INFORMATION AND COMPUTATION 75, 191–231 (1987)

# Needed Reduction and Spine Strategies for the Lambda Calculus

H. P. BARENDREGT*

*University of Nijmegen, Nijmegen, The Netherlands*

J. R. KENNAWAY†

*University of East Anglia, Norwich, United Kingdom*

J. W. KLOP‡

*Centre for Mathematics and Computer Science, Amsterdam, The Netherlands*

AND

M. R. SLEEP†

*University of East Anglia, Norwich, United Kingdom*

A redex $R$ in a lambda-term $M$ is called *needed* if in every reduction of $M$ to normal form (some residual of) $R$ is contracted. Among others the following results are proved: 1. $R$ is needed in $M$ iff $R$ is contracted in the leftmost reduction path of $M$. 2. Let $\mathscr{R}: M_0 \to M_1 \to M_2 \to \ldots$ reduce redexes $R_i: M_i \to M_{i+1}$, and have the property that $\forall i. \exists j \geqslant i. R_j$ is needed in $M_j$. Then $\mathscr{R}$ is normalising, i.e., if $M_0$ has a normal form, then $\mathscr{R}$ is finite and terminates at that normal form. 3. Neededness is an undecidable property, but has several efficiently decidable approximations, various versions of the so-called *spine* redexes. 

## 1. INTRODUCTION

A number of practical programming languages are based on some sugared form of the lambda calculus. Early examples are LISP, McCarthy *et al.* (1961) and ISWIM, Landin (1966). More recent exemplars include ML (Gordon *et al.*, 1981), Miranda (Turner, 1985), and HOPE (Burstall

* Partially supported by the Dutch Parallel Reduction Machine project.
† Partially supported by the British ALVEY project.
‡ Partially supported by ESPRIT project 432, Meteor.



0890-5401/87 $3.00

*et al.*, 1980). We use the term *lambda language* to cover these and similar languages.

Classical implementations of a lambda language have adopted an applicative order evaluation strategy, as embodied for example in the original SECD machine (Landin, 1964). It is well known that this strategy is not normalising for all lambda terms. It is also well known that the leftmost reduction strategy is normalising. However, until graph reduction was introduced by Wadsworth (1971) the leftmost strategy was not considered practicable.

The adoption of a normalising implementation of lambda languages has a number of advantages, of which the ability to specify data structures of unbounded size is most notable. Turner (1979, 1985) has argued the case for normalising implementations in a number of papers.

Recent advances in compiling techniques have led to normalising implementations of lambda languages on sequential machines which rival in performance terms applicative order implementations, e.g., Augustsson (1984). By taking advantage of the side-effect-free nature of lambda languages (at least in benign incarnations) it may be possible to achieve further improvements in performance by developing appropriate parallel architectures.

However, the best-known normalising strategy for the lambda calculus is the leftmost strategy, and this is sequential in the sense that identifying the "next" leftmost redex cannot in general be achieved without at least identifying the current leftmost redex. Equally, at least some of the identification work can be done by a compiler: recent work on strictness analysis, such as, Mycroft (1981), has exploited this observation.

The fundamental notion underlying this paper is that in every lambda term not in normal form there are a number of *needed* redexes. A redex is said to be needed in a term in a term $M$ if $R$ has to be contracted (sooner or later) when reducing $M$ to normal form. It will be shown that these redexes can be reduced in any order, or in parallel, without risking unnecessary nontermination. We will present efficient algorithms for identifying sets of needed redexes in a term. The most general concept of neededness is undecidable, as we show in Theorem 3.12. However, a family of algorithms can be identified which deliver increasingly better (but increasingly costly) approximations to the needed set. All the algorithms offered identify redexes which can be contracted safely, i.e., secure in the knowledge that such contraction will reduce the length of a leftmost reduction sequence to normal form by at least 1.

In Berry and Lévy (1979) and also in Lévy (1980) certain families of redexes are identified in order to obtain optimisations for some classes of reduction systems.

The various algorithms for detecting needed redexes are comparable to

the so-called abstract interpretations of terms, see Burn *et al.* (1986). For example, in the simplest of our algorithms the term

$$(\lambda x.(\lambda y.yPQ)R)S$$

is mapped to

$$(\lambda x.(\lambda y.y\bot\bot)\bot)\bot,$$

concluding that the two remaining redexes are needed in the original term.

Just which of the defined algorithms is appropriate for a given implementation is technology and application dependent. Our contribution is to offer a range of choices to the implementor which frees him from the sharp distinction between applicative and normal order strategies, which currently forces him to either accept wholesale the inefficiency risks associated with normal order, or to buy the known efficiency of applicative order at the cost of losing normalising properties for his implementation.

The relation with strictness analysis is as follows. There is a sharper notion of neededness: a redex $R$ is *head-needed* in a term $M$ if $R$ has to be contracted in any reduction to head normal form. For example $R$ in $\lambda x.Rx$ is needed and head-needed, but in $\lambda x.xR$ only needed. This notion of head-neededness is essentially the same as that of strictness, albeit that head-neededness refers to the argument whereas strictness refers to the function: we have for all redexes $R$ and all contexts $C[\ ]$,

$$R \text{ is head-needed in } C[R] \Leftrightarrow C[\ ] \text{ is strict in its argument } [\ ].$$

See Section 5 for further discussion on strictness.

*Plan of the paper.* In Section 2 we introduce the concepts and terminology necessary to make this paper self-contained. Section 3 contains the major new theoretical concepts and results: the main result in this section is that any strategy which eventually removes all needed redexes in a term is normalising. Section 4 develops some practical algorithms for identifying sets of needed redexes in a term. Section 5 offers some concluding remarks on strictness analysis, sequentiality, and extensions of $\lambda$-calculus. The Appendix discusses the method of Lévy-labelled $\lambda$-calculus, which can be used to provide alternative proofs of some of our results.

## 2. Preliminaries

In this paper we will use notation and terminology of Barendregt (1984). However, in order to make the paper practically self-contained, we will introduce the relevant concepts and notations in the present section. Also

some specific preparations for the sequel are included, in the form of Propositions 2.6–2.9.

2.1. DEFINITIONS. The set of *λ-terms*, notation $\Lambda$, is defined inductively by

(a) $x, y, z, \ldots \in \Lambda$;

(b) $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$;

(c) $M \in \Lambda \Rightarrow (\lambda x.M) \in \Lambda$.

If in (c) the proviso $x \in FV(M)$ is added, we get the set of *λI-terms*. Here $FV(M)$ is the set of free variables of $M$.

A term of the form $(MN)$ is an *application* (of $M$ to $N$). $M$ is the *rator* and $N$ is the *rand*. A term $(\lambda x.M)$ is an *abstraction*, $x$ is its *bound variable*, and $M$ is its *body*.

In applications the usual bracket convention of "association to the left" is employed; also outermost brackets are omitted. Repeated abstractions like $(\lambda x.(\lambda y.(\lambda z.M)))$ are written as $\lambda xyz.M$.

We use "$\equiv$" to indicate syntactical identity of terms, reserving "$=$" for the relation of reduction, which we will now define.

A term $R \equiv (\lambda x.A)B$ is called a *redex*. Given such a term, $R' \equiv A[x := B]$ denotes the result of substituting $B$ for the free occurrences of $x$ in $A$. $R'$ is called the *contractum* of $R$. A term not containing redexes is a *normal form* (or: *in normal form*). The passage from a redex to its contractum $R \to R'$ is called a *contraction*. *One step (β-)reduction* is defined by $C[R] \to C[R']$, where $R \to R'$ is a redex contraction and $C[\ ]$ is a context with one hole, i.e., a $\lambda$-term with one occurrence of a hole $[\ ]$. $C[M]$ is the result of substituting $M$ for $[\ ]$ in $C[\ ]$. The subterm relation sub is defined by

$$M \text{ sub } N \Leftrightarrow N \equiv C[M] \qquad \text{for some } C[\ ].$$

When stating that $M$ is a subterm of $N$, in this paper we will refer always to some specific occurrence of $M$ in $N$.

If we want to display which redex $R$ is contracted in the reduction step $M \to N$, we write $R: M \to N$. Again here, we refer to a specific occurrence of $R$ in $M$. The transitive reflexive closure of the one step reduction relation $\to$ is denoted by $\twoheadrightarrow$. *Reduction sequences* (or *reductions*, for short) $M_0 \to M_1 \to M_2 \to \cdots$ will be denoted by $\mathscr{R}, \mathscr{S}, \ldots$. They may be finite or infinite. Although it is an abuse of notation, we will sometimes write a reduction $\mathscr{R}: M_0 \to M_1 \to \cdots \to M_n$ as $\mathscr{R}: M_0 \twoheadrightarrow M_n$, still bearing in mind that we refer to a specific reduction from $M_0$ to $M_n$.

The equivalence relation generated by $\to$ is called *convertion* and written as "$=$." It should be distinguished from $\equiv$, syntactical equality.

*Leftmost Reductions and Head Reductions*

2.2. DEFINITIONS. If $N$ is an abstraction term $\lambda x.A$ we call the prefix $\lambda x$ the *abstractor* of $N$. Likewise if $R$ is a redex $(\lambda x.A)B$ the abstractor of $R$ is $\lambda x$. If $M$ is not a normal form, the *leftmost* redex of $M$ is that redex whose abstractor is to the left of the abstractor of every other redex in $M$. A leftmost reduction is one in which each contracted redex is leftmost. A leftmost reduction step is denoted by $\rightarrow_{\mathrm{lm}}$.

The leftmost redex $R$ is a nonnormal form $M$ is called the *head redex* if its abstractor is only preceded (in the left to right order of symbols) by abstractors of abstraction terms (not redexes). In particular, the abstractor of the head redex is not preceded by a variable. Thus in $M \equiv \lambda x.xR$, where $R$ is a redex, $R$ is the leftmost redex but not the head redex; $M$ has no head redex. On the other hand, $R$ is the head redex of $\lambda x.RABC$. A term is in *head normal form* if it has no head redex. The set of head normal forms can be defined as follows: for any terms $M_1, \ldots, M_n$, the term $\lambda x_1 \ldots x_m.yM_1 \ldots M_n$ is a head normal form. Here $n, m \geqslant 0$. A head reduction is one in which only head redexes are contracted. This is all standard terminology (apart from "abstractor"); the following is not.

2.3. DEFINITION. The *active components* of $M$ are the maximal subterms of $M$ which are not in head normal form.

Here "maximal" refers to the subterm ordering; so the active components are mutually disjoint. If a term is not in head normal form, it has one active component, namely itself. A normal form has no active components. The set of active components of $\lambda x_1 \ldots x_n.yN_1 \ldots N_k$ is the union of the sets of active components of $N_1, \ldots, N_k$. The word "active" refers to the fact that the active components are embedded in a context which is "frozen," i.e. a normal form when the holes [ ] are viewed as variables. (This frozen context of $M$ is the trivial context [ ] if $M$ is not a head normal form.)

2.4. DEFINITIONS. If $N$ is a subterm of $M$, the *descendants* of $N$ after a reduction step $M \rightarrow M'$ are those subterms in $M'$ which can be "traced back" to $N$ in $M$, in the following sense. If $N \equiv x$, the notion is clear. If $N$ is an abstraction term $(\lambda x.A)$ or an application $(AB)$, we look when tracing to the outermost pair of brackets of $N$. (For a more precise definition using labels or underlining see Klop, 1980 or Barendregt, 1984.) Our stipulation that the "identity" of the outermost bracket pair determines the descendants, entails that the contractum $R'$ of the redex $R$ in $M$ is not a descendant of $R$ after the reduction step $R$: $M \rightarrow M'$ (since in this redex contraction the original outermost pair of brackets of $R$ vanishes). Decendants of

redexes are also called *residuals*. Note that by the previous remark residuals of a redex are again redexes. The notion of descendants or residuals after one reduction step extends by transitivity to the notion of descendants or residuals after a finite reduction sequence $\mathscr{R}$.

If $\mathscr{R}: M \twoheadrightarrow N$ and $\mathscr{S}: M \twoheadrightarrow L$ are two "diverging" reductions, the well-known Church–Rosser theorem states that "converging" reductions $\mathscr{R}': L \twoheadrightarrow P$ and $\mathscr{S}': N \twoheadrightarrow P$ can be found. (In another well-known terminology, $\lambda$-calculus is said to be *confluent*.) A stronger version of this theorem asserts that these converging reductions can be found in a canonical way, by adding "elementary reduction diagrams" as suggested in Fig. 1.

The *reduction diagram* originating in this way is called $D(\mathscr{R}, \mathscr{S})$, and in Klop (1980) or Barendregt (1984) it is proved that it *closes*; i.e., the construction terminates and yields reductions $\mathscr{R}'$ and $\mathscr{S}'$ as desired. We write $\mathscr{R}' = \mathscr{R}/\mathscr{S}$ and call $\mathscr{R}'$ the *projection* of $\mathscr{R}$ by $\mathscr{S}$. Elementary reduction diagrams are obtained as follows: if $R: M \to N$ and $S: M \to L$ are two diverging reduction steps, converging reductions (making the elementary diagram complete) consist of contracting the residuals of $R$ after $S: M \to L$, resp. the residuals of $S$ after $R: M \to N$. In case one (or both) of these sets of residuals is empty, we introduce "empty" reductions as, e.g., in the elementary diagram shown in Fig. 2 (where $I \equiv \lambda x.x$).

Letting $R$ stand also for the reduction sequence $R: M \to N$ which reduces just $R$, then given any reduction $\mathscr{R}: M \twoheadrightarrow L$, the *parallel moves lemma* asserts that the projection $R/\mathscr{R}$ consists of the contraction of all residuals of $R$ after $\mathscr{R}$.

It is important to note that in general the residuals in $P$ of redex $R$ in $M$ after a reduction $M \twoheadrightarrow P$ depend on the actual reduction from $M$ to $P$. However, this is not so if $M$ and $P$ are left-upper corner and right-lower corner, respectively, of an elementary reduction diagram. As a consequence, this implies the following.

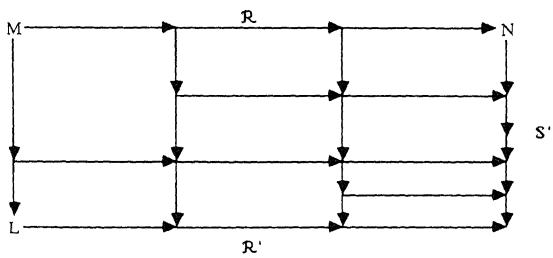


FIGURE 1

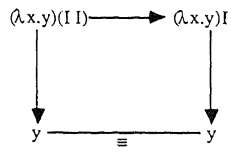


FIGURE 2

2.5. PROPOSITION. *Let* $\mathscr{R}: M \twoheadrightarrow N$ *and* $\mathscr{S}: M \twoheadrightarrow M'$. *Consider the reduction diagram of Fig.* 3. *Let R sub M be a redex. Then the residuals of R in N' are the same with respect to the two reduction paths* $\mathscr{R} * (\mathscr{S}/\mathscr{R})$ *and* $\mathscr{S} * (\mathscr{R}/\mathscr{S})$.

*Proof.* The property holds for elementary reduction diagrams and therefore also for reduction diagrams. See Barendregt (1984) for more details. ∎

In the sequel we will need the following fact.

2.6. PROPOSITION. *Consider the situation as in Fig.* 3. *Let R sub M be a redex none of whose residuals is contracted in* $\mathscr{R}$. *Let R' sub M' be a residual of R after reduction* $\mathscr{S}$. *Then no residual of R' is contracted in* $\mathscr{R}/\mathscr{S}$.

*Proof.* Suppose that a residual of $R'$ is contracted in $\mathscr{R}/\mathscr{S}$. Since in $\mathscr{R}/\mathscr{S}$ the only redexes which are contracted are residuals of redexes contracted in $\mathscr{R}$, it follows by Proposition 2.5 that a residual of $R$ in a term of $\mathscr{R}$ is contracted—contradiction. (For an alternative proof see the Appendix.) ∎

We will need the following facts about reduction diagrams and the phenomenon of "redex creation." We say that redex $S$ in $M'$ is *created* by the step $R: M \to M'$ if $S$ is not a residual of any redex in $M$. Facts like the Lemma 2.7(ii) are a good illustration of the beneficial use of Lévy's labels expounded in the Appendix, which speeds up otherwise very tedious case verifications.

2.7. LEMMA. *Let* $M \to_{\mathrm{lm}} N$ *and* $R: M \to M'$.

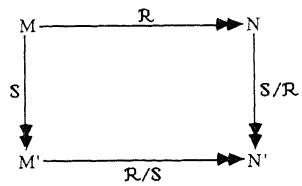


FIGURE 3

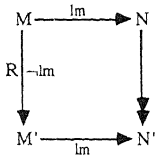


FIGURE 4

(i) *If $R$ is not leftmost, then a common reduct of $N$ and $M'$ can be found by contracting the leftmost redex in $M'$. See Fig.* 4.

(ii) *Moreover, if $S$ sub $M'$ is a redex and $S'$ sub $N'$ is a residual of $S$ in $N'$ via the contraction $M' \to_{\mathrm{lm}} N'$, then*

$$S \text{ is created in } R\colon M \to M' \Rightarrow S' \text{ is created in } N \to N'.$$

(iii) *If $R$ is arbitrary, then we have the elementary diagram of Fig.* 5, *where $M' \to_{\mathrm{lm}\equiv} N'$ denotes either the leftmost step or an empty step (in which case $M' \equiv N'$). Moreover, $M' \to_{\mathrm{lm}\equiv} N'$ is the empty step iff $R$ is the leftmost redex.*

*Proof.* (i) Routine. See, for example, Lemma 13.2.5 in Barendregt (1984).

(ii) By distinguishing some cases. (For an alternative proof see the Appendix.)

(iii) Immediate by (i). ∎

2.8. PROPOSITION. (i) *Let $M \to_{\mathrm{lm}} N$ and $M \to M'$. Then the reduction diagram looks like Fig.* 6.

(ii) *Let $M \to_{\mathrm{lm}} N$ and $M \to M'$. Then the reduction diagram looks like Fig.* 7. *Moreover, $M' \equiv N'$ iff the reduction $M' \to M'$ contracts a residual of the leftmost redex in $M$.*

*Proof.* (i) By Lemma 2.7(iii) we can make a diagram chase as in Fig. 8 and the result follows.

(ii) Again by Lemma 2.7(iii) and a simple diagram chase. ∎

## 3. NEEDED REDUCTION

3.1. DEFINITION. Let $R$ be a redex in $M$.

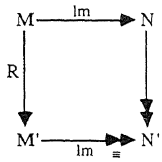


FIGURE 5

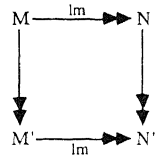


FIGURE 6

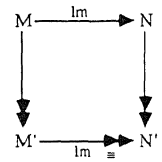


FIGURE 7

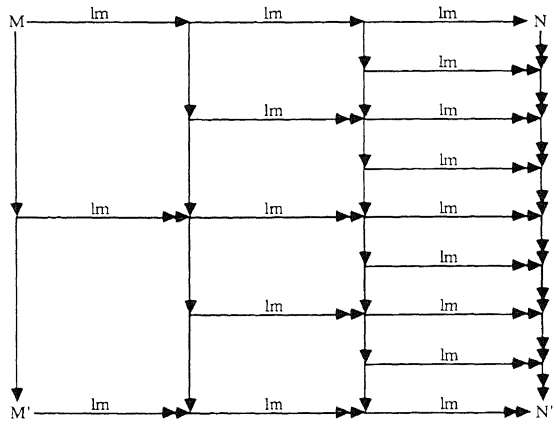


FIGURE 8

(i) $R$ is *needed* in $M$ (or just "needed" when the context makes clear which $M$ is intended) if every reduction sequence of $M$ to normal form reduces some residual of $R$.

(ii) $R$ is *head-needed* if every reduction sequence of $M$ to head normal form reduces some residual of $R$.

(iii) A reduction sequence is *(head-)needed* if every reduction step in the sequence contracts a (head-)needed redex.

3.2. EXAMPLE. Consider $\lambda xy.Ix(Ky(Iy))$. Then in this term $Ix$ is needed and head-needed, $Ky(Iy)$ is needed but not head-needed and $Iy$ is neither needed nor head-needed,

A head-needed redex is automatically needed since every reduction to normal form contains a reduction to head-normal form. If there is no reduction sequence to (head-)normal form, then every redex is (head-)-needed. Each term $M$ not in normal form has at least one needed redex, the leftmost redex. The proof requires a routine argument which we omit. Similarly the head-redex of a term (if there is one) is always head-needed.

3.3. DEFINITION. (i) Consider a reduction sequence

$$\mathscr{R}: M_0 \to M_1 \to \cdots \to M_n.$$

Let $R$ be a redex in $M_0$ such that no residual of $R$ is contracted in $\mathscr{R}$ and such that $M_n$ contains no residuals of $R$. Then we say that redex $R$ is *erased* in $\mathscr{R}$.

(ii) Redex $R$ in $M$ is *erasable* if there is a reduction sequence $\mathscr{R}$ beginning with $M$ in which $R$ is erased.

The following facts, whose proofs are immediate, give a first and simple characterisation of the needed redexes in a term, and provide an easy example of needed redexes.

3.4. PROPOSITION. (i) *Let $M$ have a normal form. Then for any redex $R$ in $M$ we have $R$ is needed in $M \Leftrightarrow R$ is not erasable in $M$.*

(ii) *The leftmost redex in any term not in normal form is not erasable, hence is needed.* ∎

Note that the restriction to terms with normal form in Proposition 3.4(i) is necessary: e.g., in $M \equiv \Omega(\lambda x.I)R)$, where $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ and $R$ is some redex; $R$ is erasable but by Definition 3.1 is also needed, as $M$ does not have a normal form.

A consequence of Proposition 3.4(i) is that in the $\lambda I$-calculus, where erasure of redexes is impossible, every redex is needed.

We will now investigate how the properties of neededness and nonneededness propagate along the lines of descendants (or residuals) of a redex in $M$, in some reduction sequence of $M$. As one may expect, nonneededness is a persistent property.

3.5. THEOREM. *Let $M \twoheadrightarrow M'$. Let $S$ be redex in $M$ and $S'$ be a residual of $S$ in $M'$. Then*

(i) *$S$ is not needed $\Rightarrow$ $S'$ is not needed;*

(ii) *$S$ is not head-needed $\Rightarrow$ $S'$ is not head-needed.*

*Equivalently*

$$S' \text{ is (head-)needed} \Rightarrow S \text{ is (head-)needed.}$$

*Proof.* (i) Suppose $S$ is not needed. Then there exists a reduction sequence $\mathscr{S}: M \twoheadrightarrow N$ to normal form $N$ in which no residual of $S$ is reduced. Let $\mathscr{S}'$ be the projection of $\mathscr{S}$ over $M \twoheadrightarrow M'$. See Fig. 9. In every reduction step in $\mathscr{S}'$ a redex is contracted which is a residual of a redex contracted in $\mathscr{S}$. Since $\mathscr{S}$ reduces no residuals of $S$ it follows by Proposition 2.6 that no residual of $S'$ is contracted in $\mathscr{S}'$. Hence $S'$ is not needed.

(ii) Similarly. ∎

We now consider how (head-)neededness propagates. If $R$ is needed in $M$, and $R$ has just one residual $R'$ in $M'$ by reduction of some other redex in $M$, then it follows immediately from Definition 3.1 that $R'$ is needed. When $R$ has more than one residual, it is easy to see that it is possible that not all of them are needed. (Consider, e.g., $(\lambda x.x(KIx))R$.) However, we do expect that at least one residual is needed. The proof of this fact is not obvious. This is because if $R$ has, say, two residuals $R_1$ and $R_2$ in $M'$, one can imagine that every reduction sequence of $M'$ to normal form might reduce a residual of $R_1$ or of $R_2$, but with some of those sequences reducing only residuals of $R_1$, and others reducing only residuals of $R_2$. This does not happen. We will obtain this fact in Proposition 3.7 as an immediate consequence of another characterisation of needed redexes, which says that

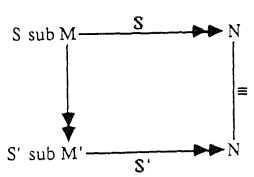


FIGURE 9

for neededness it is sufficient to look only at the leftmost (or "normal," as it is also called) reduction sequence, instead of looking at all reduction sequences of a term to normal form.

3.6. THEOREM. *Let $R$ be a redex in $M$.*

(i) *Let $\mathscr{L}: M \twoheadrightarrow N$ be a longest possible leftmost reduction starting with $M$. Then $N$ is the normal form of $M$ and*

$$R \text{ is needed} \Leftrightarrow \text{some residual of } R \text{ is contracted in } \mathscr{L}.$$

(ii) *Let $\mathscr{H}: M \twoheadrightarrow N$ be a longest possible head-reduction starting with $M$. Then $N$ is a head-normal form of $M$ and*

$$R \text{ is head-needed} \Leftrightarrow \text{some residual of } R \text{ is contracted in } \mathscr{H}.$$

*Proof.* (i) ($\Rightarrow$) By the definition of neededness. ($\Leftarrow$) Leftmost redexes are needed. If some residual of $R$ is a leftmost redex, then by Theorem 3.5, also $R$ is needed.

(ii) Similarly. ∎

This result can be reformulated as follows:

$R$ is (head-)needed iff $R$ is not erased in $\mathscr{L}(\mathscr{H})$

iff $R$ has a residual with respect to $\mathscr{L}(\mathscr{H})$ that is a leftmost redex.

In the sequel $\mathscr{G}$ will range over $\mathscr{L}$ and $\mathscr{H}$.

3.7. PROPOSITION. *Suppose $M$ has a (head-)normal form, and $R$ is a (head-)needed redex of $M$. Suppose $\mathscr{S}: M \twoheadrightarrow N$ is a reduction sequence which does not reduce any residual of $R$. Then $R$ has a (head-)needed residual in $N$.*

*Proof.* Let $\mathscr{G}: N \twoheadrightarrow N_0$ be the leftmost (head-)reduction sequence of $N$ to (head-)normal form. Because $R$ is (head-)needed in $M$, some residual $R_1$ of $R$ must be reduced in $\mathscr{S} * \mathscr{G}$. Since $\mathscr{S}$ reduces no residuals of $R$, the redex $R_1$ must descend from some residual in $N$ of $R_0$, with respect to $\mathscr{S}$. Then $R_0$ is (head-)needed in $N$, by Theorem 3.6. ∎

3.8. PROPOSITION. *If $R: M \to M'$, $Q$ is a redex of $M'$ created by $R$, and $Q$ is (head-)needed, then $R$ is (head-)needed.*

*Proof.* We will prove this for the case of head-neededness. The case of neededness is similar.

If $M$ has no head-normal form, then every redex in $M$ is (trivially) head-needed, and the theorem holds.

Otherwise, let $\mathscr{G}: M \twoheadrightarrow N$ be the head reduction path to head normal form. Let $R: M \to M'$, and let $Q$ be a redex of $M'$ created by $R$. We will prove that if $R$ is not head-needed, then neither is $Q$.

We proceed by induction on the length of $\mathscr{G}$. If $\mathscr{G}$ is the empty sequence, then $M$ is already in head normal form. Therefore $M'$ is also in head normal form. No redexes in $M'$ are head-needed, and the theorem holds.

Otherwise, $\mathscr{G} = G.\mathscr{G}'$, where $G$ is the leftmost redex of $M$. We begin by establishing some properties of the diagram $D(R, \mathscr{G})$ shown in Fig. 10.

By Theorem 3.6, $G$ is head-needed; by hypothesis $R$ is not. Therefore $R$ is not the leftmost redex of $M$. Since $R$ is not head-needed, by Theorem 3.5 none of its residuals is head-needed. Therefore every vertical reduction in Fig. 10 is nonhead-needed. By applying Lemma 2.7(i) throughout the diagram, we find that in every elementary sub-diagram, the horizontal sides each consist of a single leftmost reduction. Thus each reduction sequence $\mathscr{G}'/(R_1 \ldots R_i)$ $(0 \leqslant i \leqslant n)$ is the same length as $\mathscr{G}'$, which is shorter than $\mathscr{G}$. Since $N$ is in head normal form, so is every term occurring down the right-hand side of Fig. 10, and each sequence $\mathscr{G}'/(R_1 \ldots R_i)$ $(0 \leqslant i \leqslant n)$ is a leftmost reduction to head normal form.

Now we can show that $Q$ is not head-needed. To show this, it is enough to show that $Q$ is not reduced in $G/R$, and no residual of $Q$ is head-needed in $P'$.

$R$ creates $Q$ in $M'$, and $R$ is not leftmost, so $Q$ cannot be leftmost. Therefore $Q$ is not $G/R$.

Let $Q'$ be a residual of $Q$ in $P'$. By Lemma 2.7(ii) applied to the left-hand box of Fig. 10, $Q'$ is created by $R/G$. Let $R/G = R_1 \ldots R_n$. Then some $R_i$ creates a redex $Q''$ such that $Q'$ is a residual of $Q''$ by $R_{i+1} \ldots R_n$. We established above that $R_i$ is not a head-needed redex, and that $\mathscr{G}'/(R_1 \ldots R_{i-1})$ is a reduction to head normal form which is shorter than $\mathscr{G}$. Therefore induction applies, showing that $Q''$ is not head-needed. $Q'$ is a residual of $Q''$, so by Theorem 3.5 it is also not head-needed. ∎

A useful property is that "(head-)neededness is preserved upwards," with respect to the relation "subterm of."

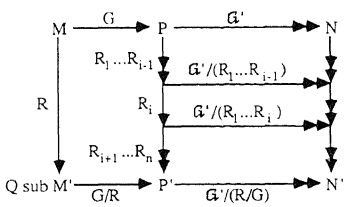


FIGURE 10

3.9. LEMMA. *Let $R$ sub $S$ sub $M$, with $R$ and $S$ redexes. Then*

$$R \text{ is (head-)needed in } M \Rightarrow S \text{ is (head-)needed in } M.$$

*Proof.* If $M$ has no (head-)normal form then this is trivial. Otherwise, let $\mathscr{G}: M \twoheadrightarrow N$ be the leftmost (head-)reduction of $M$ to its (head-)normal form. Suppose that $S$ is not (head-)needed in $M$ in order to show that $R$ is not (head-)needed in $M$. Then no residual of $S$ is contracted in $\mathscr{G}$. The redex $S$ is to the left of $R$ and the same holds for the respective residuals. It follows that no residual of $R$ is contracted in $\mathscr{G}$. Therefore $R$ is not (head-)needed by Theorem 3.5. ∎

3.10. PROPOSITION. *Let $R: M \to N$ and let $S$ be a (head-)needed redex in $M$. Suppose that $R$ is not (head-)needed. Then $S$ has a unique residual $S'$ in $N$. Moreover, $S'$ is (head-)needed.*

*Proof.* Suppose that the (head-)needed redex $S$ is multiplied by contracting $R$, then $R$ is a superredex of $S$ and therefore by Lemma 3.9 also $R$ would be (head-)needed, contradiction. Moreover, since $S$ is (head-)needed, it is different from $R$ and cannot be erased by reducing $R$. It follows that $S$ has a unique residual $S'$. This $S'$ is also (head-)needed by Proposition 3.7. ∎

3.11. LEMMA. *If $F =_\beta F'$, then*

$$R \text{ (head-)needed in } FR \Rightarrow R \text{ (head-)needed in } F'R.$$

*Proof.* By the Church–Rosser theorem one has $F \twoheadrightarrow F''$ and $F' \twoheadrightarrow F''$ for some $F''$. Then $FR \twoheadrightarrow F''R$ and $F'R \twoheadrightarrow F''R$. Suppose $R$ is (head-)needed in $FR$. Then also $R$ is (head-)needed in $F''R$ by Proposition 3.7 and hence in $F'R$ by Theorem 3.5. ∎

Intuitively, (head-)neededness is related closely to termination. Consequently, the following result comes as no great surprise.

3.12. THEOREM. *It is undecidable whether a redex in some term is (head-)needed.*

*Proof.* Scott's theorem (see Barendregt, 1984, 6.6.2) states: "Let $X$ be a set of lambda terms which is closed under conversion. Moreover, let $X$ not be the whole set of lambda terms nor be empty. Then $X$ is not recursive." Now consider the set

$$X = \{F \mid R \text{ is (head-)needed in } FR\}.$$

Then $X$ satisfies the criteria of Scott's theorem, by Lemma 3.11. Hence it is not decidable whether $R$ is (head-)needed in $FR$. From this it follows that it is in general not decidable whether a redex in some term is (head-)needed. ∎

However, just as we can determine that certain programs terminate we can hope to identify at least some (head-)needed redexes in some lambda terms. Theorem 3.12 does not say we cannot do anything; it just tells us that perfection is not possible.

We will now proceed to give a surprisingly simple characterisation of the (head-)needed redexes in $M$ in terms of their behaviour with respect to the leftmost (head-)reduction sequence of $M$ to its (head) normal form. The following definition and treatment are suggested by the analogous treatment in Huet and Lévy (1979).

3.13. DEFINITION. (i) Let $M$ be a term. Then the *norm* of $M$, notation $\|M\|$, is the length (in number of reduction steps) of the leftmost reduction sequence of $M$ to normal form if this exists, and "infinite" otherwise.

(ii) Similarly, the *head-norm* of $M$, notation $\|M\|^{\mathrm{h}}$, is the length of the head-reduction sequance of $M$ to head normal form, if this exists, and "infinite" otherwise.

Note that $\|M\|^{\mathrm{h}} \leqslant \|M\|$.

3.14. *Notation.* $\to_{\mathrm{n}}$ denotes the contraction of a needed redex and $\to_{\neg \mathrm{n}}$ the contraction of a nonneeded redex. Similarly $\to_{\mathrm{hn}}$ denotes the contraction of a head-needed redex and $\to_{\neg \mathrm{hn}}$ of a redex that is not head-needed. For each such reduction relation $\to_{\mathrm{x}}$, its transitive reflexive closure is denoted by $\twoheadrightarrow_{\mathrm{x}}$.

3.15. THEOREM. (i) *Let $M$ have normal form. Then*

$$M \xrightarrow[\mathrm{n}]{} \!\!\!\!\twoheadrightarrow N \Rightarrow \|M\| > \|N\|;$$

$$M \xrightarrow[\neg \mathrm{n}]{} \!\!\!\!\twoheadrightarrow N \Rightarrow \|M\| = \|N\|.$$

(ii) *Let $M$ have a head-normal form. Then*

$$M \xrightarrow[\mathrm{hn}]{} \!\!\!\!\twoheadrightarrow N \Rightarrow \|M\|^{\mathrm{h}} > \|N\|^{\mathrm{h}};$$

$$M \xrightarrow[\neg \mathrm{hn}]{} \!\!\!\!\twoheadrightarrow N \Rightarrow \|M\|^{\mathrm{h}} = \|N\|^{\mathrm{h}}.$$

*That is, (head-)needed reduction is (head-)norm-decreasing; non-(head-)needed reduction is (head-)norm-preserving.*

*Proof.* (i) Let $R$ sub $M$ be arbitrary and consider the leftmost reduction sequence $\mathscr{L}$: $M \equiv M_0 \twoheadrightarrow M_k$ to normal form $M_k$. Let $R$: $M \to N \equiv N_0$. By Proposition 2.8 we can erect the diagram in Fig. 11, where $\mathscr{L}'$ is again leftmost. Hence in any case $\|M\| \geqslant \|N\|$.

If $R$ is needed, at least one of its residuals is contracted in $\mathscr{L}$. Say this is in the step $M_i \twoheadrightarrow_{\mathrm{lm}} M_{i+1}$. Since $M_i \twoheadrightarrow N_i$ contracts all the residuals of $R$, it also contracts the leftmost redex in $M_i$. Therefore by Proposition 2.8 the reduction sequence $N_i \to_{\mathrm{lm}\equiv} N_{i+1}$ is the empty step. Hence $\|M\| > \|N\|$.

If $R$ is not needed, then by Theorem 3.6 no residual of $R$ is contracted in $\mathscr{L}$. Therefore again by Proposition 2.9 each step $M_i \to_{\mathrm{lm}} M_{i+1}$ in $\mathscr{L}$ gives exactly one leftmost step $N_i \to_{\mathrm{lm}} N_{i+1}$ in $\mathscr{L}'$. Thus $\mathscr{L}$ and $\mathscr{L}'$ have exactly the same length, and $\|M\| = \|N\|$.

(ii) Similarly. ∎

The next theorem collects all our equivalent characterisations of (head-)neededness.

3.16. THEOREM (equivalent characterisations of (head-)neededness). (i) *Let $M$ have normal form $N$, and let $R$ in $M$ be a redex. Let $\mathscr{L}$ be the leftmost reduction sequence from $M$ to $N$. Then*

$$R \text{ is needed in } M \Leftrightarrow R \text{ is not erasable in } M$$
$$\Leftrightarrow R \text{ has a residual contracted in } \mathscr{L}$$
$$\Leftrightarrow R \text{ is not erased in } \mathscr{L}$$
$$\Leftrightarrow R \text{ is norm-decreasing.}$$

(ii) *Let $M$ have a head-normal form, and let $R$ in $M$ be a redex. Let $\mathscr{H}$ be the maximal leftmost head-reduction sequence starting from $M$. Then*

$$R \text{ is head-needed in } M \Leftrightarrow R \text{ has a residual contracted in } \mathscr{H}$$
$$\Leftrightarrow R \text{ is head-norm-decreasing.}$$ ∎

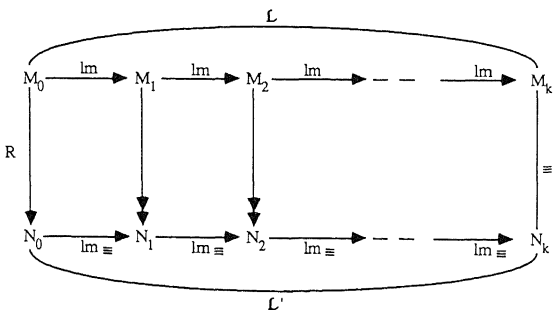


FIGURE 11

3.17. PROPOSITION. *Let $M$ have a (head-)normal form. Then the leftmost (head-)reduction sequence of $M$ has maximal length among the (head-)needed reduction sequences to (head-)normal form. (There may of course be longer sequences, but these include redexes that are not (head-)needed.)*

*Proof.* Every (head-)needed reduction reduces the (head-)norm by at least one. Therefore a (head-)needed reduction sequence from $M$ to (head-)normal form can have length no more than $\|M\|^{(h)}$. But $\|M\|^{(h)}$ is the length of the leftmost (head-)reduction sequence to (head-)normal form. ∎

This proposition implies that in the $\lambda I$-calculus the leftmost reduction sequence of $M$ has maximal length among all reduction sequences. This implies the well-known fact for the $\lambda I$-calculus, that if $M$ has a normal form, then all reductions starting with $M$ terminate.

3.18. LEMMA. (i) *Let $R: M \to N$ and $S: N \to L$, where $R$ is nonneeded and $S$ is needed. Then there exist a term $N'$, a needed redex $S': M \to N'$, and a sequence $N' \twoheadrightarrow L$ of nonneeded reduction steps. See Fig.* 12.

(ii) *An analogous statement holds for head-neededness.*

*Proof.* (i) By Proposition 3.8 nonneeded redexes never create needed redexes, so $S$ must be a residual by $R$ of some redex $S'$ in $M$. By Proposition 3.10, $S$ must be the only residual of $S'$. Therefore we can make the above reduction diagram. Hence by Theorem 3.5, the redexes reduced in $N' \twoheadrightarrow L$ are nonneeded. That $S'$ is needed follows also by 3.5.

(ii) Similarly. ∎

3.19. THEOREM. *Let $\mathscr{R}: M \twoheadrightarrow N$ be a reduction sequence. Then there are sequences $\mathscr{S}: M \twoheadrightarrow L$ and $\mathscr{T}: L \twoheadrightarrow N$ such that $\mathscr{S}$ is needed, $\mathscr{T}$ is nonneeded, and $\mathscr{R} = \mathscr{S} * \mathscr{T}$. ("Nonneeded reductions can be postponed.") Similarly for nonhead-needed reductions.*

*Proof.* By Lemma 3.18 using some "diagram chasing." ∎

The word "needed" refers to the fact that, by definition, some residual of the needed redex must be contracted in order to each normal form. Now

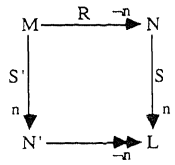


FIGURE 12

we will show that reduction of needed redexes is not only necessary, but also sufficient to reach the normal form. More generally, we will show that if an arbitrary finite number of nonneeded steps is allowed between needed steps, the resulting reduction sequence is still sufficient to reach normal form. That is, "quasi-needed reduction is normalising."

3.20. DEFINITION. Consider a (finite or infinite) reduction sequence

$$\mathscr{R} = M_0 \to M_1 \to M_2 \to \ldots$$

with $R_i: M_i \to M_{i+1}$.

(i) $\mathscr{R}$ is called a *quasi-needed* reduction sequence if

$$\forall i. \exists j \geqslant i. R_j \text{ is needed in } M_j.$$

(ii) Similarly we define *quasi-head-needed.*

So the quasi-needed reduction discipline has the nice property that one is free to perform, between needed reduction steps, an arbitrary finite reduction sequence.

3.21. THEOREM. (i) *Let $M$ have a normal form. Then every quasi-needed reduction sequence starting with $M$ terminates.*

(ii) *Similarly, if $M$ has a head-normal form, then every quasi-head-needed reduction starting with $M$ terminates.*

*Proof.* (i) By Theorem 3.15, needed reductions are norm-decreasing, while nonneeded reductions are norm-preserving. Hence a quasi-needed reduction sequence starting from a term with a finite norm (i.e., having a normal form), must end in a term with norm 0 (i.e., a normal form).

(ii) Similarly. ∎

It follows that if a term has a (head)-normal form, then a quasi-(head-)-needed reduction is able to find it (one).

## 4. SPINE STRATEGIES

As shown in 3.12, neededness of a redex $R$ in $M$ is undecidable in general. In practical cases we usually work with terms having a (head) normal form. In these cases we can decide whether $R$ is (head) needed: reduce $M$ by the leftmost reduction path $L$ to (head) normal form; if (a residual of) $R$ is reduced in $L$, then $R$ is (head-)needed, otherwise not. (A leftmost reduction to head normal form is a head reduction.) This is, however, not a practical algorithm: it uses unpredictably long look-ahead.

Practical algorithms for identifying needed redexes should be efficient: the number of steps required should be bounded by some linear function of the size of a term. This motivates the various notions of *spine redex* introduced below. These come in two groups: various notions of *head-spine redex* and *spine redex*. These are generalisations of the notions of head redex and leftmost redex, respectively. The redexes belonging to these families are all needed. Moreover, we will give efficient algorithms to test whether a redex in a term belongs to one of the classes.

4.1. DEFINITION. The set $\mathrm{HS}(M)$ of *head spine redexes* in a lambda-expression $M$ is defined as

$$\mathrm{HS}(M) = \varnothing \qquad \text{if } M \text{ is in head normal form}$$

$$\mathrm{HS}(M) = \{(\lambda y.P)Q\} \cup \mathrm{HS}(P) \qquad \text{if } M \equiv \lambda x_1 .. x_n .((\lambda y.P)Q)R_1 ... R_m, \text{ for some } n, m \geqslant 0.$$

We will see that there is an efficient algorithm for identifying the head spine redexes, by computing the *head spine* of the given term.

4.2. DEFINITION. For a lambda-expression $M$ we define $\mathrm{hs}(M)$; this will be the same term with some underlining:

$$\mathrm{hs}(x) = \underline{x}$$

$$\mathrm{hs}(\lambda x.P) = \underline{\lambda x}.\mathrm{hs}(P)$$

$$\mathrm{hs}(PQ) = \mathrm{hs}(P)Q$$

It is easy to see that a redex $R$ in $M$ is a head spine redex iff the $\lambda$ of $R$ is underlined in $\mathrm{hs}(M)$.

4.3. DEFINITION. Every lambda-expression can be written in the form

$$M \equiv \underline{\lambda \mathbf{x}_0.(\lambda \mathbf{x}_1.(...(\lambda \mathbf{x}_n.y}\mathbf{P}_{n+1})\mathbf{P}_n)...)\mathbf{P}_1,$$

where $n \geqslant 0$, the $\mathbf{x}_0, ..., \mathbf{x}_n, \mathbf{P}_1, ..., \mathbf{P}_{n+1}$ are vectors, i.e., $\mathbf{x}_0 = x_{01}, x_{02}, ...$, etc. Note that such vectors are not subterms, but lists of subterms. The vectors $\mathbf{x}_0$ and $\mathbf{P}_{n+1}$ may be empty, but the remaining $\mathbf{x}_i$ and $\mathbf{P}_i$ are nonempty. Note that with more parentheses we have

$$M \equiv (\underline{\lambda \mathbf{x}_0.((\lambda \mathbf{x}_1.((...((\lambda \mathbf{x}_n.(y}\mathbf{P}_{n+1}))\mathbf{P}_n))...))\mathbf{P}_1).$$

The *head spine* corresponds to the underlined portion of $M$. The variable $y$ is called the *head variable* of $M$. By analogy with the notion of "spine," the terms $P_{ij}$ are called the *ribs* of $M$.

EXAMPLE. A term with $n=2$ looks like

$$\underline{\lambda \mathbf{x}_0.(\lambda \mathbf{x}_1.(\lambda \mathbf{x}_2.\, y\mathbf{P}_3)\mathbf{P}_2)\mathbf{P}_1}$$

and when the vectors are written out, e.g., like

$$\underline{\lambda a_1\, a_2.(\lambda b_1 b_2 b_3.(\lambda c_1 c_2.\, y D_1 D_2)C_1)B_1 B_2}.$$

In tree notation the term looks like Fig. 13. The thick lines pick out the head-spine.

Looking at a term $M$ in this way makes it clear that its leftmost reduction sequence begins by reducing all the head spine redexes, from the outermost inwards. If in a leftmost reduction some rib $P_{ij}$ is substituted for the head variable of $M$, the head spine of $M$ is extended by the head spine of $P_{ij}$, and the head spine redexes of $P_{ij}$ will have residuals on the spine of the resulting expression. It follows from the contrapositive formulation of Theorem 3.6 that all the head spine redexes of $P_{ij}$ are needed in $M$. These observations provide a basis for a better approximation to neededness than that offered by head spine redexes.

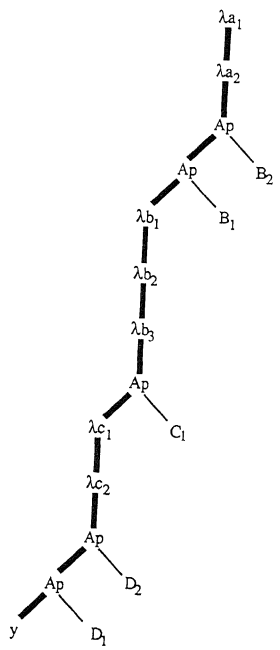


FIGURE 13

4.4. DEFINITION. (i) Let $M$ be as above and let $\mathbf{P}_i = P_{i1}.P_{i2}...$ and $\mathbf{x}_i = x_{i1}, x_{i2}....$ Then $M$ has the *head spine target* $P_{i1}$ if $y \equiv x_{i1}$ with $i > 0$. This subterm will be substituted for the head spine variable when normalising $M$.

(ii) $M$ has the *polyadic head spine target* $P_{ij}$ if $y \equiv x_{ij}$ with $i > 0$ and $P_{ij}$ exists (i.e., $\mathbf{P}_i$ has at least $j$ elements).

4.5. EXAMPLE. Consider $M \equiv (\lambda x_1 x_2 x_3.((\lambda y_1 y_2.((\lambda z_1 z_2 z_3 z_4.yW_1W_2)Z_1))Y_1Y_2Y_3))$; then

$y \equiv y_1$ $\Rightarrow Y_1$ is the (polyadic) head spine target;

$y \equiv y_2$ $\Rightarrow Y_2$ is the polyadic head spine target;

$y \equiv z_1$ $\Rightarrow Z_1$ is the (polyadic) head spine target;

$y \equiv z_2$ $\Rightarrow$ there is no (polyadic) head spine target;

$y$ is free $\Rightarrow$ there is no (polyadic) head spine target.

4.6. DEFINITION. Let $M$ be a term with head spine target $N$. An *extended head spine redex* of $M$ is a head spine redex of $M$ or an extended head spine redex of $N$.

Let $M$ be a term with polyadic head spine target $N$. A *polyadic head spine redex* of $M$ is a head spine redex of $M$ or a simple polyadic head spine redex of $N$.

Recall that the active components of $M$ are the maximal subterms of $M$ not in head normal-form. For example, if $M \equiv \lambda\mathbf{x}.(\lambda y.P)Q\mathbf{R}$, then $M$ is the only active component of itself. If $M \equiv \lambda\mathbf{x}.yR_1...R_m$, then the active components of $M$ are those of $R_1, ..., R_m$ together.

4.7. DEFINITION. (i) A *spine redex* of $M$ is a head spine redex of an active component of $M$.

(ii) Similarly, *extended* or *polyadic spine redexes* are respectively the extended or polyadic head-spine redexes of an active component in $M$.

4.8. LEMMA. *Let $A$ sub $M$ be an active component. Then*

$$R \text{ is head-needed in } A \Rightarrow R \text{ is needed in } M.$$

*Proof.* Induction on the length of $M$. If $M$ is not in head-normal form, then $A \equiv M$ and the statement is trivial. Otherwise, $M \equiv \lambda x_1...x_n.yA_1...A_m$ and $A$ is an active component of say $A_i$. By the induction hypothesis $R$ is needed in $A_i$, hence in $M$. ∎

4.9. THEOREM. (i) *Head spine, extended head spine, and polyadic head spine redexes are all head-needed.*

(ii) *Spine, extended spine, and polyadic spine redexes are all needed.*

*Proof.* (i) From the definition of head spine redexes it is clear that these will be contracted on the head reduction path. Hence by 3.6(ii) such redexes are head-needed. From the definition of extended or polyadic head spine redexes it also is clear that these will become head redexes; therefore they are also needed.

(ii) By (i) and Lemma 4.8. ∎

Figure 14 summarises the relations between the various classes of redexes.

Now we will turn to the algorithms that detect the various classes of needed redex. First we give a noncomputable version in order to make the idea clear.

4.10. DEFINITION. The *selection number* of a $\lambda$-term $M$, notation $\mathrm{Sel}(M)$, is defined as

$$\begin{aligned}
\mathrm{Sel}(M) &= \uparrow(\text{undefined}), && \text{if } M \text{ has no head-normal form;}\\
&= 0, && \text{if } M \text{ has a head-normal form with a free head variable;}\\
&= i, && \text{if } M \text{ has a head-normal form } \lambda x_1 \ldots x_n.x_i M_1 \ldots M_m,\ i:1..n.
\end{aligned}$$

Clearly Sel is a partial recursive function on (the codes of) terms.

The selection number is related to the notion of head-neededness. In the following definition $\Lambda\bot$ stands for the set of lambda terms extended with a new constant $\bot$.

| | | |
|---|---|---|
| head spine | $\subseteq$ | spine |
| $\cap$ | | $\cap$ |
| extended head spine | $\subseteq$ | extended spine |
| $\cap$ | | $\cap$ |
| polyadic head spine | $\subseteq$ | polyadic spine |
| $\cap$ | | $\cap$ |
| head-needed | $\subseteq$ | needed |

FIGURE 14

4.11. DEFINITION. The map $\langle\ \rangle: \Lambda \to \Lambda\bot$ is defined as

$$\begin{aligned}
\langle x \rangle &= x; \\
\langle \lambda x.P \rangle &= \lambda x.\langle P \rangle; \\
\langle PQ \rangle &= \langle P \rangle \langle Q \rangle, \qquad \text{if } \mathrm{Sel}(P) = 1, \\
&= \langle P \rangle \bot, \qquad \text{otherwise.}
\end{aligned}$$

4.12. EXAMPLE. Let

$$\begin{aligned}
M_1 &\equiv \lambda w.(\lambda xy.yAB)((\lambda z.w)B); \\
M_2 &\equiv \lambda w.(\lambda xy.xAB)((\lambda z.w)B).
\end{aligned}$$

Then

$$\begin{aligned}
\langle M_1 \rangle &\equiv \lambda w.(\lambda xy.y\bot\bot)\bot; \\
\langle M_2 \rangle &\equiv \lambda w.(\lambda xy.x\bot\bot)((\lambda z.w)\bot).
\end{aligned}$$

4.13. DEFINITION. Let $R$ be a redex in $M$.

(i) $R$ is called $\langle\ \rangle$-*preserved* if the abstractor of $R$ is still present in $\langle M \rangle$.

(ii) $R$ is called $\langle\ \rangle$-*needed* if $R$ is $\langle\ \rangle$-*preserved* in an active component of $M$.

4.14. PROPOSITION. (i) *$R$ is $\langle\ \rangle$-preserved $\Rightarrow$ $R$ is head-needed.*

(ii) *$R$ is $\langle\ \rangle$-needed $\Rightarrow$ $R$ is needed.*

*Proof.* (i) By induction on the term $M$ of which $R$ is a subterm. If $M \equiv x$, then the result is trivial. If $M \equiv \lambda x.P$, then the result follows from the induction hypothesis. If $M \equiv PQ$, then there are three subcases:

$R \equiv PQ$. Then $R$ is head needed.
$R$ sub $P$. Then

$R\langle\ \rangle$-preserved $\Rightarrow$ $R$ visible in $\langle M \rangle$

$\Rightarrow$ $R$ head-needed in $P$, by the induction hypothesis

$\Rightarrow$ $R$ head-needed in $M$.

$R$ sub $Q$. Then

$$R\langle\ \rangle\text{-preserved} \Rightarrow R \text{ visible in } \langle M\rangle \equiv \langle P\rangle\langle Q\rangle \ \&\ \mathrm{Sel}(P)=1$$

$$\Rightarrow R \text{ visible in } \langle Q\rangle \ \&\ \mathrm{Sel}(P)=1$$

$$\Rightarrow R \text{ head-needed in } Q \ \&\ P \twoheadrightarrow_h \lambda x_1 \ldots x_n . x_1 N_1 \ldots N_k$$

$$\Rightarrow R \text{ head-needed in } PQ \equiv M.$$

(ii) By (i) and Lemma 4.8. ∎

The converse does not hold; $R$ in $(\lambda x.xR(\lambda y.y))(\lambda pq.p)$ is head-needed, but not $\langle\ \rangle$-preserved.

Now we will define several computable approximations to Sel: $\mathrm{Sel}_{1\ldots4}$. These partial recursive functions are computable in the sense that their domains are recursive. The definition of $\mathrm{Sel}_i$ is simultaneous with that of $\langle\ \rangle_i$.

In the following definition $\uparrow$ denotes "undefined" and $a \dot{-} b$ is $a-b$ if this is not negative and 0 otherwise.

4.15. DEFINITION. (i) $\langle\ \rangle_i$ is defined by replacing Sel in the definition of $\langle\ \rangle$ by $\mathrm{Sel}_i$.

(ii) $\mathrm{Sel}_1(P) = \uparrow$, for all $P$.

(iii)
$$\begin{aligned}
\mathrm{Sel}_2(x) &= 0;\\
\mathrm{Sel}_2(PQ) &= \uparrow;\\
\mathrm{Sel}_2(\lambda x.P) &= 1, && \text{if } x \in FV(\langle P\rangle_2),\\
&= \uparrow, && \text{otherwise.}
\end{aligned}$$

(iv)
$$\begin{aligned}
\mathrm{Sel}_3(x) &= 0;\\
\mathrm{Sel}_3(PQ) &= \mathrm{Sel}_3(P) \dot{-} 1, && \text{if } \mathrm{Sel}_3(P) \neq 1,\\
&= \uparrow, && \text{otherwise;}\\
\mathrm{Sel}_3(\lambda x.P) &= 1, && \text{if } x \in FV(\langle P\rangle_3),\\
&= \mathrm{Sel}_3(P)+1, && \text{if } x \notin FV(\langle P\rangle_3)\\
& && \quad \text{and } \mathrm{Sel}_3(P) > 0,\\
&= 0, && \text{otherwise.}
\end{aligned}$$

(v)
$$\begin{aligned}
\mathrm{Sel}_4(x) &= 0;\\
\mathrm{Sel}_4(PQ) &= \mathrm{Sel}_4(P) \dot{-} 1, && \text{if } \mathrm{Sel}_4(P) \neq 1,\\
&= \mathrm{Sel}_4(Q) - \mathrm{lengthtail}(P), && \text{if } \mathrm{Sel}_4(P) = 1\\
& && \quad \text{and } \mathrm{Sel}_4(Q) > \mathrm{lengthtail}(P)\\
&= \uparrow, && \text{otherwise;}\\
\mathrm{Sel}_4(\lambda x.P) &= 1, && \text{if } x \in FV(\langle P\rangle_4),\\
&= \mathrm{Sel}_4(P)+1, && \text{if } x \notin FV(\langle P\rangle_4)\\
& && \quad \text{and } \mathrm{Sel}_4(P) > 0,\\
&= 0, && \text{otherwise.}
\end{aligned}$$

Here lengthtail($P$) is defined by:

> if $P$ has as head-normal form $\lambda x_1 \ldots x_m . yQ_1 \ldots Q_n$ (with $n, m \geqslant 0$), then lengthtail($P$) $= n$, otherwise $\uparrow$

Moreover, we have the property

$$\mathrm{Sel}_4(P) = 1 \Rightarrow \text{lengthtail}(P) \text{ is defined.}$$

It follows that $\mathrm{Sel}_4$ is computable, even although its definition uses the uncomputable—more precisely, not always computable—function lengthtail. We will not prove this fact here as it will follow from a more precise analysis later on. See Definition 4.22 below.

4.16. DEFINITION. Let $R$ be a redex in $M$.

(i) $R$ is $\langle\ \rangle_i$-*preserved* if $R$ is visible in $\langle M \rangle_i$.

(ii) $R$ is $\langle\ \rangle_i$-*needed* if $R$ is $\langle\ \rangle_i$-preserved in an active component of $M$.

4.17. DEFINITION. Let $R$ be a redex in $M$.

(i) $R$ is called a *generalised head spine redex* if $R$ is $\langle\ \rangle_4$-preserved.

(ii) $R$ is a *generalised spine redex* if $R$ is a generalised head spine redex of an active component of $M$.

It is clear that for the partial functions $\mathrm{Sel}_i$ we have

$$\mathrm{Sel} \supseteq \mathrm{Sel}_4 \supseteq \mathrm{Sel}_3 \supseteq \mathrm{Sel}_2 \supseteq \mathrm{Sel}_1,$$

i.e., the $\mathrm{Sel}_i$ are successively better approximations of Sel.

In the next proposition $\supseteq$ denotes Böhm-tree inclusion of $\lambda\perp$-terms, that is $M \supseteq N$ iff $M$ results by replacing some occurrences of $\perp$ by arbitrary $\lambda\perp$-terms. E.g., $\lambda x.xy \supseteq \lambda x.x\perp$.

4.18. PROPOSITION. *For all terms $M$ we have*

$$\langle M \rangle \supseteq \langle M \rangle_4 \supseteq \langle M \rangle_3 \supseteq \langle M \rangle_2 \supseteq \langle M \rangle_1.$$

*Proof.* By the previous remark. ∎

EXAMPLE. Let $M \equiv I(K_* I((\lambda x.xI)\, K_*(II)))(\omega I(II))$, where $K_* \equiv \lambda xy.y$ and $\omega \equiv \lambda x.xx$. Then

$$\langle M\rangle_1 \equiv I\bot\bot;$$

$$\langle M\rangle_2 \equiv I(K_*\bot\bot)\bot;$$

$$\langle M\rangle_3 \equiv I(K_*\bot((\lambda x.x\bot)K_*\bot))\bot;$$

$$\langle M\rangle_4 \equiv I(K_*\bot((\lambda x.x\bot)K_*(II)))((\lambda x.x\bot)I\bot);$$

$$\langle M\rangle \equiv I(K_*\bot((\lambda x.x\bot)K_*(II)))((\lambda x.x\bot)I(II)).$$

4.19. PROPOSITION. (i) *$R\langle\ \rangle_i$-preserved $\Rightarrow$ $R$ head-needed.*

(ii) *$R\langle\ \rangle_i$-needed $\Rightarrow$ $R$ needed.*

*Proof.* (i) By Proposition 4.17 it follows that if $R$ is visible in $\langle M\rangle_i$, then also in $\langle M\rangle$ and therefore $\langle\ \rangle$-preserved. Hence by Proposition 4.14 we are done.

(ii) By (i) and Lemma 4.8. ∎

4.20. PROPOSITION. (i) *$R$ is a head-spine redex $\Leftrightarrow$ $R$ is $\langle\ \rangle_1$-preserved.*

(ii) *$R$ is a spine redex $\Leftrightarrow$ $R$ is $\langle\ \rangle_1$-needed.*

(iii) *$R$ is an extended head spine redex $\Rightarrow$ $R$ is $\langle\ \rangle_2$-preserved.*

(iv) *$R$ is an extended spine redex $\Rightarrow$ $R$ is $\langle\ \rangle_2$-needed.*

(v) *$R$ is a polyadic head spine redex $\Rightarrow$ $R$ is $\langle\ \rangle_3$-preserved.*

(vi) *$R$ is a polyadic spine redex $\Rightarrow$ $R$ is $\langle\ \rangle_3$-needed.*

*Proof.* For the statements including "head-" this follows by induction on the structure of $M$ in which $R$ occurs. The case distinctions are best made according to the shape of $M$ displayed in Definition 4.3. As a typical example let us show (v) with $M \equiv ((\lambda x_1 x_2.(\lambda y_1\, y_2.x_2\mathbf{Z})\mathbf{Y})X_1)X_2$ and let $R$ sub $M$ in fact be sub $X_2$. Then

$R$ is polyadic head spine redex of $M$

$\Rightarrow$ $R$ polyadic head-needed in $X_2$

$\Rightarrow$ $R$ visible in $\langle X_2\rangle_3$, by the induction hypothesis

$\Rightarrow$ $R$ visible in $\langle M\rangle_3 \equiv ((\lambda x_1 x_2.(\lambda y_1\, y_2.x_2\bot)\bot)\bot)\langle X_2\rangle_3$,

since $\mathrm{Sel}_3((\lambda x_1 x_2.(\lambda y_1\, y_2.x_2\mathbf{Z})\mathbf{Y})X_1) = \mathrm{Sel}_3((\lambda y_1\, y_2.x_2\mathbf{Z})\mathbf{Y}) + 1 + 1 - 1 = 1$.

For the statements without "head-," the validity follows from Lemma 4.8. ∎

The reverse implications in (iii) and (v) do not hold. Consider, e.g., $M \equiv (\lambda x.(\lambda y.yA)x)R$. Then $R$ is not an extended (nor polyadic) head spine redex, although $R$ is visible in $\langle M\rangle_2$.

As to the length of the different spine reductions, we can state the following simple observation.

4.21. PROPOSITION. *All spine reduction sequences of a given term to normal form have the same length.*

*Proof.* Note that if $R_0$ and $R_1$ are two different spine redexes in $M$, then $R_0$ and $R_1$ can neither multiply nor erase each other. Hence we have the elementary reduction diagram of Fig. 15. Now the statement follows by a simple diagram chase. ∎

For extended and polyadic spine reductions this is not true, since, for example, a polyadic spine redex may be duplicated by a spine redex to its left.

The corresponding theorem for head reductions would be trivial, since there is only at most one head redex in any term.

*An Algorithm for Detecting Generalised (Head) Spine Redexes*

The definition of $\mathrm{Sel}_4$ contains an unsatisfactory element, namely the appeal to lengthtail($M$), for which the head-normal form of $M$ must be determined. It would be better to have a more explicit algorithm to determine lengthtail($M$). Such an algorithm is given by the following definition. The operation $L$ gives what was called above lengthtail, in those (computable) cases where $\mathrm{Sel}_4$ needs it. $K$ is an auxiliary function; see Theorem 4.31 below. $K(M)$, $\mathrm{Sel}_4(M)$, $L(M)$ are defined simultaneously; therefore it is convenient to work with triples $(K(M), \mathrm{Sel}_4(M), L(M))$, abbreviated as $KSL(M)$ and varying over $\mathbb{N}^3 \cup \{(*, *, *)\}$. The operation $+$ on this set works coordinatewise with the understanding that $n + * = *$.

4.22. DEFINITION. (i) $KSL(\bot) = (*, *, *)$

(ii) $KSL(x) = (0, 0, 0)$

(iii) $KSL(\lambda x.P) = KSL(P) + (1, 1, 0)$ if $x \in FV(\langle P \rangle_4)$ or $FV(\langle P \rangle_4) = \varnothing$, $KSL(\lambda x.P) = KSL(P) + (1, 0, 0)$ otherwise.

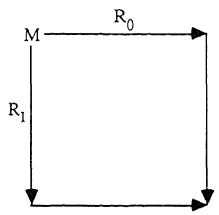


FIGURE 15

(iv) $KSL(PQ) = KSL(P) \oplus KSL(Q)$, where $\oplus$ is recursively defined by

$$
\begin{array}{lll}
(1) & (*, *, *) \oplus (x, y, z) & = (*, *, *) \\
(2) & (0, 0, j) \oplus (x, y, z) & = (0, 0, j+1) \\
(3) & (k+1, 0, j) \oplus (x, y, z) & = (k, 0, j) \\
(4) & (k+1, 1, j) \oplus (*, *, *) & = (*, *, *) \\
(5) & (k+1, 1, j) \oplus (0, 0, j') & = (k, 0, j+j') \\
(6) & (k+1, n+2, j) \oplus (x, y, z) & = (k, n+1, j) \\
(7) & (k+1, 1, 0) \oplus (k'+1, 0, j') & = (k+k'+1, 0, j') \\
(8) & (k+1, 1, j+1) \oplus (k'+1, 0, j') & = (k+1, 1, j) \oplus (k', 0, j') \\
(9) & (k+1, 1, 0) \oplus (k'+1, 1, j') & = (k+k'+1, k+1, j') \\
(10) & (k+1, 1, j+1) \oplus (k'+1, 1, j') & = (*, *, *) \\
(11) & (k+1, 1, 0) \oplus (k'+1, n'+2, j') & = (k+k'+1, k+n'+2, j') \\
(12) & (k+1, 1, j+1) \oplus (k'+1, n'+2, j') & = (k+1, 1, j) \oplus (k', n'+1, j').
\end{array}
$$

As to the intuition for $KSL(M)$, the following will be proved:

$$KSL(M) = (k, s, j) \Rightarrow M \text{ has a head-normal form of the form } \lambda z_1 \ldots z_k . z_s N_1 \ldots N_j.$$

The reverse implication does not hold; $M = (\lambda x.xI)I$ has a head-normal form $I$, but

$$
\begin{aligned}
KSL(M) &= KSL(\lambda x.xI) \oplus KSL(I) = (KSL(xI) + (1, 1, 0)) \oplus KSL(I) \\
&= ((KSL(x) \oplus KSL(I)) + (1, 1, 0)) \oplus KSL(I) \\
&= (((0, 0, 0) \oplus (1, 1, 0)) + (1, 1, 0)) \oplus (1, 1, 0) \\
&= ((0, 0, 1) + (1, 1, 0)) \oplus (1, 1, 0) \\
&= (1, 1, 1) \oplus (1, 1, 0) \\
&= (*, *, *).
\end{aligned}
$$

The reason is that the computation of the head-normal form of $M$ uses the underlined subterm in $(\lambda x.x\underline{I})I$ whereas the definition of $\langle\ \rangle_4$ (for which $KSL$ is a subroutine) is such that every vector $xP_1 \ldots P_j$ is replaced by $x\perp \ldots \perp$ ($j$ times $\perp$; this is abbreviated as $x\perp^j$). One can formulate a restricted $\lambda$-calculus embodying these restrictions (namely that no information is visible of a vector $xP_1 \ldots P_j$ except the head variable and the length $j$ of the tail) in the calculation of $KSL$ and therefore of $\langle\ \rangle_4$, and obtain a precise characterisation of when $KSL(M) = (*, *, *)$ as follows.

4.23. DEFINITION. $\lambda\bot$-calculus has as terms the set $\Lambda\bot$ and as rules

(i) $\bot M \to \bot$,

(ii) $\lambda x.\bot \to \bot$,

(iii) $xP_1 \ldots P_j \to x\bot^j$, $j \geqslant 0$,

(iv) $(\lambda x_1 \ldots x_k . x_1 \bot^j)P \to \lambda x_2 \ldots x_k . P\bot^j$, if $k \geqslant 1$, $j \geqslant 0$,

(v) $(\lambda x_1 \ldots x_k . x_n \bot^j)P \to \lambda x_2 \ldots x_k . x_n \bot^j$, if $k \geqslant 1$, $j \geqslant 0$, $n \neq 1$.

Note again that the $\bot^j$ are not subterms.

These rules generate a reduction relation $\to$, a reflexive-transitive closure $\twoheadrightarrow$, and an equality $=$, just as for ordinary $\lambda$-calculus. When it is necessary to distinguish these relations from those of $\lambda$-calculus, we write $\lambda\bot \vdash M \to N$, $\lambda\bot \vdash M \twoheadrightarrow N$, and $\lambda\bot \vdash M = N$.

EXAMPLE. In $\lambda\bot$: $(\lambda x.xx)(\lambda x.xx) \twoheadrightarrow (\lambda x.x\bot)(\lambda x.x\bot) \twoheadrightarrow (\lambda x.x\bot)\bot \to \bot\bot \to \bot$.

4.24. PROPOSITION. *$\lambda\bot$-calculus is terminating and Church–Rosser. The normal forms are $\bot$ and $\lambda x_1 \ldots x_k . x_n \bot^j$.*

*Proof.* Every reduction decreases the length of a term, hence the system is terminating. The Church–Rosser property follows via Newman's lemma (see Proposition 3.1.25 of Barendregt, 1984), since the system is easily proved to be weakly Church–Rosser. ∎

4.25. LEMMA. (i) $KSL(\bot) = (*, *, *)$.

(ii) $KSL(\lambda x_1 \ldots x_k . x_n \bot^j) = (k, n, j)$, *if* $1 \leqslant n \leqslant k$.

(iii) $KSL(\lambda x_1 \ldots x_k . x\bot^j) = (k, 0, j)$, *if* $x \notin \{x_1, \ldots, x_k\}$.

*Proof.* By the definition. ∎

4.26. LEMMA. *KSL is substitutive. That is, if $KSL(E) = KSL(F)$, then for any $G$, $KSL(G[y := E]) = KSL(G[y := F])$ (where by the usual variable convention the substitution automatically renames variables in $E$ to avoid captures).*

*Proof.* By induction on the structure of $G$.

(i) $G \equiv x$. Trivial

(ii) $G \equiv \lambda x.P$. Write $G^E \equiv G[y := E]$. Now

$$\begin{aligned} KSL(G^E) &= KSL(\lambda x.P^E) + (1, 1, 0) \text{ or } (1, 0, 0) \\ &= KSL(P^E) + (1, 1, 0) \text{ or } (1, 0, 0) \\ &= KSL(P^F) + (1, 1, 0) \text{ or } (1, 0, 0) \quad \text{(by the induction hypothesis)} \\ &= KSL(G^F), \end{aligned}$$

since $x \in FV(\langle P^E \rangle_4 \Leftrightarrow x \in FV(\langle P^F \rangle_4)$ because $x \neq y$ and $x$ is not free in $E$ or $F$

(iii) $G \equiv PQ$. Similar but easier. ∎

4.27. LEMMA. *Let $E$ be a $\lambda\perp$-redex without proper subredexes. Let $F$ be the contractum of $E$. Then $KSL(E) = KSL(F)$.*

*Proof.* By cases of $\lambda\perp$-reduction, and by induction, firstly on the size of $E$, and then (when $E$ is an application) on the size of the rator of $E$. See Definitions 4.22 and 4.23.

(i) $E \equiv \perp M$. Then $F \equiv \perp$. Now $KSL(E) = (*, *, *) \oplus KSL(M) = (*, *, *) = KSL(F)$.

(ii) $E \equiv \lambda x.\perp$. Then $F \equiv \perp$. Now $KSL(E) = (*, *, *) = KSL(F)$.

(iii) $E \equiv xP_1 \ldots P_j$. Then $F \equiv x\perp^j$. Now $KSL(E) = (0, 0, j) = KSL(F)$.

(iv) $E \equiv (\lambda x_1 \ldots x_k . x_1 \perp^j)P$. Then $F \equiv \lambda x_2 \ldots x_k . P\perp^j$. Now

$$\begin{aligned} KSL(E) &= KSL(\lambda x_1 \ldots x_k . x_1 \perp^j) \oplus KSL(P) \\ &= (k, 1, j) \oplus KSL(P). \end{aligned}$$

We compute $KSL(F)$ according to the following subsubcases (4), (5), (7)–(12) corresponding to the definition of $\oplus$.

(4) $KSL(P) = (*, *, *)$. Then $P \equiv \perp$. Now $F \equiv \lambda x_2 \ldots x_k . \perp\perp^j$ and $KSL(F) = (*, *, *) = KSL(E)$.

(5) $KSL(P) = (0, 0, j')$. Then $P \equiv y\perp^{j'}$. Now $F \equiv \lambda x_2 \ldots x_k, y\perp^{j'}\perp^j$ and

$$KSL(E) = (k, 1, j) \oplus (0, 0, j') = (k - 1, 0, j + j') = KSL(F).$$

(7) $KSL(P) = (k' + 1, 0, j') \,\&\, j = 0$. Then $P \equiv \lambda y_1 \ldots y_{k'+1} . y\perp^{j'}$. Now $F \equiv \lambda x_2 \ldots x_k . P$ and

$$KSL(E) = (k, 1, 0) \oplus (k' + 1, 0, j') = (k + k', 0, j') = KSL(F).$$

(8) and (12) can be treated simultaneously. $KSL(P) = (k' + 1, n', j')$, where $n' \neq 1$, and $k, j > 0$. $P \equiv \lambda y_1 \ldots y_{k'+1} . y\perp^{j'}$, where if $n' \geqslant 2$ then $y = y_n$, and if $n' = 0$ then $y$ is not equal to any of $y_1 \ldots y_{k'+1}$. Now $F = \lambda x_2 \ldots x_k . P\perp^j$. Let $G = (\lambda x_1 \ldots x_k . x_1 \perp^{j-1})(P\perp)$. Then $G \to F$. From the definition of $\oplus$ we have

$$KSL(E) = (k, 1, j) \oplus (k' + 1, n', j') = (k, 1, j - 1) \oplus (k', n'', j') = KSL(G),$$

where if $n' = 0$, then $n'' = 0$, and if $n' \geqslant 1$, then $n'' = n' - 1$. We must prove $KSL(G) = KSL(F)$. Since $P$ is an abstraction, $P\perp$ is a redex. Let $P'$ be the

result of reducing it. It is clear that $P\perp$ contains no proper subredexes and is smaller than $E$; therefore by the induction hypothesis $KSL(P\perp) = KSL(P')$. By substitutivity of $KSL$ one has $KSL(G) = KSL(G')$, where $G' \equiv (\lambda x_1 \ldots x_k . x_1 \perp^{j-1}) P'$. Let $F' \equiv \lambda x_2 \ldots x_k . P' \perp^{j-1}$. Then $G' \to F'$ and $F \to F'$. Both $G'$ and $F'$ are smaller than $E$, and are both redexes not containing subredexes. Therefore by the induction hypothesis $KSL(G') = KSL(F') = KSL(F)$, and the result is proved.

(9) and (11) can be treated simultaneously. $KSL(P) = (k'+1, n'+1, j')$ & $k > 0$, $j = 0$. Then $P \equiv \lambda y_1 \ldots y_{k'+1} . y_{n'+1} \perp^{j'}$. Now $F \equiv \lambda x_2 \ldots x_k$, $P \equiv \lambda x_2 \ldots x_k\, y_1 \ldots y_{k'+1} . y_{n'+1} \perp^{j'}$, and we have

$$KSL(E) = (k, 1, 0) \oplus (k'+1, n'+1, j') = (k+k', k+n'+1, j') = KSL(F).$$

(10) $KSL(P) = (k'+1, 1, j')$ & $k, j > 0$. Then $P \equiv \lambda y_1 \ldots y_{k'+1} . y_1 \perp^{j'}$. Now $F \equiv \lambda x_2 \ldots x_k . P \perp^j$:

$$\begin{aligned} KSL(E) &= (k, 1, j) \oplus (k'+1, 1, j') = (*, *, *) \\ KSL(F) &= (KSL(P) \oplus KSL(\perp) \oplus \ldots \oplus KSL(\perp)) + \ldots \\ &= ((k'+1, 1, j') \oplus (*, *, *) \oplus \ldots \oplus (*, *, *)) + \ldots \\ &= (*, *, *) + \ldots \\ &= (*, *, *) \end{aligned}$$

(v) $E \equiv (\lambda x_1 \ldots x_k . x_n \perp^j) P$ & $k \geqslant 1$, $n \neq 1$. Now $F \equiv \lambda x_2 \ldots x_k . x_n \perp^j$:

$$\begin{aligned} KSL(E) &= KSL(\lambda x_1 \ldots x_k . x_n \perp^j) \oplus KSL(P) \\ &= (k, n, j) \oplus KSL(P). \end{aligned}$$

We compute $KSL(F)$ according to cases (3) and (6) of the definition of $\oplus$.

(3) $n = 0$. $KSL(E) = (k, n, j) \oplus KSL(P) = (k-1, 0, j) = KSL(F)$

(6) $n \geqslant 2$. $KSL(E) = (k, n, j) \oplus KSL(P) = (k-1, n-1, j) = KSL(F)$. ∎

4.28. PROPOSITION. *If* $\lambda\perp \vdash E = F$, *then* $KSL(E) = KSL(F)$. *In particular,* $KSL(E) = KSL(E^{\mathrm{nf}})$, *where* $E^{\mathrm{nf}}$ *is the normal form of* $E$.

*Proof.* From Lemmas 4.26 and 4.27, if $\lambda\perp \vdash E \to F$, then $KSL(E) = KSL(F)$. The proposition follows. ∎

4.29. PROPOSITION. (i) $KSL(M) = (k, n, j) \Leftrightarrow \lambda\perp \vdash M \twoheadrightarrow \lambda x_1 \ldots x_k . x_n \perp^j$.

(ii) $KSL(M) = (*, *, *) \Leftrightarrow \lambda\perp \vdash M \twoheadrightarrow \perp$.

*Proof.* The normal forms of $\lambda\bot$ are $\lambda x_1 \ldots x_k.x_n \bot^j$ and $\bot$. Every $\lambda\bot$-expression has a normal form. Hence the proposition follows from Lemma 4.25 and Proposition 4.28. ∎

4.30. PROPOSITION. *If* $\lambda\bot \vdash M \twoheadrightarrow \lambda x_1 \ldots x_k.x_n \bot^j$, *then* $\lambda \vdash M \twoheadrightarrow \lambda x_1 \ldots x_k.x_n P_1 \ldots P_j$, *for some expressions* $P_1 \ldots P_j$.

*Proof.* All $\lambda\bot$-reductions can be mimicked in $\lambda$. ∎

4.31. THEOREM. $KSL(M) = (k, s, j) \Rightarrow M$ *has a hnf of the form* $\lambda z_1 \ldots z_k.z_s N_1 \ldots N_j$.

*Proof.* From Proposition 4.29 and Proposition 4.30. ∎

## 5. CONCLUDING REMARKS

We will make some remarks on the relation of the present work with "strictness analysis" and with the various concepts of "sequentiality."

*Strictness*

As the bare essence of "strictness analysis" we understand the following. Given a domain $D$ of data, including an undefined element $\bot$, and some space $\mathbb{F}$ of functions over $D$ (not necessarily only unary functions) we will understand "strictness analysis" to designate the endeavour of (1) giving characterisations of some classes of strict functions from $\mathbb{F}$ and (2) giving computable approximations (that is, subclasses) of some classes of strict functions from $\mathbb{F}$. Here a unary function $f$ in $\mathbb{F}$ is strict if $f(\bot) = \bot$, meaning that nonzero information output can only be obtained by nonzero information input. Further, a binary function $g$ in $\mathbb{F}$ is strict in both arguments if $g(\bot, x) = g(x, \bot) = \bot$, and likewise for $n$-ary functions.

In our setting, the data domain $D$ is the set of $\lambda$-terms modulo equality as obtained by $\beta$-reduction plus the rule $M \to \bot$ for all $M$ without head normal form. Thus all terms without head normal form are considered to be meaningless and identified with the undefined element $\bot$. In Barendregt (1984, Chap. 16) this lambda theory is called $\mathscr{H}$. The space of $n$-ary functions $\mathbb{F}$ consists of contexts $C[\ , \ldots, \ ]$ with $n$ (or fewer) holes; here $n \geqslant 1$. We now have the following result, due to H. Mulder (oral communication).

5.1. PROPOSITION. *For every context* $C[\ ]$ *and every redex* $R$ *we have*:

*the unary function associated with* $C[\ ]$ *is strict* $\Leftrightarrow$ $R$ *is head needed in* $C[R]$.

*Proof.* We show that the negations of both sides are equivalent:

$$C[\bot] \neq \bot \Leftrightarrow C[\bot] \text{ has a head normal form}$$
$$\Leftrightarrow C[\bot] \xrightarrow[\text{head}]{} \lambda x_1 \ldots x_n . x_i M_1 \ldots M_m$$
$$\Leftrightarrow C[R] \xrightarrow[\text{head}]{} \lambda x_1 \ldots x_n . x_i M_1^* \ldots M_m^*$$

(without reducing $R$, where $M_i^*$ is the result of substituting $R$ for $\bot$ in $M_i$)

$$\Leftrightarrow R \text{ not head needed in } C[R]. \quad \blacksquare$$

It follows that

$$C[\ ] \text{ is strict in } [\ ] \Leftrightarrow \forall R. \quad R \text{ is head-needed in } C[R]$$
$$\Leftrightarrow \exists R. \quad R \text{ is head-needed in } C[R].$$

Thus our computable approximations of the concept of head-needed redex, such as head spine redex, generalized head spine redex, etc., can be perceived as strictness analysis.

*Berry Sequentiality*

At this point it is worthwhile to note that in the pure $\lambda$-calculus there are no nontrivial $n$-ary functions (with $n \geqslant 2$) which are strict in all their arguments. That is, if $C[M, \bot] = C[\bot, M] = \bot$ for all $M$, then the function associated with this binary context is identically $\bot$. This follows from a theorem of G. Berry (1978) who refers to this fact as the "sequentiality" of $\lambda$-calculus. It is therefore slightly puzzling that an operator like $+$ can be defined in $\lambda$-calculus by a term PLUS such that PLUS $\underline{n}\underline{m} \twoheadrightarrow \underline{n+m}$; apparently the operator $+$ which is strict in both arguments in some setting $(D, F)$ can only be implemented in $\lambda$-calculus such that the dependence on one argument is nonstrict; indeed, the usual definition of PLUS will be such that PLUS $\bot\ \underline{m} = \bot$, whereas PLUS $\underline{n}\bot \neq \bot$. The "Berry-sequentiality" of $\lambda$-calculus entails that PLUS reads in and processes its input in a sequential way.

Of course the concept of strictness depends entirely on what is taken to be $\bot$; a typical example is the following: in $\lambda I$-calculus with "having no head-normal form" standing for "undefined" we have, as we just saw, no binary contexts strict in both "arguments." However, if we take as notion of undefined: "having no normal form" (so $M \to \bot$ if $M$ has no nf) then there are binary functions strict in both arguments; just take the context $\lambda z.z[\ ][\ ]$. (The restriction to $\lambda I$-calculus is necessary for this example,

since in $\lambda$-calculus it is not possible to identify all terms without normal form.) See the discussion on "undefined" in Barendregt (1977).

The remark above on the nonexistence of binary functions (as given by contexts) strict in both arguments can be paraphrased in another way. In a $\lambda$-term $M$ we can discern the head-needed redexes $R_1,\ldots, R_n$. Each redex $R_i$ can be replaced by an arbitrary redex which still is head-needed *if the other redexes are kept the same.* So we have determined head-needed "places"; but the place occupied by $R_i$, while independent of $R_i$, does depend on the other redexes. In fact, Berry's sequentiality theorem states that there is no binary context such that both places are head-needed regardless of the contents; the head-neededness of one place depends on the actual content of the other place.

*Huet–Lévy Sequentiality*

The terminology of "needed places" brings us to another concept of sequentiality, that of Huet and Lévy (1979), which should not be confused with Berry's notion of sequentiality. While Berry's notion refers to the way in which data are read in and processed in a $\lambda$-term, regardless of any "reduction strategy," the notion of Huet and Lévy says that a sequential reduction strategy (as opposed to a parallel one) is adequate for reaching (head) normal forms. This in contrast with some rewriting systems for which no adequate sequential reduction strategy exists and for which one must adopt a parallel strategy in order to be sure of finding (head) normal forms whenever they exist. In the terminology of Huet and Lévy, a rewrite system is sequential if for every $n$-ary context $C[\ ,\ldots,\ ]$ in normal form and for every substitution with redexes $R_i$ such that the result $C[R_1,\ldots, R_n]$ has a normal form, there exists at least one redex $R_i$ which is needed. A shortcoming of this notion is that, in general, it cannot be decided whether a rewrite system has this property; and second, that even if the rewrite system has this sequentiality property, such a needed redex cannot always be indicated in a computable way. Therefore they introduce a stronger concept: a rewrite system is strongly sequential if for every $n$-ary context $C[\ ,\ldots,\ ]$ in normal form there exists a needed *place*, say the $i$th place. This means that after filling up the context with redexes $R_i$ such that $C[R_1,\ldots, R_n]$ has a normal form, the $i$th redex is needed. Clearly, $\lambda$-calculus is strongly sequential in this sense: the leftmost place in $C[\ ,\ldots,\ ]$ is always needed.

*Summarizing.* (i) $\lambda$-calculus is strongly sequential in the sense of Huet and Lévy;

(ii) $\lambda$-calculus with identification of terms without head-normal form is sequential in the sense of Berry.

To see the difference between the two notions even more sharply, one may consider the extension of $\lambda$-calculus with an new constant $+$ satisfying $\underline{n}+\underline{m} \to \underline{n+m}$, $\bot + \underline{n} \to \bot$, and $\underline{n} + \bot \to \bot$. This extension is still strongly sequential in the sense of Huet and Lévy, but it is not Berry-sequential. Another extension of $\lambda$-calculus, with $\underline{\text{or}}(T, x) \to T$, $\underline{\text{or}}(x, T) \to T$, and $\underline{\text{or}}(F, F) \to F$ is neither Huet–Lévy-sequential nor Berry-sequential. The first operator, $+$, is strict in both arguments, the second, $\underline{\text{or}}$, is strict in neither of its arguments.

It is interesting to note that for lambda calculus with $\underline{\text{or}}$, there nevertheless exists a strategy which is sequential, in the weak sense that it chooses a single redex to reduce at each step, without memory of the past history of the computation (Kennaway, 1987).

*Extending Lambda Calculus with Strict Operators*

Our algorithms for the determination of sets of needed and head-needed redexes can easily be extended to extensions of $\lambda$-calculus with strict operators such as $+$. We will show that the algorithms for $\text{Sel}_i$ and $\langle\ \rangle_i$ (see Definition 4.15) can easily be extended to the case where a "demand-forking" operator like "$+$" is present. We will only do this for $i = 3$.

Consider the extension of $\lambda$-calculus with a *binary* operator $+$, and numerals $\underline{n}$ for each natural number $n$. Apart from the $\beta$-reduction rule there are the rules $+(\underline{n}, \underline{m}) \to \underline{n+m}$ for all $n, m$. An expression $+(\underline{n}, \underline{m})$ is a "$+$-redex." Call this extension $\lambda^+$-calculus. An example of a $\lambda^+$-term is $(\lambda x.+(x, x))\underline{3}$. (Note that $+(x, x)$ is not a redex.)

We have to define what a *head-normal form* in $\lambda^+$-calculus is: it is a term such that neither a $\beta$-redex nor a $+$-redex is in "head position." More precisely:

5.2. DEFINITION. (i) Let $M$ be a $\lambda^+$-term. A redex $R$ sub $M$ is in *head-position* if the leading symbol of $R$ (that is, $\lambda$ or $+$) is only preceded by occurrences of $+$ or $\lambda$ where the latter are not redex-$\lambda$s. Here the precedence ordering is as follows: (1) if $s, t$ are symbol occurrences in an application $PQ$, $s$ in $P$ and $t$ in $Q$, then $s$ precedes $t$; (2) in $+(P, Q)$ the $+$ precedes all symbols of $P, Q$, but there is no relation between $s$ in $P$ and $t$ in $Q$.

(ii) A $\lambda^+$-term $M$ is a *head-normal form* if there is no redex $R$ sub $M$ in head-position.

EXAMPLE. $+((\lambda x.+(\underline{3}, \underline{2})), \underline{1})$ is not a head-normal form; $\lambda xy.+(+(x, +(1, 1)), y)$ is a head-normal form but not a normal form.

The notion of (head-)needed is analogous to the case without $+$.

Now $\mathrm{Sel}_3(M)$, for a $\lambda^+$-term $M$, is defined as follows. It will be a *set of nonzero natural numbers.* First some notation; if $X$ is such a set, then

$$X--1 = \{n \mid n+1 \in X\} - \{0\}$$
$$X++1 = \{n+1 \mid n \in X\}.$$

Simultaneously with $\mathrm{Sel}_3(M)$, we define $\langle M\rangle_3$.

5.3. DEFINITION. (a)

$$\begin{aligned}
\langle x\rangle_3 &= x;\\
\langle \underline{n}\rangle_3 &= \underline{n};\\
\langle \lambda x.P\rangle_3 &= \lambda x.\langle P\rangle_3;\\
\langle PQ\rangle_3 &= \langle P\rangle_3\langle Q\rangle_3, \qquad \text{if } 1 \in \mathrm{Sel}_3(P),\\
&= \langle P\rangle_3 \perp, \qquad \text{otherwise;}\\
\langle +(P,Q)\rangle_3 &= +(\langle P\rangle_3, \langle Q\rangle_3).
\end{aligned}$$

(b)
$$\begin{aligned}
\mathrm{Sel}_3(\underline{n}) &= \mathrm{Sel}_3(x) = \varnothing;\\
\mathrm{Sel}_3(PQ) &= \mathrm{Sel}_3(P)--1;\\
\mathrm{Sel}_3(\lambda x.P) &= (\mathrm{Sel}_3(P)++1)\cup\{1\}, \qquad \text{if } x \in FV(\langle P\rangle_3),\\
&= \mathrm{Sel}_3(P)++1, \qquad \text{otherwise;}\\
\mathrm{Sel}_3(+(P,Q)) &= \mathrm{Sel}_3(P)\cup\mathrm{Sel}_3(Q).
\end{aligned}$$

(Note that the role of $\uparrow$ in Definition 4.15 is now played by $\varnothing$).

EXAMPLE. (i) $\mathrm{Sel}_3(\lambda xyz.+(z,+(x,z))) = \{1,3\}$.

(ii) $\langle(\lambda xyz.+(z,+(x,z)))PQRS\rangle_3 = (\lambda xyz.+(z,+(x,z)))\langle P\rangle_3\perp\langle R\rangle_3\perp$.

The proof of the following fact follows the same lines as the case without $+$, and is omitted.

5.4. THEOREM. *All redexes visible in $\langle M\rangle_3$, where $M$ is a $\lambda^+$-term, are (head-)needed.* ∎

*Summary of Results*

The Introduction motivated the precise identification of the concept of needed redex in a lambda term and the requirement for efficient algorithms which yield approximations to this undecidable notion. Section 3 developed the main technical results characterising neededness and proving that quasi-needed reduction sequences are normalising. Section 4 begins the

work of identifying efficient algorithms for computing neededness. Whether such algorithms are best employed at compile or run time is very much a matter for the implementor, and the technology available to him. At the time of writing he will achieve some benefit from including an algorithm for detecting neededness in a compiler for a sequential machine. Future implementors may find it useful to embody algorithms for recognising needed redexes in hardware. Section 5 illustrates how the approach can be extended to $\lambda$-calculus with built-in operators.

## APPENDIX: LÉVY'S LABELLED LAMBDA CALCULUS

Lévy's labelled $\lambda$-calculus is a powerful instrument to trace in a precise way what happens in a reduction sequence. Many arguments using the terminology of reduction diagrams, residuals of redexes and creation of redexes as explained in the Introduction can be dealt with in a more succinct way using Lévy's labels. In this Appendix we will introduce Lévy's labelled $\lambda$-calculus and use it to obtain some alternative proofs for propositions in this paper, in particular, those propositions which are required for a complete proof have very verbose arguments and elaborate case distinctions, which, therefore, we have only sketched. Besides giving additional credibility to some of those technical propositions, we feel that Lévy-labelled $\lambda$-calculus can play a beneficial role in investigations similar to the present one. Lévy-labelled $\lambda$-calculus was introduced in Lévy (1975); we will present and use the simplified version in Klop (1980) (in Barendregt, 1984, Exercise 14.5.5).

6.1. DEFINITION. (i) Let $L_0 = \{a, b, c, \ldots\}$ be an infinite set of symbols. The set $L$ of (Lévy) labels is defined inductively by

$$w \in L_0 \Rightarrow w \in L$$
$$w, v \in L \Rightarrow wv \in L$$
$$w \in L \Rightarrow \underline{w} \in L.$$

Here $wv$ is the concatenation (without brackets) of the words $w$ and $v$. Note that labels may have nested underlinings, as in $\underline{ab\underline{c}ab\underline{c}a}$.

(ii) The set $\Lambda_L$ of labelled $\lambda$-terms is inductively defined by

$$x \in \Lambda_L,$$
$$M, N \in \Lambda_L \Rightarrow (MN) \in \Lambda_L,$$
$$M \in \Lambda_L \Rightarrow (\lambda x.M) \in \Lambda_L,$$
$$M \in \Lambda_L \Rightarrow (M^w) \in \Lambda_L.$$

Since the first three clauses generate unlabelled terms, this means that we have defined terms with *partial* labellings (i.e., not every subterm bears a label; equivalently, some subterms may have the "empty label").

Multiple labellings as in $(M^w)^v$ will be simplified to $M^{wv}$; this simplification is executed as soon as possible.

(iii) Labelled $\beta$-reduction $\to_L$ (where the subscript $L$ will often be dropped) is defined by

$$(\lambda x.M[x])^w N \underset{L}{\longrightarrow} (M[(N)^w])^w,$$

i.e., each occurrence of $x$ in the (labelled) term $M(x)$ is replaced by $N^w$ (note that $N$ may have some labels itself; see example below) and the result is labelled by $w$. The label $w$ appearing in this definition is called the *degree* of the redex in the LHS. Here is an example of a labelled reduction step, where we have omitted parentheses as allowed by part (i) of the definition:

$$((\lambda x.(x^a A)^b)^c(\lambda y.B)^d)^e \underset{L}{\longrightarrow} ((\lambda y.B)^{dca}A)^{bce}.$$

Note that this step has taken place in the labelled context $[\ ]^e$; and that substituting $(\lambda y.B)^{dc}$ in $x^a$ has yielded $(\lambda y.B)^{dca}$.

The following fact is proved in Lévy (1975) and Klop (1980).

6.2. THEOREM. *Labelled $\lambda$-calculus is confluent.* ∎

6.3. EXAMPLE. Figure 16 shows an elementary reduction diagram of labelled $\lambda$-calculus.

*Residuals* of redexes are defined in the case of labelled reductions just as in the unlabelled case. We can now state the first benefit of the labelled version of reductions: let $R$ in the unlabelled term $M$ be a redex, and suppose $M \twoheadrightarrow N$. To determine the residuals of $R$ in $N$, we attach an atomic label, say "$a$," as the degree of $R$ (that is, $R \equiv (\lambda x.A)B$ is replaced by $(\lambda x.A)^a B$). The result is a (partially) labelled term $M^I$, where $I$ denotes the labelling. The given reduction $M \twoheadrightarrow N$ can now in the obvious way be "lifted" to the labelled case; we find a labelled reduction $M^I \twoheadrightarrow_L N^J$, where $N^J$ is $N$ together with a labelling $J$. Now all redexes $R'$, $R''$, ... in $N^J$ with degree "$a$,"

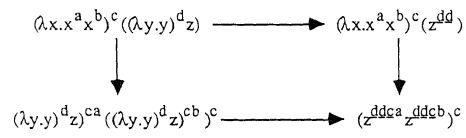


FIGURE 16

are residuals of the original redex $R$, and they are the only ones. (The proof is a routine exercise.)

*Creation of redexes* can also neatly be expressed in the formalism of labelled reduction. Given an unlabelled reduction step $R: M \to N$ and a redex $S$ of $N$, we say that $S$ is created by the $R$-contraction if $S$ is not the residual of any redex in $M$. Now if the present reduction step takes place in the labelled setting: $M^I \to N^J$, it turns out that the degree of the created redex $S$ in $N^J$ contains the underlined degree of the creator redex $R$ as a subword. We give an example.

6.4. EXAMPLE. (i) $M \equiv R \equiv (\lambda x.x^u B)^v(\lambda x.A)^w \to ((\lambda x.A)^{w\underline{v}u}B)^{\underline{v}} \equiv S \equiv N$. Indeed the degree $w\underline{v}u$ of the created redex contains the underlined degree $v$ as a subword.

(ii) $(\lambda x.x^u)^v(\lambda y.A)^w B \to (\lambda y.A)^{w\underline{v}uv}B$

(iii) $(\lambda x.\lambda y.u^u)^v CB \to (\lambda y.C^{\underline{v}u})^{\underline{v}}B$.

(Essentially these are all "types of creation" that exist.) Theorem 6.2 can in fact be strengthened in the same way as for unlabelled reductions, as in 2.35 of the preliminary section: the common reduct can be found by completing a reduction diagram (now for the labelled case) by adding "elementary labelled reduction diagrams" of which one is displayed in Fig. 16. In such elementary diagrams the redexes contracted in opposite sides have the same degree; so one might say that degrees propagate without changing in horizontal and vertical direction, in the construction of a reduction diagram. Therefore, in a completed composite labelled reduction diagram, the degrees of the redexes contracted in the top side of the diagram coincide exactly with the degrees of the redexes contracted in the bottom side, and likewise for left side and right side. Bearing in mind that residuals have the same degree as their ancestor redex, we have an immediate proof of Proposition 2.7(ii) in the preliminary section.

Finally, an alternative proof for Proposition 2.7(i) can be obtained easily using the above mentioned facts for labelled reductions. However, with the available power of labelled reductions, it is just as easy to skip Proposition 2.7 and prove Proposition 2.6 directly; 2.6 follows at once from the following.

6.5. PROPOSITION. *Let a reduction as in Fig.* 17 *be given, such that no residual of redex $R$ in $M$ is contracted in the reduction $\mathscr{R} = M \twoheadrightarrow N$. Let redex $S$ in $M$ be created by the step $R: M \to M'$. Then in the projected reduction $\mathscr{R}/R$ no residual of $S$ is contracted.*

*Proof.* Label $M$ partially by assigning degree "$a$" to $R$ and degree "$b$" to all other redexes in $M$. Then every redex contracted in the reduction

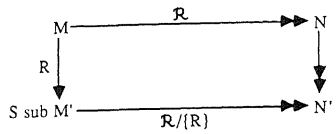


FIGURE 17

$M \twoheadrightarrow N$ has degree containing "$b$" as its only atomic label. The same therefore holds true for the projected reduction $M' \twoheadrightarrow N'$. This means that no residual of $S$ is contracted in that reduction, since in the labelled reduction diagram (obtained by lifting the given reduction diagram starting with the before mentioned labelling of $M$) the degree of redex $S$ in $M'$ contains an occurrence of the symbol "$a$." ∎

## ACKNOWLEDGMENTS

Inspiration for the paper was obtained on the island of Ustica at the *First autumn seminar and conference on reduction machines* (organised by C. Böhm) where we had stimulating discussions with Arvind and others. The various algorithms in Section 4 are also described in van Eekelen and Plasmeijer (1985) who introduced them independently.

RECEIVED June 20, 1986; ACCEPTED May 6, 1987

## REFERENCES

AUGUSTSSON, L. (1984), A compiler for lazy ML, *in* "Proceedings, ACM Conf. on LISP and Functional Programming," pp. 218–227, Assoc. Comput. Mach., New York.

BARENDREGT, H. P. (1977), Solvability in lambda calculi, *in* "Colloque International de Logique," pp. 209–220, Éditions du Centre National de la Recherche Scientifique, Paris.

BARENDREGT, H. P. (1984), "The Lambda Calculus," North-Holland, Amsterdam.

BERRY, G. (1978), Séquentialité de l'évaluation formelle des $\lambda$-expressions, *in* "Proceedings, 3-e Colloque International sur la Programmation," Dunod, Paris.

BERRY, G., AND LÉVY, J.-J. (1979), Minimal and optimal computations of recursive programs, *J. Assoc. Comp. Mach.* **26**, 148–175.

BURN, G. L., HANKIN, C. L., AND ABRAMSKY, S. (1986), Strictness analysis for higher-order functions, *Sci. Comput. Programming* **7**, 249–278.

BURSTALL, R. M., MACQUEEN, D. B., AND SANNELLA, D. T. (1981), HOPE: An experimental applicative language, *in* "Proceedings, First LSP Conference, Stanford," pp. 136–143.

VAN EEKELEN, M. C. J. D., AND PLASMEIJER, M. J. (1985), Avoiding redex copying in lambda reduction, preprint, Department of Informatics, University of Nijmegen, Tournooiveld 1, 6525 ED Nijmegen, The Netherlands.

GORDON, M. J., MILNER, A. J. R. G., AND WADSWORTH, C. P. (1979), "Edinburgh LCF," Lecture Notes in Computer Science Vol. **78**, Springer, Heidelberg/Berlin/New York.

HUET, G., AND LÉVY, J.-J. (1979), Call by need computations in non-ambiguous linear term rewriting systems, preprint 359, INRIA, B.P. 105, Le Chesnay 78150, France.

KENNAWAY, J. R. (1987), "Sequential Evaluation Strategies for Parallel-or and Related Reduction Systems, Report SYS-C87-05, School of Information Systems, University of East Anglia, Norwich NR47TJ, England.

KLOP, J. W. (1980), "Combinatory Reduction Systems," Mathematical Centre Tracts Vol. **127**, Mathematical Centre, Kruislaan 413, 1098 SJ Amsterdam.

LANDIN, P. (1964), The mechanical evaluation of expressions, *Comput. J.* **6**, 308–320.

LANDIN, P. (1966), The next 700 programming languages, *Comm. ACM* **9**, 157–166.

LÉVY, J.-J. (1975), An algebraic interpretation of the $\lambda\beta K$-calculus and a labelled $\lambda$-calculus, *in* "Lambda Calculus and Compuer Science Theory," Lecture Notes in Computer Science Vol. **137**, pp. 147–165, Springer-Verlag, New York/Berlin.

LÉVY, J.-J. (1980), Optimal reductions in the lambda calculus, *in* "To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus, and Formalism" (J. Seldin and R. Hindley, Eds.), Academic Press, New York.

MCCARTHY, J., ABRAHAMS, P. W., EDWARDS, D. J., HART, T. P., AND LEVIN, M. I. (1962), "The LISP 1.5 Programmers' Manual," MIT Press, Cambridge, MA.

MYCROFT, A. (1981), "Abstract Interpretation and Optimising Transformations for Applicative Programs," Ph. D. thesis, University of Edinburgh.

TURNER, D. (1979), A new implementation technique for applicative languages, *Software: Practice Experience* **9**, 31–49.

TURNER, D. (1985), Miranda: A non-strict functional language with polymorphic types, *in* "Functional Programming Languages and Computer Architecture," (J.-P. Jouannaud, Ed.), Lecture Notes in Computer Science Vol. **201**, pp. 1–16, Springer, Heidelberg/Berlin/New York.

WADSWORTH, C. P. (1971), "Semantics and Pragmatics of the Lambda Calculus," D. Phil. thesis, Oxford University.