Postponement, Conservation and Preservation of Strong Normalisation for Generalised Reduction

Fairouz Kamareddine*

Abstract

Postponement of $\beta_K$-contractions and the conservation theorem do not hold for ordinary $\beta$ but have been established by de Groote for a mixture of $\beta$ with another reduction relation. In this paper, de Groote’s results are generalised for a single reduction relation $\beta$, which generalises $\beta$. We show moreover, that $\beta$ has the Preservation of Strong Normalisation property.

Keywords: Generalised $\beta$-reduction, Postponement of $K$-contractions, Generalised Conservation, Preservation of Strong Normalisation.

1 The $\lambda$-calculus with generalized reduction

In the term $((\lambda_x . \lambda_y . N)P)Q$, the abstraction starting with $\lambda_x$ and the argument $P$ form the redex $(\lambda_x . \lambda_y . N)P$. When this redex is contracted, the abstraction starting with $\lambda_y$ and $Q$ will in turn form a redex. It is important to note that $Q$ (or some residual of $Q$) is the only argument that the abstraction (or some residual of the abstraction) starting with $\lambda_y$ can ever have. This fact has been exploited by many researchers. Reduction has been extended so that the implicit redex based on the matching $\lambda_y$ and $Q$ is given the same priority as the intervening redex.

An initial attempt to generalize the notion of redex might be to define a rule like the following:

$$(\lambda_x . \lambda_y . N)PQ \rightarrow (\lambda_x . N[\gamma = Q])P$$

It quickly becomes evident that this is not sufficient as the following example shows:

Example 1 The proposed rule does not allow directly reducing the binding of $y$ to $Q$ in the term $A \equiv (\lambda_z . (\lambda_x . \lambda_y . N)P)RQ$.

We shall exploit the notion of a well balanced segment (sometimes known as a $\beta$-chain), which is the special case of one-hole contexts given by this grammar:

$$S ::= [\bullet] \mid (S[\lambda_x . [\bullet]])M \mid S[S]$$

*Department of Computing and Electrical Engineering, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, Scotland, email: fairouz@coe.hw.ac.uk, fax + 44 131 451 3327.
Using balanced segments, **generalized reduction** is then given by this rule:

\[ S[\lambda_x.M]N \rightarrow S[M[x:=N]] \]

We find the above definition of well-balanced segments and generalised reduction rather cumbersome and believe that a more elegant definition can be given. In order to do so, we change from the classical notation to the **item notation**. Instead of writing \( \lambda_x.M \), we write \([x]M\) and instead of \(MN\) we write \((N)M\). Item notation has many advantages as shown in [7, 8]. Let us illustrate here with term \(A\) of Example 1, which we write in item notation as in Figure 1. We see immediately that the redices

![Figure 1: Redices in item notation in term A](image)

originate from the couples \((Q)[y] \), \((R)[z]\) and \((P)[x]\). Moreover, \((Q)(R)[z](P)[x][y]\) is a well-balanced segment. This natural matching was not present in the classical notation. We call items of the form \((P)\) and \([x]\), application and abstraction items respectively. With item notation, generalised reduction is written as:

\[(M)\bar{\pi}[x]N \rightarrow_{g_{\beta}} \bar{\pi}[N[x:=M]]\] for \(\bar{\pi}\) well-balanced.

(Here, \{ and \} are used for grouping purposes so that no confusion arises.) For example:

\[(Q)(R)[z](P)[x][y]N \rightarrow_{g_{\beta}} (R)[z](P)[x][y][N[y:=Q]]\]

Surely this is clearer than writing \((\lambda_z.(\lambda_x.N)P)RQ \rightarrow_{g_{\beta}} (\lambda_z.(\lambda_x.N[y:=Q]P)R\).

## 2 An overview of generalised reduction in the literature

Generalized reduction was first introduced by Nederpelt in 1973 to aid in proving the strong normalization of AUTOMATH [19]. Kamareddine and Nederpelt have shown how generalised reduction makes more redices visible, allowing flexibility in reducing a term [7]. Bloor, Kamareddine, and Nederpelt show that with generalised reduction one may indeed avoid size explosion without the cost of a longer reduction path and that simultaneously the \(\lambda\)-calculus can be elegantly extended with definitions which result in shorter type derivations [5]. Generalised reduction is strongly normalising [5] for all systems of the \(\lambda\)-cube [3].

An alternative approach to generalised reduction which has been followed by many researchers is to use one of these two local transformations:

\[ \begin{align*}
\theta & \quad (\lambda_x.N)PQ \rightarrow (\lambda_x.NQ)P \\
\gamma & \quad (\lambda_x.\lambda_y.N)P \rightarrow \lambda_y.(\lambda_x.N)P
\end{align*} \]

1Note that putting the argument before the function was first introduced by de Bruijn in his Automath project [20] and has been used by many researchers since. For example, Krivine in [17] also puts the argument before the function.
These rules transform terms to make more redexes visible to the ordinary notion of β-reduction. For example, both the γ and θ rules make sure that λy and Q in the term A of Example 1 can form a redex before the redex based on λx and P is contracted. That is:

\[(\theta_C) \quad (\lambda_x. (\lambda_y. N)PQ \rightarrow (\lambda_x. (\lambda_y. N)Q)P)\]

\[(\gamma_C) \quad (\lambda_x. (\lambda_y. N)PQ \rightarrow (\lambda_y. (\lambda_x. N)P)Q)\]

Hence both θ and γ put λy next to its matching argument. The θ rule moves the argument next to its matching λ whereas γ moves the λ next to its matching argument.

Obviously, θ and γ are related to generalised reduction. In fact, θ and γ transform terms in order to make more potential redexes visible and then conventional β-reduction can be used to contract those newly visible redexes. Generalised reduction on the other hand, performs reduction on the potential redexes without having to bother to make them into classical redexes. The following example illustrates:

**Example 2** Take again the term \( A \equiv (\lambda_z. (\lambda_x. \lambda_y. N)P)RQ \). With generalised reduction we get: \( (\lambda_z. (\lambda_x. \lambda_y. N)P)RQ \rightarrow_{\theta} (\lambda_z. (\lambda_x. \lambda_y. N)[y := Q])P \). We illustrate how θ and γ work:

\[\theta \quad (\lambda_z. (\lambda_x. \lambda_y. N)P)RQ \rightarrow_{\theta} (\lambda_z. (\lambda_x. \lambda_y. N)PQ)R \rightarrow_{\theta} (\lambda_z. (\lambda_x. N)[y := Q])P)R \]

\[\gamma \quad (\lambda_z. (\lambda_x. \lambda_y. N)P)RQ \rightarrow_{\gamma} (\lambda_z. \lambda_y. (\lambda_x. N)P)RQ \rightarrow_{\gamma} (\lambda_y. (\lambda_z. (\lambda_x. N)P)RQ \rightarrow_{\beta} (\lambda_z. (\lambda_x. N)[y := Q])P)R \]

Note that in item notation it is easier to describe θ and γ. We illustrate with θ.

**Example 3** We can reshuffle \((Q)(R)[\bar{z}][\bar{P}][\bar{x}][\bar{y}]N\) to \((R)[\bar{z}][\bar{P}][\bar{x}][\bar{y}]N\) in order to transform the bracketing structure \{ \} \{ \} \{ \} \{ \} into \{ \} \{ \} \{ \} \{ \}, where all the redexes correspond to adjacent ‘{’ and ‘}’. Figure 1 can be redrawn using the θ-reduction twice in Figure 2.

\[
\begin{array}{c}
\text{(R)} \\
\text{[z]} \\
\text{(P)} \\
\text{[x]} \\
\text{(Q)} \\
\text{[y]} \\
\text{N}
\end{array}
\]

**Figure 2:** θ-normal forms in item notation for term A

The θ rule can be applied to both explicitly and implicitly typed systems. However, the transfer of γ to explicitly typed systems is not straightforward, since in these systems the type of \( y \) in the term \( A \) may be affected by the reducible pair of \( \lambda_x \) and \( P \). For example, it is fine to write \(((\lambda_x. \lambda_y. x. y)z)u \rightarrow_{\gamma} (\lambda_y. (\lambda_x. x. y)u)z\) but not to write \(((\lambda_x. \lambda_j. x. y)z)u \rightarrow_{\gamma} (\lambda_y. (\lambda_x. y)z)u\).

Local transformations like γ and θ began to appear in the literature during the eighties. (See [15] for a summary). Regnier [21] introduces the notion of a premier redex which is similar to the redex based on \( \lambda_y \) and \( Q \) above (which we call a generalised redex). Later, he uses θ and γ (and calls the combination σ) to show that

\footnote{An alternative is to apply γ to the type erase of the term, which may be quite complicated to express in terms of the type-annotated term.}
the perpetual reduction strategy finds the longest reduction path when the term is
Strongly Normalising (SN) [22]. Vidal also introduces similar reductions [25]. Kfoury,
Tiuryn, and Urzyczyn use θ (and other reductions) to show that typability in ML is
equivalent to acyclic semi-unification [12]. Sabry and Felleisen describe a relationship
between a reduction similar to θ and a particular style of CPS [23]. De Groote [6]
uses θ and Kfoury and Wells [14] use γ to reduce the problem of β-strong normali-
sation to the problem of weak normalisation (WN) for related reductions. Kfoury
and Wells use θ and γ to reduce typability in the rank-2 restriction of system F to
the problem of acyclic semi-unification [13]. Klop, Sørensen, and Xi [16, 26, 24] use
related reductions to reduce SN to WN. Finally, Ariola, Felleisen, Maraist, Odetsky
and Wadler use θ (called “let-C”) in [1], as a part of an analysis of how to represent
sharing in a call-by-need language implementation in a formal calculus.

All the research mentioned above is a living proof for the importance and usefulness
of generalised reduction (from now on, βe). For this reason, properties of this
reduction must be studied. Confluence of βe is a direct consequence of the fact that
Μ =β N ⇔ Μ =βe N. Subject reduction for βe has been established in [5] (with the
condition that explicit definitions must be added for some systems of the cube). And,
as we mentioned earlier, Strong Normalisation of βe has been established for the whole
Cube and type derivation paths have been analysed. Other important properties of
βe have however remained unanswered. These properties are:

1. PreserVation of Strong Normalisation PSN. This property is: if M is strongly
   normalising for ordinary β-reduction (written M is β-SN), then M remains
   strongly normalising for generalised reduction βe (i.e. M is also βe-SN). PSN
   makes βe a useful extension of β. This parallels the work on extending λ-calculi
   with explicit substitutions which satisfy the PSN property.

2. Conservation of βe-reduction. This property is: if a term is βe-I-normalisable
   (i.e. βe-normalisable reducing only redexes that don’t erase their arguments,
   so called I-redexes, or strict redexes), then it is strongly normalisable. This is
   interesting in view of the ongoing interest of showing that strong normalisation
can be reduced to weak normalisation [16, 24, 26].

3. Postponement of K-reduction. Generalised reduction allows the postponement
   of K-reduction (which discards their arguments) after I-reductions (which use
   their arguments in at least one place). Hence, generalised reduction allows
   unnecessary K-redexes to be bypassed. From the implementation point of view,
   this results in flexibility in work. Unnecessary work can be delayed or even
   avoided completely.

In this paper, we show these three properties for the generalised reduction βe. We
use item notation to be able to write generalised reduction in a really general way
and to be able to describe proofs and proof objects elegantly. We believe that if this
paper was written in classical notation, then the proofs would have been cumbersome
to present.
3 Contributions of this paper and related work

Because we still have not introduced all the machinery of item notation, we shall use classical notation in this section.

Let us recall the three basic reduction rules of the \( \lambda \)-calculus \((FV(M) \) stands for the free variables of \( M \)):

\[
\begin{align*}
(\beta) & \quad (\lambda_x.M)N \rightarrow M[x := N] \\
(\beta_I) & \quad (\lambda_x.M)N \rightarrow M[x := N] \quad \text{if } x \notin FV(M) \\
(\beta_K) & \quad (\lambda_x.M)N \rightarrow M \quad \text{if } x \notin FV(M)
\end{align*}
\]

Redexes based on the \( \beta_I \) rule are called \( \beta_I \)- or \( I \)-redexes. Similarly, those based on the \( \beta_K \) rule are called \( \beta_K \)- or \( K \)-redexes. For any relation \( r \), we write \( r_K \) and \( r_I \) for the corresponding \( K \)- and \( I \)-reductions.

In this paper, we show that the generalised reduction \( \beta \) satisfies PSN, the postponement of \( K \)-contractions and conservation. Of course the latter two properties fail for ordinary \( \beta \) as shown by the following example:

**Example 4** \((\lambda_y.(\lambda_x.x))MN \rightarrow_{\beta_K} (\lambda_x.x)N \rightarrow_{\beta_I} N \) and it is impossible to \( \beta_I \)-reduce \((\lambda_y.(\lambda_x.x))MN \). Moreover, \((\lambda_x.(\lambda_y.y(\lambda_z.zz))u)\lambda_z.zz \) is \( \beta_I \)-normalising but not strongly \( \beta \)-normalising.

Attempts have been made at establishing some reduction relations for which postponement of \( K \)-contractions and conservation hold ([2] and [6]). The picture is as follows (-N stands for normalising and \( r \in \{ \beta_I, \theta_K \} \) where \( \theta \) was defined earlier):

\[
\begin{align*}
(\beta_K \text{-postponement for } r) & \quad \text{If } M \rightarrow_{\beta_K} N \rightarrow_r O \text{ then } \exists P \text{ such that } M \rightarrow_{\beta_I, \theta_K} P \rightarrow_{\beta_K} O \\
(\text{Conservation for } \beta_I) & \quad \text{If } M \text{ is } \beta_I \text{-N then } M \text{ is } \beta_I \text{-SN} \\
(\text{Conservation for } \beta + \theta) & \quad \text{If } M \text{ is } \beta_I, \theta_K \text{-N then } M \text{ is } \beta \text{-SN}
\end{align*}
\]

Conservation for \( \beta_I \) is found in [2]. Conservation for \( \beta + \theta \) and \( \beta_K \)-postponement for \( r \in \{ \beta_I, \theta_K \} \) are established by de Groot in [6]. However, de Groot does not produce these results for a single reduction relation, but for \( \beta \) in which another relation \( \theta \) is used. This paper establishes \( \beta_K \)-postponement and conservation for a single relation \( \beta \), and is hence the first to do so. Moreover, the relation \( \theta \) is more restrictive than the generalised reduction of this paper.

Let us now list the postponement and conservation properties for \( \beta_K \):

\[
\begin{align*}
(\beta_K \text{-postponement for } \beta_k) & \quad \text{If } M \rightarrow_{\beta_k} N \rightarrow_{\beta_\mathbf{SN}} O \text{ then } \exists P \text{ such that } M \rightarrow_{\beta_I, \theta_K} P \rightarrow_{\beta_K} O \\
(\text{Conservation for } \beta_k) & \quad \text{If } M \text{ is } \beta_k \text{-N then } M \text{ is } \beta_k \text{-SN}
\end{align*}
\]

These two properties are important because here we have the first reduction relation which generalises \( \beta \) (yet \( M =_\beta N \iff M =_{\beta_k} N \)) and which satisfies them.

Now we come to the PSN property which is as follows:

\[
(\text{PSN for } \beta_k) \quad M \text{ is } \beta \text{-SN } \iff M \text{ is } \beta_k \text{-SN}.
\]

PSN not only means that \( \beta_k \) does not change the set of \( \beta \)-SN terms, but also that we can actually use \( \beta_k \) with explicit substitution. In fact, explicit substitution is an important topic of research and PSN is an important property for any \( \lambda \)-calculus.
extended with explicit substitution. In fact, lately, much research has been carried out ([4, 9]) in order to find systems of explicit substitution which are both confluent and have the PSN property (if \( M \) is \( \beta \)-SN then \( M \) is \( \lambda_e \)-SN where \( \lambda_e \) is the lambda calculus extended with explicit substitution). This is the reason for our interest in PSN of \( \beta_e \) (which is confluent by the way). After all, generalised reductions à la \( \beta_e \) have been extensively used as we saw in Section 2 for both theoretical and practical reasons. Furthermore, systems of explicit substitution have been the subject of much recent research. Both generalised reduction and explicit substitution are of practical importance and combining them both in one system may turn out to be very useful. The main benefits of these concepts are similar: both emphasize flexibility in the ordering of operations. In particular, both generalised reduction and explicit substitution allow the postponement of work, but in different, complementary ways. On one side, generalised reduction always allows unnecessary \( K \)-redexes to be bypassed. Explicit substitution will not in general allow this, since reducing the \( K \)-redex might be necessary to expose an essential \( I \)-redex. Similarly, on the other side, explicit substitution allows bypassing any work inside a subterm that will be discarded later. However, generalised reduction does not provide any means for performing only those parts of a substitution that will be used later. Thus, we can see that their benefits are complementary.

We claim that a system with the combination of generalised reduction and explicit substitution is more advantageous than a system with each concept separately. Obviously, if the benefits of both are desired simultaneously, it is important to study the combination, a task which this paper performs. Before the combination can be safely used, it must be checked that this combination is sound and safe exactly like it has been checked that each of explicit substitutions and generalised reductions separately are sound and safe.

Once PSN is established we can study extending the \( \lambda \)-calculus with both explicit substitution and generalised reduction. This means that we can combine the advantages of the two different extensions in one system [10, 11].

We had established in [9] property (1) below, and in [10, 11] property (2) below (\( \lambda_{\beta_e} \) stands for the lambda calculus extended with explicit substitution and generalised reduction and for reasons of uniformity, we write \( \lambda \)-SN for \( \beta \)-SN and \( \lambda_{\beta_e} \)-SN for \( \beta_{e,SN} \):

(1) \quad M \text{ is } \lambda \text{-SN} \iff M \text{ is } \lambda_{\beta_e} \text{-SN}

(2) \quad M \text{ is } \lambda_{\beta_e} \text{-SN} \iff M \text{ is } \lambda_{\beta_{e,SN}} \text{-SN}

The proofs for (1) and (2) are similar. Now with PSN, we get (3) below and then (4) comes for free.

(3) \quad M \text{ is } \lambda \text{-SN} \iff M \text{ is } \lambda_{\beta_e} \text{-SN}

(4) \quad M \text{ is } \lambda_{\beta_e} \text{-SN} \iff M \text{ is } \lambda_{\beta_{e,SN}} \text{-SN}

Hence, one gets: \( M \text{ is } \lambda \text{-SN} \iff M \text{ is } \lambda_{\beta_e} \text{-SN} \iff M \text{ is } \lambda_{\beta_{e,SN}} \iff M \text{ is } \lambda_{\bar{e}} \text{-SN} \).

Based on the above discussion, this article shows \( \beta_{eK} \) postponement (Section 5), the generalised conservation for \( \beta_e \) (Section 6), and the PSN property for \( \beta_e \) (Section 7).
4 The formal machinery

We assume the reader familiar with the λ-calculus whose terms are

\[ \Lambda := \mathcal{V}[(\Lambda \Lambda)(\Lambda \mathcal{V}, \Lambda)] \]

We take terms modulo α-conversion and use the variable convention VC (as in [3]) which avoids any clash of variables. We use \( x, y, z, x_1, x_2, \ldots \) and \( M, N, P, Q, A, B, A_1, \ldots \) to range over \( \mathcal{V} \) and \( \Lambda \) respectively. We assume the usual definition of substitution and use \( FV(M) \) for the set of free variables of \( M \). Because we need to see redexes (ordinary and generalised) we shall write terms in item notation (see [8] or [7]). In this notation, \( \lambda x \) is written as \([x]\) and \((MN)\) is written \((N)M\) (note that following de Bruijn, we put the argument before the function). \([x]\) and \((N)\) are called items. A sequence of items is called a segment. We use \( I, I_1, \ldots \) to range over items and \( S, S_1, S_2, \ldots \) to range over segments. A well-balanced segment (w.b for short) is defined as the empty segment or \((P)_S[x]_S_2\) where \( S_1 \) and \( S_2 \) are w.b. Note that the concatenation of w.b segments is a well-balanced segment.

One particular advantage of this notation is that redexes are more clear than in the usual notation. For example, \( \gamma_C \) of Section 2 becomes:

\[(\gamma_C) \quad (Q)(P)[x][y]N \rightarrow (Q)[y](P)[x]N\]

where it is clear that \((P)\) matches \([x]\) and \((Q)\) matches \([y]\). So, an ordinary redex starts with a \( ( ) \) adjacent to \( [] \). A generalised redex starts with \( ( )S[] \) where \( S \) is w.b. When \( S = \emptyset \), a generalised redex is an ordinary redex. In \((Q)(P)[x][y]N\), we say that \((P), [x], (Q)\) and \([y]\) are partnered, \((P)\) is the partner of \([x]\) (or \([x]\) is the partner of \((P))\) and \((Q)\) is the partner of \([y]\). \((P)\) and \([x]\) are also said to be \( \beta \)-partnered whereas \((Q)\) and \([y]\) are \( \beta \)-partnered. In general, we say that \((P)\) (or \([x]\)) is partnered in \( M \) if:

- \( M \equiv (P)S[x]N \) where \( S \) is w.b (in this case \((P)\) and \([x]\) are partners), or
- \( M \equiv [y]N \) and \((P)\) (or \([x]\)) is partnered in \( N \), or
- \( M \equiv (N_1)N_2 \) and \((P)\) is either partnered in \( N_1 \) or in \( N_2 \).

We may also talk of \( \beta_\gamma, \beta, \beta_\gamma, \beta_\gamma, \beta_\gamma \)-partnered items with the obvious meaning. Note that if \( S_1(A)S_2[x]S_3 \) is w.b where \((A)\) and \([x]\) are partnered then \( S_2 \) and \( S_1S_3 \) are w.b.

If an item is not partnered in a term we say that it is bachelor (and may talk of \( \beta, \beta_\gamma, \beta_\gamma, \beta_\gamma, \beta_\gamma \)-bachelor items). A segment consisting of bachelor items only is called bachelor. Note that a term will always be written as \( I_1I_2 \ldots I_nx \). Each \( I_i \) is said to be a main-item in \( M \). A main item can of course have items inside it but these will not be main in \( M \). For example, \([(y)[x]x][z]z \) has the main items \([(y)[x]x]\) and \([z]\). The redex \([(y)[x][z]z \) is said to be a main-redex. The other redex \((y)[x]x \) is not main. The weight of a segment is defined to be the number of its main items. We write \([x := N]M \) instead of \( M[x := N] \) which stands for substituting \( N \) for the free occurrences of \( x \) in \( M \).
We assume the reader familiar with the basic machinery of reduction ([2], p. 50-59). In particular, if $R$ is a binary relation $\subset \Lambda \times \Lambda$, and $(M,N) \in R$, we call $M$ the $R$-redex and $N$ the contractum of $M$. Given $R \subset \Lambda \times \Lambda$, we define $\rightarrow_R$ to be the least compatible relation containing $R$, $\rightarrow_R$ to be its reflexive transitive closure and $=R$ to be its reflexive, symmetric and transitive closure. A term $M$ is said to be in $R$-normal form ($R$-nf) if there is no $N$ such that $M \rightarrow_R N$. $M$ is said to have a $R$-nf, if there is $N$ in $R$-nf such that $M \rightarrow_R N$. We say $M$ is $R$-normalising or is $R$-N if $M$ has a $R$-nf. We say that $M$ is strongly $R$-normalising and write $M$ is $R$-SN if there is no infinite $R$-reduction path starting at $M$. We may use $M \rightarrow_R^* N$ to indicate the existence of one or more steps from $M$ to $N$ and $M \rightarrow_R^* N$ to mean that there are $n$ reduction steps. Ordinary $\beta$, $\beta_I$ and $\beta_K$-reduction are defined as the reduction relations generated by the corresponding rules below:

\[
\begin{align*}
(\beta) & \quad (N)[x]M \rightarrow [x := N]\!M \\
(\beta_I) & \quad (N)[x]M \rightarrow [x := N]\!M \quad \text{if } x \not\in \text{FV}(M) \\
(\beta_K) & \quad (N)[x]M \rightarrow M \quad \text{if } x \not\in \text{FV}(M)
\end{align*}
\]

As explained in Example 4, postponement of $K$-contractions and conservation do not hold for $\beta$. De Groote in [6] introduces different reduction relations for which he establishes these properties. First, [6] uses

\[
\theta_K (O)(N)[x]M \rightarrow (N)[x](O)M \quad \text{if } x \not\in \text{FV}(M)
\]

Note that by VC, in $\theta_K$, $x \not\in \text{FV}(O)$. Then, de Groote moves $(O)$ to the right of $(N)[x]$ so that it can eventually occur adjacent to its partner in $M$ if it exists. De Groote establishes the following two results ($r \in \{\beta_I, \theta_K\}$):

\[
\begin{align*}
(\beta_K\text{-postponement for } \beta_I) & \quad \text{If } M \rightarrow_{\beta_I} N \rightarrow_r O \text{ then } \exists P \text{ such that } M \rightarrow_{\beta_I}^* P \rightarrow_{\beta_K} O \\
(\text{Conservation for } \beta + \theta) & \quad \text{If } M \text{ is } \beta_I\theta_K\text{-N then } M \text{ is } \beta\text{-SN}.
\end{align*}
\]

In this paper, we will improve both results. We will define a $\beta$-reduction relation (see Definition 5) whose $\beta_I$ and $\beta_{K}$ stand for its $I$ and $K$-reductions. We shall show that:

\[
\begin{align*}
(\beta_K\text{-postponement for } \beta_I) & \quad \text{If } M \rightarrow_{\beta_K} N \rightarrow_{\beta_I} O \text{ then } \exists P \text{ such that } M \rightarrow_{\beta_I}^* P \rightarrow_{\beta_K}^* O \\
(\text{Conservation for } \beta_I) & \quad \text{If } M \text{ is } \beta_I\text{-N then } M \text{ is } \beta_I\text{-SN}.
\end{align*}
\]

**Definition 5** (Generalised $\beta$-reduction $\beta_e$) We generalise $\beta$, $\beta_I$ and $\beta_K$ to the reduction relations generated by the corresponding rules of what follows:

\[
\begin{align*}
(\beta_e) & \quad (N)[x]S[x]M \rightarrow S[x := N]\!M \quad \text{if } S \text{ is w.b} \\
(\beta_{eI}) & \quad (N)[x]S[x]M \rightarrow S[x := N]\!M \quad \text{if } S \text{ is w.b and } x \in \text{FV}(M) \\
(\beta_{eK}) & \quad (N)[x]S[x]M \rightarrow SM \quad \text{if } S \text{ is w.b and } x \not\in \text{FV}(M)
\end{align*}
\]

Note that $\beta_e$ is more generalised than the reduction relation introduced by combing de Groote’s $\beta + \theta_K$. In fact, $\beta_e$ is not restricted to $K$-redexes and one unique step can do the work of many in Groote’s sense. For example, if $S = (A_1)[x_1][A_2][x_2] \cdots (A_n)[x_n]$ and all the redexes starting with $(A_1), (A_2), \ldots (A_n)$ are $K$-redexes in $S[x]M$, then $(N)[x]S[x]M \rightarrow_{\beta_e} S[x := N]\!M$ iff $(N)[x]S[x]M \rightarrow_{\beta_K}^* S(N)[x]M \rightarrow_{\beta} S[x := N]\!M$.

Now, here is a basic lemma about terms:
Lemma 6

1. Let \( r \in \{ \beta_e, \beta_{eI}, \beta_{eK} \} \). If \((A)\) is \( r \)-bachelor in \((A)M\) then \((B)\) is also \( r \)-bachelor in \((B)(A)M\).

2. If \( M \) is in \( \beta \)-nf, then \( M \equiv [x_1][x_2] \ldots [x_n](A_1)(A_2) \ldots (A_m)z \) where \( n \geq 0, m \geq 0 \) and \( \forall i, 1 \leq i \leq m \Rightarrow A_i \) is in \( \beta \)-nf.

3. If \( A \rightarrow^r A' \) then \( SA \rightarrow^r SA' \) for any segment \( S \) and any reduction relation \( r \in \{ \beta, \beta_I, \beta_K, \beta_e, \beta_{eI}, \beta_{eK} \} \).

Proof

1. If \( (B) \) was \( r \)-partnered, then \( (B)(A)M \equiv (B)(A)S[x]N \) where \( (A)S \) is w.b (and hence \( (A)S \equiv (A)(S_1[y]S_2 \) where \( S_1, S_2 \) are w.b) contradicting the fact that \( (A) \) is \( r \)-bachelor.

2. By induction on the structure of \( M \).

3. By induction on the weight of \( S \).

In order to show the Preservation of Strong Normalisation for \( \beta_e \), we need a reduction strategy where a \( \beta_K \)-redex \([M][x]N\) is contracted only if \( M \) is in \( \beta \)-nf. This strategy is actually the perpetual strategy (see [2] and [22]):

**Definition 7** We define the perpetual strategy \( F \) as follows:

\[
\begin{align*}
F([x]M) &= F(M) \\
F((M)N) &= F(N) & \text{if } N \not\equiv [x]P \text{ and } N \text{ is not in } \beta \text{-nf} \\
F((M)N) &= F(M) & \text{if } N \not\equiv [x]P \text{ and } N \text{ is in } \beta \text{-nf} \\
F((M)[x]N) &= (M)[x]N & \text{if } x \in FV(N) \text{ or } M \text{ is in } \beta \text{-nf} \\
F((M)[x]N) &= F(M) & \text{if } x \not\in FV(N) \text{ and } M \text{ is not in } \beta \text{-nf}
\end{align*}
\]

We call perpetual reduction the reduction associated with this strategy. When \( M \) \( \beta \)-reduces to \( N \) by contracting \( F(M) \), we write, \( M \rightarrow^F N \). This strategy has been shown in [22] to give the longest path for a SN term. It was moreover, shown in [2] that \( M \) is \( \beta \)-SN iff its perpetual reduction terminates. With the result of this paper, it will also be the case that \( M \) is \( \beta_e \)-SN iff its perpetual path terminates.

The following lemma is informative about where \( F \)-reduction takes place in a term in the case of \( K \)-redexes:

**Lemma 8** If \( M \rightarrow^F N \) where \( F(M) \) is a \( \beta_K \)-redex, then one of the following holds:

1. \( M \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A)[x]P \) and

\[
N \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)P
\]

where \( x \not\in FV(P) \), \( A \) is in \( \beta \)-nf, \( n \geq 0 \) and \( m \geq 0 \).

2. \( M \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A)[x]P \) and

\[
N \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A')[x]P
\]

where \( x \not\in FV(P) \), \( A \) is not in \( \beta \)-nf, \( A \rightarrow^F A' \), \( n \geq 0 \) and \( m \geq 0 \).
3. \( M \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A)(B_1)(B_2) \ldots (B_r)z \) and 
\( N \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A')(B_1)(B_2) \ldots (B_r)z \) and 
\( A \) is not in \( \beta\)-nf, \( A \rightarrow F A' \), \( n \geq 0 \), \( m \geq 0 \) and \( r \geq 0 \) and \( \forall i, 1 \leq i \leq r, B_i \) is in \( \beta\)-nf.

**Proof** By induction on \( M \rightarrow_F N \) where \( F(M) \) is a \( \beta_K \)-redex.

5 Postponement of \( \beta_{e,K} \)-reduction

In this section, we establish in lemma 10 the postponement of \( \beta_{e,K} \)-reduction. The proof of postponement is similar to that of de Groote. For us, however, we can get away with only one step \( \beta_{et} \) reduction in the postponement lemma (Lemma 10). De Groote, had to have many steps in order to accommodate the slow process of moving an item ( ) next to its matching [ ] (see for example his proof of Lemma 11, (c) ii). We could also in Lemma 10, replace \( \beta_{e,K} \) with ordinary \( \beta_K \) in \( P \rightarrow^+ O \) but we won’t bother doing so in this paper as it is not needed.

The following lemma establishes that substitution preserves \( \beta_{e,K} \)-reduction.

**Lemma 9** If \( M \rightarrow_{\beta_{e,K}} N \) then the following hold:

1. \( [x := M]P \rightarrow_{\beta_{e,K}} [x := N]P \).
2. If \( x \in FV (P) \) then \( [x := M]P \rightarrow_{\beta_{e,K}} [x := N]P \).
3. \( [x := P]M \rightarrow_{\beta_{e,K}} [x := P]N \).

**Proof** Use induction on the structure of \( P \) for 1 and 2, and on the derivation of \( M \rightarrow_{\beta_{e,K}} N \) for 3.

Now we come to the postponement lemma. Note that in this lemma, \( P \rightarrow^+ O \) and not \( P \rightarrow_{\beta_{e,K}} O \) nor \( P \rightarrow_{\beta_{et}} O \). This is due to Lemma 9.

**Lemma 10** If \( M \rightarrow_{\beta_{e,K}} N \rightarrow_{\beta_{et}} O \) then \( \exists P \) such that \( M \rightarrow_{\beta_{et}} P \rightarrow^+_{\beta_{e,K}} O \).

**Proof** By induction on the derivation of \( M \rightarrow_{\beta_{e,K}} N \).

- Case \( (A)S[x]B \rightarrow_{\beta_{e,K}} SB \), \( S \) w. b, \( x \not\in FV (B) \), check where in \( SB \) the \( \beta_{et} \)-redex appears (note that if \( S_1, S_2, S_3 \) and \( S_4 \) are w. b, then \( S_1, S_3 \) is w. b).

  We only treat the case where \( S \equiv S_1(A_1)S_2[y]S_3 \) with \( S_2 \) w. b and 
  \( S_1(A_1)S_2[y]S_3 \rightarrow_{\beta_{et}} S_1S_2 \{[y := A_1]S_3 \} \{y := A_1\}B \).

  Then \( (A)S_1(A_1)S_2[y]S_3[x]B \rightarrow_{\beta_{et}} (A)S_1S_2 \{[y := A_1]S_3 \} \{x := A_1\} \rightarrow_{\beta_{e,K}} S_1S_2 \{[y := A_1]S_3 \} \{y := A_1\}B \) as \( x \not\in FV \). (\( S_1S_2 \) due to VC.

- Case \( [x]M \rightarrow_{\beta_{e,K}} [x]N \rightarrow_{\beta_{et}} O \), then \( O \equiv [x]Q \). Use IH on \( M \rightarrow_{\beta_{e,K}} N \rightarrow_{\beta_{et}} Q \).

- Case \( (A)B \rightarrow_{\beta_{e,K}} (A')B \rightarrow_{\beta_{et}} O \), investigate how \( (A')B \rightarrow_{\beta_{et}} O \). We only treat the case where \( (A')B \equiv (A')S[x]B_1 \rightarrow_{\beta_{et}} S[x := A']B_1 \). Then 
  \( (A)B \equiv (A)S[x]B_1 \rightarrow_{\beta_{et}} S[x := A]B_1 \rightarrow^+_{\beta_{e,K}} S[x := A']B_1 \) by Lemma 9.
• Case $(A)B \rightarrow \beta_{\text{k}} (A)B' \rightarrow \beta_{\text{sl}} O$, we only treat the case where $B' \equiv S[x]C$, $S$ is w.h., and $O \equiv S[x]C$ is w.h. I.e. $(A)B \rightarrow \beta_{\text{k}} (A)S[x]C \rightarrow \beta_{\text{sl}} S[x]C \rightarrow \beta_{\text{k}} S[x]C$ (note that it cannot occur that $B \equiv (Q)S[x][y]C \rightarrow \beta_{\text{k}} S[x][y]C$). Then one of the following holds:

- Case $B \equiv S[x]C_1$ and $C_1 \rightarrow \beta_{\text{k}} C$, then
  $(A)S[x]C \rightarrow \beta_{\text{k}} S[x]C \rightarrow \beta_{\text{k}} S[x]C$ by Lemma 9, case 3.
- Case $B \equiv S_1[x]C$ and the $\beta_{\text{k}}$-redex is in $S_1$, i.e.
  $(A)S_1[x]C \rightarrow \beta_{\text{k}} (A)S_1[x]C \rightarrow \beta_{\text{k}} S_1[x]C \rightarrow \beta_{\text{k}} S_1[x]C$ by VC.

6 The generalised conservation for $\beta_{\text{c}}$

In this section, we establish in Theorem 28, the generalised conservation for $\beta_{\text{c}}$, the same relation for which we established in the previous section, the postponement of its $K$-reduction. This is an extension of de Groote’s work which established the postponement and conservation properties for two different relations. We start by defining the set of labelled terms which will help us in establishing the generalised conservation. Labels will used as counters to record the number of contracted redexes when reducing a term.

**Definition 11** The set $^N\Lambda$ of labelled $\lambda$-terms is inductively defined as follows:

1. $n \in \mathbb{N}, x \in \mathbb{V} \Rightarrow \, ^nx \in ^N\Lambda$.
2. $n \in \mathbb{N}, x \in \mathbb{V}, M \in ^N\Lambda \Rightarrow \, ^nxM \in ^N\Lambda$.
3. $n \in \mathbb{N}, M, N \in ^N\Lambda \Rightarrow \, ^nm(M)N \in ^N\Lambda$

We take $M, N, O, A, B, \ldots$ to range over labelled $\lambda$-terms. We use $^nM$ to stress that the outermost label of a $\lambda$-term $M$ is $n$. Hence, $M$ and $^nM$ stand for the same labelled $\lambda$-term. We write $^nmM$ for the labelled $\lambda$-term obtained by adding $m$ to the outermost label of a labelled $\lambda$-term $M$. Hence if the outermost label of $M$ is $n$ then $^nmM$ denotes $^n+M$.

For $M \in ^N\Lambda$, we write $[M]$ for the (unlabelled) $\lambda$-term in $\Lambda$ obtained by erasing all labels in $M$. Moreover, if $M \in \Lambda$, we identify $M$ with $M'$ in $^N\Lambda$ such that $[M'] \equiv M$ and all labels in $M'$ are 0. Hence, $\Lambda \subset ^N\Lambda$.

We use in this section, the notations and techniques of de Groote adapted however to our generalised reduction. Basically the idea is as follows:

1. **Church Rosser CR:** We introduce a labelled reduction relation $\rightarrow_{\beta_{\text{c}}^+}$ which we prove Church Rosser. $\rightarrow_{\beta_{\text{c}}^+}$ is shown CR by showing that a related reduction relation $\rightarrow_1$ is CR. Hence, if a labelled term $M$ has a $\beta_{\text{c}}^+$-nf, it must be unique.

2. **Increasing property Inc:** We then introduce the weight of a term $M, \Theta[M]$, which is used to limit the length of $\beta_{\text{c}}^+$-reductions starting at normalising terms. That is, the length of any sequence of $\beta_{\text{c}}^+$-reductions starting at a normalising term $M$ is bounded by $\Theta[M] - \Theta[M]$ where $M'$ is the (unique) $\beta_{\text{c}}^+$-nf of $M$. This implies that any $\beta_{\text{c}}^+$-N term is $\beta_{\text{c}}^+$-SN. This will be extended to $\beta_{\text{c}}$ by showing that any $\beta_{\text{c}}$-N term is $\beta_{\text{c}}$-SN.
3. **Weak Normalisation ⇒ Strong Normalisation** (WN ⇒ SN) Next we show that if $M$ is $\beta_f$-N then it is $\beta_f$-SN by using the fact that $M$ is $\beta_f$-SN, postponement and that there can only be a finite $\beta_{iK}$-redexes. This establishes the generalised conservation.

Note that the structure of our proof can be seen as: CR+Inc+WN ⇒ SN. This is as we said a generalisation of the proof of de Groote. One could however use Corollary 5.19 in Klopf’s thesis [16] which states that WCR+Inc+WN ⇒ SN & CR where WCR is Weak Church Rosser. We leave this alternative to the reader to establish.

Here is the definition of substitution on labelled terms:

**Definition 12** Let $M, N \in \mathcal{N}_\Lambda$. $[x := N]M$ is defined as follows:

\[
[x := nN]^m x \quad \equiv \quad n^m N \\
[x := nN]^m y \quad \equiv \quad m^y \quad \text{if} \quad x \neq y \\
[x := nN]^m(P)Q \quad \equiv \quad m([x := nN]^m[x := nN]^m)[x := nN]Q \\
[x := nN]^m[y]M \quad \equiv \quad m[y][x := nN]M
\]

Now we define $\rightarrow_{\beta_{st}^+}$ which will be used to show conservation.

**Definition 13** $M \rightarrow_{\beta_{st}^+} N$ is defined inductively as follows:

1. $n(iN)S^i[x]iM \rightarrow_{\beta_{st}^+} n^{i+1}S[x := iN]iM$ if $x \in FV(M), S$, w.b.

2. If $M \rightarrow_{\beta_{st}^+} N$ then $n[x]M \rightarrow_{\beta_{st}^+} n[x]N, n(M)P \rightarrow_{\beta_{st}^+} n(P)N$ and $n(P)M \rightarrow_{\beta_{st}^+} n(P)N$.

$\rightarrow_{\beta_{st}^+}$ is defined as the transitive reflexive closure of $\rightarrow_{\beta_{st}^+}$.

We define $\rightarrow_1$ for which CR is easier to show than for $\rightarrow_{\beta_{st}^+}$.

**Definition 14** $M \rightarrow_1 N$ is defined inductively as follows:

1. $M \rightarrow_1 M$

2. If $M \rightarrow_1 N$ then $n[x]M \rightarrow_1 n[x]N$

3. If $M \rightarrow_1 O$ and $N \rightarrow_1 P$ then $n(M)N \rightarrow_1 n(O)P$

4. If $S^p[x]M \rightarrow_1 S'[x]O, N \rightarrow_1 P, S, S'$ w.b., and $x \in FV(M)$ then $n(N)S^p[x]M \rightarrow_1 n^{i+1}S'[x := P]O$.

$\rightarrow_1$ is defined as the transitive reflexive closure of $\rightarrow_1$.

The following lemma shows that labels can be increased for both $\rightarrow_1$ and $\rightarrow_{\beta_{st}^+}$.

**Lemma 15** Let $M, N \in \mathcal{N}_\Lambda$ and $r \in \{1, \beta_{st}^+\}$. If $M \rightarrow_r N$ then $^+nM \rightarrow_r ^+nN$.

**Proof** By induction on the derivation $M \rightarrow_r N$.

The following lemma shows that $\rightarrow_1$ and $\rightarrow_{\beta_{st}^+}$ are closed under substitution.
Lemma 16  Let $M, N, P, O \in \mathcal{N}_{\lambda}$. The following hold:

1. If $M \rightarrow_1 N$, then $[x := M^{\rightarrow_1} mO \rightarrow_1 [x := N]^{\rightarrow_1} O$.
2. If $M \rightarrow_{\beta_{st}^+} N$, then $[x := M]^{\rightarrow_{\beta_{st}^+}} mO \rightarrow_{\beta_{st}^+} [x := N]^{\rightarrow_{\beta_{st}^+}} O$.
3. If $M \rightarrow_1 N$ and $O \rightarrow_1 P$ then $[x := O]M \rightarrow_1 [x := P]N$.
4. If $M \rightarrow_{\beta_{st}^+} N$ and $O \rightarrow_{\beta_{st}^+} P$ then $[x := O]M \rightarrow_{\beta_{st}^+} [x := P]N$.

Proof  1 and 2 are similar and are by induction on the structure of $O$. 3 and 4 are by induction on the derivation $M \rightarrow, N$ where $r$ is the corresponding reduction.

Here is the relationship between $\rightarrow_1$ and $\rightarrow_{\beta_{st}^+}$:

Lemma 17  $M \notightarrow_1 N$ if $M \notightarrow_{\beta_{st}^+} N$.

Proof  $\Rightarrow$ By induction on the derivation of $M \rightarrow_1 N$ show that:

$M \rightarrow_1 N \Rightarrow M \notightarrow_{\beta_{st}^+} N$.

$\Leftarrow$ By induction on the derivation $M \notightarrow_{\beta_{st}^+} N$, show that $M \notightarrow_{\beta_{st}^+} N \Rightarrow M \notightarrow_1 N$.

The following two lemmas enable us to establish that $\rightarrow_1$ is CR.

Lemma 18  If $S, S'$ w.r.t. none of the binding variables of $S[x]$ occurs free in $N$, none of the binding variables of $S'[x]$ occurs free in $P$, none of the binding variables of $N$ are free in $M$ and none of the binding variables of $P$ are free in $O$, $S'[x]M \rightarrow_1 S'[x]O$ and $N \rightarrow_1 P$ then $\notightarrow S[x := N]M \rightarrow_1 \notightarrow S'[x := P]O$.

Proof  Note that if $\text{weight}(S) = \text{weight}(S')$ then $p = q$, $M \rightarrow_1 O$ and if $(A_i)$ and $(A'_i)$ are the $i$th main application items of $S$ and $S'$ respectively, then $A_i \rightarrow_1 A'_i$. Hence the result is shown by Lemmas 15 and 16 and the def. of $\rightarrow_1$. If $\text{weight}(S) > \text{weight}(S')$, then we prove the lemma by induction on $\text{weight}(S)$.

Lemma 19  Let $M, N, O \in \mathcal{N}_{\lambda}$ such that $M \rightarrow_1 N$ and $M \rightarrow_1 O$ then $\exists P \in \mathcal{N}_{\lambda}$ such that $N \rightarrow_1 P$ and $O \rightarrow_1 P$.

Proof  By induction on the derivation of $M \rightarrow_1 N$.

Now we have the Church Rosser property for $\rightarrow_1$:

Corollary 20 (Church Rosser of $\rightarrow_1$) Let $M, N, O \in \mathcal{N}_{\lambda}$ such that $M \rightarrow_1 N$ and $M \rightarrow_1 O$ then $\exists P \in \mathcal{N}_{\lambda}$ such that $N \rightarrow_1 P$ and $O \rightarrow_1 P$.

Hence, the first part of this section (CR of $\rightarrow_{\beta_{st}^+}$) is done:

Lemma 21 (Church Rosser of $\rightarrow_{\beta_{st}^+}$) Let $M, N, O \in \mathcal{N}_{\lambda}$ such that $M \rightarrow_{\beta_{st}^+} N$ and $M \rightarrow_{\beta_{st}^+} O$ then $\exists P \in \mathcal{N}_{\lambda}$ such that $N \rightarrow_{\beta_{st}^+} P$ and $O \rightarrow_{\beta_{st}^+} P$.

Proof  By Corollary 20 and Lemma 17.

In order to show Lemma 25, we introduce the following definition:
Definition 22 The weight $\Theta[M]$ of a labelled $\lambda$-term $M$ is defined as follows:

$$
\begin{align*}
\Theta^{}[x] &= n \\
\Theta^{}[y] &= n + \Theta[M] \\
\Theta^{}(M)N &= n + \Theta[M] + \Theta[N]
\end{align*}
$$

The following two lemmas enable us to establish that the weight as we defined will help us to measure terminating reductions:

Lemma 23 If $x \in FV(M)$ then $\Theta[x := N]M \geq \Theta[M] + \Theta[N]$.

Proof By induction on the structure of $M$ showing first that $\Theta[\text{+}_M] = m + \Theta[M]$.

Lemma 24 Let $M, N \in N\Lambda$ and $M \rightarrow^+_{\beta^+_J} N$ then $\Theta[M] < \Theta[N]$.

Proof By induction on the derivation $M \rightarrow^+_{\beta^+_J} N$ using Lemma 23.

Now, $\beta^+_I$-N and $\beta^+_J$-SN are the same:

Lemma 25 If $M$ is $\beta^+_I$-N then $M$ is $\beta^+_J$-SN.

Proof Since $M$ is $\beta^+_I$-N, and since $\beta^+_I$ is Church-Rosser by Lemma 21, then $M$ has a unique $\beta^+_I$-nf $M'$. According to Lemma 24, the length of any sequence of $\beta^+_I$-reduction starting at $M$ is bounded by $\Theta[M'] - \Theta[M]$.

Here is the relationship between $\rightarrow_{\beta_J}$ and $\rightarrow_{\beta^+_J}$:

Lemma 26 Let $M, N \in \Lambda$ such that $M \rightarrow_{\beta_J} N$, then there exist $M', N' \in N\Lambda$ such that $|M'| \equiv M, |N'| \equiv N$ and $M' \rightarrow^+_{\beta^+_J} N'$. Furthermore, if $N$ is in $\beta^+_I$-nf then $N'$ is in $\beta^+_I$-nf.

Proof Easy. Put the right labels on $M$ and $N$ obtaining $M', N'$ where $M' \rightarrow_{\beta^+_J} N'$.

Now, we generalise Lemma 25 to $\rightarrow_{\beta^+_J}$

Theorem 27 If $M$ is $\beta^+_I$-N then $M$ is $\beta^+_J$-SN.

Proof $M \beta^+_I$-N $\Rightarrow L\text{emma 26}$ $M \beta^+_J$-N $\Rightarrow L\text{emma 25}$ $M \beta^+_J$-SN $\Rightarrow M \beta^+_J$-SN (otherwise there exists an infinite $\beta^+_J$-path).

Finally, from the above theorem and postponement of $K$-contractions, we can establish conservation:

Theorem 28 (Conservation) If $M$ is $\beta^+_I$-N then $M$ is $\beta^+_J$-SN.

Proof If $M$ is not $\beta^+_J$-SN then there is an infinite $\beta^+_J$-path starting at $M$. But by postponement of $\beta^+_K$ redexes, and by the fact that there can only be a finite $\beta^+_K$-contractions, there must be an infinite $\beta^+_J$-path. But $M$ is $\beta^+_I$-N and so it is $\beta^+_J$-SN by Theorem 27. Contradiction.
7 Preservation of Strong Normalisation

To show PSN, we show that if $M \rightarrow^{*}_F N$ (using the perpetual strategy of Definition 7) and if $N$ is $\beta_\perp$-N then $M$ is $\beta_\perp$-N. Now, we take $M$ which is $\beta$-SN, and its perpetual path to its normal form $N$. As $N$ is $\beta_\perp$-N, then $M$ is $\beta_\perp$-N and hence by conservation, $M$ is $\beta_\perp$-SN. It is possible however to show PSN in many different ways and without using conservation. For example, one may use a result of Regnier in [22] which states that the length of the longest reduction of a term is invariant by $\sigma$-equivalence noting that the $\beta$-reduction modulo $\sigma$-equivalence is isomorphic to the $\beta_\perp$-reduction.\(^3\)

In this section, we will use the already available conservation theorem and following Theorem 27, we interchange $\beta_\perp$-SN and $\beta_\perp$-N at liberty. We shall show PSN of $\beta_\perp$. Note that the derivation:

$$M \ \beta$-SN $\Rightarrow M \ \beta$-N $\Rightarrow M \ \beta_\perp$-N $\Rightarrow M \ \beta_\perp$-SN$

is incorrect because $M \ \beta$-N $\not\Rightarrow M \ \beta_\perp$-N. For example, $(\lambda x.y)\Omega$ is $\beta$-N but not $\beta_\perp$-N for $\Omega \equiv (\lambda x. \ v v) (\lambda x. \ v v)$. To show PSN, we take $M$ that is $\beta$-SN. Then $M \rightarrow^{*}_F N$ where $N$ is the $\beta$-nf of $M$ and $\rightarrow^{*}_F$ is the perpetual strategy. As $N$ is in $\beta$-nf, then $N$ is $\beta_\perp$-N. But the inverse of $\rightarrow^{*}_F$ preserves $\beta_\perp$-N. Hence, $M$ is $\beta_\perp$-N and by conservation, $M$ is $\beta_\perp$-SN.

In order to establish that the inverse of $\rightarrow^{*}_F$ preserves $\beta_\perp$-normalisation (Theorem 33), we need the following three lemmas which will be combined with the three forms of perpetual reduction for $K$-redexes as in Lemma 8.

**Lemma 29** If $(A_1) \ldots (A_n)(A)[x]P$ has $\beta_\perp$-nf, $x \not\in FV(P)$, then its $\beta_\perp$-nf is of the form $(B_1) \ldots (B_j)(A')[x]Q$ where $A'$ is the $\beta_\perp$-nf of $A$, $0 \leq j \leq n$, $B_j$ is the $\beta_\perp$-nf of some $A_i$ for $1 \leq i \leq n$. Moreover, $(A_1) \ldots (A_n)P$ has $(B_1) \ldots (B_j)Q$ as its $\beta_\perp$-nf.

**Proof** By induction on $n \geq 0$.

- $n = 0$, the $\beta_\perp$-nf is $(A')[x]Q$ where $Q$ is the $\beta_\perp$-nf of $P$.
- Assume the property holds for $n \geq 0$.
  
  As $(A_1) \ldots (A_n)(A_{n+1})(A)[x]P$ has $\beta_\perp$-nf, then it is $\beta_\perp$-SN and so $(A_2) \ldots (A_n)(A_{n+1})(A)[x]P$ and $A_1$ have $\beta_\perp$-nf. Call the $\beta_\perp$-nf of $A_1$, $A_1'$. Now, by IH, $(B_1) \ldots (B_j)(A')[x]Q$ is the $\beta_\perp$-nf of $(A_2) \ldots (A_n)(A_{n+1})(A)[x]P$ and $(B_1) \ldots (B_j)Q$ is the $\beta_\perp$-nf of $(A_2) \ldots (A_n)(A_{n+1})P$.

  - If $(A_1')$ is $\beta_\perp$-bachelor in $(A_1')(B_1) \ldots (B_j)(A')[x]Q$ (and hence it is bachelor in $(A_1')(B_1) \ldots (B_j)(B_j')Q$), then:
    
    $(A_1')(B_1) \ldots (B_j)(A')[y]Q$ and $(A_1')(B_1) \ldots (B_j)Q$ are the $\beta_\perp$-nfs required.

  - If $(A_1')$ is $\beta_\perp$-partnered in $(A_1')(B_1) \ldots (B_j)(A')[x]Q$ then all $(B_1), \ldots (B_j)$ start $\beta_K$-redexes and $Q \equiv [x_{j'}] \ldots [x_1][y]R$. Now, for $B$ the $\beta_\perp$-nf of $[y:=A_1'][x]R$ we have:
    
    $(A_1) \ldots (A_n)(A_{n+1})(A)[x]P \rightarrow^{*}_F \beta_\perp$,
    
    $(A_1')(B_1) \ldots (B_j)(A')[x_1] \ldots [x_1][y]R \rightarrow^{*}_F \beta_\perp$,
    
    $(B_1) \ldots (B_j)(A')[x_1] \ldots [x_1][y:=A_1'][x]R \rightarrow^{*}_F \beta_\perp$.

\(^3\)Thanks for an anonymous referee who drew my attention to this point.
\( (B_1) \ldots (B_j)(A')[x][x_j] \ldots [x_1]B. \)
Moreover, \((A_1) \ldots (A_n)(A_{n+1})P \stackrel{\rightarrow}{\beta_{I}} (A'_1)(B_1) \ldots (B_j)[x_j] \ldots [x_1][y]R \rightarrow_{\beta_{I}} \)
\((B_1) \ldots (B_j)[x_j] \ldots [x_1][y] := A'[x]R \rightarrow_{\beta_{I}} (B_1) \ldots (B_j)[x_j] \ldots [x_1]B. \) Now, we are done (note that \(B_1) \ldots (B_j\) start \(\beta_{K}\)-redexes).

**Lemma 30** If \((A_1) \ldots (A_n)P\) and \(A\) have \(\beta_{I}\)-nf, \(x \notin FV((A_1) \ldots (A_n)(A)P)\), then:
\((A_1) \ldots (A_n)(A)[x]P\) has \(\beta_{I}\)-nf.

**Proof** By induction on \(n \geq 0\).

- Case \(n = 0\), \(P\) and \(A\) have \(P'\) and \(A'\) as \(\beta_{I}\)-nfs, then \((A)[x]P\) has \((A')[x]P'\) as \(\beta_{I}\)-nf.

- Assume the property holds for \(n \geq 0\). Let \((A_1) \ldots (A_n)(A_{n+1})P\) have \(\beta_{I}\)-nf, hence it is \(\beta_{I}\)-SN and so \((A_2) \ldots (A_{n+1})P\) has \(\beta_{I}\)-nf and \(A_1\) has \(\beta_{I}\)-nf. By IH, \((A_2) \ldots (A_{n+1})(A)[x]P\) has \(\beta_{I}\)-nf which is by Lemma 29, \(M \equiv (B_1) \ldots (B_j)(A')[x]Q\) and \((A_2) \ldots (A_{n+1})P\) has \((B_1) \ldots (B_j)Q\) as its \(\beta_{I}\)-nf.

- If \((A'_1)\) is \(\beta_{I}\)-bachelor in \((A'_1)M\) then \((A'_1)M\) is the \(\beta_{I}\)-nf of \((A_1) \ldots (A_{n+1})(A)[x]P\).

- If \((A'_1)\) is \(\beta_{I}\)-partnered in \((A'_1)M\) then \(Q \equiv [x_j] \ldots [x_1][y]R\). Now, \((A_1) \ldots (A_n)(A_{n+1})P \rightarrow_{\beta_{I}} (A'_1)(B_1) \ldots (B_j)[x_j] \ldots [x_1][y]R \rightarrow_{\beta_{I}} \).

**Lemma 31** If \(A_i \rightarrow_{\beta_{K}} B_i, (A_1) \ldots (A_i-1)(B_i)(A_{i+1}) \ldots (A_n)z\) and \(A_i\) have \(\beta_{I}\)-nf then \((A_1) \ldots (A_{i-1})(A_i)(A_{i+1}) \ldots (A_n)z\) has \(\beta_{I}\)-nf.

**Proof** \(A_1, \ldots , A_{i-1}, A_i, A_{i+1}, \ldots , A_n\) all have \(\beta_{I}\)-nf, \(A'_1, \ldots , A'_{i-1}, A'_i, A'_{i+1}, \ldots , A'_n\).

Hence, \((A_1) \ldots (A_{i-1})(A_i)(A_{i+1}) \ldots (A_n)z \rightarrow_{\beta_{I}} (A'_1) \ldots (A'_{i-1})(A'_i)(A'_{i+1}) \ldots (A'_n)z\) which is in \(\beta_{I}\)-nf.

**Lemma 32** If \(M \rightarrow_{F} N\) using a \(\beta_{K}\)-reduct, and \(N\) has a \(\beta_{I}\)-nf, then \(M\) has \(\beta_{I}\)-nf.

**Proof** By induction on the depth of the \(F\)-reduct (following Lemma 8).

- If \(M \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A)[x]P\), where \(x \notin FV(P)\), \(A\) is in \(\beta\)-nf and \(m \geq 0\), and if \(N \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)P\) where \(n \geq 0\), use Lemma 30 (\(A\) in \(\beta\)-nf \(\Rightarrow A\) in \(\beta_{I}\)-nf).

- Let \(S \equiv [x_1][x_2] \ldots [x_m](A)(A_2) \ldots (A_n)\). If \(M \equiv S(\mathcal{A})(x)P, N \equiv S(A')[x]P\) where \(x \notin FV(P)\), \(A\) is not in \(\beta\)-nf, \(A \rightarrow_{F} A'\), \(n \geq 0\) and \(m \geq 0\). Use IH to deduce that \(A\) has \(\beta_{I}\)-nf. As \(N\) has \(\beta_{I}\)-nf then \([x_1][x_2] \ldots [x_m](A)((A_2) \ldots (A_n)P\) has \(\beta_{I}\)-nf by Lemma 29. Hence, \([x_1][x_2] \ldots [x_m](A)(A_2) \ldots (A_n)(A)[x]P\) has \(\beta_{I}\)-nf by Lemma 30.
\begin{itemize}
\item If \(M \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A)(B_1)(B_2) \ldots (B_r)z\),
\[N \equiv [x_1][x_2] \ldots [x_m](A_1)(A_2) \ldots (A_n)(A')(B_1)(B_2) \ldots (B_r)z\] where
\(A\) is not in \(\beta\)-nf, \(A \rightarrow_F A'\), \(n \geq 0\), \(m \geq 0\) and \(r \geq 0\) and \(\forall i, 1 \leq i \leq r, B_i\) is in \(\beta\)-nf. As \(N\) has \(\beta_{\ell}\)-nf, so does \(A'\) and by IH, so does \(A\). By Lemma 31, \(M\) has \(\beta_{\ell}\)-nf.
\end{itemize}

Now, here is the key theorem to PSN:

**Theorem 33 (The inverse of \(\rightarrow_F\) preserves \(\beta_{\ell}\)-N)**

If \(M \rightarrow_F N\) and \(N\) is \(\beta_{\ell}\)-N, then \(M\) is \(\beta_{\ell}\)-N.

**Proof** We show it for \(M \rightarrow_F N\). Note that if \(M \rightarrow_F N\) and \(F(M)\) is a \(\beta_{\ell}\)-redex, then the theorem is obvious as a \(\beta_{\ell}\)-redex is a \(\beta_{\ell}\)-redex. Hence, we only need to prove the theorem for the case when \(F(M)\) is a \(\beta_\kappa\)-redex. But this is already done in Lemma 32.

Finally, here is the PSN result.

**Corollary 34 (Preservation of Strong Normalisation)**

If \(M\) is \(\beta\)-SN then \(M\) is \(\beta_{\ell}\)-SN.

**Proof** As \(M\) is \(\beta\)-SN, the perpetual strategy of \(M\) terminates. Let \(M \rightarrow_F N\) where \(N\) is in \(\beta\)-nf. As \(N\) has no \(\beta\)-redexes, \(N\) is \(\beta_{\ell}\)-N. Hence, by Theorem 33, \(M\) is \(\beta_{\ell}\)-N.

So, by Theorem 28 \(M\) is \(\beta_{\ell}\)-SN.

### 8 Conclusion

In this paper, we established that there is indeed a reduction relation which satisfies both postponement of \(K\)-contractions and conservation. This reduction relation is a generalisation of the ordinary \(\beta\)-reduction and has been extensively used since '73 for theoretical and practical reasons (see Section 2). We showed moreover that this generalised reduction (called \(\beta_{\ell}\)) is indeed a desirable generalisation of \(\beta\)-reduction by showing that \(\beta_{\ell}\) preserves strong normalisation in the sense that if \(M\) is \(\beta\)-SN then \(M\) is \(\beta_{\ell}\)-SN. Preservation of Strong Normalisation (PSN) is a property that has to be established for any extension of a reduction relation in the sense that: if a term is strongly normalising for a reduction relation, then it must remain strongly normalising for its extension. For example, a lot of research has been carried out lately to establish PSN for \(\beta\)-reduction extended with explicit substitution (see [4], [9] and [18]). The results of this paper establish that \(\beta_{\ell}\) is indeed a safe extension of \(\beta\).

It is worth noting that we used item notation in this paper in order to reach the results desired. There is a reason for this. In the usual notation, generalised redexes are not easily visible whereas they are in item notation (see [8]). For example, in 
\[
(\_)(\_)(\_)(\_)(\_)
\]
we can clearly see that the leftmost (\_) matches the rightmost (\_)]. Using item notation enables us to write the proofs clearly. Compare with [6] who used a more restricted generalised reduction but it was still hard to discuss where generalised redexes occurs in a term. For more information on the simplicity and usefulness of item notation, the reader is referred to [8]. It should be noted moreover, that using
item notation is not restrictive and that the results of this paper would still hold if we used the classical notation. Only the proofs will be cumbersome to write as the classical notation cannot easily enable us to express generalised redexes.

The following is an itemised summary of this paper:

1. Postponement and conservation are shown for the same reduction relation $\beta_e$.

2. This reduction relation is a generalisation of that presented in de Groote’s article and of many of the existing generalisations of $\beta$-reduction. Because of this, it enables greater flexibility in the ordering of evaluation.

3. The fact that this reduction relation preserves strong normalisation and enables the postponement of some work allowing unnecessary $K$-redexes to be bypassed, means that one can investigate a programming language evaluation strategy based on this reduction together with explicit substitutions (which allow bypassing any work inside a subterm that will be discarded later). This means that one can combine the advantages of the complementary ways of postponing work due to both explicit substitutions and generalised reduction. Hence, one can achieve an even greater flexibility in the order of reduction and evaluation, something very welcome in the implementation and compilation of programming languages.

4. The syntax of this paper may well be the answer to the existence of a syntax that realises Regnier’s $\sigma$-reduction. We leave this for future explorations.

5. It is our belief that automating proofs written in this fashion may be more efficient than automating the proofs written using $\sigma$-equivalence (which was said to need a good syntax to describe it in [22]).

Acknowledgements

I am grateful to Joe Wells and the anonymous referees for their comments on the paper. This work is supported by EPSRC grants number GR/L15685 and GR/L36963. Anonymous referees provided useful comments for which I am grateful.

References


