Lambda calculus with postponed substitution

_Citation for published version (APA):_

_Document status and date:_
Published: 01/01/1983

_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: 02. May. 2019
LAMBDA CALCULUS WITH POSTPONED SUBSTITUTION

by

N.G. de Bruijn

May 1983
1. Introduction.

This note extends the notational system of [1], where namefree lambda calculus (with depth references) was enriched by means of reference transformers. Here we go somewhat further: we also admit references to formulas (the depth references refer to variables). The idea behind this is that we want to indicate substitutions without actually carrying them out. At the same time we provide facilities for a stronger form of postponement of substitution: we create "substitutional atoms" which have, at some point of a tree, the purpose to express that references to a certain variable, made at far-out endpoints, have to be replaced by references to a certain formula.

2. The syntax.

We discuss planar trees, which are intended to represent formulas. Every node in a tree has a label that identifies it, and a case identifier that describes its role in the game. The labels will be represented by lower case letters, and will be written to the left of the nodes. We have the following kind of case identifiers:

a) arity 0

```
\[ p \]
```

"pointer"

```
\[ n \]
```

"depth reference"

```
\[ c \]
```

"constant"

where \( p \) is a label, \( n \) an element of \( \mathbb{N} = \{1, 2, \ldots \} \), and \( c \) is an element of a set of "constants".

b) arity 1

```
\[ \lambda \]
```

"lambda"

```
\[ \phi \]
```

"reference transformer"

```
\[ (n, \rho) \]
```

"substitutional atom"
where λ is the symbol λ, φ is a mapping N → N, n ∈ N, and p is a label.

c) further nodes with arity 1, 2, 3, ... ad libitum, but in particular

\[ \bigwedge \text{"application"} \quad \bigwedge \text{"typing"} \quad \bigwedge \text{"parking"}. \]

We shall say that a label p is out of use if the formula nowhere contains \( p \) or \( n, p \).

3. Primary and secondary reductions.

We shall consider "primary reductions" which are directed to the goal of getting rid of the \( \bigwedge, \bigvee, \bigwedge_p \) and \( \phi \). A tree that does not have anything of those can be called "primary normal". We cannot state in general that every tree can be reduced to a primary normal tree by a finite sequence of reductions, but if we start from primary normal trees and apply substitution or beta reduction as described in sections 6 and 7, a sequence of primary reductions will lead back to a primary normal tree.

As a secondary reduction we have the beta reduction which is of a less trivial nature as far as normal forms are concerned.


P1. A parking node may be skipped if there is nothing in the whole formula that refers to it. Skipping the parking node means replacing

\[ \bigwedge \text{by} \]

\( (p, q, r \text{ are labels}; \sigma, \varphi \text{ are out of use}). \)
if \( p \) is the label of \( \varphi \).

This rule holds for every case identifier, including those with arity 0 and 1. For example, if \( p \) is the label of \( \neg \), we have the reduction

P3.

\[
\begin{array}{c}
\text{n} \\
\varphi \\
r
\end{array}
\quad\rightarrow\quad
\begin{array}{c}
\varphi(n) \\
r
\end{array}
\]

if \( q \) is out of use.

P4.

\[
\begin{array}{c}
\text{n} \\
\varphi \\
m,p
\end{array}
\quad\rightarrow\quad
\begin{array}{c}
\text{n} \\
r
\end{array}
\]

if \( n \neq m \),

\[
\begin{array}{c}
p \\
\varphi \\
r
\end{array}
\]

if \( n = m \), with

\[
\forall_k (\varphi(k) = k + m),
\]

and in both cases under the condition that \( q \) is out of use.
P5. \( c \quad \rightarrow \quad c \) if \( q \) is out of use.

P6. \( c \quad \rightarrow \quad c \) if \( q \) is out of use.

P7. With the nodes mentioned under c) we have the trivial shifts, like

without any change in the contents of \( \square \) and \( \triangle \), provided that \( q \) is out of use.
5. Extra reductions.

We might use some extra reductions $Q_1 - Q_5$ that possibly speed up the calculation of the primary normal form. We do not take them as primary reductions, since they are not essential for reaching the normal form; we also note that $Q_5$ would kill the strong normalization. In all cases it is to be required that $q$ is out of use.
Q1: \( \forall_k \varphi(k) = k \)

Q2: \( \forall_k \chi(k) = \varphi(\varphi(k)) \)

Q3: \( n \notin \varphi(\mathbb{N}) \)

Q4: \( \begin{cases} \forall_m (\varphi(m) = n \rightarrow m = n) \\ \forall_k (k \neq n \rightarrow \psi(k) = \varphi(k)) \\ \psi(n) = n \end{cases} \)

Q5
6. Beta reduction. In typed lambda-calculus beta-reduction is

\[ C_{\phi(k)} \]

where \( \phi(k) = k - 1 \) for all \( k > 1 \); \( \phi(1) \) is irrelevant. The mapping is invariant under primary equivalence.

7. Substitution. Substitution of tree \( A \) for the \( k \)-th free variable of \( B \) gives a primary equivalence class of which

is a representative. Here \( q \) is a freshly created label.
8. Remark.

1. The calculus can be extended by adding facilities for substitutional references to "segments" (like "strings" and "telescopers") and more general procedure-like abbreviations. This does not necessarily involve extensions on the "lambda-level": in the first approach we can restrict ourselves to formulas without string references.

2. In order to be able to answer questions like which labels are "out of use" at a certain moment, it may be worthwhile to keep record of the number of references to $p$ or $p.m$ which are made to a certain $p$, and to update such numbers at every reduction step. One might also think of keeping track of the complete set of references to $p$, for instance by means of a linked list. (We might store at every $d$ or $o.m$ the label of the previous and the label of the next reference to $p$.)

3. One might also try to keep track of the references to $\lambda$'s in a way similar to the one described in the previous remark. This might help saving work in carrying out beta reduction. A warning: Referring to the label of the root of an expression starting with $\lambda$ is not the same thing as referring to the $\lambda$ itself. The latter thing means referring to the "variable attached to $\lambda$". Or, stated in other words, referring to a formula $\lambda x F(x)$ is not the same thing as referring to $x$.

Reference.