BARENDREGT'S LEMMA

ROEL DE VRIJER

Vrije Universiteit Amsterdam

Abstract. Barendregt's Lemma in its original form is a statement on Combinatory Logic that holds also for the lambda calculus and gives important insight into the syntactic interplay between substitution and reduction. Its origin lies in undefinablity proofs, but there are other applications as well. It is connected to the so-called Square Brackets Lemma, introduced by van Daalen in proofs of strong normalization of typed lambda calculi and of the Hyland–Wadsworth labelled lambda calculus. In the generalization of the latter result to higher-order rewriting systems, finite family developments, van Oostrom introduced the property "Invert", which is also related.

In this short note we state the lemma, try to put it in perspective, and discuss the mentioned connections. We also present a yet unpublished alternative proof of SN of the Hyland–Wadsworth labelled lambda calculus, using a computability argument.

Dedicated to Henk Barendregt, in celebration of his 60th anniversary

Introduction

There is a nice little story to be told about a beautiful lemma that Henk conjectured, that has interesting applications and inspired a line of further discovery, but that never got properly published and is probably known by just a few insiders. Although the lemma appears in some form in Henk's book on the lambda calculus, it is not traced back there to its source.

At the end of [Bar72], a handwritten note of Henk on the undefinability of Church's δ in CL, he scribbled as an afterthought an interesting theorem. The ink shows that it was added at a later moment, after the rest of the paper had been completed. We quote:

> **Theorem 12.** *If $CL \vdash FM \twoheadrightarrow N$, then there are subterm occurrences A_i of N such that $CL \vdash Fx \twoheadrightarrow N'$ where N' is the result of substituting $x\vec{N}_i$ for the subterm occurrence A_i and such that $CL \vdash [x/M]N' \twoheadrightarrow N$.*
>
> *Proof.* Same method as the proof of 9. □

2000 ACM Subject Classification: F.4.1 [Mathematical logic and formal languages]: Lambda calculus and related systems.

Key words and phrases: combinatory logic, strong normalization, computability, surjective pairing, Hyland–Wadsworth, Church's delta.

REFLECTIONS ON TYPE THEORY, λ-CALCULUS, AND THE MIND

Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday

It is a statement that needs careful reading in order to grasp it. Jokingly, and with some exaggeration, we would sometimes refer to it as that lemma that is almost as difficult to formulate as it is to prove. As to the quoted proof, "Same method as the proof of 9" says so much as: by the method used earlier in the paper, which used a variant of underlining.

I am not sure, but I think the note dates from 1972. A few years later, I was working in Eindhoven at that time, I studied the note and I was also looking for a project assigment for a student, S. de Boer. I then asked him to transfer the results of the note from the original setting of CL to the lambda calculus and to complete the proof of the conjectured result at the end of the note. This resulted in the report [dB75].

In the same period Diederik van Daalen, who was also in Eindhoven, working on the theory of the type systems used in the Automath project, found a new and original method to give proofs of strong normalization (SN). A crucial lemma in that method was the so-called Square Brackets Lemma. Diederik already had a proof for that lemma, but it turned out that Barendregt's Lemma, which was the name we had adopted for Henk's conjecture, could also be used to give a quick proof of the SqBL. This inspired Diederik to look into the proof of BL, which was still quite involved and used an ad hoc method of underlining. He found an elegant proof of what is actually a slightly stronger variant of BL, which he called the Reduction-under-Substitution Lemma. We will refer to it as the substitution variant of Barendregt's Lemma, but there would be good reasons to just call it van Daalen's Lemma.

In 1975, at a lambda calculus workshop that Roger Hindley had organized in Swansea and that Diederik, Henk and I attended, J.-J. Lévy told us that he needed SN for the Hyland–Wadsworth labelled lambda calculus (HW), but had trouble finding a proof. Diederik and I immediately tried and both of us came up with a proof, Diederik using his new method and I using a computability argument. Diederik's proof was adopted by J.-J. in his paper [Lév75] and later it also found its way into Henk's book [Bar84]. Henk gives two proofs of the SqBL there, one direct, based on the original proof of Diederik, the other one using Diederik's substitution variant of BL, which ended up as Exercise 15.4.8 in the book. Both J.-J. and Henk also mention my alternative proof of SN for HW, and sometimes I get questions about it, as it was never published, except as a "Stelling" accompanying my PhD-thesis [dV87].

The origin of Barendregt's Lemma lies in undefinability. In accordance, in [Bar74] Exercise 15.4.8 is also employed as one of two methods to obtain the undefinability of Church's δ, the other method using a Böhm-out technique. However, in the book there is no direct reference to BL, neither to the proof method, with underlining, that was used in the unpublished [Bar72]. In my thesis [dV87] I used Barendregt's Lemma for a quick proof of the undefinability of surjective pairing in the lambda calculus, which is one of the early results of Henk in [Bar74].

Van Oostrom [vO97] generalizes SN for HW to higher order rewriting sytems and calls the result Finite Family Developments, stressing the obvious relation with the Finite Developments Theorem. His proof method is a generalization of van Daalen's method and a key lemma, "Invert", is directly related to BL and the SqBL (only SqBL is mentioned).

In sum, Barendregt's Lemma has many traces, but there are not many visible tracks left of BL in the literature: two Ph.D.-theses mention it, [vD80][1] and [dV87] and there is the report [dB75]. Barendregt's Lemma influenced several parts of Henk's book, but no direct mention is made of it. This seems a good opportunity to put Barendregt's Lemma on the record.

[1]Large parts of Daalen's thesis, including the treatment of BL, have been reproduced in [NGV94].

Outline. We start out by presenting BL and connecting it to van Daalen's substitution variant. In subsequent sections we pay attention to the use of BL in proofs of undefinability and to the use in proofs of strong normalization, via the SqBL. We show the undefinability of surjective pairing in the lambda calculus and we briefly present the SN proof for Hyland–Wadsworth labelled lambda calculus. The final section contains the above-mentioned alternative proof of SN for HW, using a computability argument.

Preliminaries. We are concerned with Combinatory Logic and the pure lambda calculus and we assume familiarity with these systems. We adopt the notations and conventions of the standard text [Bar84], which we will sometimes refer to as "Henk's book", or shorter "the book".

1. Barendregt's Lemma

Look once again at Theorem 12 above, the original form of Barendregt's Lemma. We will give a rendering of BL that, in several aspects, is somewhat more explicit. We start with brief comments upon each of these aspects separately.

First, the prefix that remains invariant in passing from N' to N can be specified as a multi-hole context C (with 0 or more holes!) So we have $N \equiv C[A_1, \ldots, A_n]$ and $N' \equiv C[x\vec{N}_1, \ldots, x\vec{N}_n]$, with $n \geq 0$.

Secondly, the notation $x\vec{N}_i$ should be elucidated. If we for the moment define an x-vector as a term of the form $xP_1 \ldots P_k$ $(k \geq 0)$, then what is meant is that each $x\vec{N}_i$ is an x-vector, that is, a term $x\vec{N}_i \equiv xN_{i,1} \ldots N_{i,k_i}$.

Thirdly, we can be more specific about the reduction $N'[x := M] \twoheadrightarrow N$. It takes place below the prefix C, so it can be divided into reductions $(x\vec{N}_i)[x := M] \twoheadrightarrow A_i$. Making this explicit rules out syntactic accidents.

Lastly, as BL has been established for both CL and for the lambda calculus, in our new rendering we leave out the explicit reference to CL.

Lemma 1.1 (Barendregt's Lemma)**.** *Let $FM \twoheadrightarrow N$ and let x be a variable not occurring in F.*

Then there are a term N', an n-hole context C (with $n \geq 0$), x-vectors $B_1, \ldots, B_n$ and terms $A_1, \ldots, A_n$, such that $Fx \twoheadrightarrow N' \equiv C[B_1, \ldots, B_n]$, $N \equiv C[A_1, \ldots, A_n]$ and $B_i[x := M] \twoheadrightarrow A_i$ $(1 \leq i \leq n)$. □

Heuristics. Barendregt's Lemma can be understood as providing an answer to the question:

> If $FM \twoheadrightarrow N$, then what can we say about the contribution of the argument M to the result N?

Answer: N can be decomposed in two parts.

(1) A prefix C of N that is independent of M.
(2) Subterm occurrences A_i, immediately below C, that depend on M in an essential way, namely as reducts of x-vectors in which M has been substituted for x.

The substitution variant. Barendregt's Lemma can be cast in a different way, in terms of substitution instead of function application. This is the form that originates with Diederik van Daalen [vD80] and that found its way into Henk's book, as Exercise 15.4.8. It is slightly stronger than BL and easier to prove.

Lemma 1.2 (van Daalen)**.** *Let $L[x := M] \twoheadrightarrow N$. Then there are a term N', an n-hole context C (with $n \geq 0$), x-vectors $B_1, \ldots, B_n$ and terms $A_1, \ldots, A_n$, such that $L \twoheadrightarrow N' \equiv C[B_1, \ldots, B_n]$, $N \equiv C[A_1, \ldots, A_n]$ and $B_i[x := M] \twoheadrightarrow A_i$ $(1 \leq i \leq n)$.* □

For the proof we refer to Exercise 15.4.8 in [Bar84]. Here we only point out that Lemma 1.1 immediately follows from Lemma 1.2 by taking Fx for L.

2. Undefinability of Church's δ

The manuscript [Bar72] is about the undefinability in CL of Church's discriminator δ, satisfying the conversion equations:

$$\begin{array}{rl} \delta MM = \mathsf{T} & \text{if } M \text{ is a closed normal form} \\ \delta MM' = \mathsf{F} & \text{if } M, M' \text{ are closed normal forms and } M \not\equiv M' \end{array}$$

This result is obtained there as a corollary of the undefinability of an operator F that discriminates $\mathbf{0}$ from the other numerals, with the encoding $\mathbf{0} \equiv \boldsymbol{I}$, $\mathbf{n+1} \equiv \boldsymbol{K}\mathbf{n}$. In [Bar72] the undefinability of F is proved using an underlining technique.

Exercise 15.4.9 in the book uses van Daalen's substitution variant of Barendregt's Lemma (Exercise 15.4.8) for showing that F, called there the *signum* function sg, is not definable in the lambda calculus. In the lambda calculus we prefer the equivalent encoding $\mathbf{0} \equiv \lambda x.x$, $\mathbf{n+1} \equiv \lambda x.\mathbf{n}$, by which the numerals remain distinct normal forms. We prove the undefinability of F directly from BL.

First note that with the given numeral encoding we have

$$(\mathbf{n} + \mathbf{m})P_1 \ldots P_n \twoheadrightarrow \mathbf{m} \qquad (*)$$

Theorem 2.1. *There is no lambda term F such that $F\mathbf{0} = \mathbf{0}$ and $F(\mathbf{n+1}) = \mathbf{1}$ for all $\mathbf{n}$.*

Proof. Suppose such an F exists. Then by CR we have $F\mathbf{0} \twoheadrightarrow \mathbf{0} \equiv \lambda y.y$, since $\lambda y.y$ is a normal form. Apply BL to this reduction. There are three possibilities for N', it is either $\mathbf{0}$ itself, or $\lambda y.B$ or B, with B an x-vector $xP_1 \ldots P_k$.

Case 1. $N' \equiv \mathbf{0}$. Then $Fx \twoheadrightarrow \mathbf{0}$ and by substitutivity of reduction also $F\mathbf{1} \twoheadrightarrow \mathbf{0}$. Since also $F\mathbf{1} \twoheadrightarrow \mathbf{1}$ this contradicts CR.

Case 2. $N' \equiv \lambda y.xP_1 \ldots P_k$. We have $Fx \twoheadrightarrow N'$ and substituting the numeral $\mathbf{k+1}$ for x throughout this reduction we get

$$F(\mathbf{k+1}) \twoheadrightarrow (\lambda y.xP_1 \ldots P_k)[x := \mathbf{k+1}] \equiv \lambda y.(\mathbf{k+1})P'_1 \ldots P'_k$$

By $(*)$ we have $(\mathbf{k+1})P'_1 \ldots P'_k \twoheadrightarrow \mathbf{1}$ and hence $F(\mathbf{k+1}) \twoheadrightarrow \lambda y.\mathbf{1} \equiv \mathbf{2}$. Since we also have $F(\mathbf{k+1}) \twoheadrightarrow \mathbf{1}$, this contradicts CR.

Case 3. $N' \equiv xP_1 \dots P_k$. Now substituting $\mathbf{k}+\mathbf{2}$ for x we get by similar reasoning as in the previous case

$$F(\mathbf{k}+\mathbf{2}) \twoheadrightarrow (\mathbf{k}+\mathbf{2})P_1' \dots P_k' \twoheadrightarrow \mathbf{2}$$

Since we also have $F(\mathbf{k}+\mathbf{2}) \twoheadrightarrow \mathbf{1}$, this again contradicts CR. □

An alternative proof of Theorem 2.1 is given in the book in Section 20.3, where the undefinability of F is proved using the Böhm-out technique.

Recall that in the lambda calculus we have $\mathsf{T} \equiv \mathsf{K} \equiv \lambda xy.x$ and $\mathsf{F} \equiv \lambda xy.y$.

Corollary 2.2. *Church's δ is not definable in the lambda calculus.*

Proof. If we would have such a term δ, then defining $F \equiv \lambda x.\delta x\mathbf{001}$ would yield an F that contradicts Theorem 2.1. □

3. Undefinability of surjective pairing

A *surjective pairing* would consist of a triple of lambda terms D, D_1, D_2, such that for arbitrary M, N we have:

$$\begin{aligned} D_1(DMN) &= M \\ D_2(DMN) &= N \\ D(D_1M)(D_2M) &= M \end{aligned}$$

The undefinability of surjective pairing in the lambda calculus is the central result of [Bar74]. The proof is by underlining. Here we present the short proof from [dV87] using BL.

We use the fact that the term $\Omega \equiv (\lambda x.xx)\lambda x.xx$ has order 0, that is, if $\Omega M_1 \dots M_p \twoheadrightarrow N$, then $N \equiv \Omega M_1' \dots M_p'$ and $M_i \twoheadrightarrow M_i'$. See [Bar84], 17.3.2-3.

Theorem 3.1. *In the lambda calculus a surjective pairing does not exist.*

Proof. Assume there were D, D_1, D_2 satisfying the equations for surjective pairing. Define $F \equiv \lambda x.D(D_1\Omega)(D_2x)$. Then $F\Omega = D(D_1\Omega)(D_2\Omega) = \Omega$ and hence by the Church–Rosser Theorem the terms $F\Omega$ and Ω have a common reduct, which can only be Ω itself. So $F\Omega \twoheadrightarrow \Omega$ and BL can be applied to yield an N' with the ascribed properties (taking $M \equiv N \equiv \Omega$). Because Ω has order 0 one easily verifies that there are but two possibilities for N', namely either $N' \equiv \Omega$ or $N' \equiv x$. We investigate both cases.

Case 1. $N' \equiv \Omega$. Then $Fx \twoheadrightarrow \Omega$ and so $Fx = \Omega$ and by substitutivity of conversion $FM = \Omega$ for an arbitrary term M. So for any M we have $D_2M = D_2(D(D_1\Omega)(D_2M)) = D_2(FM) = D_2\Omega$ and hence for arbitrary N we have $N = D_2(DNN) = D_2\Omega$. It follows that all terms are equal, contradicting consistency of the lambda calculus.

Case 2. $N' \equiv x$. Then $Fx \twoheadrightarrow x$ and so $Fx = x$ and we have $FM = M$ for an arbitrary term M. Hence $D_1M = D_1(FM) = D_1(D(D_1\Omega)(D_2M)) = D_1\Omega$ for any M. From this a contradiction is derived in the same way as in Case 1. □

4. The Square Brackets Lemma

The interest of van Diederik van Daalen in the substitution variant of BL, his Reduction-under-Substitution Lemma, was because of the Square Brackets Lemma, which he used in his new proof of strong normalization.

Lemma 4.1 (Square Brackets Lemma, [vD80])**.** *Let $L[x := M] \twoheadrightarrow \lambda y.P$. Then we have one of the following two cases.*

(1) *$L \twoheadrightarrow \lambda y.P'$ for a P' such that $P'[x := M] \twoheadrightarrow P$*
(2) *$L \twoheadrightarrow x\vec{Q}$ and $(x\vec{Q})[x := M] \twoheadrightarrow \lambda y.P$*

Proof. The prefix C found by Lemma 1.2 can either be of the form $\lambda y.C'$ or it must be the empty context. If $C \equiv \lambda y.C'$ then $N' \equiv \lambda y.P'$ for some P' and we are in Case 1. If $C \equiv [\,]$ then N' is an x-vector and we are in Case 2. □

Why "square brackets"? The lemma analyses the contribution of the substitution to an abstraction term. In the notation of Automath square brackets were used to denote lambda abstraction.

It is noted in [vD80] that the lemma can be generalized to situations where the outer shape of the reduct is not an abstraction. In [vO97] a similar lemma is stated for arbitrary patterns, the generalization is called there "Invert". Contribution to a pattern is formalized in [vO97] via labels in the style of Hyland–Wadsworth, see the next section.

5. SN of Hyland–Wadsworth labelled lambda calculus

We will briefly sketch van Daalen's SN-method in the setting of Hyland–Wadsworth labelled lambda calculus (HW). The method has been applied to this system in [Lév75, Lév78] and a similar proof can be found in Section 14.1 of Henk's book. Although the proof was originally pointed out by van Daalen, there is no written score of him available.[2]

We just give a short introduction to HW. For more technical details we refer to [Bar84], Section 14.1. The terms of HW are lambda terms of which subterms carry natural numbers as labels. Actually, we require any subterm to be labelled.[3] Labels behave as unary function symbols in the sense that reduction is supposed to be compatible with labelling: $M \to N \implies M^l \to N^l$. Multiple labels like in $(M^l)^k$, also written $M^{l,k}$, are allowed, but there is a reduction rule **label** by which they can be contracted.

The only other reduction rule of HW is the β-rule. It is restricted to redexes where the λ-abstraction has a (single) positive label and involves a label decrease. We specify the two reduction rules.

$$\begin{array}{rlcl} \mathbf{label}: & (M^l)^k & \to & M^{min(l,k)} \\ \mathbf{beta}_+: & (\lambda x.M)^{k+1}N & \to & (M[x := N^k])^k \end{array}$$

[2]Diederik communicated the proof for HW to J.-J. Lévy, as acknowledged in [Lév75]. However, although the reference in [Bar84] (and later also in [vO97]) suggests that the proof for HW can be found in [vD80], the proof method is in fact applied there to a class of typed lambda calculi which does not include HW.

[3]This requirement is not made in [Bar84], but it makes life much easier, without affecting the applicability of HW. Moreover, SN can easily be seen to carry over from HW with to HW without the requirement.

The label $k+1$ of the λ-abstraction is called the *degree* of the redex. So the degree of the contracted redex in the following example is 4.

$$((\lambda x.(x^4y^6)^2)^4z^5)^7 \to ((z^{5,3,4}y^6)^{2,3,7} \twoheadrightarrow (z^3y^6)^2$$

We note that in [Lév75, Lév78] a β-reduction step induces a label *in*crease, just the opposite of the β-rule we just defined. SN is then proved for bounded reduction, i.e. reduction with contraction of only redexes with degree smaller than some fixed natural number N. The decreasing variant is the original one of Hyland and Wadsworth [Hyl76], [Wad76], who used it as a syntactic tool to analyze Scott's D_∞-models, and also the one that one finds in [Bar84]. Increasing labels are natural if they are used to trace the creation history of redexes. From the perspective of SN it is more convenient to have decreasing labels.

SN of HW will be proved by induction on the length of labelled terms, which is straightforward if we know that SN is preserved under substitution with SN terms. Note that label reduction (only the rule $\mathbf{beta}_+$) is itself SN and moreover that SN is preserved under adding labels: one easily verifies that if M is SN, then so is M^n (cf. Lemma 6.2(4)).

So proving the following lemma is the hard part. In our presentation we highlight the application of the SqBL, which is only natural anyhow, as it forms the heart of the proof.

Lemma 5.1. *If M, N are both SN, then so is $M[x := N]$*

Proof. The proof is by induction on the triple $\langle l(N), d(M), \|M\|\rangle$, where l takes the label, d the depth of the reduction tree, and $\|\ \|$ the length of a term.

Distinguishing cases according to the form of M, the critical case is when $M \equiv (M_1M_2)^n$ and $M_1[x := N]$ reduces to an abstraction $(\lambda y.P)^{k+1}$. SN has then to be established for $(P[y := Q^k])^k$, given that $M_2[x := N] \twoheadrightarrow Q$. Apply the SqBL to the reduction $M_1[x := N] \twoheadrightarrow (\lambda y.P)^{k+1}$.

Case 1. $M_1 \twoheadrightarrow (\lambda y.P')^{k+1}$ and $P'[x := N] \twoheadrightarrow P$. Then the substitution can be postponed: $M \twoheadrightarrow (\lambda y.P')^{k+1}M_2 \to (P'[y := (M_2)^k])^k \equiv M'$ and $M'[x := N] \twoheadrightarrow (P[y := Q^k])^k$. The induction hypothesis can be used on M' since it has smaller d.

Case 2. $M_1 \twoheadrightarrow x\vec{P}$ and $(x\vec{P})[x := N] \twoheadrightarrow (\lambda y.P)^{k+1}$ We know that Q is SN because it is a reduct of $M_2[x := N]$ and the induction hypothesis can be used on M_2, similarly P is SN. Now an analysis of the labels learns that k must be smaller than $l(N)$, and hence SN of $(P[y := Q^k])^k$ follows by the induction hypothesis due to a decrease in the first component. □

Theorem 5.2. *Every term M is strongly normalizable.*

Proof. Easy induction on $\|M\|$ using Lemma 5.1 for the case $M \equiv (M_1M_2)^n$. □

Van Daalen's proof of SN for HW has been generalized to prove Finite Family Developments in the setting of higher-order pattern rewriting in [vO97].

6. A computability proof of SN for HW

The proof using a computability argument is referred to in [Lév75] and [Bar84] and it is indicated in [dV87] Stelling 1, but details have never been published. Here we will present the proof.

For technical reasons we add one extra rule to HW.

$$\textbf{decrease}: \quad M^p \ \to \ M^q \quad \text{if } q < p$$

Note that conversely we have that if $M \twoheadrightarrow N$, then $l(N) \leq l(M)$. Of course, if we prove SN for HW with the extra rule, this immediately transfers to HW without it.

Definition 6.1. Define by induction on $l(M)$ that M is *computable* if the following two conditions are satisfied.

(1) M is SN.
(2) If $M \twoheadrightarrow (\lambda x.M_0)^{k+1}$ and N^k is computable, then also $(M_0[x := N^k])^k$ is computable.

Lemma 6.2.

(1) *Computability is closed under reduction.*
(2) *If M is not of the form $(\lambda x.M_0)^{k+1}$, and all M' such that $M \to M'$ are computable, then M is computable.*
(3) *Labelled variables x^n are computable.*
(4) *If M is computable, then so is M^n.*
(5) *If M, N are computable, then so is $(MN)^n$.*

Proof. (1) and (2) are immediate by the definition of computability, and then (3) follows by induction on n, using (2).

(4) Strong normalization of M is preserved under adding a label since reduction of the labels at the outside of M using the rules **label** and **decrease** is SN and, moreover, commutes stepwise with other reduction steps. Further observe that if M^n reduces to $(\lambda x.M_0)^{k+1}$, then so does M. Here the rule **decrease** may be needed.
(5) Assume M, N are computable. We prove computability of $(MN)^n$ by induction on $d(M) + d(N) + n$. By (2) it is sufficient to prove computability for an arbitrary P such that $(MN)^n \to P$. Distinguish three cases.

Case 1. The reduction to P is internal, that is, $P \equiv (M'N')^n$ and either $M \to M'$ and $N \equiv N'$, or $M \equiv M'$ and $N \to N'$. Apply the induction hypothesis.

Case 2. The reduction to P is an outer label decrease, that is, $P \equiv (MN)^l$ with $l < n$. Apply the induction hypothesis.

Case 3. $M \equiv (\lambda x.M_0)^{k+1}$ and $P \equiv (M_0[x := N^k])^{k,n}$. By (4) also N^k is computable, so computability of $(M_0[x := N^k])^k$ is assured by the second clause of Definition 6.1. Then computability of P follows by again (4). □

Definition 6.3. M is *computable under substitution* if $M[\vec{x} := \vec{N}]$ is computable for all substitutions with computable terms $N_1, \dots N_n$.

Theorem 6.4. *Every term M is computable under substitution.*

Proof. Induction on $\|M\|$. The only case that is not immediate by the induction hypothesis and the previous lemma is that M is an abstraction term $M \equiv (\lambda x.M')^n$. We check (1) and (2) of Definition 6.1 to prove computability of $M[\vec{x} := \vec{N}]$. (We may assume that $x \notin FV(\vec{N})$.)

(1) $M[\vec{x} := \vec{N}]$ is SN since the induction hypothesis for M' implies that $M'[\vec{x} := \vec{N}]$ is SN.

(2) Suppose $M[\vec{x} := \vec{N}] \twoheadrightarrow (\lambda x.M_0)^{k+1}$ and N^k is computable. We have $M'[\vec{x} := \vec{N}] \twoheadrightarrow M_0$ and hence also $M'[\vec{x}, x := \vec{N}, N^k] \twoheadrightarrow M_0[x := N^k]$. We must check computability of $(M_0[x := N^k])^k$.

As $\vec{N}, N$ are computable by assumption, the induction hypothesis for M' yields that $M'[\vec{x}, x := \vec{N}, N^k]$ is also computable, and hence also $M_0[x := N^k]$, by Lemma 6.2(1). For the last k apply Lemma 6.2(4). □

Corollary 6.5. *Hyland–Wadsworth-labelled lambda calculus is strongly normalizing.*

Proof. Immediate from Theorem 6.4 □

7. Concluding remarks

We mention two more proofs of SN for HW-labelled β-reduction. First there is the proof in the Ph.D.-thesis of J.W. Klop [Klo80]. It uses the method of passing from WN to SN via an interpretation in λI-calculus, where there is no erasure and WN and SN are equivalent. Yet another proof is by J. Terlouw [Ter98].

In the setting of first-order term rewriting systems [Mar92] gives a strong-normalization proof using recursive path orders (RPO). As already mentioned [vO97] has a proof in the quite general setting of higher-order rewriting (PRSs).

On intuitive grounds it seems plausible that there is an "inverse" correspondence of Barendregt's Lemma with the notions of tracing and origin tracking, and especially with the prefix property, see [BKdV00]. This relation was already indicated in [BKdV00] and, with the SqBL in the place of BL, also in [vO97] and [Ter03], Section 8.6. It would be interesting to investigate this correspondence in more detail and to compare the techniques of dynamic labelling used in tracing and origin tracking with the special underlining techniques that were employed in [Bar72] and [Bar74].

Acknowledgements

I would like to thank Henk Barendregt for introducing me to the lambda calculus and Jan Willem Klop, Vincent van Oostrom and Femke van Raamsdonk for stimulating conversations on the subject matter of this paper and helpful comments.

References

[Bar72] H.P. Barendregt. Non-definability of δ. Handwritten manuscript, unpublished, 1972.

[Bar74] H.P. Barendregt. Pairing without conventional restraints. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 20:289–306, 1974.

[Bar84] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 2nd revised edition, 1984.

[BKdV00] I. Bethke, J.W. Klop, and R. de Vrijer. Descendants and origins in term rewriting. *Information and Computation*, 159:59–124, 2000.

[dB75] S. de Boer. De ondefinieerbaarheid van Church's δ-funktie in de λ-calculus en Barendregts lemma. Stageverslag, unpublished, 1975.

[dV87] R.C. de Vrijer. *Surjective Pairing and Strong Normalization: Two Themes in Lambda Calculus*. PhD thesis, Universiteit van Amsterdam, January 1987.

[Hyl76] J.M.E. Hyland. A syntactic characterization of the equality in some models of the λ-calculus. *J. London Math. Soc. (2)*, 12:361–370, 1976.

[Klo80] J.W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts*. Mathematisch Centrum, Amsterdam, 1980.

[Lév75] J.-J. Lévy. An algebraic interpretation of the $\lambda\beta K$-calculus and a labelled λ-calculus. In C. Böhm, editor, *λ-calculus and Computer Science Theory, Proceedings of the Symposium held in Rome*, volume 37 of *Lecture Notes in Computer Science*, pages 147–165. Springer-Verlag, 1975.

[Lév78] J.-J. Lévy. *Réductions correctes et optimales dans le λ-calcul*. Thèse de doctorat d'état, Université Paris VII, 1978.

[Mar92] L. Maranget. *La stratégie paresseuse*. Thèse de doctorat, Université Paris VII, 6 juilliet 1992.

[NGV94] R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1994.

[Ter98] J. Terlouw. A proof of strong normalization for generalized labelled β-reduction by means of the successor relation method. Unpublished, 1998.

[Ter03] Terese. *Term Rewriting Systems*. Cambridge University Press, 2003.

[vD80] D.T. van Daalen. *The Language Theory of Automath*. PhD thesis, Technische Universiteit Eindhoven, 1980.

[vO97] V. van Oostrom. Finite family developments. In H. Comon, editor, *Proceedings of the Eighth International Conference on Rewriting Techniques and Applications (RTA '97)*, volume 1232 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, June 1997.

[Wad76] C.P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the λ-calculus. *SIAM Journal of Computing*, 5:488–521, 1976.