# De Bruijn's syntax and reductional equivalence of $\lambda$-terms


Fairouz Kamareddine
Computing & Electrical Eng.
Heriot-Watt University
Riccarton
Edinburgh EH14 4AS
Scotland
fairouz@cee.hw.ac.uk

Roel Bloo
Mathematics & Computing Sc.
Eindhoven University
P.O.Box 513
5600 MB Eindhoven
The Netherlands
c.j.bloo@tue.nl

Rob Nederpelt
Mathematics & Computing Sc.
Eindhoven University
P.O.Box 513
5600 MB Eindhoven
The Netherlands
r.p.nederpelt@tue.nl



## ABSTRACT

In this paper, a notation influenced by de Bruijn's syntax of the $\lambda$-calculus is used to describe canonical forms of terms and an equivalence relation which divides terms into classes according to their reductional behaviour. We show that this notation helps describe canonical forms more elegantly than the classical notation and we establish the desirable properties of our reduction modulo equivalence classes rather than single terms. Finally, we extend the cube consisting of eight type systems with class reduction and show that this extension satisfies all the desirable properties of type systems.

## Summary

In $\lambda$-calculus, a $\beta$-redex $(\lambda_x.A)B$ is characterised by the matching of $\lambda_x$ with the application argument $B$. We say that $\lambda_x$ and $B$ match or that each has the other as a partner. In a $\lambda$-term however, there can occur $\lambda_x$'s and application arguments which do not have any partners (i.e., are bachelor). In terms like $((\lambda_x.\lambda_y.A)B)C$, we see that $\lambda_y$ and $C$ are bachelors. However, after a reduction matching $\lambda_x$ and $B$, a new redex based on the then matching $\lambda_y$ and $C$ is created. This has been noted by [12, 18] who provided for each term a canonical form which shows which parts of the term are certainly partnered and which are inherently bachelor, now or in the future. This canonical form has the shape:

$$\lambda x_1 \cdots \lambda x_n.(\lambda y_1.(\lambda y_2.(\cdots.(\lambda y_m.zA_1 \cdots A_l)C_m)\cdots)C_2)C_1$$

where $\lambda x_i$ and $A_j$ are bachelor for $1 \leq i \leq n$ and $1 \leq j \leq l$ and each $C_i$ for $1 \leq i \leq m$ matches $\lambda y_i$. In addition, [18] provided the notion of $\sigma$-equivalence which identifies terms only differing by permutations of redexes, and showed that none of the standard operational classification criteria on $\lambda$-calculus (e.g., length of longest reduction) can separate two $\sigma$-equivalent terms. [18] concluded by asking if there existed a syntax that realises $\sigma$-equivalence. In this paper, we attempt to answer the question by using the item notation [5] inspired by de Bruijn's notation of the $\lambda$-calculus. Using item notation (where abstraction and application are written respectively as $(\lambda_x)A$ and $(B\delta)C$ with $C$ the function and $B$ the argument) to represent canonical forms and reductional equivalence, we find that:

- $(\lambda_{x_1})\cdots(\lambda_{x_n})(C_1\delta)(\lambda_{y_1})\cdots(C_m\delta)(\lambda_{y_m})(A_l\delta)\cdots(A_1\delta)z$ becomes the canonical form is which is clearly divided into a sequence of bachelor $\lambda$-items $(\lambda_{x_i})$ followed by a sequence of partnered pairs $(C_j\delta)(\lambda_{y_j})$ followed by a sequence of bachelor $\delta$-items $(A_k\delta)$ which is finally followed by the heart of the term $z$. This is clearer than the canonical form given in [12, 18].
- We are also able to define a decidable notion of reductional equivalence $\approx_{\texttt{equi}}$ on terms which we show to be equivalent to $\sigma$-equivalence.
- Using the result of Regnier in [18], we show that two terms have similar reduction paths and reductional behaviour if they have the same canonical forms up to a permutation of partnered pairs $(\lambda_{y_i}, C_i)$ and $(\lambda_{y_j}, C_j)$ in the canonical form, provided that no bound variables become free during the permutation.
- We proceed beyond the results of Regnier to extend the usual $\beta$-reduction $\rightarrow_\beta$ on $\lambda$-terms to $\rightsquigarrow_\beta$ on classes of terms modulo reductional equivalence which is Church-Rosser, which commutes with reductional equivalence and preserves reductional paths, and where $SN_{\rightsquigarrow_\beta}$ and $SN_{\rightarrow_\beta}$ are equivalent.

Finally, we extend the Barendregt cube with $\rightsquigarrow_\beta$ and show that this extension satisfies all the properties such as strong normalisation and subject reduction (the latter depends on allowing definitions in contexts). As far as we know this is the first account of generalising reduction in the cube using classes of terms.


## 1. INTRODUCTION

The last two decades have seen an explosion in new notions of reductions which can be summarised by four axioms:

($\theta$) $((\lambda_x.N)P)Q \rightarrow_\theta (\lambda_x.NQ)P$.
($\gamma$) $(\lambda_x.\lambda_y.N)P \rightarrow_\gamma \lambda_y.(\lambda_x.N)P$.
($g$) $((\lambda_x.\lambda_y.N)P)Q \rightarrow_g (\lambda_x.N[y := Q])P$.
($\gamma_C$) $((\lambda_x.\lambda_y.N)P)Q \rightarrow_{\gamma_C} (\lambda_y.(\lambda_x.N)P)Q$.

Note that $g$ is a combination of a $\theta$-step with a $\beta$-step. $\gamma_C$ makes sure that $\lambda_y$ and $Q$ form a redex even before the redex based on $\lambda_x$ and $P$ is contracted. By compatibility, $\gamma$ implies $\gamma_C$. Moreover, $((\lambda_x.\lambda_y.N)P)Q \rightarrow_\theta (\lambda_x.(\lambda_y.N)Q)P$

and hence both $\theta$ and $\gamma_C$ put $\lambda_y$ adjacently next to its matching argument. $\theta$ moves the argument next to its matching $\lambda$ whereas $\gamma_C$ moves the $\lambda$ next to its matching argument. For a discussion of where these reductions (let's call them *auxiliary reductions*) have been used see [11, 7]. We give here a very brief summary.

[17] introduces the notion of a *premier redex* which is similar to the redex based on $\lambda_y$ and $Q$ in the rule $(g)$ above (which we call *generalised redex*). [18] uses $\theta$ and $\gamma$ (and calls the combination $\sigma$) to show that the perpetual reduction strategy finds the longest reduction path when the term is Strongly Normalizing (SN). [21] also introduces reductions similar to those of [18]. Furthermore, [9] uses $\theta$ (and other reductions) to show that typability in ML is equivalent to acyclic semi-unification. [19] uses a reduction which has some common themes with $\theta$. [16] and [4] use $\theta$ whereas [12] uses $\gamma$ to reduce the problem of $\beta$-strong normalization to the problem of weak normalization (WN) for related reductions. [10] uses $\theta$ and $\gamma$ to reduce typability in the rank-2 restriction of the 2nd order $\lambda$-calculus to the problem of acyclic semi-unification. [14, 22, 20, 13] use related reductions to reduce SN to WN and [8] uses similar notions in SN proofs. [6] uses a more extended version of $\theta$ (called *term-reshuffling*) and of $g$ (called *generalised reduction*) where $Q$ and $N$ are not only separated by the redex $(\lambda_x.-)P$ but by many redexes (ordinary and generalised).

Looking at these attempts, one notes that auxiliary reduction can help relate $\lambda$-terms according to their evaluation behaviour. After all, auxiliary reductions turn redexes that are not immediately visible into clearly visible ones:

EXAMPLE 1. *Consider* $A \equiv (\lambda_\beta.\lambda_y.\lambda_f.fy)\alpha x$ *and* $B \equiv (\lambda_\beta.(\lambda_y.\lambda_f.fy)x)\alpha$. *Both terms have the term* $\lambda_f.fx$ *as a reduct, so* $A =_\beta B$. *However,* $B$ *has two redexes whereas* $A$ *has only one. Here are the redexes of* $B$*:*

- $r_1 = (\lambda_\beta.(\lambda_y.\lambda_f.fy)x)\alpha$. *Observe that* $B \xrightarrow{r_1} (\lambda_y.\lambda_f.fy)x$.
- $r_2 = (\lambda_y.\lambda_f.fy)x$. *Observe that* $B \xrightarrow{r_2} (\lambda_\beta.\lambda_f.fx)\alpha$.

*In* $A$*, the only obvious redex is:* $r_1' = (\lambda_\beta.\lambda_y.\lambda_f.fy)\alpha$. *Note that* $A \xrightarrow{r_1'} (\lambda_y.\lambda_f.fy)x$.
*Note that* $r_1$ *in* $B$ *and* $r_1'$ *in* $A$ *are both based on the redex* $(\lambda_\beta.-)\alpha$ *and contracting* $r_1$ *in* $B$ *results in the same term as contracting* $r_1'$ *in* $A$.

*A closer look at* $A$ *enables us to see that in* $A$ *(as in* $B$*),* $\lambda_y$ *will get matched with* $x$ *resulting in a redex* $r_2' = (\lambda_y.-)x$. *There are differences however between* $r_2$ *in* $B$ *and* $r_2'$ *in* $A$. $r_2$ *in* $B$ *is completely visible and may be contracted before* $r_1$ *in* $B$. $r_2'$ *on the other hand is a future redex in* $A$. *In fact,* $r_2'$ *is not a redex of* $A$ *itself but a redex of a contractum of* $A$*, namely* $(\lambda_y.\lambda_f.fy)x$*, the result of contracting the redex* $r_1'$ *in* $A$.

*We could guess from* $A$ *itself the presence of the future redex. That is, looking at* $A$ *itself, we see that* $\lambda_\beta$ *is matched with* $\alpha$ *and* $\lambda_y$ *is matched with* $x$. *This can be made visible via rules like* $(\theta)$ *above. Note that:* $A \equiv (\lambda_\beta.\lambda_y.\lambda_f.fy)\alpha x \rightarrow_\theta (\lambda_\beta.(\lambda_y.\lambda_f.fy)x)\alpha \equiv B$.

So, extending the $\lambda$-calculus with auxiliary reductions may lead to a better understanding of the reductional behaviour of programs [15]. Why is this important? Because the $\lambda$-calculus plays a major role in the semantics of programming languages through its mechanisms for modeling evaluation strategies (e.g., call by name, call by value, etc.). Due to this basic role, the $\lambda$-calculus must be informative not only of the final value of the program (the normal form of the $\lambda$-term representing the program), but also of the consecutive values before the final value is reached. In particular, if we have two programs $P_1$ and $P_2$ that return the same final value, we want to know if these programs have equivalent evaluation paths in the sense that each evaluation path from $P_1$ to the final value (going through all the intermediate programs), corresponds (in a strong evaluation sense) to an evaluation path from $P_2$ to the final value, and vice versa. This will mean that $P_1$ and $P_2$ are equivalent programs even though they are written differently. Each intermediate value $a_1$ along the evaluation path from one of these programs to the final value corresponds to a unique intermediate value $a_2$ along the evaluation path of the other program to the final value, and the number of evaluation steps to reach $a_1$ from the first program is equal to the number of evaluation steps to reach $a_2$ from the second program. Of course this does not constitute a formal definition of what we call *reductional equivalence*. Reductional equivalence is difficult to define and is also undecidable.

In order to discuss reductional equivalence between terms, redexes will be *extended* (cf. Definition 25) so that a potential future redex like $(\lambda_y.-)x$ in $A$ of Example 1 will be treated as a first class redex and will possibly be contracted in $A$ even before the originator $(\lambda_\beta.\lambda_y.\lambda_f.fy)\alpha$ has been contracted. Hence, with our extended notion of redexes and reduction we get in $A$ another redex:
$r_2' = (\lambda_y.\lambda_f.fy)x$, which when contracted in $A$ results in $(\lambda_\beta.\lambda_f.fx)\alpha$.
Note that $r_2'$ is $\lambda_y$ matched with $x$ (exactly as $r_2$ in $B$). Note moreover that contracting $r_2'$ in $A$ gives the same result as contracting $r_2$ in $B$.

With this notion of *extended redex*, we observe that there is a bijective correspondence between the (extended) redexes of $A$ and $B$ of Example 1. That is, $r_1$ corresponds to $r_1'$ and $r_2$ corresponds to $r_2'$. Moreover, if one redex is contracted in $A$, the reduct is syntactically equal to the reduct which results from contracting the corresponding redex in $B$ and vice versa. That is, $r_1$ and $r_1'$ yield the same values; similarly $r_2$ and $r_2'$ yield the same values. This is seen as follows:

EXAMPLE 2. *The reduction paths from* $A$ *and* $B$ *of Example 1 are as follows:*
$A\text{-}Path_1$*:* $(\lambda_\beta.\lambda_y.\lambda_f.fy)\alpha x \rightarrow_{r_1'} (\lambda_y.\lambda_f.fy)x \rightarrow \lambda_f.fx$
$A\text{-}Path_2$*:* $(\lambda_\beta.\lambda_y.\lambda_f.fy)\alpha x \rightarrow_{r_2'} (\lambda_\beta.\lambda_f.fx)\alpha \rightarrow \lambda_f.fx$
$B\text{-}Path_1$*:* $(\lambda_\beta.(\lambda_y.\lambda_f.fy)x)\alpha \rightarrow_{r_1} (\lambda_y.\lambda_f.fy)x \rightarrow \lambda_f.fx$
$B\text{-}Path_2$*:* $(\lambda_\beta.(\lambda_y.\lambda_f.fy)x)\alpha \rightarrow_{r_2} (\lambda_\beta.\lambda_f.fx)\alpha \rightarrow \lambda_f.fx$
*It is clear that* $A$ *and* $B$ *have the same number of possible paths before reaching the normal form and that there is a bijective correspondence between the paths* $A\text{-}Path_1$ *and* $B\text{-}Path_1$*, and between* $A\text{-}Path_2$ *and* $B\text{-}Path_2$.

Such equivalences have been noted in history and in particular [18] gives a nice classification of the canonical form of terms and provides a notion of $\sigma$-equivalence which identifies terms only differing by permutations of redexes, and shows that none of the standard operational classification criteria on $\lambda$-calculus (e.g., length of longest reduction) can separate two $\sigma$-equivalent terms. [18] concluded by asking if there existed a syntax that realises $\sigma$-equivalence. In this paper, we attempt to answer the question by using the item notation [5] inspired by de Bruijn's notation of the $\lambda$-calculus.

Item notation enables us to detect more redexes in a term than are immediately visible in the known "classical notation" $\lambda$-calculus.

In Section 2 we introduce what is needed of the item notation and other formal machinery in order to syntactically describe the canonical forms of terms.

In Section 3 we explain how one can achieve the *canonical forms* of terms so that the reductional behaviour is immediately visible.

In Section 4 we give our decidable notion of reductional equivalence $\approx_{\texttt{equi}}$ which we show that it coincides with the $\sigma$-equivalence of [18].

In Section 5 we extend the usual $\beta$-reduction $\rightarrow_\beta$ on $\lambda$-terms to $\rightsquigarrow_\beta$ on classes of terms modulo reductional equivalence. We establish that $\rightsquigarrow_\beta$ is Church-Rosser; that if $A \rightarrow_\beta B$ then $A \rightsquigarrow_\beta B$; and that if $A \rightsquigarrow_\beta B$ is based on a redex $(\lambda_x.-)-$, and if $A' \approx_{\texttt{equi}} A$, then there exists $B' \approx_{\texttt{equi}} B$ such that $A' \rightsquigarrow_\beta B'$ and is based on a corresponding redex $(\lambda_x.-)-$. In other words, $A$ and $A'$ have isomorphic reductional paths. We also show that $SN_{\rightsquigarrow_\beta}$ and $SN_{\rightarrow_\beta}$ are equivalent and that all reductionally equivalent terms have the same normalisation behaviour.

In Section 6 we extend the cube with class reduction and establish the desirable properties.

## 2. SOME FORMAL MACHINERY

The classical notation cannot extend the notion of redexes or enable reshuffling in an easy way. *Item notation* however can ([5] discusses various advantages of this notation). In item notation, one writes the argument before the function so $ab$ becomes $(b\delta)a$. Similarly, in item notation, one writes $(\lambda_x)a$ instead of $\lambda_x.a$. This way, a term becomes a sequence of $\lambda$-items like $(\lambda_x)$ and $\delta$-items like $(b\delta)$ followed by a variable. Moreover, a $\beta$-redex becomes in item notation a $\delta\lambda$-pair: namely, a $\delta$-item adjacent to a $\lambda$-item. We leave it to the reader to check this. Let $V$ be an infinite collection of variables over which $x, y, z, \ldots$ range. In item notation, terms of the $\lambda$-calculus are: $\mathcal{T} ::= V \mid (\mathcal{T}\delta)\mathcal{T} \mid (\lambda_V)\mathcal{T}$. We take $A, B, C, \ldots$ to range over $\mathcal{T}$. We call $(A\delta)$ a **$\delta$-item**, $A$ the **body** of the item and $(A\delta)B$ means apply $B$ to $A$ (note the order). $(\lambda_x)$ is called a **$\lambda$-item**. A redex starts with a $\delta$-item (i.e., $(A\delta)$) next to a $\lambda$-item (i.e., $(\lambda_x)$).

Here we repeat rules $(\theta)$, $(\gamma)$, $(g)$, $(\gamma_C)$ but in item notation:

$(\theta)$ $\quad (Q\delta)(P\delta)(\lambda_x)N \rightarrow_\theta (P\delta)(\lambda_x)(Q\delta)N.$

$(\gamma)$ $\quad (P\delta)(\lambda_x)(\lambda_y)N \rightarrow_\gamma (\lambda_y)(P\delta)(\lambda_x)N.$

$(g)$ $\quad (Q\delta)(P\delta)(\lambda_x)(\lambda_y)N \rightarrow_g (P\delta)(\lambda_x)\{N[y := Q]\}.$

$(\gamma_C)$ $\quad (Q\delta)(P\delta)(\lambda_x)(\lambda_y)N \rightarrow_{\gamma_C} (Q\delta)(\lambda_y)(P\delta)(\lambda_x)N.$

Note furthermore that the rules $(\theta)$, $(\gamma)$, $(g)$, $(\gamma_C)$ are not problematic because we use the Barendregt Convention (see below) 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.

In item notation, each term $A$ is the concatenation of zero or more items and a variable: $A \equiv s_1 s_2 \cdots s_n x$ where each $s_i$ is either a $\lambda$-item or a $\delta$-item, and $x \in V$. These items $s_1, s_2, \ldots, s_n$ are called the **main items** of $A$, $x$ is called the **heart** of $A$, notation $\heartsuit(A)$.[1] We use $s, s_1, s_i, \ldots$ to range over items. A concatenation of zero or more items $s_1 s_2 \cdots s_n$ is called a **segment**. We use $\overline{s}, \overline{s}_1, \overline{s}_i, \ldots$ as meta-variables for segments. We write $\emptyset$ for the empty segment. The items $s_1, s_2, \ldots, s_n$ (if any) are called the **main items** of the segment. A **$\delta\lambda$-pair** is a $\delta$-item immediately followed by a $\lambda$-item.

The **weight** of a segment $\overline{s}$, $\texttt{weight}(\overline{s})$, is the number of main items that compose the segment. Moreover, we define $\texttt{weight}(\overline{s}x) = \texttt{weight}(\overline{s})$ for $x \in V$.

In reduction, the *matching* of the $\delta$ and the $\lambda$ in question is the important thing. **Well-balanced segments (w-b)** are constructed from matching $\delta$ and $\lambda$-items. W-b segments are given inductively by: (i) $\emptyset$ is w-b, (ii) if $\overline{s}$ is w-b then $(A\delta)\overline{s}(\lambda_x)$ is w-b, (iii) if $\overline{s_1}$, $\overline{s_2}$, ... $\overline{s_n}$ are w-b, then the concatenation $\overline{s_1}\ \overline{s_2}, \cdots \overline{s_n}$ is w-b. In Figures 1 and 2, all segments that occur under a hat are w-b.

Bound and free variables and substitution are defined as usual. We write $BV(A)$ and $FV(A)$ to represent the bound and free variables of $A$ respectively. Note that in item notation, the scope of the $x$ in a $\lambda$-item $(\lambda_x)$ is anything to the right of it. We write $A[x := B]$ to denote the term where all the free occurrences of $x$ in $A$ have been replaced by $B$. We take terms to be equivalent up to variable renaming and use $\equiv$ to denote syntactical equality of terms. We assume the usual Barendregt variable convention $BC$ (which says that bound variables are always chosen distinct from free variables) and the usual definition of compatibility (cf. [2]). We say that $A$ is strongly normalizing with respect to a reduction relation $\rightarrow$ (written $\mathrm{SN}_\rightarrow(A)$) iff every $\rightarrow$-reduction path starting at $A$ terminates.

## 3. TOWARDS CANONICAL FORMS

### 3.1 Making redexes visible via $\theta$

Transformations like $(\theta)$ are rather powerful in that they can group together terms with equal reductional behavior. Let us give here this example in classical notation:

EXAMPLE 3. *Consider $E_1, E_2, E_3, E_4$ as follows:*

$$E_1 \equiv (((\lambda_f.\lambda_x.\lambda_y.fxy)+)m)n,$$
$$E_2 \equiv ((\lambda_f.(\lambda_x.\lambda_y.fxy)m)+)n,$$
$$E_3 \equiv (\lambda_f.((\lambda_x.\lambda_y.fxy)m)n)+,$$
$$E_4 \equiv (\lambda_f.(\lambda_x.(\lambda_y.fxy)n)m) + .$$

*Note that $E_1 =_\beta E_2 =_\beta E_3 =_\beta E_4$. Moreover, the visible redexes are as follows:*

*In $E_1$: $(\lambda_f.\lambda_x.\lambda_y.fxy)+$.*

*In $E_2$: $(\lambda_f.(\lambda_x.\lambda_y.fxy)m)+$ and $(\lambda_x.\lambda_y.fxy)m$.*

*In $E_3$: $(\lambda_f.((\lambda_x.\lambda_y.fxy)m)n)+$ and $(\lambda_x.\lambda_y.fxy)m$.*

*In $E_4$:*
*$(\lambda_f.(\lambda_x.(\lambda_y.fxy)n)m)+$, $(\lambda_x.(\lambda_y.fxy)n)m$ and $(\lambda_y.fxy)n$.*

*Furthermore, one can see* potential *future redexes as follows:*

*In $E_1$: $\lambda_x.-$ will eventually be applied to $m$ and $\lambda_y.-$ will be eventually be applied to $n$.*

*In $E_2$: $\lambda_y.-$ will eventually be applied to $n$.*

*In $E_3$: $\lambda_y.-$ will eventually be applied to $n$.*

*Note that $E_1 \rightarrow_\theta E_2 \rightarrow_\theta E_3 \rightarrow_\theta E_4$ and that by $\theta$-reducing $E_1$ to $E_2$ (resp. $E_3$ to $E_4$), an extra redex becomes visible. In $E_4$ all redexes are visible and $E_4$ is in $\theta$-normal form.*

Applying the item notation to Example 3 we get:

EXAMPLE 4. *$E_1$ of Example 3 reads in item notation: $(n\delta)(m\delta)(+\delta)(\lambda_f)(\lambda_x)(\lambda_y)(y\delta)(x\delta)f$. Here, the (classical)*

[1] Note that the term *head variable* used in [1] is a special case of our notion of heart. The head variable of a term in head normal form is the heart of the term. It is not the case however that the heart of a term is always a head variable.

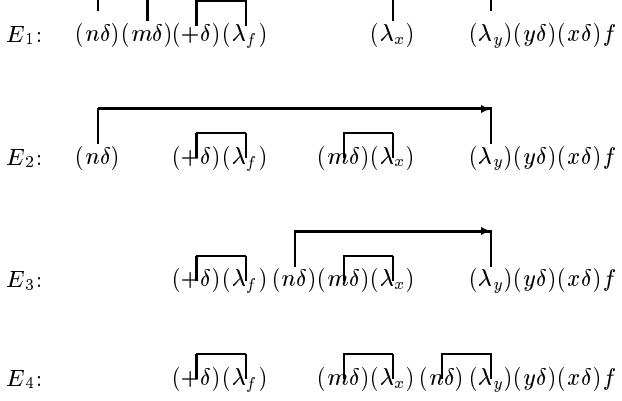


**Figure 1: $\theta$-reduction on $E_1$: $E_1 \to_\theta E_2 \to_\theta E_3 \to_\theta E_4$**

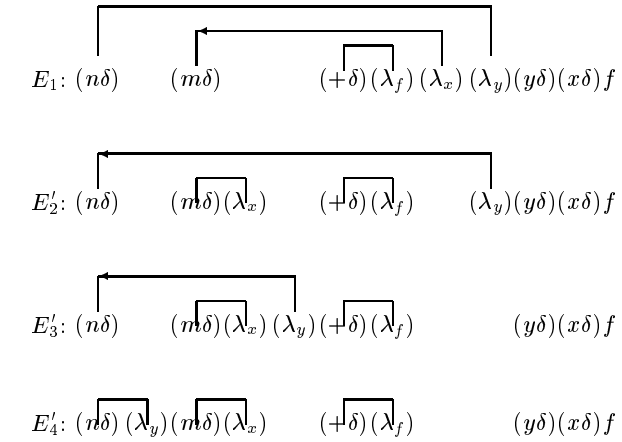


**Figure 2: $\gamma$-reduction on $E_1$: $E_1 \to_\gamma E_2' \to_\gamma E_3' \to_\gamma E_4'$**

*redex corresponds to a '$\delta\lambda$-pair', vis. $(+\delta)(\lambda_f)$, followed by the body of the abstraction, as follows:* $(\lambda_f.(\lambda_x.\lambda_y.fxy)m)+$ *becomes* $(+\delta)(\lambda_f)(m\delta)(\lambda_x)(\lambda_y)(y\delta)(x\delta)f$. *Note that the $\delta$-item $(+\delta)$ and the $\lambda$-item $(\lambda_f)$ are now adjacent, which is characteristic for the presence of a classical redex in item notation. (Cf. Figure 1). The second and third redexes of $E_1$ are obtained by matching $\delta$ and $\lambda$-items which are* not *adjacent:*

- *$(\lambda_y.fxy)n$ is visible as it corresponds to the matching $(n\delta)(\lambda_y)$ where $(n\delta)$ and $(\lambda_y)$ are separated by the segment $(m\delta)(+\delta)(\lambda_f)(\lambda_x)$ which has the bracketing structure $[\,[\,]\,]$.*
- *$(\lambda_x.\lambda_y.fxy)m$ is visible as it corresponds to the matching $(m\delta)(\lambda_x)$ where $(m\delta)$ and $(\lambda_x)$ are separated by the segment $(+\delta)(\lambda_f)$.*

We will use obvious notions throughout like **partner**, **match**, **bachelor**, etc., as follows: in the term $E_1$ of Figure 1, $(+\delta)$ and $(\lambda_f)$ match or are partnered. So are the items $(n\delta)$ and $(\lambda_y)$. $(y\delta)$ and $(x\delta)$ on the other hand are bachelor. The adjacent item pair $(+\delta)(\lambda_f)$ is called a **$\delta\lambda$-pair** and the non-adjacent partnered items $(m\delta)(\lambda_x)$ and $(n\delta)(\lambda_y)$ form **$\delta\lambda$-couples**.

$\theta$-reduction amounts to moving $\delta$-items, from left to right, in the direction of their matching $\lambda$-items, until they form a pair. This is illustrated in Figure 1.

As $\to_\theta$ is Church Rosser (CR) and Strongly Normalizing (SN), then the $\theta$-normal form $\theta(M)$ of a term $M$ is unique.

This paper will establish a method that shows that terms like $E_1, E_2, E_3$ and $E_4$ in Example 3 are reductionally equivalent. Obviously, we will use $\theta$-reduction for this purpose. Any two terms $A$ and $B$ such that $A =_\theta B$ will satisfy $A \approx_{\texttt{equi}} B$.

## 3.2 Making redexes visible via $\gamma$

Looking back at examples 3 and 4, it is possible to use $\gamma$ instead of $\theta$ in order to make more redexes visible. This is illustrated in Figure 2. $\gamma$-reduction amounts to moving $\lambda$-items from right to left, in the direction of their matching $\delta$-items until they form a pair. Also, similarly to $\to_\theta$, $\to_\gamma$ is Church Rosser and Strongly Normalizing, and hence, the $\gamma$-normal form $\gamma(M)$ of a term $M$ is unique.

However, using $\theta$ alone or $\gamma$ alone will not be comprehensive enough to capture as many cases as possible of reductional equivalence. We obviously want not only $E_1$, $E_2$, $E_3$ and $E_4$ of Example 3 to be reductionally equivalent, but also $E_1$, $E_2'$, $E_3'$ and $E_4'$. But, how do we relate $E_i$ to $E_i'$ for $2 \leq i \leq 4$? This is simple, combine the relations $\theta$ and $\gamma$ and aim to find a canonical form of terms that helps establish reductional equivalence.

## 3.3 Reaching canonical forms of $M$ via CCF($M$)

So far, we decided that for any term $A$, all elements $B$ such that $A =_\theta B$ or $A =_\gamma B$ are reductionally equivalent to $A$. But we want also that if $B =_\theta A$ and $B' =_\gamma A$ then $B$ and $B'$ are reductionally equivalent. It is obvious that $B$ and $B'$ will have the same reductional behaviour (persuade yourself of this in the case of $E_4$ and $E_4'$). In order to achieve this, and to obtain a more comprehensive notion of reductional equivalence, we combine $\theta$ and $\gamma$.

Note that $\theta(\gamma(E_1)) = E_4'$ and $\gamma(\theta(E_1)) = E_4$ and that $E_4 \not\equiv E_4'$. However, looking at $E_4$ and $E_4'$, we see that they have the following shape which we call *canonical form*: $(\lambda_{x_1})\ldots(\lambda_{x_n})(A_1\delta)(\lambda_{y_1})\ldots(A_m\delta)(\lambda_{y_m})(B_1\delta)\ldots(B_p\delta)x$ where $(\lambda_{x_i})$ and $(B_j\delta)$ are bachelor for $1 \leq i \leq n$ and $1 \leq j \leq p$ (see Table 1).

The shape of canonical forms will allow us to introduce a reduction relation $\to_p$ on them which will help us show that terms like $E_4$ and $E_4'$ are reductionally equivalent. In fact, note that $E_4$ and $E_4'$ are equivalent up to the permutation of their $\delta\lambda$-pairs. We follow this observation to define the reduction relation $\to_p$ on canonical forms $C_1$ and $C_2$ as follows: $C_1 \to_p C_2$ iff $C_1 \equiv C_2$ except for a segment $(A\delta)(\lambda_x)(B\delta)(\lambda_y)$ in $C_1$ which appears as $(B\delta)(\lambda_y)(A\delta)(\lambda_x)$ in $C_2$, on the condition that $x \notin FV(B)$.[2] We define $=_p$ as the equivalence relation of $\to_p$. We then define CCF($M$) the class of canonical forms of $M$ as $\{M' | M' =_p \theta(\gamma(M))\}$. We will show that CCF($M$) is unique and is equal to $\{M' | M' =_p \gamma(\theta(M))\}$.

With this we give for each term $M$ the class of canonical forms of $M$ modulo $=_p$, CCF($M$) and satisfying Table 1. This is indeed elegant.

Now, two terms are reductionally equivalent if they have the same canonical form modulo $=_p$. We define $[M]$, the class of terms that are reductionally equivalent to $M$, to be $\{M' \mid \theta(\gamma(M)) =_p \theta(\gamma(M'))\}$. All elements of $[M]$ are $\beta$-equal and have in some sense the same redexes.

[2]The condition that $y \notin FV(A)$ is covered by the Barendregt Convention (see Section 2).

**Table 1: The Canonical Form of terms**

| bachelor $\lambda$-items | $\delta\lambda$-pairs, $A_i$ in canon. form | bachelor $\delta$-items, $B_i$ in canon. form | end var |
|---|---|---|---|
| $(\lambda_{x_1})\dots(\lambda_{x_n})$ | $(A_1\delta)(\lambda_{y_1})\dots(A_m\delta)(\lambda_{y_m})$ | $(B_1\delta)\dots(B_p\delta)$ | $x$ |

EXAMPLE 5. *Note that in Figures 1 and 2,* $\gamma(E_1) = \theta(\gamma(E_1)) \equiv E_4'$ *and* $\theta(E_1) = \gamma(\theta(E_1)) \equiv E_4$. *Note also that* $E_4 =_p E_4'$ *and all* $E_i$, *for* $1 \le i \le 4$ *and* $E_j'$, *for* $2 \le j \le 4$ *belong to* $[E_1]$. *All* $E_i$ *and* $E_j'$ *where* $1 \le i \le 4$ *and* $2 \le j \le 4$ *are reductionally equivalent and have the same canonical form* $(+\delta)(\lambda_f)(m\delta)(\lambda_x)(n\delta)(\lambda_y)(y\delta)(x\delta)f$ *modulo* $=_p$. *That is:* $(m\delta)(\lambda_x)(+\delta)(\lambda_f)(n\delta)(\lambda_y)(y\delta)(x\delta)f$ *and* $(m\delta)(\lambda_x)(n\delta)(\lambda_y)(+\delta)(\lambda_f)(y\delta)(x\delta)f$, *etc., are all canonical forms. Note that the variable condition for permutations of pairs holds because + contains no free variables.*

In this paper, we define $A \approx_{\text{equi}} B$ iff $A \in [B]$ (i.e., $\theta(\gamma(A)) =_p \theta(\gamma(B))$). It makes sense to use $\approx_{\text{equi}}$ to represent a class of reductional equivalence on terms. We will show that $\approx_{\text{equi}}$ is decidable. We will extend the notion of $\beta$-reduction to apply to classes rather than terms. As classes capture existing extensions of reductions such as $(\theta)$, $(g)$, $(\gamma)$, etc., $\beta$-reduction over classes will capture all these notions. We say $A$ *class-reduces to* $A'$ and we write $A \rightsquigarrow_\beta A'$ iff $\exists B \in [A] \exists B' \in [A']$ such that $B \to_\beta B'$. We show (cf. Lemma 27) that $\to_\beta$ is captured by $\rightsquigarrow_\beta$.

# 4. REDUCTIONAL EQUIVALENCE

We start by defining the canonical forms (Table 1).

DEFINITION 6. *We say that a term is in canonical form if it has the form:* $(\lambda_{x_1})\dots(\lambda_{x_n})(C_1\delta)(\lambda_{y_1})\dots(C_m\delta)(\lambda_{y_m})(A_1\delta)\dots(A_l\delta)x$. *Note that here,* $(\lambda_{x_i})$ *and* $(A_j\delta)$ *are bachelor for* $1 \le i \le n$ *and* $1 \le j \le l$.

REMARK 7. *Note that canonical forms correspond in classical notation to the following:* $\lambda_{x_1}\dots\lambda_{x_n}.(\lambda_{y_1}.(\lambda_{y_2}\dots(\lambda_{y_m}.xA_l\dots A_1)C_m)\dots)C_2)C_1$ *where again it can be seen that* $\lambda_{x_i}$ *and* $A_j$ *are bachelor for* $1 \le i \le n$ *and* $1 \le j \le l$. *These are exactly the canonical forms given in [18] and represented in [18] by Figure 3 below. Note that our item notation as is seen in Definition 6 permits a more elegant representation than the one given in classical notation in Figure 3.*

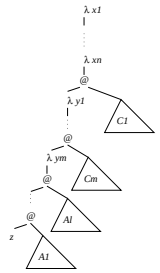


**Figure 3: Canonical forms in classical notation**

DEFINITION 8. *We define* $\to_p$ *on canonical forms as the compatible closure on canonical forms of the rule:*

$$(A_1\delta)(\lambda y_1)(A_2\delta)(\lambda y_2)B \to_p (A_2\delta)(\lambda y_2)(A_1\delta)(\lambda y_1)B \quad \text{if } y_1 \notin FV(A_2)$$

*We define* $\twoheadrightarrow_p$ *and* $=_p$ *as the reflexive, transitive respectively equivalence closures of* $\to_p$.

*We define* $\to_{\theta\gamma}$ *to be* $\to_\theta \cup \to_\gamma$ *and* $\to_{\theta\gamma p}$ *to be* $\to_\theta \cup \to_\gamma \cup \to_p$. $\twoheadrightarrow_{\theta\gamma}$, $\twoheadrightarrow_{\theta\gamma p}$, $=_{\theta\gamma}$ *and* $=_{\theta\gamma p}$ *are defined similarly to* $\twoheadrightarrow_p$ *and* $=_p$.

Intuitively, $\to_p$ transposes two adjacent $\delta\lambda$-pairs in a term if the variable bindings allow this. There is a nice correspondence between $\to_p$, $\to_\theta$ and $\to_\gamma$.

LEMMA 9. *Let* $A$ *and* $B$ *be two canonical forms. If* $A \to_p B$ *then* $\exists C[C \to_\theta A \wedge C \to_\gamma B]$.

PROOF. Induction on the structure of $A$. We take the case $A \equiv (A_1\delta)(\lambda_{y_1})(A_2\delta)(\lambda_{y_2})A_3$ and $B \equiv (A_2\delta)(\lambda_{y_2})(A_1\delta)(\lambda_{y_1})A_3$. In this case, take $C \equiv (A_2\delta)(A_1\delta)(\lambda_{y_1})(\lambda_{y_2})A_3$. □

PROPOSITION 10. $\to_\theta$ *and* $\to_\gamma$ *are SN, CR and* $\subset =_\beta$.

PROOF. SN is a simple combinatorial exercise. For CR we note that $\to_\theta$ as well as $\to_\gamma$ alone are orthogonal. $\to_\theta \subset =_\beta$ and $\to_\gamma \subset =_\beta$ are easy. □

COROLLARY 11. *For each term* $M$, $\theta(M)$, *the* $\theta$*-normal form of* $M$ *and* $\gamma(M)$, *the* $\gamma$*-normal form of* $M$ *are unique.*

We take *the* $\theta\gamma$*-normal form* of $A$ to mean $\theta(\gamma(A))$ which is unique. Similarly, we take the $\gamma\theta$*-normal form* of $A$ to mean $\gamma(\theta(A))$ which is also unique. Note that it is not necessarily the case that $\theta(\gamma(A)) = \gamma(\theta(A))$ as Example 5 shows. However, we will show in Lemma 17 that $\theta(\gamma(A)) =_p \gamma(\theta(A))$.

The following two lemmas enable us to syntactically describe $\theta$- and $\gamma$-normal forms.

LEMMA 12. *Every term has one of the three forms:*

*(i)* $(A_1\delta)\cdots(A_n\delta)x$, *where* $x \in V$ *and* $n \ge 0$,

*(ii)* $(\lambda_x)A$, *and*

*(iii)* $(A_1\delta)\cdots(A_n\delta)(B\delta)(\lambda_x)C$, *where* $n \ge 0$.

PROOF. A term has either zero main $\lambda$-items and case (i) applies, or at least one of them. In the latter case: the first main $\lambda$-item can occur in the first place in the sequence of all main items (case (ii)) or not in the first place (case (iii)). □

LEMMA 13. *Every term has one of the four forms:*

*(i)* $\overline{s}(\lambda_x)A$ *where* $\overline{s}$ *is w-b,*

*(ii)* $(A\delta)B$, *where* $B$ *has no bachelor main* $\lambda$*-items,*

*(iii)* $(A\delta)\overline{s}(\lambda_x)B$ *where* $\overline{s}$ *is w-b and* $B$ *has no bachelor main* $\lambda$*-items, and*

*(iv)* $x$.

PROOF. A term has at least one bachelor main $\lambda$-item (case (i)), or none at all. In the last case, the term may start with a bachelor $\delta$-item (case (ii)), a partnered $\delta$-item (case (iii)) or is only a variable (case (iv)). □

Now, we can syntactically characterise $\theta$- and $\gamma$-normal forms via the following two lemmas whose proof is by induction on the definition of terms as given in Lemmas 12 and 13 resp.:

LEMMA 14. *The $\theta$-normal form $\theta(M)$ of a term $M$ is:*

$$\begin{array}{ll}\theta((A_1\delta)\cdots(A_n\delta)x) & =_{df} (\theta(A_1)\delta)\cdots(\theta(A_n)\delta)x \\ & \quad \textit{if } x\in V \textit{ and } n\geq 0 \\ \theta((\lambda_x)A) & =_{df} (\lambda_x)\theta(A)\end{array}$$

$$\theta((A_1\delta)\cdots(A_n\delta)(B\delta)(\lambda_x)C) =_{df} (\theta(B)\delta)(\lambda_x)\theta((A_1\delta)\cdots(A_n\delta)C)$$

LEMMA 15. *The $\gamma$-normal form $\gamma(M)$ of a term $M$ is:*

$$\begin{array}{lll}\gamma(\overline{s}(\lambda_x)A) & =_{df} & (\lambda_x)\gamma(\overline{s}A) \quad \textit{if } \overline{s} \textit{ is w-b,} \\ \gamma((A\delta)B) & =_{df} & (\gamma(A)\delta)\gamma(B) \\ & & \textit{if } B \textit{ has no bachelor main } \lambda\textit{-items,} \\ \gamma((A\delta)\overline{s}(\lambda_x)B) & =_{df} & (\gamma(A)\delta)(\lambda_x)\gamma(\overline{s}B), \textit{ where } \overline{s} \textit{ is w-b} \\ & & \textit{and } B \textit{ has no bachelor main } \lambda\textit{-items} \\ \gamma(x) & =_{df} & x\end{array}$$

EXAMPLE 16.
*If we take the term $A$ to be:*

$$(\lambda_q)\ (\bar{j}\delta)(\bar{\lambda}_p)\ (\lambda_x)(w\delta)\ (\overset{+}{x}\delta)(\overset{\bullet}{y}\delta)\ (\overset{\bullet}{\lambda}_v)\ (\overset{\times}{v}\delta)(\overset{''}{x}\delta)(\overset{''}{\lambda}_w)(\overset{\times}{\lambda}_t)(\overset{+}{\lambda}_s)\ (s\delta)t$$

*then $\theta(A)$ and $\gamma(A)$ will be given respectively by:*

$$(\lambda_q)\ (\bar{j}\delta)(\bar{\lambda}_p)\ (\lambda_x)\ (\overset{\bullet}{y}\delta)\ (\overset{\bullet}{\lambda}_v)\ (\overset{''}{x}\delta)(\overset{''}{\lambda}_w)(\overset{\times}{v}\delta)(\overset{\times}{\lambda}_t)(\overset{+}{x}\delta)(\overset{+}{\lambda}_s)\ (w\delta)(s\delta)t$$

*and*

$$(\lambda_q)(\lambda_x)\ (\bar{j}\delta)(\bar{\lambda}_p)\ (w\delta)\ (\overset{+}{x}\delta)(\overset{+}{\lambda}_s)(\overset{\bullet}{y}\delta)\ (\overset{\bullet}{\lambda}_v)\ (\overset{\times}{v}\delta)(\overset{\times}{\lambda}_t)(\overset{''}{x}\delta)(\overset{''}{\lambda}_w)\ (s\delta)t$$

The following lemma shows that $\theta$-, $\gamma$-, and $\theta\gamma$-normal forms satisfy Table 2. In particular, all $\theta\gamma$-normal forms are in canonical form. It is interesting to note how item notation enables the clear classification of these various normal forms. Compare with [12, 18] where the classical syntax makes these normal forms cumbersome to describe.

LEMMA 17. *For a term $A$, we have:*

1. *$\theta(A) \equiv \overline{s_1}\,\overline{s_2}\heartsuit(A)$, where $\overline{s_2}$ consists of the $\theta$-normal forms of all bachelor main $\delta$-items of $A$ and $\overline{s_1}$ is a sequence of $\theta$-normal forms of main $\delta\lambda$-pairs and bachelor main $\lambda$-items.*
2. *$\gamma(A) \equiv \overline{s_0}\,\overline{s_1}\heartsuit(A)$ where $\overline{s_0}$ consists of all bachelor main $\lambda$-items of $A$ and $\overline{s_1}$ is a sequence of $\gamma$-normal forms of main $\delta\lambda$-pairs and bachelor main $\delta$-items in $\gamma$-normal form.*
3. *$\theta(\gamma(A)) \equiv \overline{s_0}\,\overline{s_1}\,\overline{s_2}\heartsuit(A)$, where $\overline{s_0}$ consists of all bachelor main $\lambda$-items of $A$ and $\overline{s_1}$ is a sequence of $\theta\gamma$-normal forms of main $\delta\lambda$-pairs and $\overline{s_2}$ consists of the $\theta\gamma$-normal forms of all bachelor main $\delta$-items of $A$.*
4. *$\gamma(\theta(A))$ has the same structure as $\theta(\gamma(A))$ in item 3, except that everywhere $\theta\gamma$-normal forms should change to $\gamma\theta$-normal forms*
5. *$\theta(\gamma(A))$ and $\gamma(\theta(A))$ are both in canonical form and we have that $\theta(\gamma(A)) =_p \gamma(\theta(A))$.*

PROOF. 1), 2) 3) and 4) are by induction on $\mathtt{weight}(A)$, distinguishing cases according to Lemmas 12 and 13 using Lemmas 14 and 15. We only prove 1).

- Case $A \equiv (\lambda_x)C$, use IH on $C$.
- Case $A \equiv (B_1\delta)\cdots(B_n\delta)x$, $x \in V$, then $\overline{s_1}$ is empty.
- $A \equiv (B_1\delta)\cdots(B_n\delta)(C\delta)(\lambda_x)E$. Then $\theta(A) \equiv (\theta(C)\delta)(\lambda_x)\theta((B_1\delta)\cdots(B_n\delta)E)$. By the induction hypothesis $\theta((B_1\delta)\cdots(B_n\delta)E)$ is of the form $\overline{s_1'}\,\overline{s_2}\heartsuit(E) \equiv \overline{s_1'}\,\overline{s_2}\heartsuit(A)$. Now take $\overline{s_1} \equiv (\theta(C)\delta)(\lambda_x)\overline{s_1'}$.

For 5) use 1) ... 4). □

Recall that both $E_4$ and $E_4'$ of Example 1 are in canonical form. They both have the same canonical form as $E_1$. Recall also that $\theta(\gamma(E_1)) \equiv E_4'$, that $\gamma(\theta(E_1)) \equiv E_4$ and that by Lemma 17.4, $E_4 =_p E_4'$. We group all canonical forms related by $=_p$ into one class:

DEFINITION 18. *We define the class of canonical forms of $M$,* $\mathrm{CCF}(M)$ *as* $\{M' \textit{ in canonical form} \mid M' =_p \theta(\gamma(M))\}$.
*Note that by Lemma 17,*
$\mathrm{CCF}(M) = \{M' \textit{ in canonical form} \mid M' =_p \gamma(\theta(M))\}$.

For example, $\mathrm{CCF}(E_1) = \{E_4, E_4'\}$.
Now we are ready to define reductional equivalence:

DEFINITION 19. *For a term $A$, we define:*

- *$[A]$, the class of terms that are reductionally equivalent to $A$, by:* $\{B \mid \theta(\gamma(A)) =_p \theta(\gamma(B))\}$.
- *We say that $B$ is reductionally equivalent to $A$, and write $B \approx_{\mathtt{equi}} A$, iff $B \in [A]$.*

The following lemma says that reductional-equivalence contains $\to_\theta$ and $\to_\gamma$.

LEMMA 20. *$\to_\theta \subset \approx_{\mathtt{equi}}$ and $\to_\gamma \subset \approx_{\mathtt{equi}}$. Moreover, these inclusions are strict.*

PROOF. If $A \to_\theta B$ or $A \to_\gamma B$ then $\theta(\gamma(A)) =_p \theta(\gamma(B))$. Example 5 gives terms $E_4$ and $E_4'$ which are $\approx_{\mathtt{equi}}$ but which are not related by $\to_\theta$ or $\to_\gamma$. □

REMARK 21. *Note that, as both $\to_\theta$ and $\to_\gamma$ are SN, we can by applying $\to_\theta$ and $\to_\gamma$ to any term $A$, reach a term $A'$ which is free of any $\theta$- and $\gamma$-redexes (it is easy to show that the combination of $\theta$- and $\gamma$-reduction is SN). The resulting term $A'$ however depends on the order of applying $\theta$ and $\gamma$. It is the case nonetheless, by Lemma 20 that all terms $A'$, which are obtained from $A$ via arbitrary $\theta$ and $\gamma$ reductions, are reductionally equivalent.*

The following proposition shows that $\approx_{\mathtt{equi}}$ is decidable and that any reductionally equivalent terms are $\beta$-equal.

PROPOSITION 22. *$\approx_{\mathtt{equi}}$ is well-defined, decidable and is an equivalence relation. Moreover, $=_\gamma, =_\theta, =_p \subset \approx_{\mathtt{equi}} \subset =_\beta$, and these inclusions are strict.*

PROOF. Well-definedness and equivalence relation are easy. Similarly, decidability is easy as $\theta$ and $\gamma$ are SN and $=_p$ is decidable. For the first $\subset$, note Lemmas 9 and 20. The second $\subset$ follows from Proposition 10. □

We will now show the equivalence between the $\sigma$-equivalence of [18] and $\approx_{\mathtt{equi}}$. First, we give the definition of [18] of $\sigma$-equivalence:

DEFINITION 23. *[18] defined $\sigma$-reduction to be the smallest compatible relation containing:*

**Table 2: $\theta$-, $\gamma$- and $\theta\gamma$-normal forms**

| | | | | |
|---|---|---|---|---|
| $\theta$-nf: | | $\delta\lambda$-pairs in $\theta$-nf and bachelor $\lambda$-items, $(A_1\delta)(\lambda_x)(\lambda_y)(\lambda_z)(A_2\delta)(\lambda_p)\ldots$ | bachelor $\delta$-items in $\theta$-nf $(B_1\delta)(B_2\delta)\ldots$ | end var $x$ |
| $\gamma$-nf: | bachelor $\lambda$-items $(\lambda_{x_1})(\lambda_{x_2})\ldots$ | $\delta\lambda$-pairs and bachelor $\delta$-items in $\gamma$-nf $(B_1\delta)(A_1\delta)(\lambda_x)(B_2\delta)\ldots$ | | end var $x$ |
| $\theta\gamma$-nf: | bachelor $\lambda$-items $(\lambda_{x_1})(\lambda_{x_2})\ldots$ | $\delta\lambda$-pairs in $\theta\gamma$-nf $(A_1\delta)(\lambda_{y_1})(A_2\delta)(\lambda_{y_2})\ldots(A_m\delta)(\lambda_{y_m})$ | bachelor $\delta$-items in $\theta\gamma$-nf $(B_1\delta)(B_2\delta)\ldots$ | end var $x$ |

- $(\theta)\quad ((\lambda_x.N)P)Q \to_\theta (\lambda_x.NQ)P$ *if* $x \notin Q$
- $(\gamma)\quad (\lambda_x.\lambda_y.N)P \to_\gamma \lambda_y.(\lambda_x.N)P$ *if* $y \notin P$

*We say that $A$ and $B$ are $\sigma$-equivalent if $A =_\sigma B$ where $=_\sigma$ is the equivalence relation associated to $\to_\sigma$.*

The following lemma is needed to establish that $\approx_{\text{equi}}$ and $=_\sigma$ are equivalent.

LEMMA 24. *$A \approx_{\text{equi}} B$ iff $A =_{\theta\gamma p} B$ iff $A =_{\theta\gamma} B$.*

PROOF. $\Longrightarrow$) Note that $\approx_{\text{equi}} \subseteq =_{\theta\gamma p}$ and that by Lemma 9, $=_{\theta\gamma p} \subseteq =_{\theta\gamma}$. $\Longleftarrow$) By Lemma 20, we have $=_{\theta\gamma} \subseteq \approx_{\text{equi}}$ and so, also $=_{\theta\gamma p} \subseteq \approx_{\text{equi}}$ □

This means that, since $\sigma$-equivalence is the same as $=_{\theta\gamma}$, we get that $\sigma$-equivalence is the same as $\approx_{\text{equi}}$. Hence, we have provided a fine grained notion of $\sigma$-equivalence.

## 5. CLASS REDUCTION

In this section, we introduce class-reduction $\rightsquigarrow_\beta$, show that it is Church-Rosser; that if $A \to_\beta B$ then $A \rightsquigarrow_\beta B$; and if $A \rightsquigarrow_\beta B$ is based on a redex $(-\delta)(\lambda_x)$ then for every $A' \approx_{\text{equi}} A$, there exists $B' \approx_{\text{equi}} B$ such that $A' \rightsquigarrow_\beta B'$ and this latter reduction is also based on a corresponding redex $(-\delta)(\lambda_x)$. In other words, $A$ and $A'$ have isomorphic reduction paths. We also show that $SN_{\rightsquigarrow_\beta}$ and $SN_{\to_\beta}$ are equivalent and that all reductionally equivalent terms have the same normalisation behaviour.

DEFINITION 25.

- *One-step class-reduction $\rightsquigarrow_\beta$ is the least compatible relation generated by:*
  *$A \rightsquigarrow_\beta B$ iff $\exists A' \in [A]\exists B' \in [B]$ such that $A' \to_\beta B'$*
  *Many-step class-reduction $\rightsquigarrow\!\!\!\!\rightsquigarrow_\beta$ is the reflexive and transitive closure of $\rightsquigarrow_\beta$ and $\approx_\beta$ is the least equivalence relation generated by $\rightsquigarrow\!\!\!\!\rightsquigarrow_\beta$.*
- *An extended redex starts with the $\delta$-item of a* **$\delta\lambda$-couple** *(i.e. is of the form $(C\delta)\overline{s}(\lambda_x)A$ where $\overline{s}$ is well-balanced and non-empty).*

EXAMPLE 26. *Let $A \equiv (z\delta)(w\delta)(\lambda_x)(\lambda_y)y$. Then $[A] = \{A, (w\delta)(\lambda_x)(z\delta)(\lambda_y)y, (z\delta)(\lambda_y)(w\delta)(\lambda_x)y\}$. Moreover, $A \rightsquigarrow_\beta (w\delta)(\lambda_x)z$ and $A \rightsquigarrow_\beta (z\delta)(\lambda_y)y$.*

The following lemma shows that $\rightsquigarrow_\beta$ captures classical $\beta$-reduction.

LEMMA 27. $\to_\beta \subset \rightsquigarrow_\beta$.

PROOF. It suffices to show $(A\delta)\overline{s}(\lambda_x)C \rightsquigarrow_\beta \overline{s}C[x := A]$. We know that $(A\delta)\overline{s}(\lambda_x)C \in [\overline{s}(A\delta)(\lambda_x)C]$, so by $\overline{s}(A\delta)(\lambda_x)C \to_\beta \overline{s}C[x := A]$ we have $(A\delta)\overline{s}(\lambda_x)C \rightsquigarrow_\beta \overline{s}C[x := A]$. It is easy to show that these inclusions are strict. For example, if $A_1 \equiv (A\delta)(B\delta)(\lambda_x)(C\delta)(D\delta)(\lambda_y)(\lambda_z)(\lambda_t)E$ and $A_2 \equiv (C\delta)(B\delta)(\lambda_x)(D\delta)(\lambda_y)(\lambda_z)(E[t := A])$ (which have respectively the bracketing structures [ [ ] [ [ ] [ ] ] ] and [ [ ] [ ] ]), then $A_1 \rightsquigarrow_\beta A_2$ but $A_1 \not\to_\beta A_2$. □

COROLLARY 28. $\twoheadrightarrow_\beta \subset \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta$.

REMARK 29. *It is not in general true that $A \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta B \Rightarrow \exists A' \in [A]\exists B' \in [B]$ such that $A' \twoheadrightarrow_\beta B'$. This can be seen by the following counterexample:*
*Let $A \equiv ((\lambda_u)(\lambda_v)v\delta)(\lambda_x)(w\delta)(w\delta)x$ and $B \equiv (w\delta)(\lambda_u)w$. Then $A \rightsquigarrow_\beta (w\delta)(w\delta)(\lambda_u)(\lambda_v)v \rightsquigarrow_\beta B$.*
*But $[A]$ has three elements: $A$, $(w\delta)((\lambda_u)(\lambda_v)v\delta)(\lambda_x)(w\delta)x$ and $(w\delta)(w\delta)((\lambda_u)(\lambda_v)v\delta)(\lambda_x)x$.*
*Moreover, $[B] = \{B\}$ and if $A' \in [A]$ then the only $\to_\beta$-reduct of $A'$ is $(w\delta)(w\delta)(\lambda_u)(\lambda_v)v$, which $\not\to_\beta$-reduce to $B$.*

The next lemma helps prove that $\rightsquigarrow_\beta$ is Church-Rosser:

LEMMA 30. *If $A \rightsquigarrow_\beta B$ then $A =_\beta B$.*

PROOF. Say $A' \in [A]$, $B' \in [B]$, $A' \to_\beta B'$. Now, by Lemma 17 and Proposition 22, $A =_\beta \theta(\gamma(A)) =_\beta \theta(\gamma(A')) =_\beta A' =_\beta B' =_\beta \theta(\gamma(B')) =_\beta \theta(\gamma(B)) =_\beta B$. □

COROLLARY 31.
*1. If $A \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta B$ then $A =_\beta B$.* *2. $A \approx_\beta B$ iff $A =_\beta B$.*

THEOREM 32 (CHURCH ROSSER THEOREM FOR $\rightsquigarrow\!\!\!\!\rightsquigarrow_\beta$).

*If $A \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta B$ and $A \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta C$, then there exists $D$ such that $B \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta D$ and $C \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta D$.*

PROOF. As $A \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta B$ and $A \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta C$ then by Corollary 31, $A =_\beta B$ and $A =_\beta C$. Hence, $B =_\beta C$ and by CR for $\twoheadrightarrow_\beta$, there exists $D$ such that $B \twoheadrightarrow_\beta D$ and $C \twoheadrightarrow_\beta D$. But, $M \twoheadrightarrow_\beta N$ implies $M \rightsquigarrow\!\!\!\!\rightsquigarrow_\beta N$. Hence we are done. □

Now we are ready to establish the isomorphism of reduction paths (via $\rightsquigarrow_\beta$) of two reductionally equivalent terms. The following lemma shows that reductional equivalence preserves the generalised reduction $\rightsquigarrow_\beta$.

LEMMA 33. *If $A \rightsquigarrow_\beta B$ then for all $A' \approx_{\text{equi}} A$, for all $B' \approx_{\text{equi}} B$, $A' \rightsquigarrow_\beta B'$.*

PROOF. As $A \rightsquigarrow_\beta B$ then $\exists A_1 \in [A]\exists B_1 \in [B]$ such that $A_1 \to_\beta B_1$. Let $A' \approx_{\text{equi}} A$ and $B' \approx_{\text{equi}} B$. Then $A', B' \in [A], [B]$ respectively. Hence $A_1 \in [A']$, $B_1 \in [B']$, $A_1 \to_\beta B_1$. So $A' \rightsquigarrow_\beta B'$. □

COROLLARY 34. *$A \rightsquigarrow_\beta B$ iff $\theta(\gamma(A)) \rightsquigarrow_\beta \theta(\gamma(B))$.*

The following remark points out that if we want to preserve reduction paths, we need to work with the reduction $\rightsquigarrow_\beta$.

REMARK 35. *[3, 6] defined $\hookrightarrow_\beta$ as the least compatible relation generated by $(B_1\delta)\overline{s}(\lambda_x)B_2 \hookrightarrow_\beta \overline{s}(B_2[x := B_1])$ for $\overline{s}$ well-balanced, that is, $\hookrightarrow_\beta$-reduction contracts an (extended) redex. $\hookrightarrow\!\!\!\!\twoheadrightarrow_\beta$ is the reflexive and transitive closure of $\hookrightarrow_\beta$. Note that $A \hookrightarrow_\beta B \not\Rightarrow \theta(\gamma(A)) \hookrightarrow\!\!\!\!\twoheadrightarrow_\beta \theta(\gamma(B))$ nor do we have $A \to_\beta B \Rightarrow \theta(\gamma(A)) \twoheadrightarrow_\beta \theta(\gamma(B))$. E.g., take $A \equiv ((\lambda_u)(\lambda_v)v\delta)(\lambda_x)(y\delta)(y\delta)x$. It is obvious that $A \to_\beta B \equiv (y\delta)(y\delta)(\lambda_u)(\lambda_v)v$ (hence $A \hookrightarrow_\beta B$) yet $\theta(\gamma(A)) \equiv A \not\twoheadrightarrow_\beta$ nor $\not\twoheadrightarrow_\beta \theta(\gamma(B)) \equiv (y\delta)(\lambda_u)(y\delta)(\lambda_v)v$.*

Finally, here is the theorem that establishes the isomorphism of reduction paths of two reductionally equivalent terms. We write $A \to_\beta^{(E\delta)(\lambda_x)} B$ for the $\beta$-reduction based on a $\beta$-redex starting with $(E\delta)(\lambda_x)$ in $A$. We write $A \rightsquigarrow_\beta^{(E\delta)(\lambda_x)} B$ for $\exists A' \in [A], \exists B' \in [B], \exists E' \in [E]$ such that $A' \to_\beta^{(E'\delta)(\lambda_x)} B'$.

THEOREM 36. *If $A \approx_{\text{equi}} C$ and $A \rightsquigarrow_\beta^{(E\delta)(\lambda_x)} B$ then there exists a $D$ and an $E'$ such that $B \approx_{\text{equi}} D, E' \approx_{\text{equi}} E$, and $C \to_\beta^{(E'\delta)(\lambda_x)} D$.*

*In other words, the following diagram commutes:*

$$\begin{array}{ccc} A & \xrightarrow{\quad (E\delta)(\lambda_x) \quad} & B \\ \approx_{\text{equi}} & & \approx_{\text{equi}} \\ C & \dashrightarrow_\beta & D \end{array}$$

PROOF. Note that $A \approx_{\text{equi}} C$ and $A \rightsquigarrow_\beta B$ implies by Lemma 33 that for all $D \approx_{\text{equi}} B$, $C \rightsquigarrow_\beta D$. What we want is to find a $D \approx_{\text{equi}} B$ and an $E' \approx_{\text{equi}} E$ such that $C \rightsquigarrow_\beta^{(E'\delta)(\lambda_x)} D$.

As $A \rightsquigarrow_\beta^{(E\delta)(\lambda_x)} B$ then $\exists A' \in [A]$, $\exists B' \in [B]$, $\exists E'' \in [E]$ such that $A' \to_\beta^{(E''\delta)(\lambda_x)} B'$. Now $C \rightsquigarrow_\beta^{(E''\delta)(\lambda_x)} B'$ due to the facts that $A' \to_\beta^{(E''\delta)(\lambda_x)} B'$, $A' \approx_{\text{equi}} C$, $B' \approx_{\text{equi}} B'$ and $E'' \approx_{\text{equi}} E''$. Hence, we have found a $D \equiv B'$ and $E' \equiv E''$ such that $B \approx_{\text{equi}} D$, $E' \approx_{\text{equi}} E$ and $C \rightsquigarrow_\beta^{(E'\delta)(\lambda_x)} D$ and we are done. □

The following two lemmas show that reductional equivalence preserves both $\rightsquigarrow_\beta$-strong normalization and $\to_\beta$-strong normalization:

LEMMA 37. *If $A \in SN_{\to_\beta}$ and $A' \in [A]$ then $A' \in SN_{\to_\beta}$.*

PROOF. If $A' \in [A]$ then $A' \approx_{\text{equi}} A$. Hence, by Lemma 24, $A' =_\sigma A$. Now, we use a result of [18] which says that if $A =_\sigma A'$ then the length of the longest reduction sequence starting from $A$ is equal to the length of the longest reduction sequence starting from $A'$. □

LEMMA 38. *If $A \in SN_{\rightsquigarrow_\beta}$ and $A' \in [A]$ then $A' \in SN_{\rightsquigarrow_\beta}$.*

PROOF. $\forall B, A' \rightsquigarrow_\beta B$ implies $A \rightsquigarrow_\beta B$ by Lemma 33. Hence, $A'$ is in $SN_{\rightsquigarrow_\beta}$. □

Finally, we show that $\rightsquigarrow_\beta$-strong normalization and $\to_\beta$-strong normalization are equivalent:

LEMMA 39. *$A \in SN_{\rightsquigarrow_\beta}$ iff $A \in SN_{\to_\beta}$.*

PROOF. As $\to_\beta \subset \rightsquigarrow_\beta$, $\Longrightarrow$ is immediate. $\Longleftarrow$ is by induction on $\mathcal{M}(A)$ where $\mathcal{M}(\mathcal{A}) = \max\{\text{maxred}_\beta(A') \mid A' \in [A]\}$; $\text{maxred}_\beta(A')$ is the maximal length of $\to_\beta$-reduction paths starting from $A'$. Note that $\mathcal{M}(A)$ is well-defined if $A \in SN_{\to_\beta}$ by Lemma 37.

Suppose $A \rightsquigarrow_\beta A'$ and $A \in SN_{\to_\beta}$. It is sufficient to prove that $A' \in SN_{\rightsquigarrow_\beta}$. Take $A_1 \in [A]$ and $A_1' \in [A']$ such that $A_1 \to_\beta A_1'$. Then also $A' \in [A_1']$, so by Lemma 38 it is sufficient to prove that $A_1' \in SN_{\rightsquigarrow_\beta}$. By Lemma 37, $A_1 \in SN_{\to_\beta}$, and since $A_1 \to_\beta A_1'$ we have $A_1' \in SN_{\to_\beta}$. Then also $\mathcal{M}(A_1') < \mathcal{M}(A_1) = \mathcal{M}(A)$, so by the induction hypothesis: $A_1' \in SN_{\rightsquigarrow_\beta}$. □

# 6. THE CUBE WITH CLASS REDUCTION

Our study of class reduction has been discussed up to now for the type free $\lambda$-calculus. But, for such reduction to be useful in practice, we need to study it within type theory. Alas, when attempting to build class reduction on the systems of the Barendregt cube of [2], we find that the subject reduction property which states that if $A \rightsquigarrow_\beta B$ then $B$ has the same type as $A$, no longer holds for six of the systems of the cube, although it holds for the systems $\lambda_\to$ and $\lambda\underline{\omega}$. This problem however can be solved by extending the cube not only with class reduction, but also with *definitions* which avoid the loss of information in the contexts needed to type terms. With this extension the subject reduction property holds for all the systems of the cube.

We show in Section 6.1 that subject reduction fails for 6 systems of the cube, but holds for $\lambda_\to$ and $\lambda\underline{\omega}$ with $\rightsquigarrow_\beta$ (without definitions). We show furthermore that reductionally equivalent terms have the same type in the sense that if $\Gamma \vdash A : B$ and $A' \in [A]$ then $\Gamma \vdash A' : B$ (see Corollary 49). Then, we add definitions in Section 6.2 and show that all the desirable properties including SR hold for all the systems of the extended cube.

## 6.1 Extending the cube with $\rightsquigarrow_\beta$

Let us start by introducing class-reduction to the cube of [2]. This means that our reduction relation now is not $\to_\beta$ but $\rightsquigarrow_\beta$ and that our extended cube of this subsection is exactly that of Barendregt in [2] with the only difference that we use $\rightsquigarrow_\beta$ instead of $\to_\beta$.

DEFINITION 40 (THE CUBE IN ITEM NOTATION). *The systems of the $\lambda$-cube are based on a set of* pseudo-expressions $\mathcal{T}$ *(also called terms) defined by (for $\mathcal{O} \in \{\lambda, \Pi\}$): $\mathcal{T} = * \mid \Box \mid V \mid (\mathcal{T}\delta)\mathcal{T} \mid (\mathcal{T}\mathcal{O}_V)\mathcal{T}$. We take $A, B, C, a, b, \ldots$, resp. $S, S_1, S_2$ to range over $\mathcal{T}$ resp. $\{*, \Box\}$. The typing rules are as follows:*

*(axiom)* $<> \vdash * : \Box$

*(start)* $$\frac{\Gamma \vdash A : S}{\Gamma(A\lambda_x) \vdash x : A} \qquad \text{if } x \text{ is fresh}$$

*(weak)* $$\frac{\Gamma \vdash A : S \qquad \Gamma \vdash D : E}{\Gamma(A\lambda_x) \vdash D : E} \qquad \text{if } x \text{ is fresh}$$

*(app)* $$\frac{\Gamma \vdash F : (A\Pi_x)B \qquad \Gamma \vdash a : A}{\Gamma \vdash (a\delta)F : B[x := a]}$$

*(abs)* $$\frac{\Gamma(A\lambda_x) \vdash b : B \qquad \Gamma \vdash (A\Pi_x)B : S}{\Gamma \vdash (A\lambda_x)b : (A\Pi_x)B}$$

*(conv)* $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : S \qquad B =_\beta B'}{\Gamma \vdash A : B'}$$

*(form)* $$\frac{\Gamma \vdash A : S_1 \qquad \Gamma(A\lambda_x) \vdash B : S_2}{\Gamma \vdash (A\Pi_x)B : S_2}(S_1, S_2)\ \textit{rule}$$

A context or a term is called *legal* with respect to a type system if it occurs as such in a type-derivation in that system. Bound and free variables and substitution are defined as usual. We write $BV(A)$ and $FV(A)$ to represent the bound and free variables of $A$ respectively. We write $A[x := B]$ to denote the term where all the free occurrences of $x$ in $A$ have been replaced by $B$. We take terms to be equivalent up to variable renaming and use $\equiv$ to denote syntactical equality of terms. We assume moreover, the usual Barendregt

variable convention $BC$ [2].

Each system of the cube is obtained by taking the $(S_1, S_2)$ rules from a subset R of $\{(*,*),(*,\Box),(\Box,*),(\Box,\Box)\}$. The basic system is the one where $(S_1, S_2) = (*,*)$ is the only possible choice. All other systems have this version of the formation rules, plus one or more other combinations of $(*,\Box)$, $(\Box,*)$ and $(\Box,\Box)$ for $(S_1, S_2)$. Table 3 presents the eight systems of the $\lambda$-cube.

The next two examples show that if our type derivation rules are those of Definition 40 and our reduction relation is $\rightsquigarrow_\beta$ instead of $\twoheadrightarrow_\beta$, then we lose the subject reduction property (SR) which states that if $\Gamma \vdash A : B$ and $A \rightsquigarrow_\beta A'$ then $\Gamma \vdash A' : B$.

EXAMPLE 41 (SR FAILS IN $\lambda 2$, $\lambda P2, \lambda\omega$ AND $\lambda C$).
$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \beta$ *but,*
$(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x \rightsquigarrow_\beta (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x$.
*Yet,* $(*\lambda_\beta)(\beta\lambda_{y'}) \not\vdash_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \beta$.
*Even,* $(*\lambda_\beta)(\beta\lambda_{y'}) \not\vdash_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \tau$ *for any* $\tau$.
*This is because* $(\alpha\lambda_x)x : (\alpha\Pi_x)\alpha$ *and* $y : \beta$ *yet* $\alpha$ *and* $\beta$ *are unrelated and hence we fail in firing the application rule to find the type of* $(y'\delta)(\alpha\lambda_x)x$. *Looking closer however, one finds that* $(\beta\delta)(*\lambda_\alpha)$ *is defining* $\alpha$ *to be* $\beta$, *yet no such information can be used to combine* $(\alpha\Pi_x)\alpha$ *with* $\beta$. *Definitions take such information into account. Finally note that failure of SR in* $\lambda 2$, *means its failure in* $\lambda P2, \lambda\omega$ *and* $\lambda C$.

EXAMPLE 42 (SR FAILS IN $\lambda P$, $\lambda P2, \lambda P\underline{\omega}$ AND $\lambda C$).
*Let* $\Gamma \equiv (*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N)$,
$A \equiv (N\delta)(t\delta)(\sigma\lambda_x)((x\delta)Q\lambda_y)(y\delta)((x\delta)Q\lambda_Z)Z$ *and*
$B \equiv (t\delta)(\sigma\lambda_x)(N\delta)((x\delta)Q\lambda_Z)Z$. *Then,* $\Gamma \vdash_{\lambda P} A : (t\delta)Q$ *and* $A \rightsquigarrow_\beta B$ *but as* $N : (t\delta)Q, y : (x\delta)Q, (t\delta)Q \neq (x\delta)Q$, $\Gamma \not\vdash_{\lambda P} B : \tau$ *for any* $\tau$.
*Here again the reason of failure is similar to the above example. At one stage, we need to match* $(x\delta)Q$ *with* $(t\delta)Q$ *but this is not possible even though we do have the definition segment:* $(t\delta)(\sigma\lambda_x)$ *which defines* $x$ *to be* $t$. *All this calls for the need to use these definitions. Finally note that failure of SR in* $\lambda P$, *means its failure in* $\lambda P2, \lambda P\underline{\omega}$ *and* $\lambda C$.

However, subject reduction holds for $\lambda_\rightarrow$ and $\lambda\underline{\omega}$. In the rest of this subsection, $\mathcal{L}$ ranges over $\lambda\underline{\omega}$ and $\lambda_\rightarrow$ and $\vdash$ ranges over type derivations in these two systems. The rest of this section proves subject reduction for $\lambda_\rightarrow$ and $\lambda\underline{\omega}$.

The first three lemmas and corollary are exactly those of the cube of [2] because $\rightsquigarrow_\beta$ does not play any role in them. Only $\approx_\beta$ (which is the same as $=_\beta$) is involved.

LEMMA 43 (THINNING FOR $\vdash$). *Let* $\Gamma$ *and* $\Delta$ *be legal contexts such that* $\Gamma \subseteq' \Delta$. *Then* $\Gamma \vdash A : B \Rightarrow \Delta \vdash A : B$.

PROOF. Induction on length of derivations $\Gamma \vdash A : B$. □

LEMMA 44 (GENERATION LEMMA FOR $\vdash$).

1. $\Gamma \vdash x : C \Rightarrow \exists S_1, S_2 \in \mathcal{S}\ \exists B =_\beta C[\Gamma \vdash B : S_1 \wedge (B\lambda_x) \in' \Gamma \wedge \Gamma \vdash C : S_2]$.
2. $\Gamma \vdash (A\Pi_x)B : C \Rightarrow \exists S_1, S_2 \in \mathcal{S}\,[\Gamma \vdash A : S_1 \wedge \Gamma(A\lambda_x) \vdash B : S_2 \wedge (S_1, S_2)$ *is a rule* $\wedge$ $C =_\beta S_2 \wedge [C \not\equiv S_2 \Rightarrow \exists S[\Gamma \vdash C : S]]]$
3. $\Gamma \vdash (A\lambda_x)b : C \Rightarrow \exists S, B\,[\Gamma \vdash (A\Pi_x)B : S \wedge \Gamma(A\lambda_x) \vdash b : B \wedge C =_\beta (A\Pi_x)B \wedge$ $C \not\equiv (A\Pi_x)B \Rightarrow \exists S \in \mathcal{S}[\Gamma \vdash C : S]]$.
4. $\Gamma \vdash (a\delta)F : C \Rightarrow \exists A, B, x[\Gamma \vdash F : (A\Pi_x)B \wedge \Gamma \vdash a : A \wedge C =_\beta B[x := a] \wedge$ $(B[x := a] \not\equiv C \Rightarrow \exists S \in \mathcal{S}[\Gamma \vdash C : S])]$.

PROOF. Induction on derivation rules using thinning. □

COROLLARY 45 (GENERATION COROLLARY FOR $\vdash$).

1. *Correctness of Types: If* $\Gamma \vdash A : B$ *then* $\exists S[B \equiv S$ *or* $\Gamma \vdash B : S]$.
2. $\Gamma \vdash A : (B_1\Pi_x)B_2 \Rightarrow \exists S[\Gamma \vdash (B_1\Pi_x)B_2 : S]$.
3. *If* $A$ *is a* $\Gamma^\vdash$*-term, then* $A$ *is* $\Box$, *a* $\Gamma^\vdash$*-kind or a* $\Gamma$*-element.*

LEMMA 46 (SUBSTITUTION FOR $\vdash$). *If* $\Gamma(B\lambda_x)\Delta \vdash C : D$ *and* $\Gamma \vdash A : B$, *then* $\Gamma\Delta[x := A] \vdash C[x := A] : D[x := A]$.

PROOF. By induction to the derivation rules, using the thinning lemma. □

Because our reduction relation $\rightsquigarrow_\beta$ is defined in terms of classes instead of terms, we cannot use the usual methods for establishing SR. For this, we need the following two lemmas which inform us that classes preserve types.

LEMMA 47 (SHUFFLE LEMMA FOR $\lambda\underline{\omega}$ AND $\lambda_\rightarrow$).

1. $\Gamma \vdash_{\mathcal{L}} \overline{s}_1(A\delta)\overline{s}_2 B : C \iff \Gamma \vdash_{\mathcal{L}} \overline{s}_1\overline{s}_2(A\delta)B : C$ *where* $\overline{s}_2$ *is well-balanced and the binding variables in* $\overline{s}_2$ *are not free in* $A$.
2. $\Gamma \vdash_{\mathcal{L}} \overline{s}(\lambda_x)A : B \iff \Gamma \vdash_{\mathcal{L}} (\lambda_x)\overline{s}A : B$ *where* $\overline{s}$ *is well-balanced and* $x$ *is not free in* $\overline{s}$.

PROOF. Sketch. The reason for this lemma to be true for $\lambda_\rightarrow$ and $\lambda\underline{\omega}$ is that in these systems, for any legal term of the form $(P\Pi_x)Q$, $x \notin FV(Q)$ (this is not true for the other systems of the cube because of the mixing of levels that comes with the rules $(*,\Box)$ and $(\Box,*)$). Therefore in 1., none of the variables of $dom(\overline{s_2})$ can occur free in the type of $B$ which means that $B$ must have a type of the form $(C\Pi_x)D$ and hence $B$ can be applied directly to $A$. □

LEMMA 48 (CLASSES PRESERVE TYPES). *If* $\Gamma \vdash A : B$ *and* $A' \in [A]$, $\Gamma'$ *results from* $\Gamma$ *by substituting some main items* $(C\lambda_x)$ *by* $(C'\lambda_x)$ *where* $C' \in [C]$, *then* $\Gamma' \vdash A' : B$.

PROOF. By induction on the derivation rules. □

COROLLARY 49 (EQUIVALENT TERMS HAVE SAME TYPES).

1. $\Gamma \vdash A : B \iff \Gamma \vdash \mathrm{CCF}(A) : B$.
2. *If* $\Gamma \vdash A : B$ *and* $A' \in [A]$, $B' \in [B]$ *then* $\Gamma \vdash A' : B'$.

PROOF. By Lemma 49, correctness of types and conversion. □

Now with Corollary 49, we can establish SR using $\vdash$ and $\rightsquigarrow_\beta$, via SR of $\vdash$ and $\twoheadrightarrow_\beta$.

COROLLARY 50 (SUBJECT REDUCTION FOR $\vdash$ AND $\rightsquigarrow_\beta$). *If* $\Gamma \vdash A : B$ *and* $A \rightsquigarrow_\beta A'$ *then* $\Gamma \vdash A' : B$.

PROOF. We prove $\Gamma \vdash A : B$, $A \rightsquigarrow_\beta A' \Longrightarrow \Gamma \vdash A' : B$. By definition of $\rightsquigarrow_\beta$, there are $A_1$, $A_1'$ such that $A_1 \in [A]$, $A_1' \in [A']$ and $A_1 \rightarrow_\beta A_1'$. By Corollary 49, $\Gamma \vdash A_1 : B$. By subject reduction for the usual $\rightarrow_\beta$ we have $\Gamma \vdash A_1' : B$. Again by Corollary 49, $\Gamma \vdash A' : B$. □

Note that although SR fails for the six remaining systems of the cube with $\vdash$ of Definition 40 and $\rightsquigarrow_\beta$, strong normalisation holds for all the systems of the cube with $\vdash$ of Definition 40 and $\rightsquigarrow_\beta$. We will not prove this here and we move immediately to the version that indeed satisfies SR and all other properties.

**Table 3: Systems of the cube**

| *System* | *Set of specific rules* | | | | *System* | *Set of specific rules* | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\lambda\!\to$ | $(*,*)$ | | | | $\lambda\underline{\omega}$ | $(*,*)$ | | | $(\Box,\Box)$ |
| $\lambda 2$ | $(*,*)$ | $(\Box,*)$ | | | $\lambda\omega$ | $(*,*)$ | $(\Box,*)$ | | $(\Box,\Box)$ |
| $\lambda P$ | $(*,*)$ | | $(*,\Box)$ | | $\lambda P2$ | $(*,*)$ | $(\Box,*)$ | $(*,\Box)$ | |
| $\lambda P\underline{\omega}$ | $(*,*)$ | | $(*,\Box)$ | $(\Box,\Box)$ | $\lambda P\omega = \lambda C$ | $(*,*)$ | $(\Box,*)$ | $(*,\Box)$ | $(\Box,\Box)$ |

## 6.2 Extending the cube with $\rightsquigarrow_\beta$ and definitions

Looking back at, for instance, Example 41, one notices that when reducing using $\rightsquigarrow_\beta$, the information that $y'$ has replaced $y$ of type $\alpha$ is lost. All we know after the reduction is that $y'$ has type $\beta$. But we need $y'$ of type $\alpha$ to be able to type the subterm $(y'\delta)(\alpha\lambda_x)x$ of the reduct. Adding definitions to our type system enables us to have extra information in our contexts such as "$\alpha$ and $\beta$ can be identified". We do this by writing in our context: $(\beta\delta)(*\lambda_\alpha)$ which expresses that $\alpha$ is defined to be $\beta$ and is of type $*$. We define now this notion of definitions and how definitions can be unfolded:

DEFINITION 51 (DEFINITIONS, UNFOLDING).

- *If $\overline{s}$ is a well-balanced segment not containing $\delta\Pi$-couples, then a segment $(B\delta)\overline{s}(C\lambda_x)$ occurring in a context is called a definition.*
- *For $\overline{s}$ well-balanced segment, we define the unfolding of $\overline{s}$ in $A$, $[A]_{\overline{s}}$, inductively as follows: $[A]_\emptyset \equiv A$, $[A]_{(B\delta)\overline{s_1}(C\mathcal{O}_x)} \equiv [A[x := B]]_{\overline{s_1}}$ and $[A]_{\overline{s_1}\,\overline{s_2}} \equiv [[A]_{\overline{s_2}}]_{\overline{s_1}}$. Note that substitution takes place from right to left.*

That is, a definition identifies a variable with a whole term. The unfolding of the definition, undoes this identification and the variable will be replaced everywhere it occurs free by the term it identifies.

DEFINITION 52.

1. *A declaration $d$ is a $\lambda$-item $(A\lambda_x)$. In this case, we define $\mathtt{subj}(d)$, $\mathtt{pred}(d)$ and $\underline{d}$ to be $x$, $A$ and $\emptyset$ resp.*
2. *For a definition $d \equiv (B\delta)\overline{s}(A\lambda_x)$ we define $\mathtt{subj}(d)$, $\mathtt{pred}(d)$, $\underline{d}$ and $\mathtt{def}(d)$ to be $x$, $A$, $\overline{s}$ and $B$ respectively.*
3. *A pseudocontext is a concatenation of declarations and definitions such that if $(A\lambda_x)$ and $(B\lambda_y)$ are two different main items of the pseudocontext, then $x \not\equiv y$. We use $\Gamma, \Delta, \Gamma', \Gamma_1, \Gamma_2, \ldots$ to range over pseudocontexts and $d, d_1, d_2, \ldots$ to range over declarations and definitions.*
4. *For $\Gamma$ a pseudocontext we define $dom(\Gamma) = \{x \in V \mid (A\lambda_x)$ is a main $\lambda$-item in $\Gamma$ for some $A\}$,*

   *$\Gamma$-$\mathtt{decl} = \{s \mid s$ is a bachelor main $\lambda$-item of $\Gamma\}$,*

   *$\Gamma$-$\mathtt{def} = \{\overline{s} \mid \overline{s} \equiv (A\delta)\overline{s}_1(B\lambda_x)$ is a main segment of $\Gamma$ where $\overline{s}_1$ is well-balanced $\}$,*

   *Note that $dom(\Gamma) = \{\mathtt{subj}(d) \mid d \in \Gamma\text{-}\mathtt{decl} \cup \Gamma\text{-}\mathtt{def}\}$.*
5. *For all contexts $\Gamma$ we define the binary relation $\Gamma \vdash \cdot =_{\mathtt{def}} \cdot$ to be the equivalence relation generated by*
   - *if $A =_\beta B$ then $\Gamma \vdash A =_{\mathtt{def}} B$*
   - *if $d \in \Gamma$-$\mathtt{def}$ and $A, B \in \mathcal{T}$ such that $B$ arises from $A$ by substituting one particular occurrence of $\mathtt{subj}(d)$ in $A$ by $\mathtt{def}(d)$, then $\Gamma \vdash A =_{\mathtt{def}} B$.*
6. *For $\Gamma$ a pseudocontext and $d \in \Gamma$-$\mathtt{def} \cup \Gamma$-$\mathtt{decl}$, $\Gamma$ invites $d$, notation $\Gamma \prec d$, iff*
   - *$\Gamma d$ is a pseudocontext*
   - *$\Gamma\underline{d} \vdash \mathtt{pred}(d) : S$ for some sort $S$.*
   - *if $d$ is a definition then $\Gamma\underline{d} \vdash \mathtt{def}(d) : \mathtt{pred}(d)$ and $FV(\mathtt{def}(d)) \subseteq dom(\Gamma)$*

Now we will in the definition below present the rules of Definition 40 differently. Note that in Definition 53, if one takes $d$ to be a meta-variable for declarations only, $=_{\mathtt{def}}$ the same as $=_\beta$ (which is independent of $\vdash$) and the reduction relation as $\to_\beta$, then one gets the known cube of [2] given in Definition 40. We invite the reader to check this.

DEFINITION 53 (AXIOMS AND RULES OF THE CUBE).

*(axiom)* $\quad <> \vdash * : \Box$

*(start)* $\quad \dfrac{\Gamma \prec d}{\Gamma d \vdash \mathtt{subj}(d) : \mathtt{pred}(d)}$

*(weak)* $\quad \dfrac{\Gamma \prec d \qquad \Gamma\underline{d} \vdash D : E}{\Gamma d \vdash D : E}$

*(app)* $\quad \dfrac{\Gamma \vdash F : (A\Pi_x)B \qquad \Gamma \vdash a : A}{\Gamma \vdash (a\delta)F : B[x := a]}$

*(abs)* $\quad \dfrac{\Gamma(A\lambda_x) \vdash b : B \qquad \Gamma \vdash (A\Pi_x)B : S}{\Gamma \vdash (A\lambda_x)b : (A\Pi_x)B}$

*(conv)* $\quad \dfrac{\Gamma \vdash A : B \quad \Gamma \vdash B' : S \quad \Gamma \vdash B =_{\mathtt{def}} B'}{\Gamma \vdash A : B'}$

*(form)* $\quad \dfrac{\Gamma \vdash A : S_1 \quad \Gamma(A\lambda_x) \vdash B : S_2}{\Gamma \vdash (A\Pi_x)B : S_2}$ *if $(S_1, S_2)$ is a rule*

In order to solve the SR problem for the six remaining systems of the cube, we extend the $\lambda$-cube with definitions, $\rightsquigarrow_\beta$ and equivalence classes modulo CCF. Contexts now consist of declarations $(A\lambda_x)$ as well as definitions. We take the typing rules $\vdash^{\mathrm{c}}$ to be exactly those of $\vdash$ of Definition 53 with the addition of the definition rule:

(def rule) $\quad \dfrac{\Gamma d \vdash^{\mathrm{c}} C : D}{\Gamma \vdash^{\mathrm{c}} dC : [D]_d}$ if $d$ is a definition

With this definition, the problem of subject reduction is solved, and all the other desirable properties hold too. The reason that subject reduction holds now whereas it did not hold in Examples 41 and 42 can be intuitively seen by showing that the counterexample given in Example 41 no longer holds. Table 4 shows how the reduct of Example 41 can now be typed.

The following lemma tells us that the use of nested definitions such as $(A\delta)(B\delta)(C\lambda_x)(D\lambda_y)$ can be replaced by

**Table 4: Definitions solve subject reduction**

$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^c y' : \beta : * : \Box$
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^c y' : \beta, \alpha : *$ (weakening resp. start)
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^c \alpha =_{\mathtt{def}} \beta$ (use the definition in the context)
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^c y' : \alpha$ (conversion)
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x) \vdash^c x : \alpha$ (start)
$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^c (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \alpha[x := y][\alpha := \beta] \equiv \beta$ (definition rule)

using linear definitions such as $(B\delta)(C\lambda_x)(A\delta)(D\lambda_y)$ and that abstractions can be postponed.

LEMMA 54. *Let $d$ be a definition.*

1. *If $\Gamma d\Delta \vdash^c C =_{\mathtt{def}} D$ then $\Gamma\underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \vdash^c C =_{\mathtt{def}} D$.*
2. *If $\Gamma d\Delta \vdash^c C : D$ then $\Gamma\underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \vdash^c C : D$.*
3. *If $\Gamma(A\lambda_x)d\Delta \vdash^c C =_{\mathtt{def}} D$ then $\Gamma d(A\lambda_x)\Delta \vdash^c C =_{\mathtt{def}} D$ if $x \notin FV(d)$.*
4. *If $\Gamma(A\lambda_x)d\Delta \vdash^c C : D$ then $\Gamma d(A\lambda_x)\Delta \vdash^c C : D$ if $x \notin FV(d)$.*

PROOF. Note that $(A\lambda_x)$ does not need to be bachelor. 1. & 3. are by induction on the generation of $=_{\mathtt{def}}$. 2. & 4. are by induction on the derivation, using 1. & 3. for conversion. □

The following three lemmas and corollary are familiar from [2], but here we take also definitions into account.

LEMMA 55 (THINNING FOR $\vdash^c$).

1. *If $\Gamma_1\Gamma_2 \vdash^c A =_{\mathtt{def}} B$, $\Gamma_1\Delta\Gamma_2$ is a legal context, then $\Gamma_1\Delta\Gamma_2 \vdash^c A =_{\mathtt{def}} B$.*
2. *If $\Gamma$ and $\Delta$ are legal contexts such that $\Gamma \subseteq' \Delta$ and $\Gamma \vdash^c A : B$, then $\Delta \vdash^c A : B$. ($\subseteq'$ is context inclusion with side effects like a bachelor $\lambda$-item becoming partnered.)*

LEMMA 56 (GENERATION LEMMA FOR $\vdash^c$).

1. *If $\Gamma \vdash^c x : A$ then for some $B$: $(B\lambda_x) \in \Gamma$, $\Gamma \vdash^c B : S$, $\Gamma \vdash^c A =_{\mathtt{def}} B$ and $\Gamma \vdash^c A : S'$ for some sort $S'$.*
2. *If $\Gamma \vdash^c (A\lambda_x)B : C$ then for some $D$ and sort $S$: $\Gamma(A\lambda_x) \vdash^c B : D$, $\Gamma \vdash^c (A\Pi_x)D : S$, $\Gamma \vdash^c (A\Pi_x)D =_{\mathtt{def}} C$ and if $(A\Pi_x)D \not\equiv C$ then $\Gamma \vdash^c C : S'$ for a sort $S'$.*
3. *If $\Gamma \vdash^c (A\Pi_x)B : C$ then for some sorts $S_1, S_2$: $\Gamma \vdash^c A : S_1$, $\Gamma \vdash^c B : S_2$, $(S_1, S_2)$ is a rule, $\Gamma \vdash^c C =_{\mathtt{def}} S_2$ and if $S_2 \not\equiv C$ then $\Gamma \vdash^c C : S$ for some sort $S$.*
4. *If $\Gamma \vdash^c (A\delta)B : C$, $(A\delta)$ bachelor in $B$ , then for some terms $D, E$, variable $x$: $\Gamma \vdash^c A : D$, $\Gamma \vdash^c B : (D\Pi_x)E$, $\Gamma \vdash^c E[x := A] =_{\mathtt{def}} C$ and if $E[x := A] \not\equiv C$ then $\Gamma \vdash^c C : S$ for some sort $S$.*
5. *If $\Gamma \vdash^c \overline{s}A : B$, then $\Gamma\overline{s} \vdash^c A : B$.*

PROOF. 1., 2., 3. and 4. follow by a tedious but straightforward induction on the derivations (use the thinning lemma). As to 5., use induction on $\mathtt{weight}(\overline{s})$. □

LEMMA 57 (SUBSTITUTION LEMMA FOR $\vdash^c$).

1. *If $\Gamma(A\lambda_x)\Delta \vdash^c B : C$, $\Gamma \vdash^c D : A$ and $(A\lambda_x)$ bachelor in $\Gamma(A\lambda_x)\Delta$ then $\Gamma\Delta[x := D] \vdash^c B[x := D] : C[x := D]$.*
2. *If $\Gamma(D\delta)\overline{s}(A\lambda_x)\Delta \vdash^c B : C$ and $\overline{s}$ well-balanced then $\Gamma\overline{s}\Delta[x := D] \vdash^c B[x := D] : C[x := D]$.*

PROOF. Induction on the derivations (straightforward). □

COROLLARY 58 (CORRECTNESS OF TYPES). *If $\Gamma \vdash^c A : B$ then $B \equiv \Box$ or $\Gamma \vdash^c B : S$ for some sort $S$.*

PROOF. By induction to the derivation rules. □

Now, firstly we prove SR for $\vdash^c$ using $\twoheadrightarrow_\beta$ rather than $\rightsquigarrow_\beta$.

THEOREM 59 (SUBJECT REDUCTION FOR $\vdash^c$ AND $\twoheadrightarrow_\beta$). *If $\Gamma \vdash^c A : B$ and $A \twoheadrightarrow_\beta A'$ then $\Gamma \vdash^c A' : B$.*

PROOF. For $\Gamma \to_\beta \Gamma'$ defined in the expected way, we show by simultaneous induction on the derivation rules that:
1. If $\Gamma \vdash^c A : B$ and $\Gamma \to_\beta \Gamma'$ then $\Gamma' \vdash^c A : B$ and
2. If $\Gamma \vdash^c A : B$ and $A \to_\beta A'$ then $\Gamma \vdash^c A' : B$
using Lemmas 56.5 and 57 when reduction is at the root. □

Similarly to Lemma 48, we have by induction on the derivation rules that:

LEMMA 60 (CLASSES PRESERVE TYPES). *If $\Gamma \vdash^c A : B$ and $A' \in [A]$, $\Gamma'$ results from $\Gamma$ by substituting some main items $(C\omega)$ by $(C'\omega)$ where $C' \in [C]$, then $\Gamma' \vdash^c A' : B$.*

COROLLARY 61 (EQUIVALENT TERMS HAVE SAME TYPES).

1. *$\Gamma \vdash^c A : B \iff \Gamma \vdash^c \mathrm{CCF}(A) : B$.*
2. *If $\Gamma \vdash^c A : B$ and $A' \in [A]$, $B' \in [B]$ then $\Gamma \vdash^c A' : B'$.*

PROOF. By Lemma 60, conversion and correctness of types. □

Here is now the proof of SR using $\vdash^c$ and $\rightsquigarrow_\beta$, via the SR of $\vdash^c$ and $\twoheadrightarrow_\beta$.

COROLLARY 62 (SUBJECT REDUCTION FOR $\vdash^c$ AND $\rightsquigarrow_\beta$). *If $\Gamma \vdash^c A : B$ and $A \rightsquigarrow_\beta A'$ then $\Gamma \vdash^c A' : B$.*

PROOF. We prove similarly to Corollary 50 that: $\Gamma \vdash^c A : B$, $A \rightsquigarrow_\beta A' \Longrightarrow \Gamma \vdash^c A' : B$. □

LEMMA 63 (UNICITY OF TYPES FOR $\vdash^c$).

1. *$\Gamma \vdash^c A : B \wedge \Gamma \vdash^c A : B' \Rightarrow \Gamma \vdash^c B =_{\mathtt{def}} B'$*
2. *$\Gamma \vdash^c A : B \wedge \Gamma \vdash^c A' : B' \wedge A =_\beta A' \Rightarrow \Gamma \vdash^c B =_{\mathtt{def}} B'$*

PROOF. 1. By induction on the structure of $A$ using the Generation Lemma. 2. By Church-Rosser and Subject Reduction using 1. □

Finally, one can establish Strong Normalisation for the $\lambda$-cube with definitions and class-reduction by using the proof of Strong Normalisation of the $\lambda$-cube extended with definitions and $\hookrightarrow\!\!\!\!\rightarrow_\beta$ as in [3] (which is related to $\theta$-reduction) and mimicking that proof for $\gamma$-reduction.

THEOREM 64 (STRONG NORMALISATION OF $\rightsquigarrow_\beta$). *Every legal term is strongly normalising with respect to* $\rightsquigarrow_\beta$.

## 7. CONCLUSION

In this paper, we attempted to understand the reductional behaviour of calculations (or programs). We looked at two calculations and be able to tell whether there is an isomorphism between the two corresponding reduction paths. We provided a notion of reductional equivalence where we define a classification of terms so that elements that belong to the same class can be said to have the same reductional behaviour.

[18] already gave a notion of reductional equivalence called $\sigma$-equivalence for which it showed that none of the standard classification criteria on $\lambda$-calculus (e.g., length of the longest reduction) can separate two $\sigma$-equivalent terms. Our paper presented a fine grained reduction relation whose congruence is $\sigma$-equivalence.

Another attractive feature of our work is that we managed to give a clear representation of the canonical forms of terms given in [18] which clearly show where redexes occur and where they do not. Table 1 shows that every $\lambda$-term can be written in canonical form. Such a canonical form can be considered as a well-organised variant of the original term, yet having a similar reductional behaviour. A canonical form of a term $M$ lists the overall (bachelor) abstractions of $M$, followed by a permutable list of redex-heads (which can also be considered as possible substitutions), followed by a list of "idle" or bachelor arguments for a single variable $x$. The idle arguments can however become active in new redex-heads after a substitution of some term for $x$, e.g., by $\beta$-reduction. Furthermore, although canonical forms are not unique, we can still find for each $\lambda$-term, the unique class of its canonical forms which are all equal modulo some simple permutation.

Finally, we extended the cube of eight type systems with class reduction and showed that subject reduction fails for six of the eight extended systems. We then established that subject reduction can be regained by adding definitions. The importance of definitions (also known as "let expressions") is witnessed by their extensive use in programming languages and theorem provers. Intuitively, definitions repair the problem of subject reduction because they save the type information that will be lost as a result of reduction.

## 8. ACKNOWLEDGEMENTS

We are grateful for enlightening discussions and useful feedback and comments received from Henk Barendregt, Twan Laan and Joe Wells. We are also grateful for the useful comments received from the anonymous referees. Kamareddine is grateful to the support received from EPSRC grants EPSRC GR/L36963 and EPSRC GR/L15685.

## 9. REFERENCES

[1] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[2] H.P. Barendregt. $\lambda$-calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 118–310. Oxford University Press, 1992.

[3] R. Bloo, F. Kamareddine, and R. P. Nederpelt. The Barendregt Cube with Definitions and Generalised Reduction. *Information and Computation*, 126 (2):123–143, 1996.

[4] P. de Groote. The conservation theorem revisited. In *International Conference on Typed Lambda Calculi and Applications, LNCS* 664. Springer-Verlag, 1993.

[5] F. Kamareddine and R. Nederpelt. A useful $\lambda$-notation. *Theoretical Computer Science*, 155:85–109, 1996.

[6] F. Kamareddine and R. P. Nederpelt. Refining reduction in the $\lambda$-calculus. *Journal of Functional Programming*, 5(4):637–651, 1995.

[7] F. Kamareddine, A. Ríos, and J.B. Wells. Calculi of generalised $\beta_e$-reduction and explicit substitution: Type free and simply typed versions. *Journal of Functional and Logic Programming*, 1998.

[8] M. Karr. Delayability in proofs of strong normalizability in the typed $\lambda$-calculus. In *Mathematical Foundations of Computer Software, LNCS*, 185. Springer-Verlag, 1985.

[9] A.J. Kfoury, J. Tiuryn, and P. Urzyczyn. An analysis of ML typability. *ACM*, 41(2):368–398, 1994.

[10] A.J. Kfoury and J.B. Wells. A direct algorithm for type inference in the rank-2 fragment of the second order $\lambda$-calculus. *Proceedings of the 1994 ACM Conference on LISP and Functional Programming*, 1994.

[11] A.J. Kfoury and J.B. Wells. Addendum to new notions of reduction and non-semantic proofs of $\beta$-strong normalisation in typed $\lambda$-calculi. Technical report, Boston University, 1995.

[12] A.J. Kfoury and J.B. Wells. New notions of reductions and non-semantic proofs of $\beta$-strong normalisation in typed $\lambda$-calculi. *LICS*, 1995.

[13] Z. Khasidashvili. The longest perpetual reductions in orthogonal expression reduction systems. $3^{rd}$ *International Conference on Logical Foundations of Computer Science, Logic at St Petersburg*, 813, 1994.

[14] J. W. Klop. Combinatory Reduction Systems. *Mathematical Center Tracts*, 27, 1980. CWI.

[15] J.-J. Lévy. Optimal reductions. In J. Hindley and J. Seldin, editors, *To H.B. Curry: Essays on combinatory logic, lambda-calculus and formalism*, pages 159–191. Academic Press, 1980.

[16] R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer. *Selected papers on Automath*. North-Holland, Amsterdam, 1994.

[17] L. Regnier. *Lambda calcul et réseaux*. PhD thesis, University Paris 7, 1992.

[18] L. Regnier. Une équivalence sur les lambda termes. *Theoretical Computer Science*, 126:281–292, 1994.

[19] A. Sabry and M. Felleisen. Reasoning about programs in continuation-passing style. *Proceedings of the 1992 ACM Conference on LISP and Functional Programming*, pages 288–298, 1992.

[20] M. H. Sørensen. Strong normalisation from weak normalisation in typed $\lambda$-calculi. *Information and Computation*, 133(1), 1997.

[21] D. Vidal. *Nouvelles notions de réduction en lambda calcul*. PhD thesis, Université de Nancy 1, 1989.

[22] H. Xi. On weak and strong normalisations. Technical Report 96-187, Carnegie Mellon University, 1996.