On normalisation

Citation for published version (APA):

Document status and date:
Published: 01/01/1995

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 15. Oct. 2019
On normalisation
by Femke van Raamsdonk and Paula Severi
Report 95-20

Computing Science Reports
Department of Computing Science
On Normalisation

by

Femke van Raamsdonk and Paula Severi

95/20

ISSN 0926-4515

All rights reserved
editors: prof.dr. J.C.M. Baeten
prof.dr. M. Rem

Computing Science Report 95/20
Eindhoven, June 1995
On Normalisation

Femke van Raamsdonk
CWI
P.O. Box 94079, 1090 GB Amsterdam
The Netherlands
femke@cwi.nl

Paula Severi
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5500 MB Eindhoven
The Netherlands
severi@win.tue.nl

Abstract

Using a characterisation of strongly normalising λ-terms, we give new and simple proofs of the following:

1. all developments and superdevelopments are finite,
2. a certain rewrite strategy is perpetual,
3. a certain rewrite strategy is maximal and thus perpetual,
4. simply typed λ-calculus is strongly normalising.

AMS Subject Classification (1991): 03B40, 03D70.
Keywords & Phrases: λ-calculus, normalisation, perpetual strategies.
Note: The research of the first author is supported by NWO/SION project 612-316-606. This report is also available as Report CS-R9545, CWI, The Netherlands.
1. INTRODUCTION

This paper represents an effort to shed some more light on various results concerning normalisation in $\lambda$-calculus. We deal with $\lambda$-calculus with only $\beta$-reduction.

As a first step towards a better understanding we characterise both the set of weakly normalising terms and the set of strongly normalising terms. Remember that a term $M$ is said to be weakly normalising if there is a rewrite sequence starting in $M$ that eventually ends in a normal form, and that a term $M$ is said to be strongly normalising if all rewrite sequences starting in $M$ end eventually in a normal form.

To give a characterisation of all weakly normalising terms is actually rather easy: a weakly normalising term is a normal form or can be obtained as the result of some expansion starting in a normal form. Or, to put it slightly differently, the set of all weakly normalising terms is exactly the smallest set of all normal forms closed under expansion.

A specialisation of this idea yields a quite elegant characterisation of the strongly normalising terms, in the form of an inductively defined set denoted as $SN$. The definition of $SN$ can be found in section 3. This set can be viewed in different ways. From the point of view of rewriting, it is the closure under expansion of the set of normal forms, where expansion is subject to two restrictions. These restrictions are the following: first, the argument of the redex introduced by the expansion step should be in the set of strongly normalising terms, and second, the expansion step should yield a new head redex or a new outermost redex in a term without a head redex. Further, those familiar with saturated sets will certainly recognise one of the clauses of the definition of $SN$ as a defining property of a saturated set.

The interesting thing of the definition of the set $SN$ is that it permits to give new proofs of important results concerning normalisation in $\lambda$-calculus, like the Finite Developments Theorem and the fact that all simply typable terms are normalising. In most cases the new proofs are essentially simpler than already existing ones. Moreover, we feel that it is important to have different proofs of important results because they may help us to understand not only the mechanics of the proofs of the results but also the reasons for their validity.

The remainder of the paper is organised as follows.

In Section 3, we prove that the set $SN$ characterises the set of $\beta$-strongly normalising terms. We also give another characterisation of the set of all strongly normalising terms.

In Section 4 we give a short and simple proof of finiteness of developments. It is different from existing proofs, like for instance the one using a decreasing labelling or the elegant proof given by De Vrijer. Our proof makes use of expansion.

In Section 5 we prove that all superdevelopments are finite. Superdevelopments are developments in which redexes that are created 'upwards' during reduction may be contracted.

In Section 6 we give a new proof of the fact that the strategy $F_{hk}$ defined by Bergstra and Klop, is perpetual (meaning that it yields an infinite rewrite sequence whenever possible). Further we prove that the strategy $F_\infty$ defined by Barendregt, Bergstra, Klop and Volken is perpetual.

For the strategy $F_\infty$, we prove in Section 7 that it is not only perpetual but also maximal. That is, it yields the longest possible reduction to normal form whenever the initial term is
strongly normalising, and an infinite rewrite sequence if possible. This is done by computing the length of the rewrite sequence to the normal form.

In Section 8 we prove that simply typed $\lambda$-calculus is strongly normalising using our characterisation of the strongly normalising terms. The definition of $SN$ clearly recalls the definition of saturation, see [Tai67] and [Gir72].

In Section 9 we consider $\lambda$-calculus with intersection types. The set of strongly normalising terms is the set of terms that are typable in $\lambda\wedge$. We felt obliged to compare both characterisations and give a direct proof of the fact that the set $SN$ coincides with the set of typable terms in $\lambda\wedge$.

Finally, we discuss related work in section 10.

We start by reviewing some notation and by formalising the concept of lifting of rewrite sequences for the case of Abstract Rewriting Systems.

2. Preliminaries

Notation. We assume familiarity with $\lambda$-calculus and just fix some (mostly standard) notation.

The set of $\lambda$-terms is denoted by $\Lambda$. We write $x, y, z, \ldots$ for variables and $M, N, P, Q, \ldots$ for terms. We assume $\alpha$-conversion to be applied whenever necessary. The symbol $[]$ is used to denote a hole in a term. A term with one or more occurrences of $[]$ is called a context and is denoted by $C[]$. The term obtained by replacing in a context $C[]$ the occurrences of $[]$ by a term $M$ is denoted by $C[M]$. If not specified otherwise, in this paper a context $C[]$ is supposed to contain one occurrence of $[]$. We suppose a term to contain no occurrences of $[]$.

The set of free variables of a term $M$ is denoted by $FV(M)$ and its set of bound variables is denoted by $BV(M)$.

We consider $\lambda$-calculus with $\beta$-reduction generated by the $\beta$-reduction rule, that is given as $(\lambda x.M)N \rightarrow M[x := N]$. We denote the $\beta$-reduction relation by $\rightarrow_{\beta}$ or by $\rightarrow_{\beta}^\phi$ if we want to specify that the $\beta$-rewrite step is obtained by contracting a $\beta$-redex at position $\phi$. The reflexive-transitive closure of $\rightarrow_{\beta}$ ($\rightarrow_{\beta}^\phi$) is denoted by $\rightarrow_{\beta}^*$ ($\rightarrow_{\beta}^{\phi*}$). Syntactic equality is denoted by $=_{\text{synt}}$.

The set of normal forms is denoted by $\mathcal{NF}$ and $\text{nf}(M)$ denotes the normal form of a term $M$.

A term $M$ is said to be strongly normalising if every rewrite sequence ends after finitely many steps in a normal form. A term $M$ is said to be weakly normalising if there is a rewrite sequence starting at $M$ that ends in a normal form.

Lifting. In rewriting one often makes use of terms that are decorated, for instance by labels. Also the rewrite relation can be decorated, in the sense that the decoration is a part of the pattern of the rewrite rule. Erasure of decoration is used in order to switch from the rewriting system with decorations to the original rewriting system without decorations. The thus obtained correspondence consists in fact of two parts: the correspondence between terms and decorated terms and the correspondence between steps and decorated steps. A decorated rewrite sequence is often called a lifting of the rewrite sequence in the original rewriting.
systems, that is obtained by erasing all decorations. In this paper we shall encounter two
different ways of lifting a $\beta$-rewrite sequence in $\lambda$-calculus. Here we formalise the concept of
lifting for the general case, that is, for Abstract Rewriting Systems that are enriched with
some more structure.

**Definition 2.1.** An Abstract Rewriting System is a pair $(A, \rightarrow)$ consisting of a set $A$ and a
relation $\rightarrow \subseteq A \times A$.

**Notation 2.2.** Abstract Rewriting Systems are denoted by $A, B, C, \ldots$.

**Definition 2.3.** A rewrite step in an Abstract Rewriting System $A = (A, \rightarrow)$ is a pair $(a, b)$
of elements of $A$ with $(a, b) \in \rightarrow$.

**Notation 2.4.** We write $a \rightarrow b$ instead of $(a, b) \in \rightarrow$.

In rewriting systems where the rewrite relation is induced by a set of rewrite rules, it may
happen that different rewrite steps between terms $a$ and $b$ exist. This is for instance the
case in $\lambda$-calculus: there are two ways to rewrite $(\lambda x.x)((\lambda x.x)y)$ to $(\lambda x.x)y$. This cannot be
expressed in Abstract Rewriting Systems. For the concept of correspondence between rewrite
sequences we have in mind it is important that not only a correspondence between terms but
also between rewrite steps can be expressed. Therefore we will consider Abstract Rewriting
Systems enriched with some more structure, called indexed Abstract Rewriting Systems.

The idea is to view $\rightarrow$ as a collection of partial functions on $A$, written as $\{\rightarrow_i\}_{i \in I}$. For instance
$\lambda$-calculus can be seen as the set of $\lambda$-terms $\Lambda$ and a collection of partial functions $\\{\rightarrow^\phi_\beta\}$. It
is clear why the functions are partial: for instance a step $\rightarrow^\delta_\beta$ is not defined on every $\lambda$-term.
We proceed by giving the definition of an indexed Abstract Rewriting System.

**Definition 2.5.** An indexed Abstract Rewriting System is a triple $(A, I, \{\rightarrow_i\}_{i \in I})$ consisting
of a set $A$, a set of indices $I$ and an indexed set of partial functions $\{\rightarrow_i\}_{i \in I}$ from $A$ to $A$.

The definition of a rewrite step in an indexed Abstract Rewriting System differs a bit from
the one in an Abstract Rewriting System.

**Definition 2.6.** A rewrite step in an indexed Abstract Rewriting System
$A = (A, I, \{\rightarrow_i\}_{i \in I})$ is a triple consisting of two elements $a, b$ of $A$ and an element $\rightarrow_i$ of
$\{\rightarrow_i\}_{i \in I}$ such that $(a, b) \in \rightarrow_i$.

**Notation 2.7.** We write $a \rightarrow_i b$ for $(a, b) \in \rightarrow_i$.

Note that in an indexed Abstract Rewriting System $a \rightarrow_i b$ and $a \rightarrow_i c$ implies $b = c$. It is
possible to have $a \rightarrow_i b$ and $a \rightarrow_j b$ with $i \neq j$.

**Notation 2.8.** We use the notation $\rightarrow_i(a) = b$ to denote that $\rightarrow_i$ is defined on $a$ and that
the result of applying $\rightarrow_i$ to $a$ equals $b$. 
2 Preliminaries

The index $i$ of a rewrite step $a \rightarrow_i b$ can be considered to be the name of the rewrite step. In the case of term rewriting, taking redexes (that is, pairs consisting of a position and a rewrite rule) as indexes yields an instance of an indexed Abstract Rewriting System.

We now give a formal definition of a rewrite sequence in an indexed Abstract Rewriting System. It can be generalised to the case of Abstract Rewriting Systems, however, we don't need a so general definition in the present paper. See [Oos94] for a general definition of conversion.

**Definition 2.9.** Let $A = (A, I, \{\rightarrow_i\}_{i \in I})$ be an indexed Abstract Rewriting System. A **rewrite sequence** of length $\alpha$ starting in $a$ is a triple $(a, \alpha, \sigma)$ satisfying the following:

1. $a \in A$,
2. $\alpha \leq \omega$,
3. $\sigma : \alpha \rightarrow I$ is a mapping that defines a sequence $\{a_n\}_{n \in \alpha}$ as follows:
   
   (a) $a_0 = a$,
   
   (b) $a_n = \rightarrow_{\sigma(n)} (a_{n-1})$ for all $n \in \alpha \setminus \{0\}$

**Notation 2.10.** We often denote a rewrite sequence $\sigma$ as in the previous definition by $\sigma : a_0 \rightarrow \sigma(1) a_1 \rightarrow \sigma(2) \cdots$

We now define the concept of morphism between Abstract Rewriting Systems. It will be used to formalise the notion of correspondence between two rewrite sequences.

**Definition 2.11.** A **morphism** between indexed Abstract Rewriting Systems $A = (A, I, \{\rightarrow_i\}_{i \in I})$ and $B = (B, J, \{\rightarrow_j\}_{j \in J})$ is a pair of mappings $f = (f_0, f_1)$ with

$$f_0 : A \rightarrow B$$

$$f_1 : I \rightarrow J$$

such that $f_0(\rightarrow_i(a)) = \rightarrow_{f_1(i)} (f_0(a))$ for all $a \in A$ such that $\rightarrow_i(a)$ is defined.

Note that in the equality $f_0(\rightarrow_i(a)) = \rightarrow_{f_1(i)} (f_0(a))$ in the previous definition it may occur that $\rightarrow_{f_1(i)} (f_0(a))$ is defined but $f_0(\rightarrow_i(a))$ isn't.

We often write $f$ for both $f_0$ and $f_1$.

Let $\sigma : a_0 \rightarrow_m a_1 \rightarrow_m a_2 \rightarrow_m \cdots$ be a rewrite sequence in an indexed Abstract Rewriting System $A = (A, I, \{\rightarrow_i\}_{i \in I})$. Let $f : A \rightarrow B$ be a morphism between indexed Abstract Rewriting Systems. We denote by $f(\sigma)$ the rewrite sequence $f(\sigma) : f_0(a_0) \rightarrow_{f_1(m_1)} f(a_1) \rightarrow_{f_1(m_2)} f_0(a_2) \rightarrow_{f_1(m_3)} \cdots$. Note that by the definition of a morphism this indeed is a well-defined rewrite sequence in $B$.

**Definition 2.12.** Let $f : A \rightarrow B$ be a morphism. A rewrite sequence $\sigma$ in an indexed Abstract Rewriting System $A = (A, I, \{\rightarrow_i\}_{i \in I})$ is an $f$-**lifting** of a rewrite sequence $\rho$ in an indexed Abstract Rewriting System $B = (B, J, \{\rightarrow_j\}_{j \in J})$ if $f(\sigma) = \rho$.

Examples of liftings can be found in Section 4 and in Section 5.
3. A Characterisation of Strongly Normalising \( \lambda \)-terms

In this section we characterise the set of \( \lambda \)-terms that are strongly normalising.

**The characterisation.**

**Definition 3.1.** The set \( SN \) is the smallest set of \( \lambda \)-terms satisfying the following:

1. if \( x \) is a variable and \( M_1, \ldots, M_n \in SN \) for some \( n \geq 0 \), then \( xM_1 \ldots M_n \in SN \),
2. if \( M \in SN \) then \( \lambda x.M \in SN \),
3. if \( M[x := N]P_1 \ldots P_n \in SN \) and \( N \in SN \), then \( (\lambda x.M)NP_1 \ldots P_n \in SN \).

We prove that the set \( SN \) characterises the strongly normalising terms.

**Theorem 3.2.** \( M \) is strongly normalising if and only if \( M \in SN \).

**Proof.**

\( \Rightarrow \). Let \( M \) be a strongly normalising term. The proof proceeds by induction on the pair \( (\maxred(M), M) \), lexicographically ordered by the usual ordering on \( \mathbb{N} \) and the subterm ordering. Here we denote by \( \maxred(M) \) the maximum length of a reduction from \( M \) to normal form.

The base case is trivial since it is easy to see that all normal forms are in \( SN \).

Suppose the maximal reduction of \( M \) to normal form takes \( k + 1 \) steps. Let \( M = \lambda x_1 \ldots \lambda x_n.PQ_1 \ldots Q_m \). There are two cases.

**Case 1.** \( P = y \). Then the normal form of \( M \) is of the form \( \lambda x_1 \ldots \lambda x_n.yQ'_1 \ldots Q'_m \) with \( Q_i \rightarrow Q'_i \) for \( i = 1, \ldots, m \). By induction hypothesis, \( Q_1 \in SN, \ldots, Q_m \in SN \). By the first and second clause of the definition of \( SN \), we have \( M = \lambda x_1 \ldots \lambda x_n.yQ_1 \ldots Q_m \in SN \).

**Case 2.** \( P = \lambda y.P_0 \). We have \( M = \lambda x_1 \ldots \lambda x_n.(\lambda y.P_0)Q_1Q_2 \ldots Q_m \rightarrow \lambda x_1 \ldots \lambda x_n.P_0[y := Q_1]Q_2 \ldots Q_m \). By induction hypothesis, \( \lambda x_1 \ldots \lambda x_n.P_0[y := Q_1]Q_2 \ldots Q_m \in SN \). Also by induction hypothesis, \( Q_1 \in SN \). By the last clause of the definition of \( SN \), we have \( M = \lambda x_1 \ldots \lambda x_n.(\lambda y.P_0)Q_1 \ldots Q_m \in SN \).

\( \Leftarrow \). Suppose \( M \in SN \). We prove by induction on the derivation of \( M \in SN \) that \( M \) is strongly normalising.

1. If \( M = xM_1 \ldots M_n \) with \( M_1, \ldots, M_n \in SN \), then the statement follows easily by induction hypothesis.
2. If \( M = \lambda x.M_0 \) with \( M_0 \in SN \), then by induction hypothesis \( M_0 \) is strongly normalising. Then also \( M = \lambda x.M_0 \) is strongly normalising.
3. Let \( M = (\lambda x.M_0)M_1M_2 \ldots M_n \) with \( M_0[x := M_1]M_2 \ldots M_n \in SN \) and \( M_1 \in SN \). Consider an arbitrary rewrite sequence \( \rho : M = P_0 \rightarrow_\beta P_1 \rightarrow_\beta P_2 \rightarrow_\beta \ldots \) starting in \( M \). There are two possibilities: in \( \rho \) either the head redex of \( M \) is contracted or the head redex of \( M \) is not contracted.
In the first case, there is an $i$ such that $P_i = M_0'[x := M'_1]M'_2 \ldots M'_n$, with $M_0 \rightarrow M'_0, \ldots, M_n \rightarrow M'_n$. Then $P_i$ is a result of rewriting the term $M_0[x := M_1]M_2 \ldots M_n$. The latter is by induction hypothesis strongly normalising. Hence $P_i$ is strongly normalising so $\rho$ is finite.

In the second case, all terms in $\rho$ are of the form $(\lambda x. M'_0)M'_1M'_2 \ldots M'_n$ with $M_0 \rightarrow M'_0, \ldots, M_n \rightarrow M'_n$. By induction hypothesis, the term $M_0[x := M_1]M_2 \ldots M_n$ is strongly normalising. Therefore $M_0, M_2, \ldots, M_n$ are strongly normalising. Moreover, we have by induction hypothesis that $M_1$ is strongly normalising. Hence all the terms in the rewrite sequence are strongly normalising and hence $\rho$ is finite.

**Background.** We would like to point out the considerations motivating the previous definition.

An easy observation is that the set that contains all normal forms and that is closed under expansion is exactly the set of all weakly normalising terms. So we have the following definition.

**Definition 3.3.** The set $W$ is the smallest set of $\lambda$-terms satisfying the following:

1. all normal forms are in $W$,
2. if $C[P[x := Q]] \in W$, then $C[(\lambda x.P)Q] \in W$. 

The first naive attempt to obtain the set of all strongly normalising terms, is to add the requirement that the argument of the redex introduced by the expansion is strongly normalising. The thus defined set $S$ is the smallest set that satisfies

1. all normal forms are in $S$,
2. if $C[P[x := Q]] \in S$ and $Q \in S$, then $C[(\lambda x.P)Q] \in S$.

However, it is easy to see that the weakly but not strongly normalising term $(\lambda x.(\lambda y.(xz))(yy))$ belongs to $S$. The problem is that expansions cannot be allowed to take place just everywhere. If the expansion as in the second clause of the definition of $S$ above is required to create a head redex or, if the result of the expansion doesn’t contain a head redex, to create an outermost redex, then we indeed obtain the set of all strongly normalising terms.

**Definition 3.4.** The set $O$ of contexts with a hole at a head or outermost position is defined as the minimal set that satisfies

1. if $C[\ ] \in O$ then $x\ M_1 \ldots C[\ ] \ldots M_n \in O$,
2. if $C[\ ] \in O$ then $\lambda x.C[\ ] \in O$,
3. $[\ ]P_1 \ldots P_n \in O$.

**Definition 3.5.** The set $SN'$ is defined as the smallest set that satisfies
4. Finite Developments

1. all normal forms are in $SN'$,
2. if $C[P[x:=Q]] \in SN'$, $Q \in SN'$ and $C[] \in \mathcal{O}$, then $C[(\lambda x.P)Q] \in SN'$.

It is not difficult to show that $SN' = SN$.

Through the paper we use the first characterisation of the strongly normalising terms because besides being easier to handle it seems more natural, as it clearly recalls the notion of saturated sets.

**Inductive Definitions.** We often describe a set by induction as the smallest set closed under some set of rules (see [Acz77] and [Ter94]). This is the way we define, for example, the class of theorems of a given system. And this is also the way we have defined the set $SN'$ in Definition 3.1.

We could have described the set $SN'$ by induction giving a monotone operator instead of giving a set of rules. In this case, we should have defined the set $SN'$ as the least fixed point of the operator $H : \mathcal{P}(\Lambda) \rightarrow \mathcal{P}(\Lambda)$ given by:

$$H(X) = \{xM_1 \ldots M_n \mid M_i \in X \text{ for all } i = 1, \ldots, n \text{ and } x \text{ a variable} \} \cup$$

$$\{((\lambda x.M) \mid M \in X) \cup$$

$$\{((\lambda x.M)NP_1 \ldots P_n \mid M[x:=N]P_1 \ldots P_n \in X \text{ and } N \in X\}$$

Another possibility is to define the set $SN'$ as the well-founded part of the order $\prec \subseteq \Lambda \times \Lambda$ defined by:

1. $M_i \prec xM_1 \ldots M_n$ for all $i = 1, \ldots, n$,
2. $M \prec \lambda x.M$,
3. $M[x:=N]P_1 \ldots P_n \prec (\lambda x.M)NP_1 \ldots P_n$,
4. $N \prec (\lambda x.M)NP_1 \ldots P_n$.

In [Acz77], these three ways of giving a definition by induction, that is giving a set of deterministic rules or clauses, giving a monotone operator and giving an order are proved to be equivalent.

4. **Finite Developments**

A *development* is a rewrite sequence in which only descendants of redexes that are present in the initial term may be contracted. In this section we give a new and short proof of the fact that in $\lambda$-calculus all $\beta$-developments terminate.

Usually, $\beta$-developments are defined via a set of underlined $\lambda$-terms and an underlined $\beta$-reduction rule. We shortly recall these definitions, for a complete formal treatment see [Bar84].

**Definition 4.1.** The set of underlined $\lambda$-terms $\Delta$ is the smallest set satisfying the following:

1. $x \in \Delta$ for every variable $x$,
2. if $M \in \Delta$, then $\lambda x.M \in \Delta$,
3. if $M \in \Delta$ and $N \in \Delta$, then $MN \in \Delta$.
4. if $M \in \Delta$ and $N \in \Delta$, then $(\Delta x.M)N \in \Delta$.

The $\beta$-rewrite relation is defined as follows:

$$(\Delta x.M)N \rightarrow M[x := N]$$

Note that $\Delta$ is closed under $\beta$-rewriting.

In order to be able to switch between $\Lambda$ and $\Delta$ a mapping $E$ that erases underlinings is defined.

**DEFINITION 4.2.**

1. The mapping $E_0 : \Delta \rightarrow \Lambda$ is defined by induction on the definition of $\Delta$.
   
   a) $E_0(x) = x$,
   
   b) $E_0(\lambda x.M) = \lambda x.E_0(M)$,
   
   c) $E_0(MN) = E_0(M)E_0(N)$,
   
   d) $E_0((\Delta x.M)N) = (\lambda x.E_0(M))E_0(N)$

2. The mapping $E_1 : \{\beta\} \times \{0,1\}^* \rightarrow \{\beta\} \times \{0,1\}^*$ is defined by $E_1(\beta, \phi) = (\beta, \phi)$.

Since it is clear that $M \overset{\phi}{\rightarrow}_\beta N$ in $\Delta$ implies $E(M) \overset{\phi}{\rightarrow}_\beta E(N)$ in $\Lambda$, the proof of the following lemma is straightforward.

**LEMMA 4.3.** The mapping $E = (E_0, E_1)$ is a morphism from $(\Delta, \{\beta\} \times \{0,1\}^*, \{\overset{\phi}{\rightarrow}_\beta\})$ to $(\Lambda, \{\beta\} \times \{0,1\}^*, \{\overset{\phi}{\rightarrow}_\beta\})$.

The definition of a development is as follows.

**DEFINITION 4.4.** A rewrite sequence $\sigma : M \rightarrow N$ in $\Lambda$ is a development if there is a rewrite sequence $\rho$ in $\Delta$ that is an $E$-lifting of $\sigma$.

We give a new and short proof of finiteness of developments by considering another inductive definition of the set of all underlined $\lambda$-terms. Like the set $SN$ of strongly normalising $\lambda$-terms, this definition makes essential use of expansion.

**DEFINITION 4.5.** The set $D$ is the smallest set of $\lambda$-terms satisfying

1. $x \in D$ for all variables $x$,
2. if $M \in D$, then $\lambda x.M \in D$,
3. if $M \in D$ and $N \in D$, then $MN \in D$.
4. if $M[x := N] \in D$ and $N \in D$, then $(\Delta x.M)N \in D$.

It is easy to prove that $\Delta = D$. 
Lemma 4.6. If $M \in \mathcal{D}$ and $N \in \mathcal{D}$ then $M[x := N] \in \mathcal{D}$.

Proof. By induction on $M \in \mathcal{D}$. □

Proposition 4.7. $\Delta = \mathcal{D}$.

Proof.

1. Let $M \in \Delta$. We prove by induction on $M$ that $M \in \mathcal{D}$. We prove the case that $M = (\lambda x. P)Q$. By induction hypothesis, $P \in \mathcal{D}$ and $Q \in \mathcal{D}$. By Lemma 4.6 we have that $P[x := Q] \in \mathcal{D}$ and by the definition of $\mathcal{D}$ we have that $(\lambda x. P)Q \in \mathcal{D}$.

2. Let $M \in \mathcal{D}$. By induction on the derivation of $M \in \mathcal{D}$ we prove that $M \in \Delta$. We prove the case that $M = (\lambda x. P)Q$. By induction hypothesis, $P[x := Q] \in \Delta$ and $Q \in \Delta$. This yields $P \in \Delta$. Hence $(\lambda x. P)Q \in \Delta$.

□

So finiteness of developments is equivalent to the fact that each term in $\mathcal{D}$ is strongly $\beta$-normalising. We use the following lemma of which the proof is immediate.

Lemma 4.8. If $P$ in $PQ$ is not of the form $\Delta x. P_0$, then all $\beta$-reducts of $PQ$ are of the form $P'Q'$ with $P \rightarrow^*_\beta P'$ and $Q \rightarrow^*_\beta Q'$.

Theorem 4.9. If $M \in \mathcal{D}$, then all $\beta$-rewrite sequences starting in $M$ are finite.

Proof. The proof proceeds by induction on the derivation of $M \in \mathcal{D}$.

1. If $M$ is a variable then it is trivial.

2. Let $M = \lambda x. P$ with $P \in \mathcal{D}$. By induction hypothesis, we have that $P$ is strongly $\beta$-normalising. So $M$ is strongly $\beta$-normalising.

3. Let $M = PQ$ with $P \in \mathcal{D}$ and $Q \in \mathcal{D}$. Note that $P$ is not of the form $\Delta x. P_0$. By lemma 4.8, every $\beta$-reduct of $M$ is of the form $P'Q'$ with $P \rightarrow^*_\beta P'$ and $Q \rightarrow^*_\beta Q'$. By induction hypothesis there are no infinite $\beta$-rewrite sequences starting in $P$ or in $Q$. Therefore $M$ is strongly $\beta$-normalising.

4. Let $M = (\Delta x. P)Q$ with $P[x := Q] \in \mathcal{D}$ and $Q \in \mathcal{D}$. Consider an arbitrary $\beta$-rewrite sequence $\rho : M = M_0 \rightarrow^*_\beta M_1 \rightarrow^*_\beta M_2 \rightarrow^*_\beta \ldots$. There are two possibilities: in $\rho$ the head redex of $M$ is contracted or the head redex of $M$ is not contracted.

In the first case there is an $i$ such that $M_i = P'[x := Q']$, with $P \rightarrow^*_\beta P'$ and $Q \rightarrow^*_\beta Q'$. The term $M_i$ is a result of rewriting $P[x := Q]$, and the latter is by induction hypothesis strongly $\beta$-normalising. Hence $\rho$ is finite.

In the second case all terms in $\rho$ are of the form $(\Delta x. P')Q'$ with $P \rightarrow^*_\beta P'$ and $Q \rightarrow^*_\beta Q'$. By induction hypothesis, $P[x := Q]$ is strongly $\beta$-normalising, which yields that $P$ is strongly $\beta$-normalising, and moreover $Q$ is strongly $\beta$-normalising. Hence all terms in $\rho$ are strongly normalising so $\rho$ is finite.
COROLLARY 4.10. All $\beta$-developments are finite.

REMARK 4.11. It is possible to prove in a different way, also using the set $SN$, that all developments are finite. We define a morphism

$$I : (\Delta, \{\beta\} \times \{0, 1\}^*, \{\phi_{\beta}\}) \to (SN, \{\beta\} \times \{0, 1\}^*, \{\phi_{\beta}\})$$

Let Abs denote a distinguished variable. The definition of position has to be adapted, such that applications of Abs are not counted. Then, for instance $(Abs \lambda x.M)\phi = (\lambda x.M)\phi$. We leave out the details.

First we define $I_0 : \Delta \to SN$ by induction on the definition of $\Delta$ as follows:

1. $I_0(x) = x$,
2. $I_0(\lambda x.M) = Abs x. I(M)$
3. $I_0(MN) = I(M)I(N)$,
4. $I_0((\Delta x.M)N) = (\lambda x.I(M))I(N)$.

Next we define the mapping $I_1 : \{\beta\} \times \{0, 1\}^* \to \{\beta\} \times \{0, 1\}^*$ by $I_1(\beta, \phi) = (\beta, \phi)$.

We have the following:

1. if $M \in \Delta$ then $I_0(M) \in SN$,
2. if $M \in \Delta$ and $M \to_{\beta} N$, then $I_0(M) \to_{\beta} I_0(N)$.

For the first point we need to prove that $I_0(M[x := N]) = I(M)[x := I_0(N)]$.

5. SUPERDEVELOPMENTS

In [Raa93], superdevelopments were introduced and proved to be finite. Superdevelopments form an extension of the notion of development. In a superdevelopment not only redexes that descend from the initial term may be contracted, but also some redexes that are created during reduction.

There are three ways of creating new redexes (see [Lév78]):

1. $$((\lambda x.\lambda y.M)N)P \to_{\beta} (\lambda y.M[x := N])P$$
2. $$(\lambda x.x)(\lambda y.M)N \to_{\beta} (\lambda y.M)N$$
3. $$(\lambda x.C[xM])(\lambda y.N) \to_{\beta} C'[\lambda y.NM']$$ where $C'$ and $M'$ are obtained from $C$ and $M$ by replacing all free occurrences of $x$ by $(\lambda y.N)$.

The first two kinds of created redexes are 'innocent' and they may be contracted in a superdevelopment. Here the redexes are created 'upwards', whereas in the last case redexes are created 'downwards'. The result that all superdevelopments are finite shows that infinite $\beta$-reduction sequences are due to the presence of the third type of redexes.

In this section we give a new proof of the fact that in $\lambda$-calculus all $\beta$-superdevelopments terminate.

First we shortly repeat the definition of a superdevelopment. The definition makes use of a set of labelled $\lambda$-terms and a notion of labelled $\beta$-reduction on it. Since application nodes will be labelled, we write them explicitly.
DEFINITION 5.1. The set $\Lambda_I$ of labelled $\lambda$-terms is defined as the smallest set satisfying the following:

1. $x \in \Lambda_I$ for every variable $x$,
2. if $M \in \Lambda_I$ and $i \in \mathbb{N}$, then $\lambda_i x . M \in \Lambda_I$,
3. if $M, N \in \Lambda_I$ and $X \subseteq \mathbb{N}$, then $@^X(M, N) \in \Lambda_I$.

On the set $\Lambda_I$, the $\beta_i$-rule is defined as follows:

$@^X(\lambda_i x . M, N) \rightarrow M[x := N]$ if $i \in X$

DEFINITION 5.2.

1. A term $M \in \Lambda_I$ is said to be well-labelled if the label $X$ of an application node never contains the label $i$ of a $\lambda$ outside the scope of the application node.
2. A term $M \in \Lambda_I$ is initially labelled if it is well-labelled and all $\lambda$'s have a different label.

The set of well-labelled terms is closed under $\beta_i$-reduction. In the sequel we shall suppose terms in $\Lambda_I$ to be well-labelled. We define a mapping from $\Lambda_I$ to $\Lambda$ that erases the labels.

DEFINITION 5.3.

1. The mapping $\mathcal{E}_0 : \Lambda_I \rightarrow \Lambda$ is defined by induction on the definition of $\Lambda_I$.
   (a) $\mathcal{E}_0(x) = x$,
   (b) $\mathcal{E}_0(\lambda_i x . M) = \lambda x . \mathcal{E}_0(M)$,
   (c) $\mathcal{E}_0(@^X(M, N)) = \mathcal{E}_0(M)\mathcal{E}_0(N)$.
2. The mapping $\mathcal{E}_1 : \{\beta_i\} \times \{0, 1\}^* \rightarrow \{\beta\} \times \{0, 1\}^*$ is defined by $\mathcal{E}_1(\beta, \phi) = (\beta, \phi)$.

The proof of the following lemma is straightforward.

LEMMA 5.4. The mapping $\mathcal{E} = (\mathcal{E}_0, \mathcal{E}_1)$ is a morphism from $(\Lambda_I, \{\beta_i\} \times \{0, 1\}^*, \{-\phi_{\beta_i}\} \phi)$ to $(\Lambda, \{\beta\} \times \{0, 1\}^*, \{-\phi\} \phi)$.

The definition of a superdevelopment is as follows.

DEFINITION 5.5. A rewrite sequence $\sigma : M \rightarrow_\beta N$ in $\Lambda$ is a superdevelopment if there is a rewrite sequence $\rho$ in $\Lambda_I$ that starts in an initially labelled term and that is an $\mathcal{E}$-lifting of $\sigma$.

We give a new proof of the fact that all superdevelopments are finite. It is similar to the proof of finite developments in section 4.

DEFINITION 5.6. The set $\mathcal{S}D$ is the smallest subset of the set of lambda terms satisfying

1. $x \in \mathcal{S}D$ for all variables $x$, 

The $\beta$-rewrite relation is defined as follows:

$$(\lambda x.M)N \rightarrow_{\beta} M[x := N]$$

We need to prove that a rewrite sequence $M \rightarrow_{\beta} N$ can be lifted to a $\beta$-rewrite sequence starting in $M$ with an initial labelling if and only if it can be lifted to a $\beta$-rewrite sequence starting with $M \in SD$. To prove this is routine. Then, proving that all superdevelopments are finite is equivalent to proving that all $\beta$-rewrite sequences in $SD$ are finite. The latter is proved by a trivial induction on the set $SD$ using the following two lemma's.

**Lemma 5.7.** ($\beta$-Closure) Let $M \in SD$. If $M \rightarrow_{\beta} M'$ then $M' \in SD$.

**Lemma 5.8.** Let $M = PQ$ with $P \in SD$ and $Q \in SD$. If $M \rightarrow_{\beta} M'$, then $M' = P'Q'$ with $P \rightarrow_{\beta} P'$ and $Q \rightarrow_{\beta} Q'$.

**Theorem 5.9.** If $M \in SD$, then all $\beta$-rewrite sequences starting at $M$ are finite.

**Proof.** The proof proceeds by induction on the derivation of $M \in SD$.

1. If $M$ is a variable then it is trivial.

2. Let $M = \lambda x.P$ with $P \in SD$. By induction hypothesis, we have that $P$ is strongly $\beta$-normalising. So $M$ is strongly $\beta$-normalising.

3. Let $M = PQ$ with $P \in SD$ and $Q \in SD$. By induction hypothesis $P$ and $Q$ are $\beta$-strongly normalising. It follows from lemma 5.8 that any $\beta$-sequence starting at $M$ is finite.

4. Let $M = (\lambda x.P)QN_1 \ldots N_n$ with $P[x := Q]N_1 \ldots N_n \in SD$. Consider an arbitrary $\beta$-rewrite sequence $\rho : M = M_0 \rightarrow_{\beta} M_1 \rightarrow_{\beta} M_2 \rightarrow_{\beta} \ldots$. There are two possibilities: in $\rho$ the head redex of $M$ is contracted or the head redex of $M$ is not contracted.

In the first case, there is an $i$ such that $M_i = P'[x := Q']N'_1 \ldots N'_n$ with $P \rightarrow_{\beta} P'$, $Q \rightarrow_{\beta} Q'$, $N_1 \rightarrow_{\beta} N'_1$, $N_n \rightarrow_{\beta} N'_n$. The term $M_i$ is obtained by rewriting $P[x := Q]N_1 \ldots N_n$ and the latter term is by induction hypothesis strongly $\beta$-normalising. Hence $\rho$ is finite.

In the second case, all terms in $\rho$ are of the form $(\lambda x.P')Q'N'_1 \ldots N'_n$ with $P \rightarrow_{\beta} P'$, $Q \rightarrow_{\beta} Q'$, $N_1 \rightarrow_{\beta} N'_1$, $N_n \rightarrow_{\beta} N'_n$. Since $P[x := Q]N_1 \ldots N_n$ and $Q$ are by induction hypothesis strongly $\beta$-normalising, we have that $P, Q, N_1, \ldots, N_n$ are strongly $\beta$-normalising. So all terms in $\rho$ are strongly $\beta$-normalising and hence $\rho$ is finite.
5. Superdevelopments

Let $M = (\lambda x.x)(\lambda y.N)PN_1 \ldots N_n$ with $(\lambda y.N)PN_1 \ldots N_n \in SD$. Consider an arbitrary $\beta$-rewrite sequence $\rho: M = M_0 \rightarrow_\beta M_1 \rightarrow_\beta M_2 \rightarrow_\beta \ldots$. There are two possibilities: in $\rho$ the head redex of $M$ is contracted or the head redex of $M$ is not contracted.

In the first case, there is an $i$ such that $M_i = (\lambda y.N')P'N_{i1} \ldots N_n'$ with $N \rightarrow_\beta N', P \rightarrow \rightarrow_\beta P', N_1 \rightarrow_\beta N_{i1}', \ldots, N_n \rightarrow_\beta N_n'$. The term $M_i$ is obtained by rewriting the term $(\lambda y.N)PN_1 \ldots N_n$ and the latter term is by induction hypothesis strongly $\beta$-normalising. So $M_i$ is strongly $\beta$-normalising and hence $\rho$ is finite.

In the second case, all terms in $\rho$ are of the form $(\lambda y.N)PN_1 \ldots N_n$ with $N \rightarrow_\beta N', P \rightarrow \rightarrow_\beta P', N_1 \rightarrow_\beta N_{i1}', \ldots, N_n \rightarrow_\beta N_n'$. By induction hypothesis, $(\lambda y.N)PN_1 \ldots N_n$ is strongly $\beta$-normalising. Hence $N, P, N_1, \ldots, N_n$ are all strongly $\beta$-normalising. This yields that $\rho$ is finite.

$\square$

Remark 5.10. Another proof of the fact that all superdevelopments are finite can be given in a way similar to the one in Remark 5.9. That is, we define a morphism from $(\Lambda_i, \{\beta_i\} \times \{0,1\}^*, \{\phi_i\}, \{\psi_i\})$ to $(SN, \{\beta\} \times \{0,1\}^*, \{\phi\}, \{\psi\})$, using the following function.

Definition 5.11. The function $| - |: \Lambda_i \rightarrow \Lambda_i$ is defined by induction on the definition of $\Lambda_i$.

1. $|x| = x$,
2. $|\lambda_i x.M| = \lambda_i x.|M|$, 
3. $|\otimes^X(M, N)| = \begin{cases} M_0[x := |N|] & \text{if } |M| = \lambda_i x.M_0 \text{ and } i \in X \\ \otimes^X(|M|, |N|) & \text{otherwise} \end{cases}$

Let App denote a distinguished variable. Like in the case of developments, the definition of position has to be adapted, such that occurrences of App do not count. Then, for instance $(\text{App}MN) |\phi = (MN) |\phi$. As in the case of developments, we leave out the details. Now we can define the morphism $J: (\Lambda_i, \{\beta_i\} \times \{0,1\}^*, \{\phi_i\}, \{\psi_i\}) \rightarrow (SN, \{\beta\} \times \{0,1\}^*, \{\phi\}, \{\psi\})$.

Definition 5.12.

1. The mapping $J_0: \Lambda_i \rightarrow SN$ is defined by induction on the definition of $\Delta$.
   (a) $J_0(x) = x$, 
   (b) $J_0(\lambda_i x.M) = \lambda x.J_0(M)$, 
   (c) $J_0(\otimes^X(M, N)) = \begin{cases} J_0(M)J_0(N) & \text{if } |M| = \lambda_i x.M_0 \text{ and } i \in X \\ \text{App}J_0(M)J_0(N) & \text{otherwise} \end{cases}$

2. The mapping $J_1: \{\beta_i\} \times \{0,1\}^* \rightarrow \{\beta\} \times \{0,1\}^*$ is defined by $E_1(\beta_i, \phi) = (\beta, \phi)$.

Lemma 5.13. Let $M \in \Lambda_i$, $J(M) \in SN$ and $N \in SN$. Then $J(M)[x := N] \in SN$.

Proof. The proof proceeds by induction on the derivation of $J(M) \in SN$. 

5. Superdevelopments

1. Suppose $J(M) = yP_1 \ldots P_n$ with $P_i \in SN$ for $i = 1, \ldots, n$. Note that it must be the case that $y = App$. By induction hypothesis, $J(P_i)[x := N] \in SN$ for $i = 1, \ldots, n$. Hence $J(M)[x := N] \in SN$.

2. Suppose $J(M) = \lambda y.P$ with $P \in SN$. Using induction hypothesis we obtain that $J(M)[x := N] \in SN$.

3. Suppose $J(M) = (\lambda y.P)Q_1Q_2 \ldots Q_n$ with $P[y := Q_1]Q_2 \ldots Q_n \in SN$ and $Q_1 \in SN$. By induction hypothesis, we have $(P[y := Q_1]Q_2 \ldots Q_n)[x := N] \in SN$ and $Q_1[x := N] \in SN$. This yields $J(M)[x := N] \in SN$.

\[\square\]

**Lemma 5.14.** Let $M \in \Lambda_1$, $J(M) \in SN$ and $N \in SN$. Then $J(M)N \in SN$.

**Proof.** The proof proceeds by induction on the derivation of $J(M) \in SN$.

1. Suppose $J = xP_1 \ldots P_n$ with $P_i \in SN$ for $i = 1, \ldots, n$. Then $J(M)N \in SN$.

2. Suppose $J(M) = \lambda x.P$ with $P \in SN$. Then $M = \lambda x.M_0$ and $J(M_0) = P$. By the previous lemma we have $P[x := N] \in SN$. Hence $J(M)N \in SN$.

3. Suppose $J(M) = (\lambda x.P)Q_1Q_2 \ldots Q_n$ with $P[x := Q_1]Q_2 \ldots Q_n \in SN$ and $Q_1 \in SN$. By induction hypothesis, we have $P[x := Q_1]Q_2 \ldots Q_nN \in SN$. Moreover $Q_1 \in SN$, hence $J(M)N \in SN$.

\[\square\]

**Theorem 5.15.** Let $M \in \Lambda_1$. Then $J(M) \in SN$.

**Proof.** The proof proceeds by induction on $M \in \Lambda_1$ and makes use of the two previous lemmas. \[\square\]

**Theorem 5.16.** Let $M \in \Lambda_1$. If $M \rightarrow^\phi \beta_1 N$ in $\Lambda_1$ then $J(M) \rightarrow^\phi \beta J(N)$ in $SN$.

It follows from Theorem 5.15 and Theorem 5.16 that $J$ is a morphism from $(\Lambda_1, \{\rightarrow^\phi \beta\}_\phi)$ to $(SN, \{\rightarrow^\phi \beta\}_\phi)$. As an immediate consequence of this, we have that all superdevelopments are finite.

Finally we would like to remark that it is easy to prove the following lemma.

**Lemma 5.17.** Let $M \in SD$.

1. $M \rightarrow^\beta [M]$.

2. If $M \rightarrow^\beta M'$ then $|M| = |M'|$.

So we have $|M|$ is the $\beta_1$-normal form of $M$ and it is unique. As a consequence of this we have that $\beta_1$ is Church-Rosser.
6. Two Perpetual Strategies

In this section we consider two rewrite strategies, $F_{bk}$ defined in [BK82] and $F_{\infty}$ introduced in [BBKV76]. Both strategies are perpetual, which means that they yield an infinite rewrite sequence whenever possible. We give for both strategies a new proof of the fact that they are perpetual. Our proofs are simpler than the original ones and make in both cases use of the characterisation of strongly normalising terms.

For the sake of self-containment we first give some definitions that can for instance be found in [Bar84].

Perpetual strategies.

**Definition 6.1.**

1. A *strategy for $\beta$-reduction* is a map $F : \Lambda \rightarrow \Lambda$ such that for all $M \in \Lambda$, $M \rightarrow_{\beta} F(M)$.

2. A *one-step strategy for $\beta$-reduction* is a map $F : \Lambda \rightarrow \Lambda$ such that for all $M \in \Lambda$ not in $\beta$-normal form, $M \rightarrow_{\beta} F(M)$.

**Definition 6.2.** A strategy is called perpetual if $F(M)$ is strongly normalising implies $M$ is strongly normalising.

A perpetual strategy finds an infinite rewrite sequence if possible. Perpetual strategies are interesting because of the easy observation that a term $M$ is strongly normalising if and only if a perpetual strategy finds a finite rewrite sequence starting from $M$.

In the sequel we will deal with one-step strategies only.

**Definition 6.3.** Let $F$ be a strategy for $\beta$-reduction. An $F$-rewrite sequence of $M$ is defined as

$$M \rightarrow_{\beta} F(M) \rightarrow_{\beta} F^2(M) \rightarrow_{\beta} \ldots$$

possibly ending in the normal form of $M$.

**The strategy $F_{bk}$.** First we consider the strategy $F_{bk}$ as introduced in [BK82]. We give a simple proof that $F_{bk}$ is perpetual using Definition 3.1.

**Definition 6.4.** Suppose that $M \in \Lambda$ is not in normal form.

Let $M = C[(\lambda x. P)Q]$ where $(\lambda x. P)Q$ is the leftmost redex of $M$.

$$F_{bk}(C[(\lambda x. P)Q]) = \begin{cases} 
C[P[x := Q]] & \text{if } Q \text{ is strongly normalising} \\
C[(\lambda x. P)F_{bk}(Q)] & \text{otherwise}
\end{cases}$$

Note that the the strategy $F_{bk}$ yields standard rewrite sequences.

**Theorem 6.5.** $F_{bk}$ is a perpetual strategy.

**Proof.** Suppose that $M$ is not in normal form and $F_{bk}(M)$ is strongly normalising. Then $F_{bk}(M) \in SN$. We prove $M \in SN$, which is equivalent to $M$ is strongly normalising. We prove $M \in SN$ by induction on the number of steps in the derivation of $F_{bk}(M) \in SN$.

The term $M$ is of the form $\lambda x_1 \ldots x_m. P Q_1 \ldots Q_m$ where $P$ can be either a variable $y$ or an abstraction $\lambda y. P_0$. We consider these two cases:
1. \( P = y \). Then
\[
F_{bk}(M) = \lambda x_1 \ldots \lambda x_n.y \; Q_1 \ldots Q_{i-1} F_{bk}(Q_i)Q_{i+1} \ldots Q_m
\]
where \( Q_1, \ldots Q_{i-1} \) are in normal form.
Since \( F_{bk}(M) \in SN \) we have that \( F_{bk}(Q_i) \in SN \). It follows from the induction hypoth­

2. \( P = \lambda y.P_0 \). Suppose \( Q_1 \) is not strongly normalising. Then \( Q_1 \not\in SN \). This yields a contradiction with the hypothesis \( F_{bk}(M) \in SN \). Hence \( Q_1 \in SN \) and
\[
F_{bk}(M) = \lambda x_1 \ldots \lambda x_n.P_0[y := Q_1]Q_2 \ldots Q_m
\]
Now \( F_{bk}(M) \in SN \) and \( Q_1 \in SN \) yield that \( M \in SN \).

The strategy \( F_\infty \). We now consider the strategy \( F_\infty \) that is defined in [BBKV76]. This strategy does not check whether the argument of the left-most redex is strongly normalising or not. Instead, it is checked whether the left-most redex is an \( I \)-redex. If it is, it is contracted. If it is not, contracting it could imply loosing the possibility of having an infinite reduction sequence. Therefore, in that case, the left-most redex is only contracted if the argument is a normal form. If the argument is not a normal form, the strategy is applied to the argument.

**Definition 6.6.** Suppose that \( M \in \Lambda \) is not in normal form.
Let \( M = C[(\lambda x.P)Q] \) where \((\lambda x.P)Q\) is the leftmost redex of \( M \).
\[
F_\infty(C[(\lambda x.P)Q]) = \begin{cases} 
C[(\lambda x.P)F_\infty(Q)] & \text{if } x \not\in \text{FV}(P) \text{ and } Q \not\in NF \\
C[P[x := Q]] & \text{otherwise}
\end{cases}
\]
The \( F_\infty \)-rewrite sequence of a term is not necessarily a standard rewrite sequence. The merit of \( F_\infty \), however, is that it is decidable. We prove that \( F_\infty \) is perpetual. Note that this proof is simpler than the proof in [BBKV76] or in [Bar84] (Chapter IV paragraph 4).

**Theorem 6.7.** \( F_\infty \) is a perpetual strategy.

**Proof.** Suppose that \( M \) is not in normal form and \( F_\infty(M) \) is strongly normalising. Then \( F_\infty(M) \in SN \). We prove that \( M \in SN \) which means that \( M \) is strongly normalising. The proof proceeds by induction on the derivation of \( F_\infty(M) \in SN \).

The term \( M \) is of the form \( \lambda x_1 \ldots \lambda x_n.PQ_1 \ldots Q_m \) where \( P \) can be either a variable \( y \) or an abstraction \( \lambda y.P_0 \). We consider these two cases:

1. \( P = y \). Then
\[
F_\infty(M) = \lambda x_1 \ldots \lambda x_n.y \; Q_1 \ldots Q_{i-1} F_\infty(Q_i)Q_{i+1} \ldots Q_m
\]
where \( Q_1, \ldots Q_{i-1} \) are in normal form.
Since \( F_\infty(M) \in SN \) we have that \( F_\infty(Q_i) \in SN \). It follows from the induction hypothesis that \( Q_i \in SN \). This yields \( M = \lambda x_1 \ldots \lambda x_n.yQ_1 \ldots Q_m \in SN \).
7. A Maximal Strategy

2. \( P = \lambda y. P_0 \). Two cases are distinguished.

(a) \( y \in \text{FV}(P_0) \). Then

\[
F_\infty (M) = \lambda x_1 \ldots \lambda x_n. P_0[y := Q_1]Q_2 \ldots Q_m \in SN
\]

Since \( y \in P_0 \) we have that \( Q_1 \in SN \). By the definition of the set \( SN \) we have that \( M \in SN \).

(b) \( y \not\in \text{FV}(P_0) \).

If \( Q_1 \) is a normal form, then

\[
F_\infty (M) = \lambda x_1 \ldots \lambda x_n. P_0 Q_2 \ldots Q_m
\]

Since \( F_\infty (M) \in SN \), that is, \( \lambda x_1 \ldots \lambda x_n. P_0[y := Q_1]Q_2 \ldots Q_m \in SN \), and moreover clearly \( Q_1 \in SN \), we can conclude \( M = \lambda x_1 \ldots \lambda x_n. (\lambda y. P_0)Q_1 Q_2 \ldots Q_m \in SN \).

If \( Q_1 \) is not a normal form, then

\[
F_\infty (M) = \lambda x_1 \ldots \lambda x_n. (\lambda y. P_0) F_\infty (Q_1) Q_2 \ldots Q_m
\]

Since \( F_\infty (M) \in SN \) we have \( F_\infty (Q_1) \in SN \) and \( P_0 Q_2 \ldots Q_m \in SN \). By induction hypothesis we have that \( Q_1 \in SN \).

We apply the last clause of Definition 3.1 in order to obtain \( (\lambda y. P_0)Q_1 Q_2 \ldots Q_m \in SN \). We have \( M = \lambda x_1 \ldots x_n. (\lambda y. P_0)Q_1 Q_2 \ldots Q_m \in SN \) by applying \( n \) times the second clause of Definition 3.1.

\[\Box\]

7. A Maximal Strategy

In this section we prove that the strategy \( F_\infty \) is maximal, which means that it computes for each term \( M \) the longest possible rewrite sequence. In particular, a maximal strategy is perpetual. The converse is not necessarily true, as witnessed by the strategy \( F_{bk} \) defined in [BK82].

Our proof that \( F_\infty \) is a maximal strategy makes use of the characterisation of strongly normalising terms. We define a mapping \( h \) that computes the length of a \( F_\infty \)-rewrite sequence of a term. Then it is proved that the mapping \( h \) computes the maximal length of a reduction to normal form.

We start by giving some definitions.

**Definition 7.1.** Let \( \sigma \) be a rewrite sequence. The length of \( \sigma \), denoted by \( \| \sigma \| \), is the number of rewrite steps in \( \sigma \). We have that \( \| \sigma \| \) is either a natural number or \( \infty \).

**Definition 7.2.** A rewrite sequence \( \sigma : M \rightarrow N \) is maximal if for all \( \rho : M \rightarrow N \) we have \( \| \sigma \| \geq \| \rho \| \).

**Definition 7.3.** A strategy \( F \) is maximal for each term \( M \) the \( F \)-rewrite sequence of \( M \) is maximal.
We define a map \( h : \Lambda \rightarrow \mathbb{N} \cup \{\infty\} \) that computes for each term the length of its \( F_\infty \)-rewrite sequence.

**Definition 7.4.**

1. The map \( h : SN \rightarrow \mathbb{N} \) is defined by induction on the definition of \( SN \).

\[
\begin{align*}
 h(xM_1 \ldots M_n) &= \begin{cases} 
 0 & \text{if } n = 0 \\
 \sum_{i=1}^{n} h(M_i) & \text{if } n \neq 0
\end{cases} \\
 h(\lambda x.M) &= h(M)
\end{align*}
\]

\[
\begin{align*}
 h((\lambda x.M)NP_1 \ldots P_n) &= \begin{cases} 
 h(M[x := N]P_1 \ldots P_n) + 1 & \text{if } x \in M \\
 h(M)P_1 \ldots P_n + h(N) + 1 & \text{if } x \notin M
\end{cases}
\end{align*}
\]

2. We extend \( h : SN \rightarrow \mathbb{N} \) to \( h : \Lambda \rightarrow \mathbb{N} \cup \{\infty\} \) by defining \( h(M) = \infty \) if \( M \notin SN \).

We prove that the map \( h \) has the following two properties:

- it computes the length of the \( F_\infty \)-rewrite sequence of a term \( M \),
- it computes the maximum length of all rewrite sequences starting in \( M \).

From these we will conclude that \( F_\infty \) is a maximal strategy.

First we prove the following lemma.

**Lemma 7.5.** Let \( M \in SN \).

1. If \( M \in NF \) then \( h(M) = 0 \).
2. If \( M \notin NF \) then \( h(M) = h(F_\infty(M)) + 1 \).

**Proof.**

1. Trivial.

2. Suppose that \( M \) is not in normal form. We prove that \( h(M) = h(F_\infty(M)) + 1 \) by induction on the number of steps in the derivation of \( M \in SN \).

The term \( M \) is of the form \( \lambda x_1 \ldots \lambda x_n.PQ_1 \ldots Q_m \) where \( P \) can be either a variable \( y \) or an abstraction \( \lambda y.P_0 \). We consider these two cases:

(a) \( P = y \). Then

\[
F_\infty(M) = \lambda x_1 \ldots \lambda x_n.y Q_1 \ldots Q_{i-1}F_\infty(Q_i)Q_{i+1} \ldots Q_m
\]
where $Q_1, \ldots, Q_{i-1}$ are in normal form. By induction hypothesis we have $h(Q_i) = h(F_{\infty}(Q_i)) + 1$. Hence we have

\[
h(M) = h(\lambda x_1 \ldots \lambda x_n, y.Q_1 \ldots Q_m) = \sum_{k=i}^{m} h(Q_k) = h(Q_i) + \sum_{k=i+1}^{m} h(Q_k) = h(F_{\infty}(Q_i)) + 1 + \sum_{k=i+1}^{m} h(Q_k) = h(F_{\infty}(M)) + 1
\]

(b) $P = (\lambda y.P_0)$. Two cases are distinguished.

i. $y \in \text{FV}(P_0)$. Then

\[
F_{\infty}(M) = \lambda x_1 \ldots \lambda x_n, P_0[y := Q_1] Q_2 \ldots Q_m
\]

We have

\[
h(M) = h(\lambda x_1 \ldots \lambda x_n, (\lambda y.P_0) Q_1 Q_2 \ldots Q_m) = h(P_0[y := Q_1] Q_2 \ldots Q_m) + 1 = h(F_{\infty}(M)) + 1
\]

ii. $y \notin \text{FV}(P_0)$. Again two cases are distinguished.

A. If $Q_1$ is not in normal form then

\[
F_{\infty}(M) = \lambda x_1 \ldots \lambda x_n, (\lambda y.P_0) F_{\infty}(Q_1) Q_2 \ldots Q_m
\]

By induction hypothesis, $h(Q_1) = h(F_{\infty}(Q_1)) + 1$. Hence we have

\[
h(M) = h(\lambda x_1 \ldots \lambda x_n, (\lambda y.P_0) Q_1 Q_2 \ldots Q_m) = h(P_0 Q_2 \ldots Q_m) + h(Q_1) + 1 = h(P_0 Q_2 \ldots Q_m) + h(F_{\infty}(Q_1)) + 1 + 1 = h(F_{\infty}(M)) + 1
\]

B. If $Q_1$ is in normal form then

\[
F_{\infty}(M) = \lambda x_1 \ldots \lambda x_n, P_0 Q_2 \ldots Q_m
\]

We have

\[
h(M) = h(\lambda x_1 \ldots \lambda x_n, (\lambda y.P_0) Q_1 Q_2 \ldots Q_m) = h(P_0 Q_2 \ldots Q_m) + h(Q_1) + 1 = h(P_0 Q_2 \ldots Q_m) + 0 + 1 = h(F_{\infty}(M)) + 1
\]
Theorem 7.6. The map \( h : \Lambda \to \mathbb{N} \cup \{\infty\} \) computes the length of the \( F_\infty \)-rewrite sequence of a term \( M \).

Proof. If \( M \in SN \) then \( M \to_\beta F_\infty(M) \to_\beta \ldots \to_\beta F_\infty^n(M) = \text{nff}(M) \). It follows by induction on \( n \) that \( h(M) = n \), using Lemma 7.5.

If \( M \notin SN \) then the \( F_\infty \)-rewrite sequence of \( M \) is infinite and indeed \( h(M) = \infty \).

Now we prove that \( h : \Lambda \to \mathbb{N} \cup \{\infty\} \) computes the maximum length of all reductions sequences starting at \( M \). Here \( \text{maxred}(M) \) denotes the length of a maximal rewrite sequence starting in \( M \).

Theorem 7.7. Let \( M \in \Lambda \). We have

\[
h(M) = \text{maxred}(M)
\]

Proof. If \( M \notin SN \), then \( h(M) = \infty \), so it is clear that the statement holds.

Suppose that \( M \in SN \) is not in normal form. We will prove that the length of an arbitrary reduction to normal form is less than or equal to \( h(M) \). The proof proceeds by induction on the number of steps in the derivation of \( M \in SN \). The term \( M \) is of the form \( \lambda x_1 \ldots x_n. P Q_1 \ldots Q_m \) where \( P \) can be either a variable \( y \) or an abstraction \( \lambda y . P_0 \). We consider these two cases:

1. \( P = y \). An arbitrary reduction from \( M \) to normal form can be transformed into a reduction sequence of the same length such that:

\[
\lambda x_1 \ldots \lambda x_n . y Q_1 \ldots Q_m \xrightarrow{n_1} \lambda x_1 \ldots \lambda x_n . y \text{nff}(Q_1) Q_2 \ldots Q_m \\
\xrightarrow{n_2} \lambda x_1 \ldots \lambda x_n . y \text{nff}(Q_1) \text{nff}(Q_2) \ldots (Q_m) \\
\cdots \\
\xrightarrow{n_m} \lambda x_1 \ldots \lambda x_n . y \text{nff}(Q_1) \text{nff}(Q_2) \ldots \text{nff}(Q_m)
\]

The number of steps of this sequence is \( n_1 + \ldots + n_m \). By induction hypothesis, we have \( h(Q_i) \geq n_i \) for \( i = 1, \ldots, m \). Hence we have

\[
h(M) = \sum_{i=1}^{m} h(Q_i) \\
\geq \sum_{i=1}^{m} n_i
\]

2. \( P = \lambda y . P_0 \). Two cases are distinguished.

(a) \( y \in \text{FV}(P_0) \). An arbitrary reduction sequence from \( M \) to normal form is of the form

\[
\begin{align*}
M &= \lambda x_1 \ldots \lambda x_n . (\lambda y . P_0) Q_1 Q_2 \ldots Q_m \\
&\xrightarrow{\rho} \lambda x_1 \ldots \lambda x_n . (\lambda y . P'_0) Q'_1 Q'_2 \ldots Q'_m \\
&\xrightarrow{\beta} \lambda x_1 \ldots \lambda x_n . P'_0[y := Q'_1] Q'_2 \ldots Q'_m \\
&\xrightarrow{\beta} \text{nff}(M)
\end{align*}
\]
It can be transformed into a rewrite sequence of the form

\[ M = \lambda x_1 \ldots \lambda x_n. (\lambda y. P_0) Q_1 Q_2 \ldots Q_m \]

\[ \rightarrow_\beta \lambda x_1 \ldots \lambda x_n. P_0[y := Q_1] Q_2 \ldots Q_m \]

\[ \rightarrow_\beta^k \lambda x_1 \ldots \lambda x_n. P_0'[y := Q'_1] Q'_2 \ldots Q'_m \]

\[ \rightarrow^l_\beta \text{nf}(M) \]

with \( k \geq p \). By induction hypothesis, \( h(P_0[y := Q_1] Q_2 \ldots Q_m) \geq k + l \). Hence

\[ h(M) = h(P_0[y := Q_1] Q_2 \ldots Q_m) + 1 \]

\[ \geq k + l + 1 \]

\[ \geq p + l + 1 \]

(b) \( y \notin P_0 \). An arbitrary reduction sequence from \( M \) to normal form can be transformed into a reduction sequence of the same length of the form:

\[ M = \lambda x_1 \ldots \lambda x_n. (\lambda y. P_0) Q_1 Q_2 \ldots Q_m \]

\[ \rightarrow_\beta \lambda x_1 \ldots \lambda x_n. (\lambda y. P_0) Q'_1 Q'_2 \ldots Q'_m \]

\[ \rightarrow_\beta \lambda x_1 \ldots \lambda x_n. P_0 Q_2 \ldots Q_m \]

\[ \rightarrow^l_\beta \text{nf}(M) \]

By induction hypothesis we have that \( h(Q_1) \geq p \) and \( h(P_0 Q_2 \ldots Q_m) \geq l \). Hence

\[ h(M) = h(P_0 Q_2 \ldots Q_m) + h(Q_1) + 1 \]

\[ \geq l + p + 1 \]

\[ \square \]

**Theorem 7.8.** The strategy \( F_{\infty} \) is maximal.

**Proof.**

1. By Theorem 7.6 we have that \( h(M) \) is the length of the \( F_{\infty} \)-rewrite sequence of \( M \).

2. By Theorem 7.7 we have that \( h(M) = \text{maxred}(M) \) is the maximum length of all reductions sequences starting at \( M \).

Hence the strategy \( F_{\infty} \) is maximal. \( \square \)

We can also define a map \( h' : \Lambda \rightarrow \mathbb{N} \cup \{\infty\} \) that computes the length from \( M \) to normal form using the strategy \( F_{uk} \).

**Definition 7.9.**

1. The map \( h' : \mathcal{SN} \rightarrow \mathbb{N} \) is defined by induction on the definition of \( \mathcal{SN} \).
8. Normalisation of Simply Typed A-calculus

\[ h'(x M_1 \ldots M_n) = \sum_{i=1}^{i=n} h'(M_i) \]
\[ h'(\lambda x. M) = h'(M) \]
\[ h'((\lambda x. M)NP_1 \ldots P_n) = h'(M[x := N]P_1 \ldots P_n) + 1 \]

2. The map \( h' : SN \rightarrow N \) is extended to \( h' : \Lambda \rightarrow N \cup \{\infty\} \) by defining \( h'(M) = \infty \) if \( M \notin SN \).

**Lemma 7.10.** The map \( h' : \Lambda \rightarrow N \cup \{\infty\} \) computes the length of the \( \beta \)-rewrite sequence of a term \( M \).

Note that \( h'(M) \leq h(M) \) for any term \( M \).

8. Normalisation of Simply Typed \( \lambda \)-calculus

In this section we give a new proof of the fact that the simply typed \( \lambda \)-calculus is \( \beta \)-strongly normalising.

In the proof we make use of the characterisation of the strongly normalising (untyped) \( \lambda \)-terms. We do not need to consider an interpretation for simply typed \( \lambda \)-terms.

First we shortly repeat the definitions of simply typed \( \lambda \)-calculus à la Curry.

Types, written as \( \alpha, \beta, \gamma, \ldots \), are built from type variables and the type constructor \( \rightarrow \).

**Definition 8.1.** The simply typed lambda calculus \( \rightarrow \)-Curry (or just \( \lambda \)) is defined by the notion of type derivation \( \Gamma \vdash \lambda \alpha \rightarrow \beta \) given by the following rules:

Start
\[ \Gamma \vdash x : \alpha \text{ if } x : \alpha \in \Gamma \]

\( \rightarrow \)-Introduction
\[ \frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x. M : \alpha \rightarrow \beta} \]

\( \rightarrow \)-Elimination
\[ \frac{\Gamma \vdash M : \alpha \rightarrow \beta \quad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta} \]

If \( A \) and \( B \) are subsets of \( \Lambda \), then we define
\[ A \rightarrow B = \{ M \in \Lambda \mid \forall N \in A : MN \in B \} \]

Note that if \( A \subset A' \) then \( A' \rightarrow B \subset A \rightarrow B \) and that if \( B \subset B' \) then \( A \rightarrow B \subset A \rightarrow B' \).

We wish to consider the set of terms that are typable by a type \( \alpha \) in a context \( \Gamma \).

**Definition 8.2.** \( T(\Gamma; \alpha) = \{ M \in \Lambda \mid \Gamma \vdash M : \alpha \} \).

**Lemma 8.3.** Let \( T(\Gamma; \alpha) \neq \emptyset \). Then \( T(\Gamma; \alpha \rightarrow \beta) = T(\Gamma; \alpha) \rightarrow T(\Gamma; \beta) \).

**Proof.**

\( \vdash \) Let \( M \in T(\Gamma; \alpha \rightarrow \beta) \) and \( N \in T(\Gamma; \alpha) \). Clearly \( \Gamma \vdash MN : \beta \).
8. Normalisation of Simply Typed $\lambda$-calculus

Let $M \in T(\Gamma; \alpha) \rightarrow T(\Gamma; \beta)$. Since $T(\Gamma, \alpha) \neq \emptyset$, there is some $N \in T(\Gamma, \alpha)$. Hence $MN \in T(\Gamma; \beta)$. This yields $\Gamma \vdash M : \alpha \rightarrow \beta$, so $M \in T(\Gamma; \alpha \rightarrow \beta)$. □

**Definition 8.4.** The set $SN(\Gamma; \alpha)$ is defined as follows.

$$SN(\Gamma; \alpha) = \{ M \in SN | M \in T(\Gamma; \alpha) \}$$

The following result is trivial.

**Theorem 8.5.** Let $SN(\Gamma; \alpha) \neq \emptyset$. Then $SN(\Gamma; \alpha \rightarrow \beta) \supset SN(\Gamma; \alpha) \rightarrow SN(\Gamma; \beta)$.

**Proof.** Let $M \in SN(\Gamma; \alpha) \rightarrow SN(\Gamma; \beta)$. Since $SN(\Gamma, \alpha) \neq \emptyset$, there is some $N \in SN(\Gamma, \alpha)$. Hence $AIN \in SN(I, \beta)$.

This yields $f ^{-1} I : \alpha \rightarrow f ^{1} I$, so $AI \in SN(\Gamma; \alpha \rightarrow \beta)$.

$\square$

The converse of this theorem is not so easy and we need the following lemma to prove it.

**Lemma 8.6.** Let $N \in SN(\Gamma; \alpha _{1}) \rightarrow \ldots \rightarrow SN(\Gamma; \alpha _{n})$ with $\alpha _{n}$ a base type. Let $P \in SN(\Gamma; \beta)$. We have

1. $M \in SN$, because $MN \in SN(\Gamma; \beta)$.
2. $\Gamma \vdash M : \alpha \rightarrow \beta$, by the previous lemma.

We make use of the Substitution Lemma for $\lambda$ à la Curry. It states that $\Gamma, x : \beta \vdash M : \alpha$ and $\Gamma \vdash N : \beta$ implies $\Gamma \vdash M[x := N] : \alpha$.

The proof proceeds by induction on the derivation of $P \in SN$.

1. Suppose $P = yP_{1} \ldots P_{k}$ with $P_{1}, \ldots , P_{k} \in SN$. By induction hypothesis, we have $P_{i}[x := N] \in SN$ for $i = 1, \ldots , k$. We write $P_{i}^{*}$ for $P_{i}[x := N]$ for $i = 1, \ldots , k$.

If $y \neq x$, then $P[x := N] \in SN$ follows from the fact that $P_{i}^{*} \in SN$ for $i = 1, \ldots , k$. Using the Substitution Lemma we obtain $P[x := N] \in SN(\Gamma; \beta)$.

If $y = x$, then we have to prove that $NP_{1}^{*} \ldots P_{k}^{*} \in SN(\Gamma; \beta)$. We have $k \leq n$ and by the induction hypothesis and the Substitution Lemma $P_{i}^{*} \in SN(\Gamma; \alpha _{i})$ for $i = 1, \ldots , k$.

Further, $N \in SN(\Gamma; \alpha _{1}) \rightarrow \ldots \rightarrow SN(\Gamma; \alpha _{n}) \subset SN(\Gamma; \alpha _{1}) \rightarrow \ldots \rightarrow SN(\Gamma; \alpha _{k}) \rightarrow SN(\Gamma; \beta)$, by Theorem 8.5. Hence we have $P[x := N] = NP_{1}^{*} \ldots P_{k}^{*} \in SN(\Gamma; \beta)$.

2. Suppose $P = \lambda y.P_{0}$ with $P_{0} \in SN$. By induction hypothesis, we have $P_{0}[x := N] \in SN$. Together with the Substitution Lemma this yields $P[x := N] = (\lambda z.P_{0})[x := N] \in SN(\Gamma; \beta)$.

3. Suppose $P = (\lambda y.P_{0})P_{1}P_{2} \ldots P_{k}$ with $P_{0}[y := P_{1}]P_{2} \ldots P_{k} \in SN$ and $P_{1} \in SN$. By induction hypothesis we have $(P_{0}[y := P_{1}]P_{2} \ldots P_{k})[x := N] \in SN$ and $P_{1}[x := N] \in SN$. Using the Substitution Lemma this yields $P[x := N] = ((\lambda y.P_{0})P_{1}P_{2} \ldots P_{k})[x := N] \in SN(\Gamma; \beta)$.

$\square$

Now we can prove the following theorem.
Theorem 8.7. \( SN(\Gamma; \alpha \rightarrow \beta) \subseteq SN(\Gamma; \alpha) \rightarrow SN(\Gamma; \beta). \)

Proof. Let \( M \in SN(\Gamma; \alpha \rightarrow \beta). \) We prove that for all \( N \in SN(\Gamma; \alpha), \) we have \( MN \in SN(\Gamma; \beta). \) Let thereto \( N \in SN(\Gamma; \alpha). \) Note that it is clear that \( \Gamma \vdash MN : \beta. \) It remains to prove that \( MN \in SN. \) This is proven by induction on \( \alpha \) and for each \( \alpha \) by induction on the derivation of \( M \in SN. \)

\( \alpha \) is a base type. The proof of this part proceeds by induction on the derivation of \( M \in SN. \)

1. Suppose \( M = xM_1 \ldots M_k \) with \( M_1, \ldots, M_k \in SN. \) We have \( N \in SN \) because \( N \in SN(\Gamma; \alpha). \) This yields \( MN = xM_1 \ldots M_k N \in SN. \)

2. Suppose \( M = \lambda x.P \) with \( P \in SN. \) Note that \( \Gamma, x : \alpha \vdash P : \beta, \) so actually \( P \in SN(\Gamma, x : \alpha; \beta). \) For proving \( (\lambda x.P)N \in SN, \) we need to prove \( P[x := N] \in SN. \) This follows from an application of Lemma 8.6.

3. Suppose \( M = (\lambda x.M_0)M_1M_2 \ldots M_k \) with \( M_0[x := M_1]M_2 \ldots M_k \in SN \) and \( M_1 \in SN. \) By induction hypothesis of the induction on the derivation of \( M \in SN, \) we have \( M_0[x := M_1]M_2 \ldots M_k N \in SN. \) Moreover \( M_1 \in SN. \) This yields \( (\lambda x.M_0)M_1M_2 \ldots M_k N \in SN. \)

\( \alpha \) is a composed type. The proof of this part proceeds as well by induction on the derivation of \( M \in SN. \)

1. Suppose \( M = xM_1 \ldots M_k \) with \( M_1, \ldots, M_k \in SN. \) Since \( N \in SN, \) we have \( MN \in SN. \)

2. Suppose \( M = \lambda x.P \) with \( P \in SN. \) For proving \( (\lambda x.P)N \in SN, \) we need to prove that \( P[x := N] \in SN. \) We have \( \alpha = \alpha_1 \rightarrow \ldots \rightarrow \alpha_n \) with \( \alpha_n \) a base type. By the induction hypothesis of the induction on \( \alpha, \) we have \( N \in SN(\Gamma; \alpha_1) \rightarrow \ldots \rightarrow SN(\Gamma; \alpha_n). \) Lemma 8.6 yields that \( P[x := N] \in SN. \)

3. Suppose \( M = (\lambda x.M_0)M_1M_2 \ldots M_k \) with \( M_0[x := M_1]M_2 \ldots M_k \in SN \) and \( M_1 \in SN. \) By induction hypothesis of the induction on the derivation of \( M \in SN, \) we have \( M_0[x := M_1]M_2 \ldots M_k N \in SN. \) Moreover \( M_1 \in SN. \) This yields \( MN \in SN. \)

\( \square \)

Corollary 8.8. \( SN(\Gamma; \alpha \rightarrow \beta) = SN(\Gamma; \alpha) \rightarrow SN(\Gamma; \beta). \)

Theorem 8.9. If \( \Gamma \vdash M : \alpha \) then \( M \in SN(\Gamma; \alpha). \)

Proof. The proof proceeds by induction on the derivation of \( \Gamma \vdash M : \alpha. \)

1. Suppose \( \Gamma, x : \alpha \vdash x : \alpha. \) A tautology.

2. Suppose \( M = \lambda x.P : \beta \rightarrow \gamma \) so the last step in the derivation of \( \Gamma \vdash M : \alpha \) is an application of the abstraction clause. By induction hypothesis, we have \( P \in SN(\Gamma, x : \beta; \gamma). \) This yields \( \lambda x.P \in SN(\Gamma; \alpha \rightarrow \beta). \)
3. Suppose $M = PQ$ so the last step in the derivation of $\Gamma \vdash M : \alpha$ follows from the application clause. Suppose $P : \beta \rightarrow \gamma$ and $Q : \beta$. By induction hypothesis, $P \in SN(\Gamma; \beta \rightarrow \gamma)$ and $Q \in SN(\Gamma; \beta)$. By the previous theorem we have $SN(\Gamma; \beta \rightarrow \gamma) = SN(\Gamma; \beta) \rightarrow SN(\Gamma; \gamma)$. Therefore $PQ \in SN(\Gamma; \gamma)$.

\[\Box\]

9. \textit{\lam-calculus with Intersection Types}

In this section we compare our characterisation of strongly normalising terms with another characterisation using intersection types. We prove that all terms in our set $SN$ are typable with intersection types, and vice versa, that all terms that are intersection typable are in our set. First we shortly recall the definition of \lam-calculus with intersection types. We consider the system without the type $\Omega$ and without the relation $\leq$ on types. Types are built from type variables and from two binary constructors $\rightarrow$ and $\land$.

The type inference system is given by the following rules:

\begin{itemize}
  \item \textbf{Start} \quad $\Gamma \vdash x : \alpha$ if $x : \alpha \in \Gamma$
  \item $\rightarrow$-Introduction \quad $\Gamma, x : \alpha \vdash M : \beta$ \quad $\frac{}{\Gamma \vdash \lambda x. M : \alpha \rightarrow \beta}$
  \item $\rightarrow$-Elimination \quad $\Gamma \vdash M : \alpha \rightarrow \beta$ \quad $\Gamma \vdash N : \alpha$ \quad $\frac{}{\Gamma \vdash MN : \beta}$
  \item $\land$-Introduction \quad $\Gamma \vdash M : \alpha$ \quad $\Gamma \vdash M : \beta$ \quad $\frac{}{\Gamma \vdash M : \alpha \land \beta}$
  \item $\land$-Elimination \quad $\Gamma \vdash M : \alpha \land \beta$ \quad $\frac{}{\Gamma \vdash M : \alpha}$ \quad $\Gamma \vdash M : \beta$
\end{itemize}

For the proof of the first result of this section we use the following notation:

\begin{itemize}
  \item $\{x : \alpha\} \land \Gamma = \Gamma_1, x : \alpha \land \beta, \Gamma_2$ if $\Gamma = \Gamma_1, x : \beta, \Gamma_2$
  \item $\{x : \alpha\} \land \Gamma = x : \alpha, \Gamma$ otherwise
\end{itemize}

Then $\Gamma \land \Gamma'$ is defined by induction on $\Gamma$.

We will make use of the following proposition that is proved in Chapter IV of [Kri93].

\textbf{Proposition 9.1.} Suppose $\Gamma \vdash M[x := N] : \alpha$ and $\Gamma \vdash N : \beta$. Suppose $x$ doesn't occur in $\Gamma$. Then $\Gamma, x : \beta \vdash M : \alpha$.

\textbf{Theorem 9.2.} If $M \in SN$ then there exist a sequent $\Gamma$ and a type $\alpha$ such that $\Gamma \vdash M : \alpha$.

\textbf{Proof.} The proof proceeds by induction on the derivation of $M \in SN$.

- Let $M = xP_1 \ldots P_n$ with $P_i \in SN$ for $i = 1, \ldots, n$. By induction hypothesis there exist for $i = 1, \ldots, n$ a sequent $\Gamma_i$ and a type $\alpha_i$ such that $\Gamma_i \vdash P_i : \alpha_i$. Define $\Gamma := (x : \alpha_1 \rightarrow \ldots \rightarrow \alpha_n \rightarrow \beta) \land \Gamma_1 \land \ldots \land \Gamma_n$. Then $\Gamma \vdash M : \beta$. 

\[\Box\]
Let $M = \lambda x. P$ with $P \in SN$. By induction hypothesis there exist a sequent $\Gamma$ and a type $\beta$ with $\Gamma \vdash P : \beta$. Then $\Gamma' \vdash \lambda x. P : \alpha \to \beta$, with $\Gamma'$ obtained from $\Gamma$ by removing a possible type declaration for $x$.

Let $M = (\lambda x. P) Q P_1 \ldots P_n$ with $P[x := Q] P_1 \ldots P_n \in SN$ and $Q \in SN$. By induction hypothesis, there exist a sequent $\Gamma_1$ and a type $\alpha$ such that $\Gamma_1 \vdash P_0[x := P_1] P_2 \ldots P_n : \alpha$ and there exist a sequent $\Gamma_2$ and a type $\beta$ such that $\Gamma_2 \vdash P_1 : \beta$. Let $\Gamma := \Gamma_1 \wedge \Gamma_2$. Then we have $\Gamma \vdash P_0[x := P_1] P_2 \ldots P_n : \alpha$ and $\Gamma \vdash P_1 : \beta$. Moreover, $\Gamma \vdash P_0[x := P_1] : \gamma$ for the appropriate type $\gamma$. There are two possibilities: $x \in \text{FV}(P_0)$ or $x \notin \text{FV}(P_0)$. In both cases we have $\Gamma, x : \beta \vdash P_0 : \gamma$; in the first case by the previous proposition and in the second case immediately. So $\Gamma \vdash \lambda x. P_0 : \beta \to \gamma$ and hence $\Gamma \vdash (\lambda x. P_0) P_1 : \gamma$. We can conclude $\Gamma \vdash M : \alpha$.

For proving the converse statement we make use of the characterisation of strongly normalising terms. Like in the previous section, we consider the intersection of the set of terms that are typable with a certain type in a certain context and the set $SN$.

**Definition 9.3.** $T(\Gamma; \alpha) = \{ M \in \Lambda \mid \Gamma \vdash M : \alpha \}$.

**Proposition 9.4.**

1. Let $T(\Gamma; \alpha) \neq \emptyset$. Then $T(\Gamma; \alpha \to \beta) \subseteq T(\Gamma; \alpha) \to T(\Gamma; \beta)$.

2. $T(\Gamma; \alpha \wedge \beta) = T(\Gamma; \alpha) \cap T(\Gamma; \beta)$.

**Proof.**

1. Let $M \in T(\Gamma; \alpha \to \beta)$ and $N \in T(\Gamma; \alpha)$. It is clear that $\Gamma \vdash MN : \beta$.

2. Trivial.

The statement $T(\Gamma; \alpha \to \beta) \supset T(\Gamma; \alpha) \to T(\Gamma; \beta)$ is not true. However, in the system with the subtype relation denoted by $\leq$, we can prove the following weaker result provided $T(\Gamma, \alpha) \neq \emptyset$:

$$T(\Gamma; \alpha \to \beta') \supset T(\Gamma; \alpha) \to T(\Gamma; \beta)$$

where $\beta'$ is some type with $\beta \leq \beta'$. A counterexample to the stronger statement is as follows. Let $\Gamma = x : (\alpha \to \alpha) \land ((\alpha \to \alpha) \to (\alpha \to \alpha))$. Then it is easy to see that $x \in T(\Gamma; \alpha \land (\alpha \to \alpha) \to (\alpha \land (\alpha \to \alpha))).$ However, we don't have $x \in T(\Gamma; (\alpha \land (\alpha \to \alpha) \to (\alpha \land (\alpha \to \alpha))).$. Note that for the $\eta$-expansion $\lambda x.xz$ of $x$, with $z : \alpha \land (\alpha \to \alpha)$ it does hold that $\lambda z.xz \in T(\Gamma; (\alpha \land (\alpha \to \alpha) \to (\alpha \land (\alpha \to \alpha))))$.

Exactly like in the previous section, we define the set $SN(\Gamma; \alpha)$.

**Definition 9.5.** $SN(\Gamma; \alpha) = \{ M \in SN \mid M \in T(\Gamma; \alpha) \}$.

Our goal is now to prove the following:
9. \( \lambda \)-calculus with Intersection Types

1. \( SN(\Gamma, \alpha \land \beta) = SN(\Gamma, \alpha) \cap SN(\Gamma, \beta) \)

2. \( SN(\Gamma, \alpha \rightarrow \beta) \subset SN(\Gamma, \alpha) \rightarrow SN(\Gamma, \beta) \)

Then it follows by a straightforward induction that \( \Gamma \vdash M : \alpha \) implies that \( M \in SN(\Gamma, \alpha) \) and hence \( M \) is strongly normalising. The first point is trivial.

**Theorem 9.6.** \( SN(\Gamma, \alpha \land \beta) = SN(\Gamma, \alpha) \cap SN(\Gamma, \beta) \).

The second point requires more care. We make use of a lemma which is a restricted form of the one used in the case of simply typed \( \lambda \)-calculus. Further, we need a Generation Lemma, which describes the types of the components of an application. For making precise what we mean by 'restricted' we need the following definition.

**Definition 9.7.** The order \( ord(\alpha) \) of a type \( \alpha \) is defined inductively as follows.

1. \( ord(\alpha) = 0 \) if \( \alpha \) is a type variable,
2. \( ord(\beta \rightarrow \gamma) = ord(\beta) + ord(\gamma) + 1 \),
3. \( ord(\beta \land \gamma) = \max\{ord(\beta), ord(\gamma)\} \).

**Lemma 9.8.** Let \( \alpha \) be a type with \( ord(\alpha) = 0 \). Let \( P \in SN(\Gamma, x : \alpha; \beta) \). Let \( N \in SN(\Gamma; \alpha) \). Then \( P[x := N] \in SN(\Gamma; \beta) \).

**Proof.** The proof proceeds by induction on the derivation of \( P \in SN \). All cases are trivial, because if \( P = yP_1 \ldots P_n \) with \( n > 0 \), it cannot be the case that \( x = y \).

**Lemma 9.9.** Let \( \Gamma \vdash MN : \beta \). Then there exist \( \alpha_1, \ldots, \alpha_n \) and \( \beta_1, \ldots, \beta_n \) such that

\[
\begin{align*}
\Gamma & \vdash M : (\alpha_1 \rightarrow \beta_1) \land \ldots \land (\alpha_n \rightarrow \beta_n) \\
\Gamma & \vdash N : \alpha_1 \land \ldots \land \alpha_n \\
\beta & = \beta_1 \land \ldots \land \beta_n
\end{align*}
\]

**Proof.** If there is a derivation of \( \Gamma \vdash MN : \beta \), then there is a subderivation of at least length one where each conclusion is of the form \( \Gamma' \vdash MN : \beta' \). The proof proceeds by induction on the length of this subderivation.

1. Suppose the subderivation is of length one. Then \( \Gamma \vdash MN : \beta \) is due to the \( \rightarrow \) Elimination rule. Then indeed \( \Gamma \vdash M : \alpha \rightarrow \beta \) and \( \Gamma \vdash N : \alpha \) for some type \( \alpha \).

2. If the subderivation is of length greater than one, then \( \Gamma \vdash MN : \beta \) can be due to either the \( \land \) Introduction rule or the \( \land \) Elimination rule.

In the first case, we have \( \beta = \beta' \land \beta'' \) and

\[
\frac{\Gamma \vdash MN : \beta', \Gamma \vdash MN : \beta''}{\Gamma \vdash MN : \beta' \land \beta''}
\]
is the last step in the subderivation we consider. It is easily seen that the statement follows from the induction hypothesis.

In the second case, we have

\[
\Gamma \vdash MN : \beta \land \beta'
\]

as the last step in the subderivation we consider. Again, a simple application of the induction hypothesis yields the desired result.

**Lemma 9.10.** Let \( \Gamma \vdash MN_1 \ldots N_m : \beta \). For all \( i \) such that \( 1 \leq i \leq m \), there are \( \alpha_i^1 \ldots \alpha_i^{n_i} \) and for all \( i \) such that \( 0 \leq i \leq m \), there are \( \gamma_i^1 \ldots \gamma_i^{m_i} \) such that for all \( i \) with \( 1 \leq i \leq m \), we have

\[
\begin{align*}
\Gamma &\vdash MN_1 \ldots N_{i-1} : \gamma_{i-1} \\
\Gamma &\vdash N_i : \alpha_i^1 \land \ldots \land \alpha_i^{n_i} \\
\gamma_{i-1} &\equiv (\alpha_i^1 \to \gamma_i^1) \land \ldots \land (\alpha_i^{n_i} \to \gamma_i^{n_i}) \\
\gamma_i &\equiv \gamma_i^1 \land \ldots \land \gamma_i^{n_i}
\end{align*}
\]

Moreover, we have

\[
\Gamma \vdash MN_1 \ldots N_m : \gamma_1^{n_1} \land \ldots \land \gamma_m^{n_m}
\]

**Proof.** For \( i = n \) the statement follows from the previous lemma. Suppose the statement holds for \( i = p \), then it holds for \( i = p - 1 \) by the previous lemma (here \( p \geq 1 \)). □

**Remark 9.11.** In the situation of Lemma 9.10, for \( i \) such that \( 1 \leq i \leq m \), we have that \( \text{ord}(\gamma_{i-1}) > \text{ord}(\gamma_i) \).

**Lemma 9.12.** If \( \Gamma \vdash x : \beta \) and \( x : \alpha \) is in \( \Gamma \), then \( \text{ord}(\beta) \leq \text{ord}(\alpha) \).

**Proof.** The lemma is proved by an easy induction on the derivation of \( \Gamma \vdash x : \beta \). □

Now we prove the following crucial result.

**Theorem 9.13.** Let \( SN(\Gamma; \alpha) \neq \emptyset \). Then \( SN(\Gamma; \alpha \to \beta) \subseteq SN(\Gamma; \alpha) \to SN(\Gamma; \beta) \).

**Proof.** Let \( M \in SN(\Gamma; \alpha \to \beta) \). Let \( N \in SN(\Gamma; \alpha) \). We prove \( MN \in SN(\Gamma; \beta) \). Note that clearly \( MN \in T(\Gamma; \beta) \) so it remains to show that \( MN \in SN \). The proof proceeds by induction on \( \text{ord}(\alpha) \).

\( \text{ord}(\alpha) = 0 \) The proof of the base step is by induction on the derivation of \( M \in SN \).

1. Let \( M = xM_1 \ldots M_n \) with \( M_1, \ldots, M_n \in SN \) Then \( MN \in SN \) by definition of \( SN \).

2. Let \( M = \lambda x.P \) with \( P \in SN \). We need to prove \( P[x := N] \). This is the case by Lemma 9.8.

3. Let \( M = (\lambda x.M_0)M_1M_2 \ldots M_n \) with \( M_0[x := M_1]M_2 \ldots M_n \in SN \) and \( M_1 \in SN \). It follows by induction hypothesis of the induction on the derivation of \( M \in SN \) that \( MN \in SN \).
ord(\alpha) > 0. The proof of the induction step is also given by induction on the derivation of \( M \in SN \).

1. Let \( M = xM_1 \ldots M_n \) with \( M_1, \ldots, M_n \in SN \) Then \( MN \in SN \) by definition of \( SN \).

2. Let \( M = \lambda x. P \) with \( P \in SN \). This is the most complicated case. It is proved by induction on the derivation of \( M \in SN \). The problematic case in this induction is if \( M = yP_1 \ldots P_m \) with \( m > 0 \) and \( x = y \). We only consider the problematic case since the other cases follow easily by induction. Let \( P_i^* \) denote \( P_i[x := N] \).

We have \( \Gamma \vdash NP_1^* \ldots P_m^* : \beta \), and we are in the situation of Lemma 9.10. That means, for \( i \) with \( 1 \leq i \leq m \) there are \( \alpha_{i1}, \ldots, \alpha_{im_i} \) and for \( i \) with \( 0 \leq i \leq m \) there are \( \gamma_{i1}, \ldots, \gamma_{im_i} \) such that for \( i \) with \( 1 \leq i \leq m \) the following holds:

\[
\begin{align*}
\Gamma &\vdash NP_1 \ldots P_{i-1}^* : \gamma_{i-1} \\
\Gamma &\vdash P_i^* : \alpha_{i1}^i \land \ldots \land \alpha_{im_i}^i \\
\gamma_{i-1} &= (\alpha_{i1}^i \rightarrow \gamma_{i1}^i) \land \ldots \land (\alpha_{im_i}^i \rightarrow \gamma_{im_i}^i) \\
\gamma_i &= \gamma_{i1}^i \land \ldots \land \gamma_{im_i}^i
\end{align*}
\]

with, moreover, \( \Gamma \vdash NP_1 \ldots P_m^* : \gamma_m^i \land \ldots \land \gamma_m^m \).

By Lemma 9.12 we have \( \text{ord}(\alpha) \geq \text{ord}(\gamma_0) \). Furthermore, we have that \( \text{ord}(\gamma_{i-1}) > \text{ord}(\gamma_i) \) for \( i \) with \( 1 \leq i \leq m \).

This yields that for all \( i \) with \( 1 \leq i \leq m \) and for all \( j \) with \( 1 \leq j \leq p_i \) we have that \( \text{ord}(\alpha) > \text{ord}(\gamma_j) \). By induction hypothesis we have that \( SN(\Gamma, \alpha_{j1}^i \rightarrow \gamma_{j1}^i) \subseteq SN(\Gamma, \alpha_{j1}^i) \rightarrow SN(\Gamma, \gamma_{j1}^i) \). Hence we have that \( NP_1 \ldots P_m^* \in SN(\Gamma, \alpha_{j1}^i) \rightarrow SN(\Gamma, \gamma_{j1}^i) \).

Hence we have \( P[x := N] \in SN \) and therefore \( MN \in SN \).

3. Let \( M = (\lambda x. M_0)M_1M_2 \ldots M_n \) with \( M_0[x := M_1]M_2 \ldots M_n \in SN \) and \( M_1 \in SN \). It follows by induction hypothesis of the induction on the derivation of \( M \in SN \) that \( MN \in SN \).

\[\square\]

The second main result of this section is the following theorem.

**Theorem 9.14.** If \( \Gamma \vdash M : \alpha \), then \( M \in SN(\Gamma, \alpha) \).

**Proof.** The proof proceeds by induction on the derivation of \( \Gamma \vdash M : \alpha \).

1. Suppose \( M = x \) and the derivation of \( \Gamma \vdash M : \alpha \) consists just of the start rule. Then the statement trivially holds.

2. Suppose we have \( \Gamma \vdash \lambda x. P : \alpha \rightarrow \beta \) as a consequence of the \( \rightarrow \)-Introduction rule with hypothesis \( \Gamma, x : \alpha \vdash P : \beta \). By induction hypothesis, we have \( P \in SN(\Gamma, x : \alpha; \beta) \). It follows that \( \lambda x. P \in SN(\Gamma, \alpha \rightarrow \beta) \).

3. Suppose we have \( \Gamma \vdash PQ : \beta \) as a consequence of the \( \rightarrow \)-Elimination rule with hypotheses \( \Gamma \vdash P : \alpha \rightarrow \beta \) and \( \Gamma \vdash Q : \alpha \). By induction hypothesis, we have \( P \in SN(\Gamma; \alpha \rightarrow \beta) \) and \( Q \in SN(\Gamma; \alpha) \). Using the previous theorem we obtain \( PQ \in SN(\Gamma; \beta) \).
10. Related Work and Conclusions

4. Suppose we have \( \Gamma \vdash P : \alpha \land \beta \) as a consequence of the \( \land \)-Introduction rule with hypotheses \( \Gamma \vdash P : \alpha \) and \( \Gamma \vdash P : \beta \). By induction hypothesis, we have \( P \in SN(\Gamma; \alpha) \) and \( P \in SN(\Gamma; \beta) \). Using Theorem 9.6 we obtain \( P \in SN(\Gamma; \alpha) \cap SN(\Gamma; \beta) = SN(\Gamma; \alpha \land \beta) \).

5. Suppose we have \( \Gamma \vdash P : \alpha \) as a consequence of the \( \land \)-Elimination rule with hypothesis \( \Gamma \vdash P : \alpha \land \beta \). By induction hypothesis, we have \( P \in SN(\Gamma; \alpha) \) and \( P \in SN(\Gamma; \beta) \). By Theorem 9.6, we have \( P \in SN(\Gamma; \alpha) \cap SN(\Gamma; \beta) \). So \( P \in SN(\Gamma; \alpha) \).

10. RELATED WORK AND CONCLUSIONS

In this section we discuss the relation between our proofs and other proofs of the same results.

The set \( SN \). Ralph Loader defines the set \( SN \) in a note distributed on the types mailinglist [Loa95], where he announces a proof of strong normalisation of system \( F \). The definitions must have been given more or less simultaneously.

Developments. The result that all \( \beta \)-developments are finite is a classical result in \( \lambda \)-calculus and various proofs already exist. There is for instance a proof that can be found in [Bar84], that makes use of a decreasing labelling. This proof is not related to ours. There is a short and elegant proof by de Vrijer [dV85], in which an exact bound for the length of a development is computed. For proving that the bound is an exact bound, he makes in fact use of the strategy \( F_\infty \). Some small observations concerning developments coincide with some small observations we make use of.

Two perpetual strategies. The original proofs of the facts that \( F_{bh} \) and \( F_\infty \) are perpetual proceed by a case analysis. In both cases it is proved that \( F(M) \) admits an infinite rewrite sequence if \( M \) does so. In our proof the equivalent statement \( F(M) \in SN \Rightarrow M \in SN \) is shown. Proving \( F(M) \in SN \Rightarrow M \in SN \) and using the definition of \( SN \) make our proofs more perspicuous.

A maximal strategy. The fact that \( F_\infty \) is a maximal strategy has been proved by Regnier [Reg94] using a relation that permits to permute redexes. In fact, in the paper Regnier shows that some operational criteria do not permit to distinguish between terms that are equivalent up to some permutation of redexes. Much more in the spirit of the present work is a paper by Sorensen ([Sor94]), who gives a proof that is very similar to ours. His work was developed independently and simultaneously. A difference is that in the present paper the number of steps of an \( F_\infty \) rewrite sequence is computed explicitly.

Normalisation of simply typed \( \lambda \)-calculus. Many proofs of strong normalisation of simply typed \( \lambda \)-calculus exist. Our proof seems to be mostly related to the proof by Tait and Girard using saturated sets. There are however some important differences.

First, in the proof by Tait and Girard, a type \( \alpha \) is interpreted as a set of \( \lambda \)-terms denoted by \( [\alpha] \). Then, the interpretation of a type \( \alpha \rightarrow \beta \) is defined to be \( [\alpha] \rightarrow [\beta] \). So \( [\alpha \rightarrow \beta] = [\alpha] \rightarrow [\beta] \) by definition, whereas in our proof we need to prove \( T(\Gamma; \alpha \rightarrow \beta) = T(\Gamma; \alpha) \rightarrow T(\Gamma; \beta) \).

On the other hand, in the proof by Tait and Girard it needs to be proved that \( [\alpha] \rightarrow [\beta] \) is a subset of the set of strongly normalising terms. In fact, it is proved that the interpretation of a type is a saturated set. A saturated set is a subset \( X \) of the set of strongly normalising \( \lambda \)-terms that satisfies
1. if $x$ is a variable and $M_1, \ldots, M_n$ are strongly normalising terms then $\pi M_1 \ldots M_n \in X$,
2. if $M[x := N]P_1 \ldots P_n \in X$ and $N$ is strongly normalising then $(\lambda x.M)NP_1 \ldots P_n \in X$.

In our proof, the set $SN(\Gamma; \alpha \rightarrow \beta)$ is a subset of the set of strongly normalising terms by definition. But the equality $SN(\Gamma; \alpha \rightarrow \beta) = SN(\Gamma; \alpha) \rightarrow SN(\Gamma; \beta)$ needs to be proved.

\textbf{\lambda-calculus with Intersection Types.} Krivine [Kri93] proved that the set of strongly normalising terms coincides with the set of terms that are typable in $\lambda\land$. For proving that a term that is typable in $\lambda\land$ is strongly normalising he makes use of saturated sets. We again make use of sets $SN(\Gamma, \alpha)$ that contain all terms that are strongly normalising and that are typable in $\Gamma$ with type $\alpha$.

\textbf{Conclusions.} We have presented a characterisation of the set of strongly normalising terms that is intuitive and elegant. Using this set we have given simple proofs of properties concerning normalisation in $\lambda$-calculus.

\textbf{Acknowledgements.} We gratefully acknowledge remarks by Morten Heine Sorensen. We thank Herman Geuvers, Jan Willem Klop, Vincent van Oostrom and the participants of the TeReSe seminar at the Vrije Universiteit Amsterdam for valuable discussions.

\textbf{References}


REFERENCES


Computing Science Reports

In this series appeared:

<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/01</td>
<td>R. van Geldrop</td>
</tr>
<tr>
<td>93/02</td>
<td>T. Verhoef</td>
</tr>
<tr>
<td>93/03</td>
<td>T. Verhoef</td>
</tr>
<tr>
<td>93/04</td>
<td>E.H.L. Aarts, J.H.M. Kort, P.I. Zwietering</td>
</tr>
<tr>
<td>93/05</td>
<td>J.C.M. Baeten, C. Verhoef</td>
</tr>
<tr>
<td>93/06</td>
<td>J.P. Velikamp</td>
</tr>
<tr>
<td>93/07</td>
<td>P.D. Moerland</td>
</tr>
<tr>
<td>93/08</td>
<td>J. Verhoosel</td>
</tr>
<tr>
<td>93/09</td>
<td>K.M. van Hee</td>
</tr>
<tr>
<td>93/10</td>
<td>K.M. van Hee</td>
</tr>
<tr>
<td>93/11</td>
<td>K.M. van Hee</td>
</tr>
<tr>
<td>93/12</td>
<td>K.M. van Hee</td>
</tr>
<tr>
<td>93/13</td>
<td>K.M. van Hee</td>
</tr>
<tr>
<td>93/14</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
</tr>
<tr>
<td>93/15</td>
<td>J.C.M. Baeten, J.A. Bergstra, R.N. Bol</td>
</tr>
<tr>
<td>93/16</td>
<td>H. Schepers, J. Hooman</td>
</tr>
<tr>
<td>93/17</td>
<td>D. Alstein, P. van der Sok</td>
</tr>
<tr>
<td>93/18</td>
<td>C. Verhoef</td>
</tr>
<tr>
<td>93/19</td>
<td>G-J. Houben</td>
</tr>
<tr>
<td>93/20</td>
<td>F.S. de Boer</td>
</tr>
<tr>
<td>93/21</td>
<td>M. Codish, D. Dams, G. Fijl, M. Bruynooghe</td>
</tr>
<tr>
<td>93/22</td>
<td>E. Poll</td>
</tr>
<tr>
<td>93/23</td>
<td>E. de Kogel</td>
</tr>
<tr>
<td>93/24</td>
<td>E. Poll, Paula Severi</td>
</tr>
<tr>
<td>93/25</td>
<td>H. Schepers, R. Gerth</td>
</tr>
<tr>
<td>93/26</td>
<td>W.M.P. van der Aalst</td>
</tr>
<tr>
<td>93/27</td>
<td>T. Klos and D. Kratsch</td>
</tr>
<tr>
<td>93/28</td>
<td>F. Kamareddine and R. Nederpelt</td>
</tr>
<tr>
<td>93/29</td>
<td>R. Post and P. De Bra</td>
</tr>
<tr>
<td>93/30</td>
<td>J. Deogun, T. Klos, D. Kratsch, H. Müller</td>
</tr>
</tbody>
</table>

Department of Mathematics and Computing Science
Eindhoven University of Technology

- Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
- A continuous version of the Prisoner's Dilemma, p. 17
- Quicksort for linked lists, p. 8.
- Deterministic and randomized local search, p. 78.
- A congruence theorem for structured operational semantics with predicates, p. 18.
- On the unavoidability of metastable behaviour, p. 29
- Exercises in Multiprogramming, p. 97
- A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
- Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
- A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
- Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
- A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
- The Design of an Online Help Facility for ExSpect, p.21.
- Freeness Analysis for Logic Programs - And Correctness, p. 24
- A Typechecker for Bijective Pure Type Systems, p. 28.
- Relational Algebra and Equational Proofs, p. 23.
- Pure Type Systems with Definitions, p. 38.
- Multi-dimensional Petri nets, p. 25.
- Finding all minimal separators of a graph, p. 11.
- A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
- GOLD, a Graph Oriented Language for Databases, p. 42.
- On Vertex Ranking for Permutation and Other Graphs, p. 11.
<table>
<thead>
<tr>
<th>Page No.</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/31</td>
<td>W. Körver</td>
<td>Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.</td>
</tr>
<tr>
<td>93/33</td>
<td>L. Loyens and J. Moonen</td>
<td>ILIAS, a sequential language for parallel matrix computations, p. 20.</td>
</tr>
<tr>
<td>93/34</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Real Time Process Algebra with Infinitesimals, p.39.</td>
</tr>
<tr>
<td>93/36</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Non Interleaving Process Algebra, p. 17.</td>
</tr>
<tr>
<td>93/38</td>
<td>C. Verhoef</td>
<td>A general conservative extension theorem in process algebra, p. 17.</td>
</tr>
<tr>
<td>93/41</td>
<td>A. Bijlana</td>
<td>Temporal operators viewed as predicate transformers, p. 11.</td>
</tr>
<tr>
<td>93/42</td>
<td>P.M.P. Rambags</td>
<td>Automatic Verification of Regular Protocols in P/T Nets, p. 23.</td>
</tr>
<tr>
<td>93/43</td>
<td>B.W. Watson</td>
<td>A taxonomy of finite automata construction algorithms, p. 87.</td>
</tr>
<tr>
<td>93/44</td>
<td>B.W. Watson</td>
<td>A taxonomy of finite automata minimization algorithms, p. 23.</td>
</tr>
<tr>
<td>93/45</td>
<td>E.J. Luit J.M.M. Martin</td>
<td>A precise clock synchronization protocol.</td>
</tr>
<tr>
<td>93/48</td>
<td>R. Genth</td>
<td>Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.</td>
</tr>
<tr>
<td>94/01</td>
<td>P. America M. van der Kammnen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart</td>
<td>The object-oriented paradigm, p. 28.</td>
</tr>
<tr>
<td>94/02</td>
<td>F. Kameredine R.P. Nederpelt</td>
<td>Canonical typing and II-conversion, p. 51.</td>
</tr>
<tr>
<td>94/04</td>
<td>J.C.M. Baeten J.A. Bergstra</td>
<td>Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.</td>
</tr>
<tr>
<td>94/06</td>
<td>T. Basten T. Kunze J. Black M. Coffin D. Taylor</td>
<td>Time and the Order of Abstract Events in Distributed Computations, p. 29.</td>
</tr>
<tr>
<td>94/08</td>
<td>O.S. van Roosmalen</td>
<td>A Hierarchical Diagrammatic Representation of Class Structure, p. 22.</td>
</tr>
<tr>
<td>94/09</td>
<td>J.C.M. Baeten J.A. Bergstra</td>
<td>Process Algebra with Partial Choice, p. 16.</td>
</tr>
</tbody>
</table>

94/11 J. Polakka
C. Huizing
C. Petersohn
   A Comparison of Ward & Mellor's Transformation

94/12 T. Klok
D. Kratke
H. Müller

94/13 R. Seljke
   A New Method for Integrity Constraint checking in Deductive Databases, p. 34.

94/14 W. Persmans
   Ups and Downs of Type Theory. p. 9.

94/15 R.J.M. Vaessens
E.H.L. Aarts
I.K. Lenstra
   Job Shop Scheduling by Local Search, p. 21.

94/16 R.C. Backhouse
H. Doombos
   Mathematical Induction Made Calculational, p. 36.

94/17 S. Mauw
M.A. Reniers
   An Algebraic Semantics of Basic Message
   Sequence Charts, p. 9.

94/18 F. Kamareddine
R. Nederpelt
   Refining Reduction in the Lambda Calculus. p. 15.

94/19 B.W. Watson
   The performance of single-keyword and multiple-keyword pattern matching
   algorithms. p. 46.

94/20 R. Bloo
F. Kamareddine
R. Nederpelt
   Beyond μ-Reduction in Church's λ→, p. 22.

94/21 B.W. Watson
   An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular
   Expressions.

94/22 B.W. Watson
   The design and implementation of the FIRE engine:
   A C++ toolkit for Finite automata and regular Expressions.

94/23 S. Mauw and M.A. Reniers
   An algebraic semantics of Message Sequence Charts, p. 43.

94/24 D. Dans
O. Grumberg
R. Gehr
   Abstract Interpretation of Reactive Systems:
   Abstractions Preserving VCTL*, SCTL* and CTL*, p. 28.

94/25 T. Klok
   K₁₅-free and W₄-free graphs, p. 10.

94/26 R.R. Hoogerwoord
   On the foundations of functional programming: a programmer's point of view. p. 54.

94/27 S. Mauw and H. Mulder

94/28 C.W.A.M. van Overveld
M. Verhoeven
   Stars or Stripes: a comparative study of finite and
   transfinite techniques for surface modelling. p. 20.

94/29 J. Hooman
   Correctness of Real Time Systems by Construction, p. 22.

94/30 J.C.M. Baeten
J.A. Bergstra
Gh. Stefanescu
   Process Algebra with Feedback, p. 22.

94/31 B.W. Watson
R.E. Watson
   A Boyer-Moore type algorithm for regular expression
   pattern matching. p. 22.

94/32 J.J. Vereijken

94/33 T. Laan
   A formalization of the Ramified Type Theory, p.40.

94/34 R. Bloo
F. Kamareddine
R. Nederpelt
   The Barendregt Cube with Definitions and Generalised
   Reduction, p. 37.

94/35 J.C.M. Baeten
S. Mauw
   Delayed choice: an operator for joining Message
   Sequence Charts, p. 15.

94/36 F. Kamareddine
R. Nederpelt
   Canonical typing and Π-conversion in the Barendregt
   Cube, p. 19.

94/37 T. Basten
R. Bol
M. Voorhoeve
   Simulating and Analyzing Railway Interlockings in
   ExSpect, p. 30.

94/38 A. Bijlsma
C.S. Scholten
   Point-free substitution, p. 10.
<table>
<thead>
<tr>
<th>Page</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>94/39</td>
<td>A. Blokhuis, T. Klosk</td>
<td>On the equivalence covering number of splitgraphs, p. 4.</td>
</tr>
<tr>
<td>94/40</td>
<td>D. Alstein</td>
<td>Distributed Consensus and Hard Real-Time Systems, p. 34.</td>
</tr>
<tr>
<td>94/41</td>
<td>T. Klosk, D. Kratsch</td>
<td>Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.</td>
</tr>
<tr>
<td>94/43</td>
<td>R.C. Backhouse, M. Bijsterveld</td>
<td>Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.</td>
</tr>
<tr>
<td>94/45</td>
<td>G.J. Houben</td>
<td>Tutorial voor de ExSpect-bibliotheek voor &quot;Administratieve Logistiek&quot;, p. 43.</td>
</tr>
<tr>
<td>94/46</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>The A-cube with classes of terms modulo conversion, p. 16.</td>
</tr>
<tr>
<td>94/47</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>On (\pi)-conversion in Type Theory, p. 12.</td>
</tr>
<tr>
<td>94/48</td>
<td>Mathematics of Program Construction Group</td>
<td>Fixed-Point Calculus, p. 11.</td>
</tr>
<tr>
<td>94/50</td>
<td>H. Geuvers</td>
<td>A short and flexible proof of Strong Normalization for the Calculus of Constructions, p. 27.</td>
</tr>
<tr>
<td>94/52</td>
<td>W. Penczek, R. Kuiper</td>
<td>Traces and Logic, p. 81</td>
</tr>
<tr>
<td>95/01</td>
<td>J.J. Lukkien</td>
<td>The Construction of a small Communication Library, p. 16.</td>
</tr>
<tr>
<td>95/02</td>
<td>M. Bezem, R. Bol, J.F. Groote</td>
<td>Formalizing Process Algebraic Verifications in the Calculus of Constructions, p. 49.</td>
</tr>
<tr>
<td>95/03</td>
<td>J.C.M. Baeten, C. Verhoef</td>
<td>Concrete process algebra, p. 134.</td>
</tr>
<tr>
<td>95/04</td>
<td>J. Hidders</td>
<td>An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.</td>
</tr>
<tr>
<td>95/05</td>
<td>P. Severi</td>
<td>A Type Inference Algorithm for Pure Type Systems, p. 20.</td>
</tr>
<tr>
<td>95/06</td>
<td>T.W.M. Vossen, M.G.A. Verhoeven, E.H.L. Aarts</td>
<td>A Quantitative Analysis of Iterated Local Search, p. 23.</td>
</tr>
<tr>
<td>95/07</td>
<td>G.A.M. de Bruijn, O.S. van Roosmalen</td>
<td>Drawing Execution Graphs by Parsing, p. 10.</td>
</tr>
<tr>
<td>95/08</td>
<td>R. Bloo</td>
<td>Preservation of Strong Normalisation for Explicit Substitution, p. 12.</td>
</tr>
<tr>
<td>95/09</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Discrete Time Process Algebra, p. 20</td>
</tr>
<tr>
<td>95/10</td>
<td>R.C. Backhouse, R. Verhoeven, O. Weber</td>
<td>Mathpad: A System for On-Line Preparation of Mathematical Documents, p. 15</td>
</tr>
</tbody>
</table>
95/11  R. Seljse  Deductive Database Systems and integrity constraint checking, p. 36.
95/12  S. Mauw and M. Reniers  Empty Interworkings and Refinement Semantics of Interworkings Revised, p. 19.
95/14  A. Ponse, C. Verhoef, S.F.M. Vlijmen (eds.)  De proceedings: ACP'95, p.
95/16  D. Dams, O. Grumberg, R. Gerth  Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.
95/17  S. Mauw and E.A. van der Meulen  Specification of tools for Message Sequence Charts, p. 36.
95/19  J.C.M. Baeten and J.A. Bergstra  Discrete Time Process Algebra with Abstraction, p. 15.