Lambda calculus with namefree formulas involving symbols that represent reference transforming mappings

P125
AUT55

N. G. DE BRUIJN

Lambda calculus with namefree formulas involving symbols that represent reference transforming mappings

Dedicated to A. Heyting at the occasion of his 80th birthday on May 9, 1978

Communicated at the meeting of March 18, 1978

Department of Mathematics, Eindhoven University of Technology, The Netherlands

1. INTRODUCTION ON NAME-CARRYING LAMBDA CALCULUS

In ordinary lambda calculus we use names both for free and for bound variables. Let us present an example that explains what kind of expressions we are after: apart from names for variables we have names for constants. We may have introduced an expression in two variables x and y, and have abbreviated it to $f(x, y)$ (now f is the "constant" we mentioned). Now $\lambda_x f(x, y)$ is a lambda expression. Its interpretation is: the function that attaches to every x the value $f(x, y)$. The letter y is a free variable and x a bound variable in the expression $\lambda_x f(x, y)$.

We can, of course, also write more complex lambda expressions like

$$(1.1) \qquad g(\lambda_x f(x, y), \lambda_t f(t, \lambda_u f(u, y)), \lambda_w f(w, \lambda_u g(w, s, u))).$$

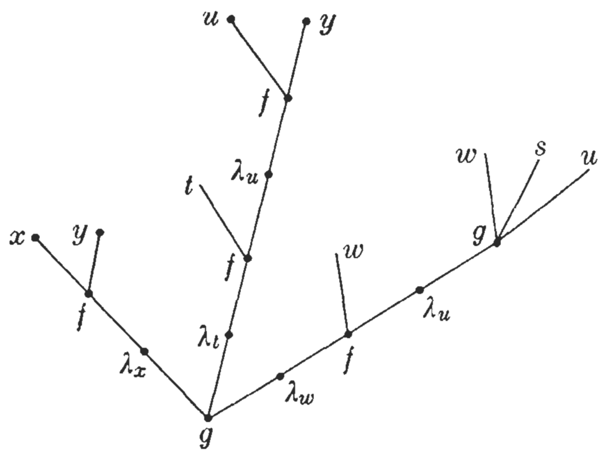
In this example the free variables are y and s.

Usual lambda calculus has a notation (in the form of concatenation) for "application" that intends to express "the value of the function y at the point x". We do not need a special notation for this, because we can devote a special constant A to this purpose, and write that value as $A(y, x)$. Now so-called beta-reduction is a kind of elimination of such an A, like the passage from $A(\lambda_x(f(x, y)), g(t))$ to $f(g(t), y)$. The latter two formulas are *not* considered to be equal (in spite of their common interpretation). On the other hand, the difference between $\lambda_x f(x, y)$ and

Reprinted from Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen, Amsterdam, series A, volume 81 (3), September 22, 1978

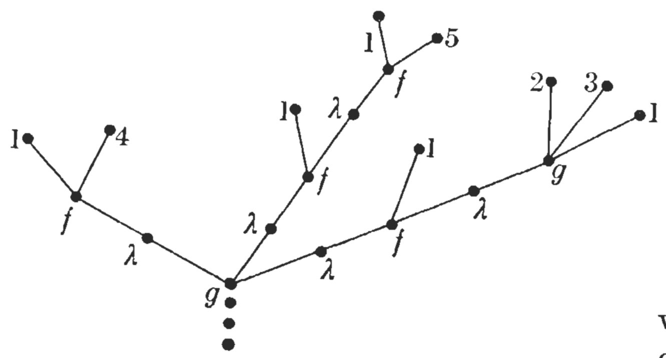
$\lambda_u f(u, y)$ is much less essential. The desire to identify them lies at the root of namefree lambda calculus.

The kind of name-carrying lambda calculus described above is exactly the same as in [1]. We close this section with the tree interpretation of the expression (1.1):



2. INTRODUCTION ON NAMEFREE LAMBDA CALCULUS

In [1] we explained a notation for lambda expressions where all occurrences of free and bound variables are replaced by positive integers that indicate their reference depth. The system is easily demonstrated at the example (1.1) in the tree form



with $s, z, y, \ldots$ as list of free variables.

The dots below the tree are unessential, but suggestive to the term "reference depth", if they are interpreted as being tied to $s, z, y, \ldots$ (the upper one refers to s, the middle one to z, the lower one to y; the fact that z is never referred to in the formula does not bother us).

The idea is that an integer k at an end-point refers to the k-th lambda we meet when travelling from that end-point to the root of the tree; if there are only j lambdas on that path, with $j < k$, then the k at the endpoint refers to the $(k-j)$-th entry of the list of free variables.

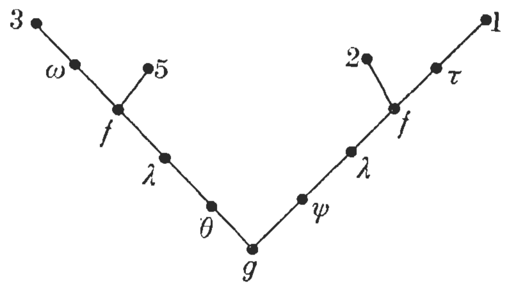
As a preparation to what follows, we express the above correspondence like this. We start at an endpoint and want to know what variable the number refers to. Now we descend the tree, taking the number along, subtracting 1 each time we pass some λ. If this subtraction leads to the value 0, we do not go any further; we have located the right lambda. Of course we act as if the free variables s, z, y, are tied to underground lambdas.

The tree at the beginning of this section did not show any names attached to the lambdas. We can assign arbitrarily names (different from the constants and the free variables) for these bound variables. There is a safe, "conservative", system where it is required that all these names are different. The "liberal" system, on the other hand, only requires that lambdas get different names if they are hierarchically related: if one lambda lies on the path down from another lambda to the root, then the two have to get different names.

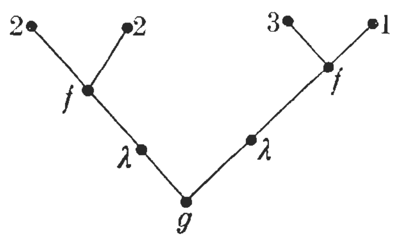
3. TREES WITH SYMBOLS THAT REPRESENT MAPPINGS

We shall now describe a new kind of namefree trees where at some places in the tree we have a symbol denoting some mapping of $\mathfrak{N}$ into $\mathfrak{N}$ ($\mathfrak{N}=\{1, 2, 3, \ldots\}$). We shall use these more complicated trees for the same purpose as the trees in section 1. What matters is, to describe to which lambda a natural number at an end-point refers. What we intend to do will be clear from an example. The letters ω, θ, τ, ψ denote mappings of $\mathfrak{N}$ into $\mathfrak{N}$.

(3.1)



If we want to know what an integer refers to we descend the tree; again we subtract 1 if we pass a λ, but if we pass one of the letters representing a mapping we do something different: we apply that mapping to our number. So the 3 in the upper left corner refers to the left-hand λ if $\omega(3)=1$. If $\omega(3)>1$ it refers to $\theta(\omega(3)-1)$-th free variable. As an excercise the reader may verify that if $\omega(3)=5$, $\theta(4)=1$, $\tau(1)=1$, $\psi(1)=2$, then this tree corresponds to the same references from end-points to lambdas or free variables as the following one:

(3.2) 

We shall say that (3.2) is the reduced form of (1). In the notation of this paper (3.1) is represented by

(3.3) $g(\theta\lambda f(\omega 3, 5), \psi\lambda f(2, \tau 1))$

and (3.2) by

(3.4) $g(\lambda f(2, 2), \lambda f(3, 1))$.

The motivation for studying the tree coding of the type (3.3) is that operations like substitution are easier described in terms of these than in terms of the mapping free codes like (3.4). This may hold both for language theory and for automatic formula manipulation. Getting rid of the mappings can be postponed until we need it; it is relatively easy.

4. METALINGUISTIC NOTATION

In [1] our way to describe linguistic operations was based on the system used in BNF (Backus' normal form). In simple cases this is quite feasable, but in more complex situations it can no longer be maintained. In the present note we prefer the system used in the theory of context-free languages, where linguistic entities like words are treated as mathematical objects, referred to by names or more complicated expressions, and never appear themselves in the language that discusses them.

We have a set A (the elements are interpreted as letters and signs like comma's, parentheses, etc.). $S(A)$ is the set of all finite sequences of elements of A (these sequences are called words). $S_1(A)$ is the subset consisting of all words of length 1.

We use the following notation for the concatenation of words: if p and q denote words then $[\,p \mid q\,]$ denotes the word we get by putting the second word directly after the first one. Similarly for three or more words: $[\,p \mid q \mid r\,]$.

The letters elt directly after an opening bracklet or a vertical bar have the meaning illustrated by the example:

$$[\,\text{elt}\,P \mid q \mid \text{elt}\,P \mid \text{elt}\,R\,] = \{[\,p \mid q \mid s \mid r\,] \mid p \in P, s \in P, r \in R\},$$

where $q \in S(A)$, $P \subset S(A)$, $R \subset S(A)$.

The following "comb" notation was introduced in [2]:

$$\overline{|\,p\,|\,q\,|\,r\,|} \qquad \text{instead of } [\,p \mid q \mid r\,],$$

$$\overline{|\underline{\,P\,}|\,q\,|\underline{\,P\,}|\underline{\,R\,}|} \qquad \text{instead of } [\,\text{elt } P \mid q \mid \text{elt } P \mid \text{elt } R\,].$$

This is quite easy for handwriting and reading, but harder to print.

5. THE SETS Z_0 AND Z

In order to give a preliminary idea we state that Z will consist of all strings of the type (3.3) (with a restriction on the constants) and Z_0 will be the subset consisting of the strings of type (3.4). (The elements of Z_0 were called *NF*-expressions in [1]).

As before, $\mathfrak{N} = \{1, 2, 3, \ldots\}$, and Γ will denote the set of all mappings of $\mathfrak{N}$ into $\mathfrak{N}$. And as before, A is a set, $S(A)$ is the set of words, and $S_1(A)$ the set of one-letter words. Furthermore ξ is some injection of $\mathfrak{N}$ into $S_1(A)$, φ some injection of Γ into $S_1(A)$, and we assume that

$$S_1(A) = C \cup \xi(\mathfrak{N}) \cup \varphi(\Gamma) \cup R$$

where the four parts on the right are disjoint, and R has exactly 4 elements. The elements r_1, r_2, r_3, r_4 of R are one-letter words with the following interpretation: r_1 is the word consisting of a lambda only, r_2 of an opening parenthesis only, r_3 of a comma only, r_4 of a closing parenthesis only. Since we never show the elements of A themselves the usual symbols for lambda, etc. are free for us to use, and as long as they are separated in the [| |] notation, confusion does not arise. We can write e.g.

$$[\,r_1 \mid r_2 \mid r_3 \mid r_4\,] = [\,\lambda \mid (\mid , \mid)\,].$$

If Y is a subset of $S(A)$ then $\sigma(Y)$ denotes the set of all strings of elements of Y separated by comma's:

$$\sigma(Y) = Y \cup [\,\text{elt } Y \mid , \mid \text{elt } Y\,] \cup [\,\text{elt } Y \mid , \mid \text{elt } Y \mid , \mid \text{elt } Y\,] \cup \ldots$$

The set Z is a subset of $S(A)$, defined as the minimal solution of the equation

$$(5.1)\quad \left\{ \begin{array}{l} Z = C \cup [\,\text{elt } C \mid (\mid \text{elt } \sigma(Z) \mid)\,] \cup \xi(\mathfrak{N}) \cup [\,\lambda \mid \text{elt } Z\,] \cup \\ \cup\, [\,\text{elt } \varphi(\Gamma) \mid \text{elt } Z\,]. \end{array} \right.$$

As an example of an element of Z we present, with $g, f \in C$ and $\omega, \theta, \tau, \psi \in \Gamma$,

$$[g|(|\varphi(\theta)|\lambda|f|(|\varphi(\omega)|\xi(3)|\,,\,|\xi(5)|)|\,,\,|\varphi(\psi)|\lambda|f|(|\xi(2)|\,,\,|\varphi(\tau)|\xi(1)|)|)].$$

Translation into (3.3) is just a matter of omitting the ['s, the]'s, the |'s, the φ's and the ξ's. In examples, one would prefer that abbreviated form (3.3), of course.

The subset Z_0 of Z can be defined as the minimal solution of

$$(5.2)\qquad Z_0 = C \cup [\,\text{elt } C \mid (\mid \text{elt } \sigma(Z_0) \mid)\,] \cup \xi(\mathfrak{N}) \cup [\,\lambda \mid \text{elt } Z_0\,].$$

6. SUBSTITUTION

Let Ω be a mapping of $\mathbb{N}$ into Z, and let z be an element of $\sigma(Z)$. We want to define subst (Ω, z). Its interpretation, first for the case that $z \in Z$, is as follows. Attach to z the free variable list $x_1, x_2, \ldots$, and to each one of $\Omega(1), \Omega(2), \ldots$ the variable list $y_1, y_2, \ldots$. Now we substitute into the name-carrying form of z, for each x_i, the name-carrying form of $\Omega(i)$. What we get is an expression with free variables $y_1, y_2, \ldots$, and the namefree form of this will be subst (Ω, z). If z is a string, $z \in \sigma(Z)$ then the substitution is effected in every entry of the string separately.

From now on we concentrate on what happens in $\sigma(Z)$ and Z, and we do not study the interpretations. (They will stay on the back of our mind, of course).

We define subst (Ω, z) for all $z \in \sigma(Z)$ by recursion on (5.1). To that end it suffices to define (note the uniqueness of parsing the elements of $\sigma(Z)$):

(i) if $z = [z_1 \mid , \mid z_2]$ with $z_1 \in \sigma(Z)$, $z_2 \in Z$ then
subst $(\Omega, z) = [\text{subst}\ (\Omega, z_1) \mid , \mid \text{subst}\ (\Omega, z_2)]$,
(ii) if $z \in C$ then subst $(\Omega, z) = z$,
(iii) if $z = [c \mid (\mid z_1 \mid)]$ with $c \in C$, $z_1 \in \sigma(Z)$ then
subst $(\Omega, z) = [c \mid (\mid \text{subst}\ (\Omega, z_1) \mid)]$,
(iv) if for some $n \in \mathbb{N}$ $z = \xi(n)$ then subst $(\Omega, z) = \Omega(n)$,
(v) if $z = [\lambda \mid z_1]$ with $z_1 \in Z$ then subst $(\Omega, z) = [\lambda \mid \text{subst}\ (\Omega^*, z_1)]$,
where Ω^* is the mapping defined by

$$\Omega^*(1) = \xi(1),\ \Omega^*(k) = [\varphi(\gamma) \mid \Omega(k-1)] \quad (k = 2, 3, \ldots)$$

with γ defined by $\gamma(k) = k+1$ $(k = 1, 2, 3, \ldots)$,
(vi) if $z = [\varphi(\theta) \mid z_1]$ with $\theta \in \Gamma$, $z_1 \in Z$ then subst $(\Omega, z) = \text{subst}\ (\Omega\theta, z_1)$
where $\Omega\theta$ is defined by $(\Omega\theta)(k) = \Omega(\theta(k))$ for all $k \in \mathbb{N}$.

7. THE REDUCED FORM

At the end of section 3 it was explained how an element z of Z leads to one of Z_0, called its reduced form. We shall denote it by $rf(z)$, to be formally defined here for all $z \in \sigma(Z)$:

(i) if $z = [z_1 \mid , \mid z_2]$ with $z_1 \in \sigma(Z)$, $z_2 \in Z$ then $rf(z) = [rf(z_1) \mid , \mid rf(z_2)]$,
(ii) if $z \in C$ then $rf(z) = z$,
(iii) if $z = [c \mid (\mid z_1 \mid)]$ with $c \in C$, $z_1 \in \sigma(Z)$ then $rf(z) = [c \mid (\mid rf(z_1) \mid)]$,
(iv) if $z = \xi(n)$ for some $n \in \mathbb{N}$ then $rf(z) = z$,
(v) if $z = [\lambda \mid z_1]$ with $z_1 \in Z$ then $rf(z) = [\lambda \mid rf(z_1)]$,
(vi) if $z = [\varphi(\theta) \mid c]$ with $\theta \in \Gamma$, $c \in C$ then $rf(z) = c$,
(vii) if $z = [\varphi(\theta) \mid c \mid (\mid x \mid)]$ with $\theta \in \Gamma$, $c \in C$, $x \in \sigma(Z)$, then
$rf(z) = [c \mid (\mid rf(p_\theta(x)) \mid)]$,
where $p_\theta(x)$ is defined recursively by

$$p_\theta([x \mid , \mid y]) = [p_\theta(x) \mid , \mid \varphi(\theta) \mid y],\ p_\theta(y) = [\varphi(\theta) \mid y] \quad (x \in \sigma(Z), y \in Z),$$

(viii) if $z=[\varphi(\theta) \mid \xi(n)]$ with $\theta \in \Gamma$, $n \in \mathfrak{N}$ then $rf(z)=\xi(\theta(n))$,

(ix) if $z=[\varphi(\theta) \mid \lambda \mid z_1]$ with $\theta \in \Gamma$, $z_1 \in Z$ then $rf(z)=[\lambda \mid w]$ with $w=rf([\varphi(\theta^*) \mid z_1])$, where θ^* is defined by $\theta^*(1)=1$, $\theta^*(k)=\theta(k-1)+1$ $(k=2, 3, \ldots)$,

(x) if $z=[\varphi(\theta) \mid \varphi(\eta) \mid z_1]$ with $\theta \in \Gamma$, $\eta \in \Gamma$, $z_1 \in Z$ then $rf(z)=rf([\varphi(\theta\eta) \mid z])$ (where, of course, $\theta\eta$ is defined by $(\theta\eta)(k)=\theta(\eta(k))$ for all $k \in \mathfrak{N}$)·

8. THEOREMS ON REDUCED FORMS

THEOREM 8.1. For all $z \in \sigma(Z)$ we have $rf(rf(z))=rf(z)$.

THEOREM 8.2. For all $z \in \sigma(Z_0)$ we have $rf(z)=z$.

THEOREM 8.3. If $\theta \in \Gamma$, $z \in Z$ then $rf([\varphi(\theta) \mid z\ ])=rf([\varphi(\theta) \mid rf(z)])$.

THEOREM 8.4. If θ_0 is the identity ($\theta_0(n)=n$ for all n), and $z \in Z$, then $rf([\varphi(\theta_0) \mid z])=rf(z)$.

These theorems are easily proved by induction with respect to the length of z. At a certain point in the proof of Theorem 8.3 it plays a role that the operation of section 7 (ix) satisfies $(\theta\eta)^*=\theta^*\eta^*$.

9. THEOREMS ON SUBSTITUTION

THEOREM 9.1. If Ω maps $\mathfrak{N}$ into Z, and if $\theta \in \Gamma$, $z \in Z$ then

$$rf(\text{subst}\ (\Omega, rf([\varphi(\theta) \mid z])))=rf(\text{subst}\ (\Omega\theta, rf(z))).$$

THEOREM 9.2. If Ω maps $\mathfrak{N}$ into Z, and if $z \in \sigma(Z)$ then

$$rf(\text{subst}\ (\Omega, z))=rf(\text{subst}\ (\Omega_1, rf(z))),$$

where Ω_1 is defined by $\Omega_1(n)=rf(\Omega(n))$ for all n.

THEOREM 9.3. If φ maps $\mathfrak{N}$ into $\mathfrak{N}$ and if $\Omega(n)=\varphi(\theta(n))$ for all n, then we have for all $z \in Z$

$$rf(\text{subst}\ (\Omega, z))=rf([\varphi(\theta) \mid z]).$$

THEOREM 9.4. If Ω maps $\mathfrak{N}$ into Z, if $z \in Z$, $\theta \in \Gamma$, and if Ω_1 is defined by $\Omega_1(n)=[\varphi(\theta) \mid \Omega(n)]$ $(n=1, 2, \ldots)$ then

$$rf(\text{subst}\ (\Omega_1, z))=rf([\varphi(\theta) \mid \text{subst}\ (\Omega, z)]).$$

THEOREM 9.5. If Ω, Σ, Λ are mappings of $\mathfrak{N}$ into Z, such that

$$\Lambda(n)=\text{subst}\ (\Omega, \Sigma(n))\quad (n=1, 2, \ldots)$$

then we have for all $z \in \sigma(Z)$

$$rf(\text{subst}(\Omega, \text{subst}(\Sigma, z)))=rf(\text{subst}\ (\Lambda, z)).$$

These theorems provide a solid background to the conviction that

$rf(\text{subst}\ (\Omega, z))$ corresponds to what we usually mean by substitution. They are easily proved by recursion on the length of z. We omit the details.

10. SUBSTITUTION IN Z_0

Right now there is not enough experience to compare the value of the present system of substitution to other systems, in particular to the system of [1].

In order to facilitate the comparison, we present the definition of substitution of [1] in our present metalanguage. It operates on Z_0 and $\sigma(Z_0)$. If $z \in \sigma(Z_0)$ and if Ω is a mapping of $\mathfrak{N}$ into Z_0, the result of the substitution will be denoted by $S(\Omega, z)$. The definition is by recursion:

(i) if $z=[z_1 \mid , \mid z_2]$ with $z_1 \in \sigma(Z_0)$ and $z_2 \in Z_0$ then

$$S(\Omega, z)=[S(\Omega, z_1) \mid , \mid S(\Omega, z_2)],$$

(ii) if $z \in C$ then $S(\Omega, z)=z$,

(iii) if $z=[c \mid (\mid z_1 \mid)]$ with $c \in C$, $z_1 \in \sigma(Z_0)$ then

$$S(\Omega, z)=[c \mid (\mid S(\Omega, z_1) \mid)],$$

(iv) if for some $n \in \mathfrak{N}$ $z=\xi(n)$ then $S(\Omega, z)=\Omega(n)$,

(v) if $z=[\lambda \mid z_1]$ with $z_1 \in Z_0$ then

$$S(\Omega, z)=[\lambda \mid S(\Omega_1, z_1)]$$

where Ω_1 is defined by its values $\Omega_1(1)=\xi(1)$ and $\Omega_1(k)=S(\Lambda, \Omega(k-1))$ $(k=2, 3, \ldots)$;
here Λ is the mapping defined by $\Lambda(k)=\xi(k+1)$ for all k.

The fact that under (v) it is required to know the effect of S on expressions that are not subexpressions of z, makes recursion proofs a bit complex.

The following two theorems can be proved straightforwardly by induction with respect to the length of z.

THEOREM 10.1. If $z \in Z_0$, $\theta \in \Gamma$, then

$$rf([\varphi(\theta) \mid z])=S(\Theta, z)$$

where $\Theta(k)=\xi(\theta(k))$ $(k=1, 2, \ldots)$.

THEOREM 10.2. If $z \in \sigma(Z_0)$, $\Omega: \mathfrak{N} \to Z_0$ then

$$S(\Omega, z)=rf(\text{subst}\ (\Omega, z)).$$

11. ALGORITHM FOR CHECKING EQUALITY OF REDUCED FORMS

Let $x, y \in \sigma(Z)$. Quite often it is possible to answer the question whether $rf(x)=rf(y)$ without evaluating $rf(x)$ and $rf(y)$.

For every $z \in \sigma(Z)$ there is a unique integer $k \geqslant 1$ such that z has the form $[z_1 \mid , \mid \ldots \mid , \mid z_k]$ with $z_1 \in Z, \ldots, z_k \in Z$. Let us call k the string length of z. It is clear that z has the same string length as $rf(z)$. So if x and y have different string length then certainly $rf(x) \neq rf(y)$.

Supposing x and y have the same string length k, we check whether $rf(x_1) = rf(y_1), \ldots, rf(x_k) = rf(y_k)$. This means that we yet have to describe how we check $rf(x) = rf(y)$ if both x and y are in Z.

If $x = [\varphi(\theta) \mid x_1]$, $y = [\varphi(\theta) \mid y_1]$ with $\theta \in \Gamma$ we just replace the question by the one whether $rf(x_1) = rf(y_1)$.

If x still has the form $x = [\varphi(\theta) \mid x_1]$ but if y does not have the form $[\varphi(\theta) \mid y_1]$, we apply one of the reduction steps (vi)–(x) of section 7, and if the result is u, we ask whether $rf(u) = rf(y)$. We do a similar thing if this applies with x and y interchanged.

Finally, if neither x nor y have such a form, we say that $rf(x) \neq rf(y)$ unless we are in one of the following four cases:

(i) $x \in C$ and $y = x$,

(ii) $x = [c \mid (\mid x_1 \mid)]$, $y = [c \mid (\mid y_1 \mid)]$ with $x_1, x_2 \in \sigma(Z)$, $c \in C$ and $rf(x_1) = rf(x_2)$.

(iii) $x = y = \xi(n)$ for some $n \in \Omega$,

(iv) $x = [\lambda \mid x_1]$, $y = [\lambda \mid y_1]$ with $x_1 \in Z$, $x_2 \in Z$, $rf(x_1) = rf(x_2)$.

12. REMARK ON STRINGS

Some of the notational effort of the previous sections went into the distinction between Z and $\sigma(Z)$, connected with the fact that we deal with n-ary expressions like $c(u_1, \ldots, u_n)$. One of the disadvantages is, that recursion over the definition of Z is not so straight-forward as it might be. It is, of course, possible to eliminate this unpleasantness, removing all cases with $n > 2$. This can be done by creating a special constant s, and replacing, e.g., $c(u_1, u_2)$ by $c(s(u_1, u_2))$, $c(u_1, u_2, u_3)$ by $c(s(u_1, s(u_2, u_3)))$, etc. The cases c and $c(u_1)$ are unaltered.

13. IMPLEMENTATION

The substitution algorithm (section 6) and the algorithm of section 11 have been implemented by Mr. R. Wieringa with the use of the programming language PASCAL.

REFERENCES

1. Bruijn, N. G. de – Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. Nederl. Akad. Wetensch. Proc. Ser. A **75**, (=Indag. Math. **34**) 381–392 (1972).

2. Bruijn, N. G. de – Notation for concatenation. Technological University Eindhoven, Department of Mathematics. Memorandum 1977–09. August 1977.