De Bruijn's syntax and reductional behaviour of $\boldsymbol{\lambda}$-terms: the untyped case[*]

Fairouz Kamareddine[†] and Roel Bloo [‡]

Abstract

In this paper, a notation influenced by de Bruijn's syntax of the λ-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. We define reduction modulo equivalence classes of terms up to the permutation of redexes in canonical forms and show that this reduction contains other notions of reductions in the literature including the σ-reduction of Regnier. We establish all the desirable properties of our reduction modulo equivalence classes for the untyped λ-calculus.

Keywords: class reduction, canonical forms, reductional behaviour.

1 Introduction

The basic operations for building terms in the λ-calculus are abstraction and application. The basic reduction operation in the λ-calculus is β-reduction where

$$(\beta) \quad (\lambda_x.A)B \to_\beta A[x := B].$$

The β-redex $(\lambda_x.A)B$ is characterised by the matching of λ_x with the argument B. We say that λ_x and B match or that each has the other as a partner. However, not all λ's of a λ-term have partners and not all arguments match a λ. We call such items with no partners, *bachelors*.

Example 1 *In $\lambda_x.((\lambda_y.A)B)C$, the* items *$\lambda_x$ and C are bachelors whereas λ_y and B are partners. Similarly, in $((\lambda_x.\lambda_y.A)B)C$, the items λ_y and C are bachelors whereas λ_x and B are partners.*

After β-reductions take place in a term, items that are bachelor may well find a partner. For example, in the second term of Example 1, a new redex based on the then matching λ_y and C is created after the reduction based on the matching of λ_x and B takes place:

$$((\lambda_x.\lambda_y.A)B)C \to_\beta (\lambda_y.A[x := B])C.$$

Many researchers noted the need to rewrite terms like $((\lambda_x.\lambda_y.A)B)C$ to either $(\lambda_y.(\lambda_x.A)B)C$ or $(\lambda_x.(\lambda_y.A)C)B$ where it is seen that the bachelor λ_y and C become partnered and the future redex based on λ_y and C becomes a present redex. We refer to such new notions of reductions as *auxiliary reductions.* These auxiliary reductions can be summarized by four axioms:

[*]This article builds on and extends the results and proofs of [7] but only for the untyped case. The typed case can be found in [6]. We are grateful for enlightening discussions and useful feedback and comments received from Henk Barendregt, Twan Laan, Rob Nederpelt and Joe Wells. An anonymous referee provided useful recommendation for which we are grateful.

[†]School of Mathematical and Computational Sciences, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, Scotland, *fairouz@macs.hw.ac.uk*

[‡]Mathematics and Computing Science, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, the Netherlands, *c.j.bloo@tue.nl*

$$\begin{array}{llll}
(\theta) & ((\lambda_x.A)B)C & \to_\theta & (\lambda_x.AC)B \\
(\gamma) & (\lambda_x.\lambda_y.A)B & \to_\gamma & \lambda_y.(\lambda_x.A)B \\
(g) & ((\lambda_x.\lambda_y.A)B)C & \to_g & (\lambda_x.A[y:=C])B \\
(\gamma_C) & ((\lambda_x.\lambda_y.A)B)C & \to_{\gamma_C} & (\lambda_y.(\lambda_x.A)B)C
\end{array}$$

Note that g is a combination of a θ-step with a β-step. γ_C makes sure that λ_y and C form a redex even before the redex based on λ_x and B is contracted. By compatibility, γ implies γ_C. Moreover, $((\lambda_x.\lambda_y.A)B)C \to_\theta (\lambda_x.(\lambda_y.A)C)B$ and hence both θ and γ_C put $\lambda_y.A$ adjacently next to its matching argument C. In this case, θ moves the argument C (inwards) next to its matching λ_y whereas γ_C moves the λ_y (outwards) next to its matching argument. For a discussion of where these reductions have been used see [14, 10]. We give here a very brief summary.

[19] introduces the notion of a *premier redex* which is similar to the redex based on λ_y and C in the left hand side of rule (g) above (which we call *generalised redex*). [20] uses θ and γ (and calls the combination σ) to show that the perpetual reduction strategy finds the longest reduction path when the term is Strongly Normalizing (SN). [23] also introduces reductions similar to those of [20]. Furthermore, [12] uses θ (and other reductions) to show that typability in ML is equivalent to acyclic semi-unification. [21] uses a reduction related to θ where $((\lambda_x.\lambda_y.x)a)b)$ is transformed into $\lambda_k.((\lambda_x.((\lambda_y.kx)b))a)$. [3] identified the extra power of the CPS transformations of [21] to enable a more effective treatment of β-redexes. [18] and [4] use θ whereas [15] uses γ to reduce the problem of β-strong normalization to the problem of weak normalization (WN) for related reductions. [13] uses θ and γ to reduce typability in the rank-2 restriction of the 2nd order λ-calculus to the problem of acyclic semi-unification. [17, 24, 22, 16] use related reductions to reduce SN to WN and [11] uses similar notions in SN proofs. [9] uses a more extended version of θ (called *term-reshuffling*) and of g (called *generalised reduction*) where C and N are not only separated by the redex $(\lambda_x.-)B$ but by many redexes (ordinary and generalised). [5] shows that generalised reduction satisfies both the postponement of K-reductions and the conservation properties and also preserves the strong normalisation of the ordinary λ-calculus.

Looking at these four axioms, one notes that auxiliary reduction can help relate λ-terms according to their present and potential redexes. After all, auxiliary reduction turns redexes that are not immediately visible but yet implicitly present, into clearly visible ones:

Example 2 *Let $A \equiv (\lambda_{\rho y f}.fy)\alpha x$ and $B \equiv (\lambda_\rho.(\lambda_{yf}.fy)x)\alpha$. Both terms have $\lambda_f.fx$ as a reduct, so $A =_\beta B$. However, B has two redexes whereas A has only one. Here are the redexes of B:*

- *$r_1 = (\lambda_\rho.(\lambda_{yf}.fy)x)\alpha$. Observe that $B \overset{r_1}{\to}_\beta (\lambda_{yf}.fy)x$.*
- *$r_2 = (\lambda_{yf}.fy)x$. Observe that $B \overset{r_2}{\to}_\beta (\lambda_{\rho f}.fx)\alpha$.*

In A, the only redex is: $r_1' = (\lambda_{\rho y f}.fy)\alpha$. Here $A \overset{r_1'}{\longrightarrow}_\beta (\lambda_{yf}.fy)x$. Note that r_1 in B and r_1' in A are both based on the redex $(\lambda_\rho.-)\alpha$ and contracting r_1 in B or r_1' in A results in the same term.

A closer look at A enables us to see that in A (as in B), λ_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. On one hand, r_2 in B is completely visible and may be contracted before r_1 in B. On the other hand, r_2' 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 $(\lambda_{yf}.fy)x$, 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 λ_ρ is matched with α and λ_y is matched with x. This can be made visible via rules like (θ) above where

$$A \equiv (\lambda_{\rho y f}.fy)\alpha x \to_\theta (\lambda_\rho.(\lambda_{yf}.fy)x)\alpha \equiv B.$$

Regnier in [20] and Kfoury and Wells in [15] went further and used the above mentioned axioms to find for each term its so-called canonical form. The canonical form shows which parts of the term are partnered, 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 λx_i and A_j are bachelor and each C_k matches λy_k for $1 \leq i \leq n$, $1 \leq j \leq l$ and $1 \leq k \leq m$.

In addition to canonical forms, [20] provided the notion of σ-equivalence which identifies terms only differing by permutations of redexes, and showed that none of the standard operational classification criteria on λ-calculus (e.g., length of longest reduction) can separate two σ-equivalent terms. [20] concluded by asking if there existed a syntax that can faithfully represent σ-equivalence.

In this paper, we attempt to answer the question by using the item notation [8] inspired by de Bruijn's notation of the λ-calculus where both the rewriting of terms to create more redexes and the canonical forms of terms are clearer than in classical notation. In item notation, abstraction and application are written respectively as $(\lambda_x)A$ and $(B\delta)C$ with C the function and B the argument (see [8] and Section 2 of this paper). In item notation, canonical forms have the following shape:

$$(\lambda_{x_1})\cdots(\lambda_{x_n})(C_1\delta)(\lambda_{y_1})\cdots(C_m\delta)(\lambda_{y_m})(A_l\delta)\cdots(A_1\delta)z.$$

Hence, a canonical form is clearly divided into a sequence of bachelor λ-items (λ_{x_i}) followed by a sequence of partnered pairs $(C_j\delta)(\lambda_{y_j})$ followed by a sequence of bachelor δ-items $(A_k\delta)$ which is finally followed by the heart of the term z. This is clearer than the canonical form of [15, 20].

When working on the rewriting of terms to make more redexes visible, we were keen to detect when two terms A and B can be defined to be reductionally equivalent in the sense that there is a bijective correspondence between reduction paths starting at A and those starting at B. We believe that such a notion of reductional equivalence (which we call $\sim_{\mathtt{equi}}$) is hard to define and that it would be undecidable. However, in this paper, we find a decidable approximation $\approx_{\mathtt{equi}}$ to reductional equivalence on strongly normalising terms, which we call semi reductional equivalence. We build classes of terms modulo θ, γ and permutation of redexes and say that $A \approx_{\mathtt{equi}} B$ when A and B are in the same class. We show that $\approx_{\mathtt{equi}}$ coincides with σ-equivalence. Armed with our classes which represent the present and future redexes in a term, we extend the usual β-reduction to reduction modulo classes. We show that the reduction modulo satisfies all the desirable properties and that it generalises other notions of generalised reduction in the literature.

This paper is divided as follows:

- In Section 2 we introduce what is needed of the item notation and other formal machinery in order to give a transparent view on the canonical forms of terms.
- In Section 3 we explain how one can achieve the *canonical forms* of terms so that an approximation of the reductional behaviour is immediately visible.
- In Section 4 we give our decidable notion $\approx_{\mathtt{equi}}$ of semi reductional equivalence. We show that $\approx_{\mathtt{equi}}$ coincides with the σ-equivalence of [20]. We also define reduction modulo σ-equivalence.
- In Section 5 we extend the usual β-reduction $\to_\beta$ on λ-terms to $\rightsquigarrow_\beta$ on classes of terms modulo $\approx_{\mathtt{equi}}$ reductional equivalence. We establish that $\rightsquigarrow_\beta$ is Church-Rosser and that $\rightsquigarrow_\beta$ subsumes other notions of reduction including $\to_\beta$ and the reduction modulo σ-equivalence. We also show that if $A \rightsquigarrow_\beta B$ is based on a redex $(\lambda_x.-)-$, and if $A' \approx_{\mathtt{equi}} A$, then there exists $B' \approx_{\mathtt{equi}} B$ such that $A' \rightsquigarrow_\beta B'$ and $A' \rightsquigarrow_\beta B'$ is based on a corresponding redex $(\lambda_x.-)-$. In other words, A and A' have isomorphic reductional paths. We also show that $\approx_{\mathtt{equi}}$ is a good approximation to the reductional equivalence $\sim_{\mathtt{equi}}$ on strongly normalising terms. Finally, we show that $SN_{\rightsquigarrow_\beta}$ and $SN_{\to_\beta}$ are equivalent and that all semi reductionally equivalent terms have the same normalisation behaviour.

2 Some formal machinery

We assume familiarity with the λ-calculus and its notions such as compatibility and reduction (see [2]). 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 the usual Barendregt variable convention BC (which says that bound variables are always chosen distinct from free variables and that whenever necessary, variables are renamed to ensure this) (cf. [2]). For any reduction relation $\to_r$, we write $\twoheadrightarrow_r$ for its reflexive transitive closure and $=_r$ for its reflexive transitive and symmetric closure. We say that A is strongly normalizing with respect to a reduction relation $\to$ (written $\mathrm{SN}_{\to}(A)$) iff every $\to$-reduction path starting at A terminates. As usual, we use SN and CR to stand respectively for strong normalisation and Church Rosser.

The classical notation cannot extend the notion of redexes in a simple way. *Item notation* however can ([8] 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 λ-items like (λ_x) and δ-items like $(b\delta)$ followed by a variable. Moreover, a β-redex becomes in item notation a $\delta\lambda$-pair: namely, a δ-item adjacent to a λ-item. I.e., $(\lambda_x.A)B$ becomes in item notation: $(B\delta)(\lambda_x)A$. Note that in item notation, the scope of the x in a λ-item (λ_x) is everything to the right of it.

Let V be an infinite collection of variables over which $x, y, z, \ldots$ range. Terms are given by:

$$\mathcal{T} ::= V \mid (\mathcal{T}\delta)\mathcal{T} \mid (\lambda_V)\mathcal{T}.$$

We take $A, B, C, \ldots$ to range over $\mathcal{T}$. We call $(A\delta)$ a **δ-item**, whose **body** is A. By $(A\delta)B$ one means apply B to A (note the order). The item (λ_x) is called a **λ-item**. A redex starts with a δ-item next to a λ-item. Here we repeat rules (β), (θ), (γ), (g), (γ_C) but in item notation:

(β)	$(B\delta)(\lambda_x)A$	$\to_\beta$	$A[x := B]$
(θ)	$(C\delta)(B\delta)(\lambda_x)A$	$\to_\theta$	$(B\delta)(\lambda_x)(C\delta)A$
(γ)	$(B\delta)(\lambda_x)(\lambda_y)A$	$\to_\gamma$	$(\lambda_y)(B\delta)(\lambda_x)A$
(g)	$(C\delta)(B\delta)(\lambda_x)(\lambda_y)A$	$\to_g$	$(B\delta)(\lambda_x)\{A[y := C]\}$
(γ_C)	$(C\delta)(B\delta)(\lambda_x)(\lambda_y)A$	$\to_{\gamma_C}$	$(C\delta)(\lambda_y)(B\delta)(\lambda_x)A$

Note that the rules (θ), (γ), (g), (γ_C) are not problematic because we use the Barendregt Convention, which means that no free variable will become unnecessarily bound after reshuffling due to the fact that renaming of bound variables can be activated at any time to ensure 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 λ-item or a δ-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 δ-item immediately followed by a λ-item. The **weight** of a segment $\overline{s}$, $\mathtt{weight}(\overline{s})$, is the number of main items that compose the segment. Moreover, we define $\mathtt{weight}(\overline{s}x) = \mathtt{weight}(\overline{s})$ for $x \in V$.

In reduction, the *matching* of the δ and the λ in question is the important thing. **Well-balanced segments (w-b)** are constructed inductively from matching δ and λ-items as follows:

(i) $\emptyset$ is w-b,

(ii) if $\overline{s}$ is w-b then $(A\delta)\overline{s}(\lambda_x)$ is w-b,

(iii) if $\overline{s_1}$, $\overline{s_2}$, $\ldots \overline{s_n}$ 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.

Let $E \equiv \overline{s_1}(A\delta)\overline{s_2}(\lambda_y)\overline{s_3}x$. We say that the items $(A\delta)$ and (λ_y) **match** or are **partners** or **partnered** if $\overline{s_2}$ is well-balanced. If an item s has no partner in a term, we say that s is **bachelor**.

[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.

For example, in the term E_1 of Figure 1, $(+\delta)$ and (λ_f) match or are partnered. So are the items $(n\delta)$ and (λ_y). On the other hand, $(y\delta)$ and $(x\delta)$ are bachelor. The pair of adjacent items $(+\delta)(\lambda_f)$ 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**.

The next remark shows that an order needs to be followed to move items next to their partners using the rules θ and γ. For example, in $(A\delta)\overline{s}(\lambda_x)B$ where $\overline{s}$ is well-balanced, each of the θ-rule and the γ-rule states that each main δ-item of $\overline{s}$ must be moved next to its λ-partner before $(A\delta)$ can be moved next to its partner (λ_x).

Remark 3 *Assume that $\overline{s}$ is well-balanced.*

- *It is not necessarily the case that $(A\delta)\overline{s}(\lambda_x)B \twoheadrightarrow_\theta \overline{s}(A\delta)(\lambda_x)B$.*[2]
  For example, $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \not\twoheadrightarrow_\theta (A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(A_1\delta)(\lambda_z)A_4$ but instead, $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \twoheadrightarrow_\theta (A_3\delta)(\lambda_x)(A_2\delta)(\lambda_y)(A_1\delta)(\lambda_z)A_4$.

- *It is not necessarily the case that $(A\delta)\overline{s}(\lambda_x)B \twoheadrightarrow_\gamma (A\delta)(\lambda_x)\overline{s}B$.*[3]
  For example, $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \not\twoheadrightarrow_\gamma (A_1\delta)(\lambda_z)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)A_4$ but instead $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \twoheadrightarrow_\gamma (A_1\delta)(\lambda_z)(A_2\delta)(\lambda_y)(A_3\delta)(\lambda_x)A_4$.

- *Note finally that using the rules θ and γ together will not solve the problem: $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \not\twoheadrightarrow_{\theta\gamma} (A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(A_1\delta)(\lambda_z)A_4$ and $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \not\twoheadrightarrow_{\theta\gamma} (A_1\delta)(\lambda_z)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)A_4$.*

2.1 Making redexes visible via θ and γ

Transformations like (θ) and (γ) are rather powerful in that they can group together terms with equal reductional behavior. Let us start with (θ):

Example 4 *Consider E_1, E_2, E_3, E_4 as follows:*

$$\begin{aligned} E_1 &\equiv (((\lambda_{fxy}.fxy)+)m)n, \\ E_2 &\equiv ((\lambda_f.(\lambda_{xy}.fxy)m)+)n, \\ E_3 &\equiv (\lambda_f.((\lambda_{xy}.fxy)m)n)+, \\ E_4 &\equiv (\lambda_f.(\lambda_x.(\lambda_y.fxy)n)m)+. \end{aligned}$$

Note that $E_1 =_\beta E_2 =_\beta E_3 =_\beta E_4$. Moreover, the visible redexes are as follows:
In E_1: $(\lambda_{fxy}fxy)+$.
In E_2: $(\lambda_f.(\lambda_{xy}.fxy)m)+$ and $(\lambda_{xy}.fxy)m$.
In E_3: $(\lambda_f.((\lambda_{xy}.fxy)m)n)+$ and $(\lambda_{xy}.fxy)m$.
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 $E_1 \to_\theta E_2 \to_\theta E_3 \to_\theta E_4$ and that by θ-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 θ-normal form.

Applying the item notation to Example 4 we get:

Example 5 *E_1 of Example 4 reads in item notation: $(n\delta)(m\delta)(+\delta)(\lambda_f)(\lambda_x)(\lambda_y)(y\delta)(x\delta)f$. The (classical) redex corresponds to a '$\delta\lambda$-*pair*', viz. $(+\delta)(\lambda_f)$, followed by the body of the abstraction.*

Note that the δ-item $(+\delta)$ and the λ-item (λ_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 δ and λ-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 (λ_y) are separated by the segment $(m\delta)(+\delta)(\lambda_f)(\lambda_x)$ which has the bracketing structure $[\,[\,]\,]$.*

[2]For instance, it is not possible to have the bracketing structure $[_1[_2[_3]]] \twoheadrightarrow_\theta [_2[_3]][_1]$.
[3]For instance, it is not possible to have the bracketing structure $[_1[_2[_3]]] \twoheadrightarrow_\theta [_1][_2[_3]]$.

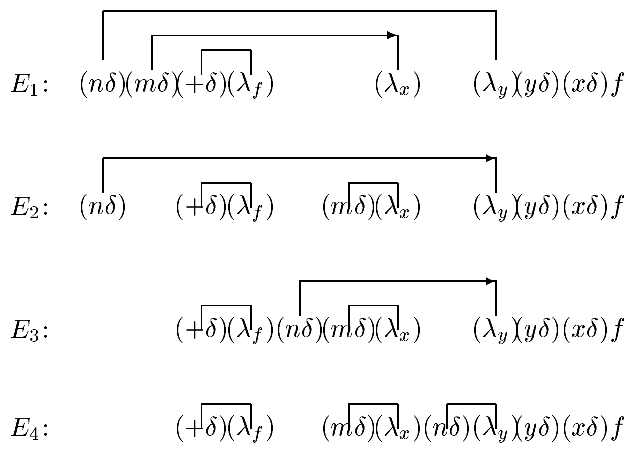
E_1: $(n\delta)(m\delta)(+\delta)(\lambda_f)$ (λ_x) $(\lambda_y)(y\delta)(x\delta)f$
E_2: $(n\delta)$ $(+\delta)(\lambda_f)$ $(m\delta)(\lambda_x)$ $(\lambda_y)(y\delta)(x\delta)f$
E_3: $(+\delta)(\lambda_f)(n\delta)(m\delta)(\lambda_x)$ $(\lambda_y)(y\delta)(x\delta)f$
E_4: $(+\delta)(\lambda_f)$ $(m\delta)(\lambda_x)(n\delta)(\lambda_y)(y\delta)(x\delta)f$

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

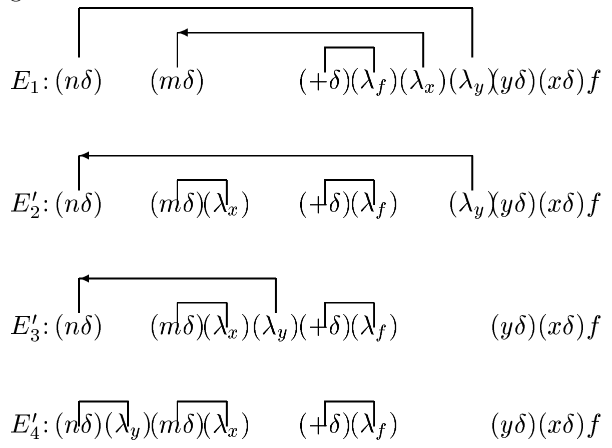
E_1: $(n\delta)$ $(m\delta)$ $(+\delta)(\lambda_f)(\lambda_x)(\lambda_y)(y\delta)(x\delta)f$
E_2': $(n\delta)$ $(m\delta)(\lambda_x)$ $(+\delta)(\lambda_f)$ $(\lambda_y)(y\delta)(x\delta)f$
E_3': $(n\delta)$ $(m\delta)(\lambda_x)(\lambda_y)(+\delta)(\lambda_f)$ $(y\delta)(x\delta)f$
E_4': $(n\delta)(\lambda_y)(m\delta)(\lambda_x)$ $(+\delta)(\lambda_f)$ $(y\delta)(x\delta)f$

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

- *$(\lambda_{xy}.fxy)m$ is visible as it corresponds to the matching $(m\delta)(\lambda_x)$ where $(m\delta)$ and (λ_x) are separated by the segment $(+\delta)(\lambda_f)$.*

θ-reduction amounts to moving δ-items, from left to right,[4] in the direction of their matching λ-items, until they form a pair (cf. Figure 1). As $\to_\theta$ is Church Rosser (CR) and Strongly Normalizing (SN), then the θ-normal form $\theta(M)$ of a term M is unique (cf. Proposition 16).

Looking back at examples 4 and 5, it is possible to use γ instead of θ in order to make more redexes visible (cf. Figure 2). γ-reduction amounts to moving λ-items from right to left, in the direction of their matching δ-items until they form a pair. Also, similarly to $\to_\theta$, $\to_\gamma$ is Church Rosser and Strongly Normalizing, and hence, the γ-normal form $\gamma(M)$ of a term M is unique.

This paper will establish a method that shows that terms like $E_1, E_2, E_2', E_3, E_3', E_4, E_4'$ in Figures 1 and 2 are reductionally equivalent.

2.2 Generalising reductions to take care of θ and γ

Look again at the rules (θ), (γ) and (g). One can say that (g) is one of the following steps:

- A θ-step followed by β-reduction where
  $(C\delta)(B\delta)(\lambda_x)(\lambda_y)A \to_\theta (B\delta)(\lambda_x)(C\delta)(\lambda_y)A \to_\beta (B\delta)(\lambda_x)\{A[y := C]\}$.

[4]This is not only for main items at the top level, but also inside the items.

- A γ-step followed by β-reduction where
  $(C\delta)(B\delta)(\lambda_x)(\lambda_y)A \to_\gamma (C\delta)(\lambda_y)(B\delta)(\lambda_x)A \to_\beta (B\delta)(\lambda_x)\{A[y := C]\}$.

So, following this, one can generalise β-reduction so that many steps θ or γ are simulated. This was done in [9] where reduction was generalised as follows:

Definition 6 (Extended redexes, $\hookrightarrow_\beta$)

- *An extended redex starts with the δ-item of a $\delta\lambda$-***couple** *(i.e. is of the form $(A\delta)\overline{s}(\lambda_x)B$ where $\overline{s}$ is well-balanced).*
- *$\hookrightarrow_\beta$ is the least compatible relation generated by $(A\delta)\overline{s}(\lambda_x)B \hookrightarrow_\beta \overline{s}\{B[x := A]\}$ for $\overline{s}$ well-balanced, that is, $\hookrightarrow_\beta$-reduction contracts an (extended) redex.*
- *$\hookrightarrow\!\!\!\!\rightarrow_\beta$ is the reflexive and transitive closure of $\hookrightarrow_\beta$ and $\sim_\beta$ the least equivalence relation closed under $\hookrightarrow\!\!\!\!\rightarrow_\beta$.*

Following [3], $\hookrightarrow_\beta$-reduction is the more effective treatment of β-redexes which identifies the power of the CPS transformations of [21]. Note furthermore, that $\hookrightarrow_\beta$-reduction is more refined than $\twoheadrightarrow_{\theta\gamma}$ followed by $\to_\beta$. In fact, recall Remark 3 and check that
$A \equiv (A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)A_4 \hookrightarrow_\beta (A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)\{A_4[z := A_1]\} \equiv B$
and that we cannot find a path of $\to_\theta$, $\to_\gamma$ and $\to_\beta$ steps starting at A and ending in B.

In this paper we show that our new notion of reduction based on classes is even more refined than and subsumes $\hookrightarrow_\beta$-reduction.

3 Reductional Equivalence and Canonical Forms

3.1 Reductional equivalence

Ideally, we would like reductional equivalence to be an equivalence relation which satisfies that terms A and B are reductionally equivalent if for every redex r in A, there is a *corresponding* redex r' in B where $A \overset{r}{\to}_\beta A'$, $B \overset{r'}{\to}_\beta B'$, and A' and B' are reductionally equivalent. Unfortunately, this relation seems hard to define in a non-involved manner; the notion of corresponding redex has to involve the position of the redex in the term for example. This section discusses those difficulties.

First, note that in order to discuss reductional equivalence between terms, auxilliary redexes must be included so that a potential future redex like $(\lambda_y.-)x$ in A of Example 2 can be treated as a present (rather than potential) redex which could possibly be contracted in A even before the originator $(\lambda_{\rho yf}.fy)\alpha$ has been contracted. Hence, with this extended notion of reduction we get in A another redex:

$r'_2 = (\lambda_{yf}.fy)x$, which when contracted in A results in $(\lambda_{\rho f}.fx)\alpha$.

Note that r'_2 is λ_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 2. 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 7 *The reduction paths from A and B of Example 2 are as follows:*

A-Path$_1$: $(\lambda_{\rho yf}.fy)\alpha x \overset{r'_1}{\to}_\beta (\lambda_{yf}.fy)x \to_\beta \lambda_f.fx$

A-Path$_2$: $(\lambda_{\rho yf}.fy)\alpha x \overset{r'_2}{\to}_\beta (\lambda_{\rho f}.fx)\alpha \to_\beta \lambda_f.fx$

B-Path$_1$: $(\lambda_\rho.(\lambda_{yf}.fy)x)\alpha \overset{r_1}{\to}_\beta (\lambda_{yf}.fy)x \to_\beta \lambda_f.fx$

B-Path$_2$: $(\lambda_\rho.(\lambda_{yf}.fy)x)\alpha \overset{r_2}{\to}_\beta (\lambda_{\rho f}.fx)\alpha \to_\beta \lambda_f.fx$

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-Path$_1$ and B-Path$_1$, and between A-Path$_2$ and B-Path$_2$.

Using auxiliary redexes, we came up with an informal definition of what we call $\sim_{\texttt{equi}}$:

Definition 8 (Reductional equivalence $\sim_{\texttt{equi}}$)
We say that A and B are reductionally equivalent and write $A \sim_{\texttt{equi}} B$ iff $A \equiv B$ or there is a bijective correspondence f between the (extended) redexes of A and B such that if $A \overset{r}{\to} A'$ and $B \overset{f(r)}{\to} B'$ then $A' \sim_{\texttt{equi}} B'$.

Note that if A is in normal form then $A \sim_{\texttt{equi}} B$ iff $A \equiv B$.

Example 9
- *$A \sim_{\texttt{equi}} B$ for A, B as in Example 2.*
- *Also $E_1 \sim_{\texttt{equi}} E_2 \sim_{\texttt{equi}} E_3 \sim_{\texttt{equi}} E_4$ for E_1, E_2, E_3, E_4 as in Example 4.*
- *However, because there is no bijective correspondence f between the (extended) redexes, it is not the case that $\mathbf{KII} \sim_{\texttt{equi}} \mathbf{KI\Omega}$ where $\mathbf{K}$ is $\lambda_{xy}.x$, $\mathbf{I}$ is $\lambda_x.x$, and $\mathbf{\Omega}$ is $(\lambda_x.xx)(\lambda_x.xx)$.*

Remark 10 *Note that Definition 8 has some limitations:*

- *We have that $(\lambda_x.\mathbf{I})\mathbf{I} \sim_{\texttt{equi}} (\lambda_x.\mathbf{I})\mathbf{K}$ although this is not desirable.*
- *$\sim_{\texttt{equi}}$ is not compositional. That is: if $A_1 \sim_{\texttt{equi}} A_2$ then it is necessarily the case that $A_1 B \sim_{\texttt{equi}} A_2 B$ and $\lambda_x.A_1 \sim_{\texttt{equi}} \lambda_x.A_2$. For example, if $A_1 \equiv \lambda_z.(\lambda_x.y)z$ and $A_2 \equiv \lambda_z.(\lambda_x.y)\mathbf{I}$, then $A_1 \sim_{\texttt{equi}} A_2$ but $A_1(\mathbf{II}) \not\sim_{\texttt{equi}} A_2(\mathbf{II})$.*

In order to deal with these limitations, we need to add the following condition to Definition 8:

If $r \equiv (\lambda_x.-)C$ is a (extended) redex of A and $f(r) \equiv (\lambda_x.-)D$ then $C \sim_{\texttt{equi}} D$.

This extra condition solves the problems raised by the above two situations. However, we will not be concerned with these situations in this paper and we will therefore not include clause 4 in Definition 8.

We conjecture that in general it is undecidable whether two terms are reductionally equivalent according to Definition 8.

Conjecture 11 (Undecidability of $\sim_{\texttt{equi}}$) *It is in general undecidable whether two terms are reductionally equivalent.*

Note that we can define $\sim_{\texttt{equi}}$ to be the infinite limit of decidable relations as suggested by Henk Barendregt, in personal communications. The idea is to define *degrees of reductional equivalence* ($\sim_n$ with $n \geq 0$ for short) in the following way:

- $M \sim_0 N$ iff $M \equiv N$.
- $M \sim_{n+1} N$ iff there is a bijective correspondence between the (extended) redexes of M and N such that contracting one in M yields a term $\sim_m$, $m \leq n$ to the result of contracting the corresponding redex in N.

It is easy to show that $\sim_{\texttt{equi}} = \bigcup_{n\geq 0} \sim_n$. Similarly to $\sim_{\texttt{equi}}$, $\sim_n$ for fixed $n \geq 2$ is not compositional. This can be seen as follows[5]:
$\lambda_{z:g}.(\lambda_{x:c}.\lambda_{y:d}.e)ba \sim_1 \lambda_{z:g}.(\lambda_{x:c}.(\lambda_{y:d}.e)a)b$ but
$(\lambda_{z:g}.(\lambda_{x:c}.\lambda_{y:d}.e)ba)f \sim_2 (\lambda_{z:g}.(\lambda_{x:c}.(\lambda_{y:d}.e)a)b)f$.

In short, reductional equivalence $\sim_{\texttt{equi}}$ is cumbersome to define. We will instead show that an approximation of reductional equivalence $\approx_{\texttt{equi}}$, called semi reductional equivalence, exists and is decidable. We will show that σ-equivalence and our equivalent decidable notion $\approx_{\texttt{equi}}$ are both incomparable to reductional equivalence of any degree $\sim_n$, $n \geq 0$. We will however show that they are both good approximations to $\sim_{\texttt{equi}}$ on strongly normalizing terms (cf. Fact 57). Canonical forms will be basic for our notion of reductional equivalence.

[5]This counterexample will be better understood if it is translated into the item-notation of Section 2.

Table 1: The Canonical Form of terms

bachelor λ-items	$\delta\lambda$-pairs	bachelor δ-items	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

3.2 Canonical forms

Consider two terms A and B. Obviously, if either $A =_\theta B$ or $A =_\gamma B$, then A and B are reductionally equivalent. But, what about if $A =_\theta C$ and $B =_\gamma C$? Would it still be the case that A and B are reductionally equivalent? The answer is yes. Look at E_4 and E'_4 of Figures 1 and 2. We want the reduction equivalence relation to capture the reductional equivalence of these terms.

Observing E_4 and E'_4, leads us to note that using θ alone or γ alone will not be comprehensive enough to capture as many cases as possible of reductional equivalence. We obviously want E_1, E_2, E_3 and E_4 of Example 4 to be reductionally equivalent, and also E_1, E'_2, E'_3 and E'_4. But, how do we relate E_i to E'_i for $2 \le i \le 4$? This is simple, combine the relations θ and γ and aim to find a canonical form of terms that helps establish semi reductional equivalence.

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 shape which we call *canonical form* (see Table 1):

Definition 12 (Canonical forms) *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, C_i and A_i are not required to be canonical forms themselves, and that for $1 \le i \le n$ and $1 \le j \le l$, (λ_{x_i}) and $(A_j\delta)$ are bachelor .

Remark 13 *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 λ_{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 [20] and represented in [20] by Figure 3 below. Note that our item notation as is seen in Definition 12 permits a more elegant representation than the one given in classical notation in Figure 3.

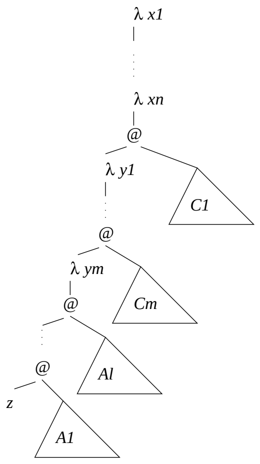

Figure 3: Canonical forms in classical notation

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 as follows:

Definition 14 *We define $\to_p$ on canonical forms as the compatible closure on canonical forms of the rule:* $(A_1\delta)(\lambda y_1)(A_2\delta)(\lambda y_2)B \to_p (A_2\delta)(\lambda y_2)(A_1\delta)(\lambda y_1)B$ *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 $\to_\theta \cup \to_\gamma \cup \to_p$. Furthermore, $\twoheadrightarrow_{\theta\gamma}$, $\twoheadrightarrow_{\theta\gamma p}$, $=_{\theta\gamma}$ 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 15 *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 $C \equiv (A_2\delta)(A_1\delta)(\lambda_{y_1})(\lambda_{y_2})A_3$. $\Box$

Proposition 16 *$\to_\theta$ and $\to_\gamma$ are SN, CR and $\subset =_\beta$. Moreover, $\to_{\theta\gamma}$ is also SN. Also, $=_{\theta\gamma p}$ and $=_{\theta\gamma}$ are the same relation.*

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. Finally, the equality of $=_{\theta\gamma p}$ and $=_{\theta\gamma}$is a consequence of Lemma 15. $\Box$

Notation 17 *Let M be a term. We use the following notation:*

- *$\theta(M)$ denotes the θ-normal form of M*
- *$\gamma(M)$ denotes the γ-normal form of M*
- *We call $\theta(\gamma(M))$ the $\theta\gamma$-normal form of M*
- *We call $\gamma(\theta(M))$ the $\gamma\theta$-normal form of M*

Corollary 18 *For each term M, the r-normal form of M for $r \in \{\theta, \gamma, \theta\gamma, \gamma\theta\}$ is unique.*

Note that it is not necessarily the case that $\theta(\gamma(A)) = \gamma(\theta(A))$ as Example 28 shows. However, we will show in Lemma 25 that $\theta(\gamma(A)) =_p \gamma(\theta(A))$.

The following two lemmas enable us to syntactically describe θ- and γ-normal forms.

Lemma 19 *Every term has one of the three forms:*

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

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

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

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

Lemma 20 *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* λ*-items,*

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

(iv) x.

Proof: A term has at least one bachelor main λ-item (case (i)), or none at all. In the last case, the term may start with a bachelor δ-item (case (ii)), a partnered δ-item (case (iii)) or is only a variable (case (iv)). $\Box$

Now, we can syntactically characterise θ- and γ-normal forms via the following two lemmas whose proof is by induction on the structure of terms as given in Lemmas 19 and 20 resp.:

Lemma 21 *The θ-normal form $\theta(M)$ of a term M is:*

$$\begin{array}{lll}
\theta((A_1\delta)\cdots(A_n\delta)x) & \equiv (\theta(A_1)\delta)\cdots(\theta(A_n)\delta)x & \text{if } x \in V \text{ and } n \geq 0 \\
\theta((\lambda_x)A) & \equiv (\lambda_x)\theta(A) & \\
\theta((A_1\delta)\cdots(A_n\delta)(B\delta)(\lambda_x)C) & \equiv (\theta(B)\delta)(\lambda_x)\theta((A_1\delta)\cdots(A_n\delta)C) &
\end{array}$$

Lemma 22 *The γ-normal form $\gamma(M)$ of a term M is:*

$$\begin{array}{llll}
\gamma(\overline{s}(\lambda_x)A) & \equiv & (\lambda_x)\gamma(\overline{s}A) & \text{if } \overline{s} \text{ is w-b,} \\
\gamma((A\delta)B) & \equiv & (\gamma(A)\delta)\gamma(B) & \text{if } B \text{ has no bachelor main } \lambda\text{-items,} \\
\gamma((A\delta)\overline{s}(\lambda_x)B) & \equiv & (\gamma(A)\delta)(\lambda_x)\gamma(\overline{s}B) & \text{where } \overline{s} \text{ is w-b and } B \text{ has no bachelor main } \lambda\text{-items} \\
\gamma(x) & \equiv & x &
\end{array}$$

Example 23 *In this example, we will decorate some items with various symbols (like $\times$, $\bullet$, etc.). Items that have the same decorations are partnered.*

If we take A to be $(\lambda_q)\ \overset{-}{(j\delta)}\overset{-}{(\lambda_p)}\ (\lambda_x)(w\delta)\ \overset{+}{(x\delta)}\overset{\bullet}{(y\delta)}\ \overset{\bullet}{(\lambda_v)}\ \overset{\times}{(v\delta)}\overset{''}{(x\delta)}\overset{''}{(\lambda_w)}\overset{\times}{(\lambda_t)}\overset{+}{(\lambda_s)}\ (s\delta)t$ *then:*

- $\theta(A)$ *is* $(\lambda_q)\ \overset{-}{(j\delta)}\overset{-}{(\lambda_p)}\ (\lambda_x)\ \overset{\bullet}{(y\delta)}\ \overset{\bullet}{(\lambda_v)}\ \overset{''}{(x\delta)}\overset{''}{(\lambda_w)}\overset{\times}{(v\delta)}\overset{\times}{(\lambda_t)}\overset{+}{(x\delta)}\overset{+}{(\lambda_s)}\ (w\delta)(s\delta)t$

- $\gamma(A)$ *is* $(\lambda_q)(\lambda_x)\ \overset{-}{(j\delta)}\overset{-}{(\lambda_p)}\ (w\delta)\ \overset{+}{(x\delta)}\overset{+}{(\lambda_s)}\overset{\bullet}{(y\delta)}\ \overset{\bullet}{(\lambda_v)}\ \overset{\times}{(v\delta)}\overset{\times}{(\lambda_t)}\overset{''}{(x\delta)}\overset{''}{(\lambda_w)}\ (s\delta)t$

Notation 24 *Let A be a term. We define the following:*

- A^{λ} *is the sequence of all bachelor main λ-items of A in the order in which they appeared in A.*

- $A^{\theta(\delta)}$ *is the sequence of the θ-normal form of all bachelor main δ-items of A in the order in which they appeared in A.*

- $A^{\theta(\delta\lambda)+\lambda}$ *is the sequence of the θ-normal form of all the main $\delta\lambda$-pairs (obtained from the $\delta\lambda$-couples) and all the main bachelor λ-items. All the λ-items are in the order in which they appeared in A and each δ-item occurs adjacent and to the left of its partner.*

- $A^{\gamma(\delta\lambda+\delta)}$ *is the sequence of the γ-normal form of all the main $\delta\lambda$-pairs (obtained from the $\delta\lambda$-couples) and of all the main bachelor δ-items. All the δ-items are in the order in which they appeared in A and each λ-item occurs adjacent and to the right of its partner.*

- $A^{\theta\gamma(\delta\lambda)}$ *is the sequence of the $\theta\gamma$-normal forms of all the main $\delta\lambda$-pairs (obtained from the $\delta\lambda$-couples).*

- $A^{\theta\gamma(\delta)}$ *is the sequence of the $\theta\gamma$-normal forms of all the main bachelor δ-items in the order in which they appeared in A.*

- $A^{\gamma\theta(\delta\lambda)}$ *is the sequence of the $\gamma\theta$-normal forms of main $\delta\lambda$-pairs (obtained from the $\delta\lambda$-couples).*

- $A^{\gamma\theta(\delta)}$ *is the sequence of the $\gamma\theta$-normal forms of all the main bachelor δ-items in the order in which they appeared in A.*

The following lemma shows that θ-, γ-, 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 [15, 20] where the classical syntax makes these normal forms cumbersome to describe.

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

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

Lemma 25 *For any term A, we have:*

1. $\theta(A) \equiv A^{\theta(\delta\lambda)+\lambda}A^{\theta(\delta)}\heartsuit(A)$.
2. $\gamma(A) \equiv A^{\lambda}A^{\gamma(\delta\lambda+\delta)}\heartsuit(A)$.
3. $\theta(\gamma(A)) \equiv A^{\lambda}A^{\theta\gamma(\delta\lambda)}A^{\theta\gamma(\delta)}\heartsuit(A)$.
4. $\gamma(\theta(A)) \equiv A^{\lambda}A^{\gamma\theta(\delta\lambda)}A^{\gamma\theta(\delta)}\heartsuit(A)$.
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 19 and 20 using Lemmas 21 and 22. 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 $A^{\theta(\delta\lambda)+\lambda}$ 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 $\theta((B_1\delta)\cdots(B_n\delta)E) \equiv \overline{s'_1}\ A^{\theta(\delta)}\heartsuit(E) \equiv \overline{s'_1}\ A^{\theta(\delta)}\heartsuit(A)$ where $\overline{s'_1}$ is the sequence of the θ-normal form of all the main $\delta\lambda$-pairs (obtained from the $\delta\lambda$-couples) and all the main bachelor λ-items. All the λ-items are in the order in which they appeared in A and each δ-item occurs adjacent and to the left of its partner. Hence, $\theta(A) \equiv A^{\theta(\delta\lambda)+\lambda}A^{\theta(\delta)}\heartsuit(A)$.

For 5) use 1) ... 4). □

Recall that both E_4 and E'_4 of Figures 1 and 2 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 25.4, $E_4 =_p E'_4$. We group all canonical forms related by $=_p$ into one class:

Definition 26 (Class of canonical forms CCF**)** *We define the class of canonical forms of M,* CCF(M) *as $\{M' \mid M' =_p \theta(\gamma(M))\}$.*

Note that by Lemma 25 we have CCF(M) $= \{M' \mid M' =_p \gamma(\theta(M))\}$.

For example, CCF$(E_1) = \{E_4, E'_4\}$.

4 Semi Reductional Equivalence and σ-equivalence

4.1 The relation $\approx_{\mathtt{equi}}$

Now, we are ready to define our notion of semi reductional equivalence. We say that two terms are semi reductionally equivalent if they have the same canonical form modulo $=_p$:

Definition 27 ($\approx_{\mathtt{equi}}$) *For a term A, we define:*

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

Note that, by Lemma 25.5, $[A] = \{B \mid \gamma(\theta(A)) =_p \gamma(\theta(B))\}$.

Example 28 *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.*

The following lemma says that semi reductional-equivalence $\approx_{\texttt{equi}}$ contains $\rightarrow_\theta$ and $\rightarrow_\gamma$.

Lemma 29 *$\rightarrow_\theta \subset \approx_{\texttt{equi}}$ and $\rightarrow_\gamma \subset \approx_{\texttt{equi}}$. Moreover, these inclusions are strict.*

Proof: If $A \rightarrow_\theta B$ or $A \rightarrow_\gamma B$ then $\theta(\gamma(A)) =_p \theta(\gamma(B))$. Example 28 gives terms E_4 and E'_4 which are $\approx_{\texttt{equi}}$ but which are not related by $\rightarrow_\theta$ or $\rightarrow_\gamma$. □

Remark 30 *Note that, as both $\rightarrow_\theta$ and $\rightarrow_\gamma$ are SN, we can by applying $\rightarrow_\theta$ and $\rightarrow_\gamma$ to any term A, reach a term A' which is free of any θ- and γ-redexes (it is easy to show that the combination of θ- and γ-reduction is SN). The resulting term A' however depends on the order of applying θ and γ. It is the case nonetheless, by Lemma 29 that all terms A', which are obtained from A via arbitrary θ and γ reductions, are semi reductionally equivalent.*

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

Proposition 31 *$\approx_{\texttt{equi}}$ is well-defined, decidable and is an equivalence relation. Moreover, $=_\gamma$, $=_\theta$, $=_p \subset \approx_{\texttt{equi}} \subset =_\beta$, and these inclusions are strict.*

Proof: Well-definedness and equivalence relation are easy. Similarly, decidability is easy as θ and γ are SN and $=_p$ is decidable. For the first $\subset$, note Lemmas 15 and 29. The second $\subset$ follows from Proposition 16. □

The next section gives the definition of σ-equivalence of [20] and establishes its equivalence to $\approx_{\texttt{equi}}$. Then, it defines β-reduction modulo σ-equivalence.

4.2 σ-equivalence and reduction modulo σ-equivalence

In [20], Regnier defined an equivalence relation called σ-equivalence on λ-terms which identified terms modulo the permutation of their redexes. [20] showed that none of the standard operational classifications of the λ-calculus can distinguish two σ-equivalent terms. In particular, [20] showed that any two σ-equivalent terms have the same normal form, the same length of head reduction, the same length of normalisation by leftmost reduction, and the same length of longest reduction. [20] also used σ-equivalence to generalise the theorem of perpetual strategy and to find the canonical forms of a term. This section establishes that our $\approx_{\texttt{equi}}$ is equivalent to σ-equivalence and defines β-reduction modulo σ-equivalence. First, we give the definition of σ-equivalence:

Definition 32 (σ-equivalence)
[20] defined σ-reduction $\rightarrow_\sigma$ to be the smallest compatible relation containing:

(θ) $((\lambda_x.A)B)C \rightarrow_\theta (\lambda_x.AC)B$ *if* $x \notin C$

(γ) $(\lambda_x.\lambda_y.A)B \rightarrow_\gamma \lambda_y.(\lambda_x.A)B$ *if* $y \notin B$

A and B are σ-equivalent if $A =_\sigma B$ whre $=_\sigma$ is the equivalence relation associated to $\rightarrow_\sigma$.

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

Lemma 33 *$A \approx_{\texttt{equi}} B$ iff $A =_{\theta\gamma p} B$ iff $A =_{\theta\gamma} B$.*

Proof: $\Longrightarrow$) Note that $\approx_{\texttt{equi}} \subseteq =_{\theta\gamma p}$ and that by Lemma 15, $=_{\theta\gamma p} \subseteq =_{\theta\gamma}$.
$\Longleftarrow$) By Lemma 29, we have $=_{\theta\gamma} \subseteq \approx_{\texttt{equi}}$ and so, also $=_{\theta\gamma p} \subseteq \approx_{\texttt{equi}}$ $\Box$
Hence, we have provided a fine grained notion of σ-equivalence:

Corollary 34 *σ-equivalence and $\approx_{\texttt{equi}}$ are the same relation.* **Proof:** *This holds because σ-equivalence is the same as $=_{\theta\gamma}$.* $\Box$

Now we define β-reduction modulo σ-equivalence:

Definition 35 (Reduction modulo σ, $\mapsto_\beta$)

- *One-step reduction modulo σ, $\mapsto_\beta$ is the least compatible relation generated by:*
  $A \mapsto_\beta B$ iff $\exists C =_\sigma A$ such that $C \to_\beta B$.
- *Many-step class-reduction modulo σ, $\mapsto\!\!\!\!\twoheadrightarrow_\beta$ is the reflexive and transitive closure of $\mapsto_\beta$ and $\cong_\beta$ is the least equivalence relation generated by $\mapsto_\beta$.*

The next Lemma and its Corollary help establish that $\mapsto\!\!\!\!\twoheadrightarrow_\beta$ is Church Rosser.

Lemma 36 *If $A \mapsto_\beta B$ then $A =_\beta B$.*

Proof: If $A \mapsto_\beta B$ then $C \to_\beta B$ for some $C =_\sigma A$. Hence, $C =_\beta B$ and by σ-equivalence, $C =_\beta A$. Hence $A =_\beta B$. $\Box$

Corollary 37 *1. If $A \mapsto\!\!\!\!\twoheadrightarrow_\beta B$ then $A =_\beta B$.* *2. $A \cong_\beta B$ iff $A =_\beta B$.*

Theorem 38 (Church Rosser theorem for $\mapsto\!\!\!\!\twoheadrightarrow_\beta$)
If $A \mapsto\!\!\!\!\twoheadrightarrow_\beta B$ and $A \mapsto\!\!\!\!\twoheadrightarrow_\beta C$, then there exists D such that $B \mapsto\!\!\!\!\twoheadrightarrow_\beta D$ and $C \mapsto\!\!\!\!\twoheadrightarrow_\beta D$.

Proof: As $A \mapsto\!\!\!\!\twoheadrightarrow_\beta B$ and $A \mapsto\!\!\!\!\twoheadrightarrow_\beta C$ then by Corollary 37, $A =_\beta B$ 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, $M \twoheadrightarrow_\beta A$ implies $M \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta A$ (cf. Corollary 42). Hence we are done. $\Box$

5 Class Reduction

In this section, we introduce class-reduction $\rightsquigarrow_\beta$, show that it is Church-Rosser and that it is more general than (i.e., subsumes) other notions of reduction including $\to_\beta$ and the reduction relation based on the σ-equivalence. We also show that if $A \rightsquigarrow_\beta B$ is based on a redex $(-\delta)(\lambda_x)$ then for every $A' \approx_{\texttt{equi}} A$, there exists $B' \approx_{\texttt{equi}} B$ 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 semi reductionally equivalent terms have the same normalisation behaviour.

Definition 39 (Class-reduction $\rightsquigarrow_\beta$)

- *One-step class-reduction $\rightsquigarrow_\beta$ is the least compatible relation generated by:*
  $A \rightsquigarrow_\beta B$ iff $\exists A' \in [A]$ (i.e., $A' \approx_{\texttt{equi}} A$) $\exists B' \in [B]$ (i.e., $B' \approx_{\texttt{equi}} B$) such that $A' \to_\beta B'$.
- *Many-step class-reduction $\rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta$ is the reflexive and transitive closure of $\rightsquigarrow_\beta$ and $\approx_\beta$ is the least equivalence relation generated by $\rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta$.*
- *We write $A \to_\beta^{(E\delta)(\lambda_x)} B$ for the β-reduction based on a β-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 $A' \to_\beta^{(E'\delta)(\lambda_x)} B'$.*

Example 40 *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 various other notions of reduction including classical β-reduction and reduction modulo σ-equivalence.

Lemma 41 $\to_\beta \subset \to_g \subset \hookrightarrow_\beta \subset \mapsto_\beta \subset \rightsquigarrow_\beta$*, and all these inclusions are strict.*

Proof: If we show $(A\delta)(\lambda_x)C \to_g C[x := A]$, $(C\delta)(B\delta)(\lambda_x)(\lambda_y)A \hookrightarrow_\beta (B\delta)(\lambda_x)\{A[y := C]\}$, and $(A\delta)\overline{s}(\lambda_x)C \mapsto_\beta \overline{s}C[x := A]$, this would be sufficient for the first three inclusions.

- First note that $(A\delta)(\lambda_x)C \equiv (A\delta)\emptyset(\lambda_x)C \to_g \emptyset C[x := A] \equiv C[x := A]$.
- Also, by Definition 6, $(C\delta)(B\delta)(\lambda_x)(\lambda_y)A \hookrightarrow_\beta (B\delta)(\lambda_x)\{A[y := C]\}$.
- Finally, we can easily show that $(A\delta)\overline{s}(\lambda_x)C =_\sigma \overline{s}(A\delta)(\lambda_x)C$, and since $\overline{s}(A\delta)(\lambda_x)C \to_\beta \overline{s}C[x := A]$ we have $(A\delta)\overline{s}(\lambda_x)C \mapsto_\beta \overline{s}C[x := A]$.

As for $\mapsto_\beta \subset \rightsquigarrow_\beta$, then note that if $A \mapsto_\beta B$ then for some $C =_\sigma A$ we have: $C \to_\beta B$. Then by Corollary 34, $C \in [A]$ and hence, as $B \in [B]$ we get $A \rightsquigarrow_\beta B$.

It is easy to show that these inclusions are strict:

- For $\rightsquigarrow_\beta \not\subseteq \mapsto_\beta$, take[6] $A_1 \equiv ((\lambda_y)(C\delta)(\lambda_z)y\delta)(\lambda_x)(B\delta)x$ where $x \notin FV(B)$ en $y \notin FV(C)$, and take $A_2 \equiv (C\delta)(\lambda_z)(B\delta)(\lambda_y)y$. Then, $A_1 \to_\beta (B\delta)(\lambda_y)(C\delta)(\lambda_z)y =_\sigma A_2$, but there is no term D such that $A_1 =_\sigma D \to_\beta A_2$. Hence, $A_1 \rightsquigarrow_\beta A_2$ but $A_1 \not\mapsto_\beta A_2$.
- For $\mapsto_\beta \not\subseteq \hookrightarrow_\beta$, construct for instance terms A_1 and A_2 with bracketing structures [[ ] [[ ]]] and [[ ] [ ]] respectively. E.g., if $A_1 \equiv (A\delta)(B\delta)(\lambda_x)(C\delta)(D\delta)(\lambda_y)(\lambda_z)(\lambda_t)E$ and $A_2 \equiv (C\delta)(B\delta)(\lambda_x)(D\delta)(\lambda_y)(\lambda_z)\{E[t := A]\}$ then $A_1 \mapsto_\beta A_2$ but $A_1 \not\hookrightarrow_\beta A_2$.
- For $\hookrightarrow_\beta \not\subseteq \to_g$, note that $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)C \hookrightarrow_\beta (A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)\{C[z := A_1]\}$ but $(A_1\delta)(A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)(\lambda_z)C \not\to_g (A_2\delta)(A_3\delta)(\lambda_x)(\lambda_y)\{C[z := A_1]\}$.
- For $\to_g \not\subseteq \to_\beta$, note that $(A\delta)(B\delta)(\lambda_x)(\lambda_y)C \to_g (B\delta)(\lambda_x)C[y := A]$ but $(A\delta)(B\delta)(\lambda_x)(\lambda_y)C \not\to_\beta (B\delta)(\lambda_x)\{C[y := A]\}$. $\Box$

Corollary 42 $\twoheadrightarrow_\beta \subset \twoheadrightarrow_g \subset \hookrightarrow\!\!\!\!\rightarrow_\beta \subset \mapsto\!\!\!\!\rightarrow_\beta \subset \rightsquigarrow\!\!\!\!\rightarrow_\beta$.

Remark 43 *It is not in general true that* $A \rightsquigarrow\!\!\!\!\rightarrow_\beta B \Rightarrow \exists A' \in [A]\exists B' \in [B]$ *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* $\not\twoheadrightarrow_\beta$*-reduce to* B.

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

Lemma 44 *If* $A \rightsquigarrow_\beta B$ *then* $A =_\beta B$.

Proof: Say $A' \in [A]$, $B' \in [B]$, $A' \to_\beta B'$. Now, by Lemma 25 and Proposition 31, $A =_\beta \theta(\gamma(A)) =_\beta \theta(\gamma(A')) =_\beta A' =_\beta B' =_\beta \theta(\gamma(B')) =_\beta \theta(\gamma(B)) =_\beta B$. $\Box$

Corollary 45 *1. If* $A \rightsquigarrow\!\!\!\!\rightarrow_\beta B$ *then* $A =_\beta B$. *2.* $A \approx_\beta B$ *iff* $A =_\beta B$.

Theorem 46 (Church Rosser theorem for $\rightsquigarrow_\beta$)
If $A \rightsquigarrow\!\!\!\!\rightarrow_\beta B$ *and* $A \rightsquigarrow\!\!\!\!\rightarrow_\beta C$*, then there exists* D *such that* $B \rightsquigarrow\!\!\!\!\rightarrow_\beta D$ *and* $C \rightsquigarrow\!\!\!\!\rightarrow_\beta D$.

Proof: As $A \rightsquigarrow\!\!\!\!\rightarrow_\beta B$ and $A \rightsquigarrow\!\!\!\!\rightarrow_\beta C$ then by Corollary 45, $A =_\beta B$ 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, $M \twoheadrightarrow_\beta A$ implies $M \rightsquigarrow\!\!\!\!\rightarrow_\beta A$. Hence we are done. $\Box$

[6]This counterexample is due to Rob Nederpelt.

Remark 47 *[9] gave similar properties for $\hookrightarrow\!\!\!\!\twoheadrightarrow_\beta$ where $A \sim_\beta B$ iff $A =_\beta B$ and $\hookrightarrow\!\!\!\!\twoheadrightarrow_\beta$ is CR.*

Now we are ready to establish the isomorphism of $\rightsquigarrow_\beta$-reduction paths of reductionally equivalent terms. The next lemma shows that reductional equivalence preserves $\rightsquigarrow_\beta$.

Lemma 48 *If $A \rightsquigarrow_\beta B$ then for all $A' \approx_{\mathtt{equi}} A$, for all $B' \approx_{\mathtt{equi}} B$, $A' \rightsquigarrow_\beta B'$.*

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 $A' \approx_{\mathtt{equi}} A$ and $B' \approx_{\mathtt{equi}} B$. 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 49 *$A \rightsquigarrow_\beta B$ iff $\theta(\gamma(A)) \rightsquigarrow_\beta \theta(\gamma(B))$.*

Remark 50 *Note that in the above lemma we cannot replace $\rightsquigarrow_\beta$ by $\to_\beta$:*

1. *Let A and B be in $\theta\delta$-normalform. Let $A_1 \equiv (A\delta)(B\delta)(\lambda_x)(\lambda_y)y$ and $A_2 \equiv (B\delta)(\lambda_x)A$. Note that $A_2 \equiv \theta(\gamma(A_2))$.*
   Now, $\theta(\gamma(A_1)) \equiv (B\delta)(\lambda_x)(A\delta)(\lambda_y)y \to_\beta (B\delta)(\lambda_x)A \equiv \theta(\gamma(A_2))$. But, $A_1 \not\to_\beta A_2$.
2. *For the other direction, take $A_1 \equiv ((\lambda_x)x\delta)(\lambda_y)(A\delta)(B\delta)y$ where $y \notin FV(A) \cup FV(B)$ and both A and B are in $\theta\gamma$-normalform, and take $A_2 \equiv (A\delta)(B\delta)(\lambda_x)x$. Then, $A_1 \to_\beta A_2$. Now, $\theta(\gamma(A_1)) \equiv A_1$, but $\theta(\gamma(A_2)) \equiv (B\delta)(\lambda_x)(A\delta)x$. Hence, $\theta(\gamma(A_1)) \not\to_\beta \theta(\gamma(A_2))$.*

Note also that the above lemma does not hold if we only replace the second $\rightsquigarrow_\beta$ by $\to_\beta$. In fact, the example used in 2 above can be used to show that $A_1 \rightsquigarrow_\beta A_2$ but $\theta(\gamma(A_1)) \not\to_\beta \theta(\gamma(A_2))$. Note however of course that if $\theta(\gamma(A_1)) \to_\beta \theta(\gamma(A_2))$ then definitely $A_1 \rightsquigarrow_\beta A_2$.

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

Remark 51 *Note that[7] $A \hookrightarrow_\beta B$ does not necessarily imply $\theta(\gamma(A)) \hookrightarrow\!\!\!\!\twoheadrightarrow_\beta \theta(\gamma(B))$, nor do we have that $A \to_\beta B$ implies $\theta(\gamma(A)) \twoheadrightarrow_\beta \theta(\gamma(B))$. 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 $\theta(\gamma(A)) \equiv A \not\hookrightarrow\!\!\!\!\twoheadrightarrow_\beta$ 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.

Theorem 52 *For all $A' \approx_{\mathtt{equi}} A$, for all $B' \approx_{\mathtt{equi}} B$, for all $E' \approx_{\mathtt{equi}} E$, if $A \rightsquigarrow_\beta^{(E\delta)(\lambda_x)} B$ then we have $A' \rightsquigarrow_\beta^{(E'\delta)(\lambda_x)} B'$.*

In other words, the following diagram commutes:

$$\begin{tikzcd}
A \arrow[r, rightsquigarrow, "(E\delta)(\lambda_x)"] \arrow[d, phantom, "\approx_{\mathtt{equi}}"] & {}_\beta\, B \arrow[d, phantom, "\approx_{\mathtt{equi}}"] \\
A' \arrow[r, rightsquigarrow, "(E'\delta)(\lambda_x)"] & {}_\beta\, B'
\end{tikzcd}$$

Proof: By definition of $\rightsquigarrow_\beta$, $A \rightsquigarrow_\beta^{(E\delta)(\lambda_x)} B$ implies there exist $A'' \approx_{\mathtt{equi}} A$, $B'' \approx_{\mathtt{equi}} B$ and $C'' \approx_{\mathtt{equi}} C$ such that $A'' \to_\beta^{(E''\delta)(\lambda_x)} B''$. But, as $\approx_{\mathtt{equi}}$ is an equivalence relation, $A'' \approx_{\mathtt{equi}} A'$, $B'' \approx_{\mathtt{equi}} B'$ and $E'' \approx_{\mathtt{equi}} E'$. Finally, by definition of $\rightsquigarrow_\beta$, $A' \rightsquigarrow_\beta^{(E'\delta)(\lambda_x)} B'$. □

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

Lemma 53 *If $A \in SN_{\to_\beta}$ and $A' \in [A]$ then $A' \in SN_{\to_\beta}$.*

[7]This counterexample is due to Rob Nederpelt.

Proof: If $A' \in [A]$ then $A' \approx_{\mathtt{equi}} A$. Hence, by Lemma 33, $A' =_\sigma A$. Now, we use a result of [20] 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'. $\Box$

Lemma 54 *If $A \in SN_{\leadsto_\beta}$ and $A' \in [A]$ then $A' \in SN_{\leadsto_\beta}$.*

Proof: $\forall B, A' \leadsto_\beta B$ implies $A \leadsto_\beta B$ by Lemma 48. Hence, A' is in $SN_{\leadsto_\beta}$. $\Box$

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

Lemma 55 *$A \in SN_{\leadsto_\beta}$ iff $A \in SN_{\to_\beta}$.*

Proof: As $\to_\beta \subset \leadsto_\beta$, $\Longrightarrow$ 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 53.

Suppose $A \leadsto_\beta A'$ and $A \in SN_{\to_\beta}$. It is sufficient to prove that $A' \in SN_{\leadsto_\beta}$. 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 54 it is sufficient to prove that $A'_1 \in SN_{\leadsto_\beta}$. By Lemma 53, $A_1 \in SN_{\to_\beta}$, 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: $A'_1 \in SN_{\leadsto_\beta}$. $\Box$

Now we show that semi reductional equivalence for SN terms implies reductional equivalence:

Lemma 56 *Let $A \in SN_{\leadsto_\beta}$. If $A' \approx_{\mathtt{equi}} A$ then $A' \sim_{\mathtt{equi}} A$.*

Proof: It is sufficient to show that:

1. $(B\delta)\overline{s}C$ is reductionally equivalent to $\overline{s}(B\delta)C$ if $\overline{s}$ is well-balanced and $(B\delta)\overline{s}C \in SN_{\leadsto_\beta}$.

2. $\overline{s}(\lambda_x)C$ is reductionally equivalent to $(\lambda_x)\overline{s}C$ if $\overline{s}$ is well-balanced and $\overline{s}(\lambda_x)C \in SN_{\leadsto_\beta}$.

We only prove 1. The proof is by induction on the maximal length of $\leadsto_\beta$-reduction paths of $(B\delta)\overline{s}C$.

If $(B\delta)\overline{s}C$ is in normal form then $\overline{s} \equiv \emptyset$ so $(B\delta)\overline{s}C \equiv \overline{s}(B\delta)C$. If $(B\delta)\overline{s}C$ is not in normalform then contraction of some redex yields a term which is either of the form $(B'\delta)\overline{s'}C'$ (if the redex was inside B, $\overline{s}$ or C) or of the form $\overline{s}C'$ if the redex consisted of $(B\delta)$ and its partnered item.

Then in the first case $\overline{s}(B\delta)C$ can reduce to $\overline{s'}(B'\delta)C'$ by contracting the corresponding redex, now by the induction hypothesis $(B'\delta)\overline{s'}C'$ is reductionally equivalent to $\overline{s'}(B'\delta)C'$. In the second case, $\overline{s}(B\delta)C$ also reduces to $\overline{s}C'$.

Hence $(B\delta)\overline{s}C$ is reductionally equivalent to $\overline{s}(B\delta)C$. $\Box$

Hence we have provided a relation between terms which approximates reductional equivalence. Here are some facts on this relation and on reductional equivalence:

Fact 57 *The following holds:*

1. *Let $A \in SN_{\leadsto_\beta}$. If $A \approx_{\mathtt{equi}} B$ then $A \sim_{\mathtt{equi}} B$ (Lemma 56).*

2. *$A \approx_{\mathtt{equi}} B$ does not imply $A \sim_{\mathtt{equi}} B$ (Example 58 below).*

3. *$A \sim_{\mathtt{equi}} B$ does not imply $A \approx_{\mathtt{equi}} B$ (Example 59 below).*

4. *$A \approx_{\mathtt{equi}} B$ is decidable (Proposition 31).*

5. *Let $A \in SN_{\leadsto_\beta}$. Then for all $A' \approx_{\mathtt{equi}} A$, $A' \in SN_{\leadsto_\beta}$ (Lemma 54).*

Example 58 *Let $\mathbf{\Omega} \equiv ((\lambda_z)(z\delta)z\delta)(\lambda_z)(z\delta)z$. Take A and B where $A \equiv (a\delta)(b\delta)(\lambda_x)(\lambda_y)\mathbf{\Omega}$ and $B \equiv (b\delta)(\lambda_x)(a\delta)(\lambda_y)\mathbf{\Omega}$. These terms read in classical notation $(\lambda_x.\lambda_y.\mathbf{\Omega})ba$ respectively $(\lambda_x.(\lambda_y.\mathbf{\Omega})a)b$. Now, $A \approx_{\mathtt{equi}} B$ but $A \not\sim_{\mathtt{equi}} B$. This example shows in 1. of Fact 57 that one cannot drop the assumption that A is strongly normalising.*

Example 59 *Let $A \equiv ((a\delta)(\lambda_x)x\delta)(\lambda_y)y$ and $B \equiv (a\delta)(\lambda_x)(x\delta)(\lambda_y)y$. $A \sim_{\mathtt{equi}} B$ but $A \not\approx_{\mathtt{equi}} B$. The same holds for the terms $(a\delta)(\lambda_y)(y\delta)y$ and $(a\delta)(\lambda_y)(y\delta)a$. This shows that the converse of 1. in Fact 57 does not hold.*

6 Conclusion

In this paper, we attempted to understand the reductional behaviour of calculations (or programs). We looked at two calculations and wanted to 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.

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

Another attractive feature of our work is that we managed to give a clear representation of the canonical forms of terms given in [20] which transparently shows where redexes occur and where they do not. Table 1 shows that every λ-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 β-reduction. Furthermore, although canonical forms are not unique, we can still find for each λ-term, the unique class of its canonical forms which are all equal modulo some simple permutation.

References

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

[2] H.P. Barendregt. λ-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] Olivier Danvy and Lasse R. Nielsen. CPS transformation of beta-redexes. In Amr Sabry, editor, *Proceedings of the Third ACM SIGPLAN Workshop on Continuations*, Technical report 545, Computer Science Department, Indiana University, pages 35–39, London, England, January 2001. Also available as the technical report BRICS RS-00-35.

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

[5] F. Kamareddine. Postponement, conservation and preservation of strong normalisation for generalised reduction. *Logic and Computation*, 10(5):721–738, 2000.

[6] F. Kamareddine and R. Bloo. De Bruijn's syntax and reductional equivalence of lambda terms: the typed case. *Logic and Algebraic Programming*, 2004.

[7] F. Kamareddine, R. Bloo, and R. P. Nederpelt. De Bruijn's syntax and reductional equivalence of lambda terms. In *Proc. 3rd Int'l Conf. Principles & Practice Declarative Programming*, pages 16–27, 5–7 September 2001.

[8] F. Kamareddine and R. Nederpelt. A useful λ-notation. *Theoretical Computer Science*, 155:85–109, 1996.

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

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

[11] M. Karr. Delayability in proofs of strong normalizability in the typed λ-calculus. In *Mathematical Foundations of Computer Software, LNCS*, volume 185. Springer-Verlag, 1985.

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

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

[14] A.J. Kfoury and J.B. Wells. Addendum to new notions of reduction and non-semantic proofs of β-strong normalisation in typed λ-calculi. Technical report, Boston University, 1995.

[15] A.J. Kfoury and J.B. Wells. New notions of reductions and non-semantic proofs of β-strong normalisation in typed λ-calculi. *LICS*, 1995.

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

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

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

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

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

[21] 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.

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

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

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