other items. I.e., the relevant question is whether they *may* together become a $\delta\lambda$-segment after a number of β-steps. This depends solely on the structure of the intermediate segment. If such an intermediate segment is well-balanced then the δ-item and the λ-item match and β-reduction based on these two items may take place. Here is the definition of well-balanced segments:

Definition 2.1 *(well-balanced segments in Λ)*

- *An empty segment is a well-balanced segment.*
- *If $\overline{s}$ is a well-balanced segment, then $(t'\delta)\overline{s}(\lambda_v)$ is a well-balanced segment.*
- *The concatenation of well-balanced segments is a well-balanced segment.*

A well-balanced segment has the same structure as a matching composite of opening and closing brackets, each δ- (or λ-)item corresponding with an opening (resp. closing) bracket.

Example 2.2

1. $(x_3\delta)(\lambda_{x_6})$ forms a well-balanced segment as it corresponds to [ ].
2. $(x_2\delta)(\lambda_{x_7})$ forms a well-balanced segment as it corresponds to [ ].
3. $(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})$ forms a well-balanced segment as it corresponds to [ ][ ].
4. $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})$ forms a well-balanced segment as it corresponds to [[ ][ ]]; the $\delta\lambda$-couples in this segment are $(x_2\delta)(\lambda_{x_7})$, $(x_3\delta)(\lambda_{x_6})$ and $(x_1\delta)(\lambda_{x_5})$.

Now we can easily define what matching $\delta\lambda$-couples are. Namely, they are a δ-item and a λ-item separated by a well-balanced segment. Such couples are reducible couples. The δ-item and λ-item of the $\delta\lambda$-couple are said to match and each of them is called a partner or a partnered item. The items in a segment that are not partnered are called bachelor items. The following definition summarizes all this:

Definition 2.3 *(match, $\delta\lambda$- or reducible couple, partner, partnered item, bachelor item, bachelor segment)*
Let t be a Λ-term. Let $\overline{s} \equiv s_1 \cdots s_n$ be a segment occurring in t.

- *We say that the main items s_i and s_j* **match** *in $\overline{s}$, when $1 \leq i < j \leq n$, s_i is a δ-item, s_j is a λ-item, and the sequence $s_{i+1} \cdots s_{j-1}$ forms a well-balanced segment.*
- *When s_i and s_j match, we call $s_i s_j$ a* **$\delta\lambda$-couple** *or* **reducible couple.**
- *When s_i and s_j match, we call both s_i and s_j the* **partners** *in the $\delta\lambda$-couple. We also say that s_i and s_j are* **partnered** *items in $\overline{s}$ (or in t).*
- *All λ- (or δ-)items s_k that are not partnered in $\overline{s}$, are called* **bachelor** *λ- (resp. δ-)items in $\overline{s}$ (or in t).*
- *A segment $\overline{s}$ consisting of only bachelor items (in $\overline{s}$), is called a* **bachelor segment.**
- *The segment $s_{i_1} \cdots s_{i_m}$ consisting of all bachelor main λ- (or δ-)items of $\overline{s}$ is called the* **bachelor λ-** *(or δ-)***segment** *of $\overline{s}$*

Example 2.4 Let $\overline{s} \equiv (\lambda_{v_1})(\lambda_{v_2})(t_3\delta)(\lambda_{v_3})(\lambda_{v_4})(t_6\delta)(t_7\delta)(t_8\delta)(\lambda_{v_5})(\lambda_{v_6})(t_{11}\delta)$. Then:

- $(t_3\delta)$ matches with (λ_{v_3}), $(t_8\delta)$ matches with (λ_{v_5}) and $(t_7\delta)$ matches with (λ_{v_6}). The segments $(t_3\delta)(\lambda_{v_3})$ and $(t_8\delta)(\lambda_{v_5})$ are $\delta\lambda$-segments (also $\delta\lambda$-couples), and there is another $\delta\lambda$-couple in $\overline{s}$, viz. the couple of $(t_7\delta)$ and (λ_{v_6}).
- $(t_3\delta)$, (λ_{v_3}), $(t_7\delta)$, $(t_8\delta)$, (λ_{v_5}) and (λ_{v_6}), are the partnered main items of $\overline{s}$. (λ_{v_1}), (λ_{v_2}), (λ_{v_4}), $(t_6\delta)$ and $(t_{11}\delta)$, are bachelor items.
- $(\lambda_{v_1})(\lambda_{v_2})$ and $(\lambda_{v_4})(t_6\delta)$ are bachelor segments. $(t_7\delta)(t_8\delta)(\lambda_{v_5})$ and $(t_7\delta)(t_8\delta)(\lambda_{v_5})(\lambda_{v_6})$ are non-bachelor segments, the latter also being a well-balanced segment.

Remark 2.5
Observe that a *$\delta\lambda$-segment* is a $\delta\lambda$-couple and that the δ-item and the λ-item in a *$\delta\lambda$-couple* are separated by zero, one or more $\delta\lambda$-couples.

2.2 Extending β-reduction and the Church Rosser theorem

Having argued above that β-reduction should not be restricted to the $\delta\lambda$-segments but may take into account other candidates, we can extend our notion of β-reduction in this vein. That is to say, we may allow *$\delta\lambda$-couples* to have the same "reduction rights" as *$\delta\lambda$-segments.* That is, the β-reduction of Definition 1.10 changes to the following:

Definition 2.6 *(General β-reduction $\leadsto_\beta$ in Λ)*
General β-reduction $\leadsto_\beta$, is the least compatible relation generated out of the following axiom:

$$(general\ \beta) \qquad (t''\delta)\overline{s}(\lambda_v)t' \leadsto_\beta \overline{s}t'[v := t''] \qquad \text{if } \overline{s} \text{ is well-balanced}$$

Example 2.7 Take Example 1.4. As $(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})$ is a well-balanced segment, then $(x_1\delta)(\lambda_{x_5})$ is a $\delta\lambda$-couple and

$$\begin{array}{ll} t \equiv (x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5 & \leadsto_\beta \\ (x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})((x_4\delta)x_5)[x_5 := x_1] & \equiv \\ (x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(x_4\delta)x_1 & \end{array}$$

The *reducible couple* $(x_1\delta)(\lambda_{x_5})$ also has a corresponding ("generalized") redex in the traditional notation, which will appear after two one-step β-reductions, leading to $(\lambda_{x_5}.x_5x_4)x_1$. With our generalised one-step β-reduction we could reduce $((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2)x_1$ to $(\lambda_{x_7}.(\lambda_{x_6}.x_1x_4)x_3)x_2$. This reduction is difficult to carry out in the classical λ-calculus. The item notation enables a new and important sort of reduction which has not yet been studied in relation to the standard λ-calculus up to date. We believe that this generalised reduction (introduced in [Nederpelt 73]) can only be obtained tidily in a system formulated using our item notation. In fact, one is to compare the bracketing structure of the classical term t of Example 1.4, with the bracketing structure of the corresponding term in item notation:

Example 2.8 The "bracketing structure" of the maximal main segment of t of Example 1.4, (that is, of $((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.--)x_3)x_2)x_1$), is compatible with '$[_1\ [_2\ [_3\ ]_2\ ]_1\ ]_3$', where '$[_i$' and '$]_i$' match. In item notation however, t has the bracketing structure [[ ][ ]].

We strongly believe that it is the item notation which enables us to extend reduction smoothly beyond the existing $\twoheadrightarrow_\beta$. Because a well-balanced segment may be empty, the general β-reduction rule presented above is really an extension of the classical β-reduction rule.

Lemma 2.9 *Let t_1, t_2 be Λ-terms. If $t_1 \twoheadrightarrow_\beta t_2$ then $t_1 \rightsquigarrow_\beta t_2$. Moreover, if $t_1 \rightsquigarrow_\beta t_2$ comes from contracting a $\delta\lambda$-segment then $t_1 \twoheadrightarrow_\beta t_2$.*

Proof: *Obvious as a $\delta\lambda$-segment is a $\delta\lambda$-couple by Remark 2.5.* □

The proof of the Church Rosser theorem is simple. The idea is to show that if $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t'$ then $t =_\beta t'$ (where $=_\beta$ is the least equivalence relation closed under $\twoheadrightarrow_\beta$ and $\rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta$ the reflexive transitive closure of $\rightsquigarrow_\beta$) and to use the Church Rosser property for $=_\beta$.

Lemma 2.10 *If $t \rightsquigarrow_\beta t'$ then $t =_\beta t'$.*

Proof: *It suffices to consider the case $t \equiv \overline{s_1}(t_1\delta)\overline{s}(\lambda_v)t_2$ where the contracted redex is based on $(t_1\delta)(\lambda_v)$, $t' \equiv \overline{s_1}\,\overline{s}t_2[v := t_1]$, and $\overline{s}$ is balanced (hence* $\mathtt{weight}(\overline{s})$ *is even). We shall prove the lemma by induction on* $\mathtt{weight}(\overline{s})$.

- *Case* $\mathtt{weight}(\overline{s}) = 0$ *then obvious as $\rightsquigarrow_\beta$ coincides with $\twoheadrightarrow_\beta$ in this case.*
- *Assume the property holds when* $\mathtt{weight}(\overline{s}) = 2n$. *Take $\overline{s}$ such that* $\mathtt{weight}(\overline{s}) = 2n+2$. *Now, $\overline{s} \equiv (t_3\delta)\overline{s'}(\lambda_{v'})\overline{s''}$ where $\overline{s'}$, $\overline{s''}$ are well-balanced.*
  - *As $\overline{s}t_2[v := t_1] \rightsquigarrow_\beta \overline{s'}(\overline{s''}t_2[v := t_1])[v' := t_3]$, we get by IH and compatibility that $t' =_\beta \overline{s_1}\,\overline{s'}(\overline{s''}t_2[v := t_1])[v' := t_3] \equiv \overline{s_1}\,\overline{s'}\,\overline{s''}[v' := t_3]t_2[v := t_1][v' := t_3]$.*
  - *Moreover, $t \equiv \overline{s_1}(t_1\delta)(t_3\delta)\overline{s'}(\lambda_{v'})\overline{s''}(\lambda_v)t_2 \rightsquigarrow_\beta \overline{s_1}(t_1\delta)\overline{s'}(\overline{s''}(\lambda_v)t_2)[v' := t_3] \equiv^{BC} \overline{s_1}(t_1\delta)\overline{s'}\,\overline{s''}[v' := t_3](\lambda_v)t_2[v' := t_3] \equiv t_4$. Hence by III, $t =_\beta t_4$.*
  - *Now, $t_4 \rightsquigarrow_\beta \overline{s_1}\overline{s'}\,\overline{s''}[v' := t_3]t_2[v' := t_3][v := t_1]$. But by BC, v, $v' \notin FV(t_1) \cup FV(t_3)$. Hence, by III and substitution, $t_4 =_\beta \overline{s_1}\overline{s'}\,\overline{s''}[v' := t_3]t_2[v := t_1][v' := t_3]$.*

  Therefore $t =_\beta t'$. □

Corollary 2.11 *If $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t'$ then $t =_\beta t'$.* □

Theorem 2.12 *The general β-reduction is Church-Rosser. I.e. If $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t_1$ and $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t_2$, then there exists t_3 such that $t_1 \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t_3$ and $t_2 \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t_3$.*

Proof: *As $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t_1$ and $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t_2$ then by Corollary 2.11, $t =_\beta t_1$ and $t =_\beta t_2$. Hence, $t_1 =_\beta t_2$ and by the Church Rosser property for the classical lambda calculus, there exists t_3 such that $t_1 \twoheadrightarrow_\beta t_3$ and $t_2 \twoheadrightarrow_\beta t_3$. But, $t \twoheadrightarrow_\beta t'$ implies $t \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta t'$. Hence the Church-Rosser theorem holds for the general β-reduction.* □

3 Term reshuffling

Let us go back to the definition of $\delta\lambda$-couples. Recall that if $\overline{s} \equiv s_1 \cdots s_m$ for $m > 1$ where $s_1 s_m$ is a $\delta\lambda$-couple then $s_2 \cdots s_{m-1}$ is a well-balanced segment, $s_1 \equiv (t_1\delta)$ is the δ-item of the $\delta\lambda$-couple and $s_m \equiv (\lambda_v)$ is its λ-item. Now, we can move s_1 in $\overline{s}$ so that it occurs adjacently to s_m. That is, we may rewrite $\overline{s}$ as $s_2 \cdots s_{m-1} s_1 s_m$.

Example 3.1 In our item notation, the term $(x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$ can be easily rewritten as $(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(x_1\delta)(\lambda_{x_5})(x_4\delta)x_5$ by moving the item $(x_1\delta)$ to the right. Hence, we can rewrite (or *reshuffle*) a term so that all δ-items stand next to their matching λ-items. This means that we can keep the old β-axiom and we can contract redexes in any order. Such an action of reshuffling is not easy to describe in the classical notation. That is, it is difficult to describe how $((\lambda_{x_7}.(\lambda_{x_6}.\lambda_{x_5}.x_5x_4)x_3)x_2)x_1$, is rewritten as $(\lambda_{x_7}.(\lambda_{x_6}.(\lambda_{x_5}.x_5x_4)x_1)x_3)x_2$. This is another advantage of our item notation.

Note furthermore that in Λ, the shuffling is not problematic due to the Barendregt Convention which means that no free variable will become unnecessarily bound after reshuffling due to the fact that names of bound and free variables are distinct.

Lemma 3.2
If v° is a free occurrence of v in $s\overline{s_1}\,t$, then v° is free in $\overline{s_1}s\,t$.
Proof: *By BC as λ_v does not occur in $s\overline{s_1}t$.* □

Example 3.3 Note that in Example 3.1, reshuffling does not affect the "meaning" of the term. In fact, in $t \equiv (x_1\delta)(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(\lambda_{x_5})(x_4\delta)x_5$, the free variable x_1 cannot be captured by λ_{x_7} or λ_{x_6}. Moreover, t is equivalent, semantically and procedurally, to $(x_2\delta)(\lambda_{x_7})(x_3\delta)(\lambda_{x_6})(x_1\delta)(\lambda_{x_5})(x_4\delta)x_5$.

That is, the δ-items of $\delta\lambda$-couples can occupy different positions in a term, without disturbing the meaning of the term, both semantically and procedurally. We call this process of moving δ-items of $\delta\lambda$-couples in a term to occupy positions adjacent to their λ-partners, *term reshuffling*. This term reshuffling should be such that *all* the δ-items of well-balanced segments in a term are shifted to the right until they meet their λ-partners. Before we define term reshuffling, we need to understand better the structure of terms. Therefore the following section.

3.1 Partitioning the term into bachelor and well-balanced segments

Convention 3.4 In what follows, we will use $(t\omega)$ to refer to $(t\delta)$ when $\omega = \delta$ and to $(\varepsilon\lambda_v)$ or (λ_v) when $\omega \equiv \lambda_v$. I.e. we extend Λ with ε and we take ε to represent the empty term. At the moment, this is only to enable us to use a unified notation for items in our definitions. (However, this opens the possibility for an extension to typed λ-calculus with items $(t\lambda_v)$, where t is the type of v.)

With Definition 2.3 and Example 2.4, we may categorize the main items of a term t into different classes:

1. The "partnered" items (i.e. the δ- and λ-items which are partners, hence "coupled" to a matching one).

2. The "bachelors" (i.e. the bachelor λ-items and bachelor δ-items).

Lemma 3.5
Let $\overline{s}$ be the body of a term t. Then the following holds:

1. *Each bachelor main λ-item in $\overline{s}$ precedes each bachelor main δ-item in $\overline{s}$.*
2. *The removal from $\overline{s}$ of all bachelor main items, leaves behind a* well-balanced *segment.*
3. *The removal from $\overline{s}$ of all main $\delta\lambda$-couples, leaves behind a $\underbrace{\lambda\ldots\lambda}_{n}\underbrace{\delta\ldots\delta}_{m}$-segment, consisting of all bachelor main λ- and δ-items.*
4. *If $\overline{s} \equiv \overline{s_1}(t'\delta)\overline{s_2}(\lambda_v)\overline{s_3}$ where (λ_v) and $(t'\delta)$ match, then $\overline{s_2}$ is well-balanced.*

Proof: *1 is by induction on* $\mathtt{weight}(\overline{s'})$ *for $\overline{s} \equiv \overline{s'}(\lambda_v)\overline{s''}$ and (λ_v) bachelor in $\overline{s}$. 2 and 3 are by induction on* $\mathtt{weight}(\overline{s})$. *4 is by induction on* $\mathtt{weight}(\overline{s_2})$. □

Note that we have assumed ε well-balanced. We assume it moreover non-bachelor.

Corollary 3.6 *For each non-empty segment $\overline{s}$, there is a unique* partitioning *in segments $\overline{s_0}, \overline{s_1}, \cdots, \overline{s_n}$, such that*

1. $\overline{s} \equiv \overline{s_0}\,\overline{s_1}\cdots\overline{s_n}$,
2. *For all $0 \leq i \leq n$, $\overline{s_i}$ is well-balanced in $\overline{s}$ for even i and $\overline{s_i}$ is bachelor in $\overline{s}$ for odd i.*
3. *each bachelor λ-segment $\overline{s_j}$ precedes each bachelor δ-segment $\overline{s_k}$ in $\overline{s}$.*
4. *If $i \geq 1$ then $\overline{s_{2i}} \not\equiv \varepsilon$.*

□

Example 3.7
The segment $\overline{s} \equiv (\lambda_{v_1})(\lambda_{v_2})(t_3\delta)(\lambda_{v_3})(\lambda_{v_4})(t_6\delta)(t_7\delta)(t_8\delta)(\lambda_{v_5})(\lambda_{v_6})(t_{11}\delta)$, has the following partitioning:

- well-balanced segment $\overline{s_0} \equiv \varepsilon$
- bachelor segment $\overline{s_1} \equiv (\lambda_{v_1})(\lambda_{v_2})$,
- well-balanced segment $\overline{s_2} \equiv (t_3\delta)(\lambda_{v_3})$,
- bachelor segment $\overline{s_3} \equiv (\lambda_{v_4})(t_6\delta)$,
- well-balanced segment $\overline{s_4} \equiv (t_7\delta)(t_8\delta)(\lambda_{v_5})(\lambda_{v_6})$.
- bachelor segment $\overline{s_5} \equiv (t_{11}\delta)$.

3.2 The reshuffling procedure

Definition 3.8 *TS and T are defined mutually recursively such that:*

$$
\begin{array}{lll}
TS(\varepsilon) & =_{df} & \varepsilon \\
TS(\overline{s}v) & =_{df} & TS(\overline{s})v \\
TS((t_1\omega_1)\cdots(t_n\omega_n)) & =_{df} & (TS(t_1)\omega_1)\cdots(TS(t_n)\omega_n) \quad \textit{if } (t_1\omega_1)\cdots(t_n\omega_n) \textit{ is bachelor} \\
TS(\overline{s}) & =_{df} & T(\varepsilon,\overline{s}) \quad \textit{if } \overline{s} \textit{ is well-balanced} \\
TS(\overline{s_0}\cdots\overline{s_n}) & =_{df} & TS(\overline{s_0})\cdots TS(\overline{s_n}) \quad \textit{If } \overline{s_0}\cdots\overline{s_n}\textit{, is the unique partitioning of Corollary 3.6} \\
T(\overline{s}(t\delta),(\lambda_v)\overline{s'}) & =_{df} & (t\delta)(\lambda_v)T(\overline{s},\overline{s'}) \\
T(\overline{s},(t\delta)\overline{s'}) & =_{df} & T(\overline{s}(TS(t)\delta),\overline{s'}) \\
T(\varepsilon,\varepsilon) & =_{df} & \varepsilon
\end{array}
$$

Note that in this definition, we use $\overline{s}$ bachelor to mean $\overline{s}$ bachelor in $\overline{s}$.

Lemma 3.9

1. *If $\overline{s}$ is well-balanced, then $T(\overline{s_1},\overline{s}\,\overline{s_2}) \equiv TS(\overline{s})T(\overline{s_1},\overline{s_2})$.*
2. *If $\overline{s}$ is well-balanced and none of its binding variables are free in t, then $TS((t\delta)\overline{s}(\lambda_v)\overline{s'}) \equiv TS(\overline{s}(t\delta)(\lambda_v)\overline{s'})$.*
3. *If $\overline{s}$ contains no items which are partnered in t then $TS(\overline{s}t) \equiv TS(\overline{s})TS(t)$.*
4. *If $\overline{s}$ is well-balanced or is bachelor in $\overline{s}t$ then $TS(\overline{s}t) \equiv TS(\overline{s})TS(t)$.*

Proof: *1: by induction on* $\mathtt{weight}(\overline{s})$. *Case* $\mathtt{weight}(\overline{s}) = 0$ *then obvious. Case* $\overline{s} \equiv (t\delta)\overline{s'}(\lambda_v)\overline{s''}$ *then*

$$
\begin{array}{lcll}
T(\overline{s_1},(t\delta)\overline{s'}(\lambda_v)\overline{s''}\,\overline{s_2}) & \equiv & T(\overline{s_1}(TS(t)\delta),\overline{s'}(\lambda_v)\overline{s''}\,\overline{s_2}) & =^{IH} \\
TS(\overline{s'})T(\overline{s_1}(TS(t)\delta),(\lambda_v)\overline{s''}\,\overline{s_2}) & \equiv & TS(\overline{s'})(TS(t)\delta)(\lambda_v)T(\overline{s_1},\overline{s''}\,\overline{s_2}) & =^{IH} \\
TS(\overline{s'})(TS(t)\delta)(\lambda_v)TS(\overline{s''})T(\overline{s_1},\overline{s_2}) & \equiv & TS(\overline{s'})T((TS(t)\delta),(\lambda_v)\overline{s''})T(\overline{s_1},\overline{s_2}) & =^{IH} \\
T((TS(t)\delta),\overline{s'}(\lambda_v)\overline{s''})T(\overline{s_1},\overline{s_2}) & \equiv & TS((t\delta)\overline{s'}(\lambda_v)\overline{s''})T(\overline{s_1},\overline{s_2}) &
\end{array}
$$

2: using 1. 3: let $t \equiv \overline{s_0}\cdots\overline{s_n}v$ and $\overline{s} \equiv \overline{s'_0}\cdots\overline{s'_m}$ be partitionings. Use cases on $\overline{s_0}$ being empty or not and on $\overline{s'_m}$ being bachelor or well-balanced. 4: This is a corollary of 3 above in case $\overline{s}$ is bachelor. If $\overline{s}$ is well-balanced then let $t \equiv \overline{s_0}\cdots\overline{s_n}v$. If $\overline{s_0} \equiv \varepsilon$, nothing to prove, else note from 1, that $TS(\overline{s}\,\overline{s_0}) \equiv TS(\overline{s})TS(\overline{s_0})$. □

The following lemma shows that $TS(t)$ changes all $\delta\lambda$-couples of t to $\delta\lambda$-segments.

Lemma 3.10 *For every subterm t' of a term t, the following holds:*

1. *$TS(t')$ is well-defined.*
2. *If $\overline{s} \equiv (t_1\delta)\overline{s'}(\lambda_v)$ is a subsegment of t' and $(t_1\delta)$ matches (λ_v), then $TS(\overline{s}) \equiv TS(\overline{s'})(TS(t_1)\delta)(\lambda_v)$.*
3. *If $\overline{s} \equiv (t_1\omega_1)\cdots(t_n\omega_n)$ is a bachelor subsegment of t', then $TS(\overline{s}) \equiv (TS(t_1)\omega_1)\cdots(TS(t_n)\omega_n)$ is a bachelor subsegment of $TS(t')$.*
4. *If $\overline{s}$ is a subsegment of t' which is well-balanced, then $TS(\overline{s})$ is well-balanced.*

Proof: *By induction on t.*

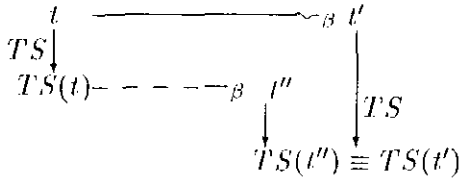
- *Case $t \equiv v$ then t is the unique subterm of t and all $1 \cdots 4$ hold.*
- *Assume $t \equiv (t_1\omega)t_2$ where IH holds for t_1 and t_2. Let t' be a subterm of t. If t' is a subterm of t_1 or t_2 then use IH. If $t' \equiv t$ then:*
  - *Case $(t_1\omega)$ is bachelor then $TS(t) \equiv^{Lemma\ 3.9} (TS(t_1)\omega)TS(t_2)$. Here all $1 \cdots 4$ hold by IH on t_1 and t_2.*
  - *Case $(t_1\delta)$ matches (λ_v) in t. I.e. $t \equiv (t_1\delta)\overline{s}(\lambda_v)t_3$ then $TS(t) \equiv^{Lemma\ 3.9} TS(\overline{s})(TS(t_1)\delta)(\lambda_v)TS(t_3)$. Now use IH to show $1 \cdots 4$.* □

Lemma 3.11 *For all variables v and terms t, t' we have:*
$TS(t) \equiv TS(TS(t))$ and $TS(t[v := t']) \equiv TS(TS(t)[v := TS(t')])$.

Proof: *By induction on t we show that for all subterms t'' of t, $TS(t'') \equiv TS(TS(t''))$ and $TS(t''[v := t']) \equiv TS(TS(t'')[v := TS(t')])$.* □

Note that if $t \to_\beta t'$ and if all the $\delta\lambda$-couples in t are $\delta\lambda$-segments, then it is not necessary that all the $\delta\lambda$-couples of t' are $\delta\lambda$-segments. In other words, we can have $TS(t_1) \to_\beta t_2$ where $t_2 \not\equiv TS(t_2)$. For example, $(x_1\delta)(x_2\delta)(\lambda_{x_3})((\lambda_{x_4})x_4\delta)(\lambda_{x_5})x_5 \to_\beta (x_1\delta)(x_2\delta)(\lambda_{x_3})(\lambda_{x_4})x_4$. Following this remark, we show that in a sense, term reshuffling preserves β-reduction.

Lemma 3.12 *If $t, t' \in \Lambda$ and $t \sim_\beta t'$ then $(\exists t'')[(TS(t) \to_\beta t'') \wedge TS(t'') \equiv TS(t')]$. In other words, the following diagram commutes:*



Proof: *By induction on the general $\sim_\beta$. As the compatibility case is easy, we will only consider the case where $t \equiv \overline{s'}(t_1\delta)\overline{s}(\lambda_v)t_3 \sim_\beta t' \equiv \overline{s'}\,\overline{s}t_3[v := t_1]$. Here, we use induction on the number n of bachelor δ-items of $\overline{s'}$ that are partnered in t_3. Recall that $\overline{s}$ is well-balanced.*

- *Case $n = 0$ then*

$$
\begin{array}{ll}
TS(\overline{s'}(t_1\delta)\overline{s}(\lambda_v)t_3) & \equiv^{Lemma\ 3.9\,(4)} \\
TS(\overline{s'})TS((t_1\delta)\overline{s}(\lambda_v))TS(t_3) & \equiv^{Lemma\ 3.9\,(2,4)} \\
TS(\overline{s'})TS(\overline{s})(TS(t_1)\delta)(\lambda_v)TS(t_3) & \to_\beta \\
TS(\overline{s'})TS(\overline{s})TS(t_3)[v := TS(t_1)] & \equiv t''. \\
TS(t'') & \equiv^{Lemmas\ 3.10,\ 3.11,\ 3.9\,(4)} \\
TS(\overline{s'})TS(\overline{s})TS(TS(t_3)[v := TS(t_1)]) & \equiv^{Lemma\ 3.11} \\
TS(\overline{s'})TS(\overline{s})TS(t_3[v := t_1]) & \equiv \\
TS(\overline{s'}\,\overline{s}t_3[v := t_1]) &
\end{array}
$$

- *Assume the property holds for n and let us show it for the case where $\overline{s'}$ contains $n+1$ δ-items which match λ-items of t_3. Let $(t''\delta)$ be the leftmost such δ-item of $\overline{s'}$. Take $\overline{s'} \equiv \overline{s'_1}(t''\delta)\overline{s''_1}$ and $t_3 \equiv \overline{s'_2}(\lambda_{v'})t_2$ where $(t''\delta)$ matches $(\lambda_{v'})$. By Lemma 3.5, $(t''\delta)\overline{s''_1}(t_1\delta)\overline{s}(\lambda_v)\overline{s'_2}(\lambda_{v'})$ is well-balanced. Moreover, no item of $\overline{s'_1}$ has a partner in $(t''\delta)\overline{s''_1}(t_1\delta)\overline{s}(\lambda_v)t_3$. As $\overline{s''_1}(t_1\delta)\overline{s}(\lambda_v)\overline{s'_2}(t''\delta)(\lambda_{v'})t_2 \sim_\beta \overline{s''_1}\overline{s}(\overline{s'_2}(t''\delta)(\lambda_{v'})t_2)[v := t_1]$, we find by III, t''' such that*
  $TS(\overline{s''_1}(t_1\delta)\overline{s}(\lambda_v)\overline{s'_2}(t''\delta)(\lambda_{v'})t_2) \rightarrow_\beta t''' \wedge TS(t''') \equiv TS(\overline{s''_1}\overline{s}(\overline{s'_2}(t''\delta)(\lambda_{v'})t_2)[v := t_1])$.
  Now, $TS(\overline{s'_1})t'''$ is the wanted term because:
  $TS(t) \equiv^{Lemma\ 3.9(4)} TS(\overline{s'_1})TS((t''\delta)\overline{s''_1}(t_1\delta)\overline{s}(\lambda_v)\overline{s'_2}(\lambda_{v'})t_2) \equiv^{Lemma\ 3.9(2)}$
  $TS(\overline{s'_1})TS(\overline{s''_1}(t_1\delta)\overline{s}(\lambda_v)\overline{s'_2}(t''\delta)(\lambda_{v'})t_2) \rightarrow_\beta TS(\overline{s'_1})t'''$
  and $TS(TS(\overline{s'_1})t''') \equiv^{Lemma\ 3.11}$
  $TS(\overline{s'_1})TS(\overline{s''_1}\overline{s}(\overline{s'_2}(t''\delta)(\lambda_{v'})t_2)[v := t_1]) \equiv^{Lemma\ 3.9(2),BC}$
  $TS(\overline{s'_1})TS((t''\delta)\overline{s''_1}\overline{s}\overline{s'_2}[v := t_1](\lambda_{v'})t_2[v := t_1]) \equiv^{Lemma\ 3.9(4)}$
  $TS(\overline{s'_1}(t''\delta)\overline{s''_1}\overline{s}(\overline{s'_2}(\lambda_{v'})t_2)[v := t_1]) \equiv TS(t')$. □

Corollary 3.13 *If $t \rightsquigarrow_\beta t'$ then there exists exists $t_1.t_2.\cdots t_n$ such that:*
$[(TS(t) \rightarrow_\beta t_1) \wedge (TS(t_1) \rightarrow_\beta t_2) \wedge \cdots \wedge (TS(t_{n-1}) \rightarrow_\beta t_n) \wedge (TS(t_n) \equiv TS(t'))]$.

Proof: *By induction on $\rightsquigarrow_\beta$.*

- *Case $t \sim_\beta t'$ use Lemma 3.12.*
- *Case $t \rightsquigarrow_\beta t$ then obvious ($n = 1 \wedge t_1 \equiv TS(t)$).*
- *Case $t' \rightsquigarrow_\beta t'' \wedge t'' \rightsquigarrow_\beta t'''$, then by III. there exists $t_1, t_2, \cdots, t_n, t'_1, t'_2, \cdots t'_m$ such that $(TS(t') \rightarrow_\beta t_1) \wedge (TS(t_1) \rightarrow_\beta t_2) \wedge \cdots \wedge (TS(t_{n-1}) \rightarrow_\beta t_n) \wedge (TS(t_n) \equiv TS(t'')) \wedge$ $(TS(t'') \rightarrow_\beta t'_1) \wedge (TS(t'_1) \rightarrow_\beta t'_2) \wedge \cdots \wedge (TS(t'_{m-1}) \rightarrow_\beta t'_m) \wedge (TS(t'_m) \equiv TS(t''')))$. Hence, $(TS(t') \rightarrow_\beta t_1) \wedge \cdots \wedge (TS(t_{n-1}) \rightarrow_\beta t_n) \wedge (TS(t_n) \rightarrow_\beta t'_1) \wedge \cdots \wedge (TS(t'_{m-1}) \rightarrow_\beta t'_m) \wedge (TS(t'_m) \equiv TS(t'''))$.*
- *The compatibility case is easy.*

Note that in the basic case and the reflexive case we get $n = 1$ for sure. In the transitive case, this may not be the case. For example, $t \equiv (\lambda_{x_1})((\lambda_{x_2})(\lambda_{x_3})x_1\delta)(\lambda_{x_4})(x_1\delta)(x_1\delta)x_4 \rightsquigarrow_\beta t' \equiv (\lambda_{x_1})(x_1\delta)(\lambda_{x_2})x_1$ and does not satisfy Lemma 3.12. There is however t_1 and t_2 such thatt$_1 \equiv (\lambda_{x_1})(x_1\delta)(x_1\delta)(\lambda_{x_2})(\lambda_{x_3})x_1$ and $t_2 \equiv t'$ such that $TS(t) \rightarrow_\beta t_1 \wedge TS(t_1) \rightarrow_\beta t_2$ and $TS(t_2) \equiv TS(t')$. □

4 Conclusion

Classical reduction is in certain respects unattractive, not only because we should be able to discuss the existing redexes in a term at first sight, but also for at least two more reasons:

1. It may be the case that in some applications, we may want to contract a certain redex before other ones. This is the case for example in lazy evaluation where we may be interested in freezing some redexes but in working with the term as usual. That is for example, in the term $t \equiv ((\lambda_{x_7}.(\lambda_{x_6}.x_5x_4)x_3)x_2)x_1$, we may want to substitute x_1 for x_5 in $(\lambda_{x_7}.(\lambda_{x_6}.x_5x_4)x_3)x_2$ before x_3 has been substituted for x_6 and x_2 has been substituted for x_7. I.e. we need to carry out the reduction which substitutes x_1 for x_5 while the other two reductions which substitute x_3 and x_2 for x_6 and x_7 respectively are frozen.

2. Having a term like t above, we want to be able to discuss its *needed* redexes for reasons that are explained in [BKKS 87]. In fact, the needed redexes in a term t' as introduced in that paper are those redexes that are contracted in every reduction of t' to normal form. Now, [BKKS 87] provides many results which are important especially for the implementor in that it frees him from having to stick to either an inefficient but terminating normal order strategy, or an efficient but non terminating applicative strategy. Our approach enhances the work of [BKKS 87].

In this paper, we presented a notation which enables us to extend the classical notion of a redex and of β-reduction. This extension helps us to see more needed redexes than in classical calculus. The notation moreover allows us to reshuffle the term in hand so that more redexes can become visible and be contracted even using the old β-reduction. Both the generalised reduction and the reshuffling of the term are difficult to describe in the classical notation.

The notation presented in this paper has further advantages than generalising reduction and term reshuffling. These advantages are studied in our articles mentioned in the bibliography. Of these advantages however, we mention the ability to describe substitution explicitly as in [KN 93] and of generalising type systems as in [NK 94]. There is moreover the advantage of being able to make normal order reduction more efficient than in the classical calculus. The reason for this being that when searching for the leftmost outermost redex in a term, we need to make less recursive calls in item notation than in classical notation because in item notation, a term has a more linear structure. This and other advantages are investigated further in [KN 9z].

References

[BKKS 87] Barendregt, H.P., Kennaway, J.R., Klop, J.W., and Sleep M.R., Needed reduction and spine strategies for the λ-calculus, *Information and Computation 75 (3)*, 1191-231, 1987.

[de Bruijn 93] Bruijn, N.G. de, Algorithmic definition of lambda-typed lambda calculus, in Huet, G. and Plotkin, G. eds. *Logical Environments*, 131-146, Cambridge University Press, 1993.

[KN 93] Kamareddine, F., and Nederpelt, R.P., On stepwise explicit substitution, *International Journal of Foundations of Computer Science 4 (3)*. 197-240, 1993.

[KN 9z] Kamareddine, F., and Nederpelt, R.P., *The beauty of the λ-calculus*, in preparation.

[Nederpelt 73] Nederpelt, R.P., *Strong normalisation in a typed lambda calculus with lambda structured types*, Ph.D. thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science, 1973, to appear in Nederpelt, R.P., Geuvers, J.H. and de Vrijer, R.C.: *Selected Papaers on Automath*, North Holland, 1994.

[NK 94] Nederpelt, R.P., and Kamareddine, F., A unified approach to type theory through a refined λ-calculus, paper presented at the 1992 conference on *Mathematical Foundations of Programming Semantics*, to appear in the proceedings.

Computing Science Notes

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.

91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.
91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.

91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.
92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.

92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.

93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness?, p. 24.
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.
93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.

93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.
94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doombos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.