A namefree lambda calculus with facilities for internal

definition of expressions and segments.

by

N.G. de Bruijn.

Abstract.

The paper starts with a formal definition of a lambda calculus with abbreviation facilities, including a set of single-step reductions which can be used to effectuate substitution and beta-reduction. A novelty is the abbreviation of segments of formulas. Later reference to such a segment involves duplication of the segment variables. These variables can be local, global, formal or somehow disguised by other segment abbreviations, and therefore it would be hard to describe without namefree references what the system would be.

A large part of the paper consists of informal explanations.

1. Introduction. We deviate from the usual way of presentation, shifting introductory material to the end of the paper (section 6). This makes it clear that such material, although it provides a background, is in no way used in the main sections.

The calculus consists of a syntax (section 3) and a set of primitive reductions. At present there is not much available in the direction of a theory about these reductions. It is quite conceivable that it may turn out to be better to alter syntax and reductions in order to make room for a more satisfactory theory. Therefore this note has to be considered as a first step rather than as something final.

2. Notation. We adopt the metalinguistic notation explained in [6]. (The notation is essentially the same as in [5], except for the fact that in [5] the "combs" have been replaced by something else, just for printability. In [4] the notation is essentially different.) A short explanation follows here.

$\mathcal{A}$ is a finite non-empty set, called the alphabet. $S(\mathcal{A})$ is the set of all finite $\mathcal{A}$-words. In $S(\mathcal{A})$ we have concatenation as basic operation; with this operation $S(\mathcal{A})$ is what is usually called the free monoid over $\mathcal{A}$. For k=0,1,2,... we denote by $S_k(\mathcal{A})$ the subset consisting of all words of k letters. There is a trivial bijection from $\mathcal{A}$ to $S_1(\mathcal{A})$. $S_0(\mathcal{A})$ consists of the empty word only. That empty word is denoted by ϵ.

From now on we put $S(\mathcal{A}) = S$, $S_1(\mathcal{A}) = S_1$, and we no longer mention $\mathcal{A}$ and its elements. The elements of S are to be denoted by mathematical symbols (like the elements of any other set), and a symbol used for an object in S usually does not reveal the elements of $\mathcal{A}$ the object consists of.

If $p \in S$ then the index k with $p \in S_k$ is called the length of p.

Concatenation of p and q (with $p \in S$, $q \in S$) is denoted by $\boxed{p \mid q}$, and similarly we write $\boxed{p \mid q \mid r}$, etc.

We also use notations like this one: if $P \subset S$, $q \in S$, $R \subset S$, $s \in S$, $t \in S$ then

$$\boxed{P \mid q \mid R \mid s \mid t} = \{ \boxed{p \mid q \mid r \mid s \mid t} \mid p \in P,\ r \in R \} \qquad (2.1)$$

3. <u>The set T</u>. We write $\mathbb{N} = \{1,2,3,\ldots\}$, $\mathbb{N}_o = \{0,1,2,\ldots\}$; $\mathbb{N}^{\mathbb{N}}$ is the set of all mappings of $\mathbb{N}$ into $\mathbb{N}$.

M is the set of all pairs (k,θ) where $k \in \mathbb{N}_o$, and where θ is a mapping of $\{1,\ldots,k\}$ into $\mathbb{N}_o$. (If $k=0$ then θ can only be the empty mapping).

We introduce a number of elements and subsets of S_1 and mappings into S_1 which will be fixed from now on.

$X_0, X_1, X_2, \ldots$ are mutually disjoint subsets of S_1.

ξ is an injection of $\mathbb{N}$ into X_o.

ω is an injection of M into X_o.

ϕ is an injection of $\mathbb{N}^{\mathbb{N}}$ into X_1.

η is an injection of $\mathbb{N}_o \times \mathbb{N}$ into X_1.

λ, δ are elements of X_1, X_2, respectively.

The sets

$$\xi(\mathbb{N}), \omega(M), \phi(\mathbb{N}^{\mathbb{N}}), \eta(\mathbb{N}_o \times \mathbb{N}), \{\lambda\}, \{\delta\}$$

are assumed to be pairwise disjoint. The whole situation is pictured in fig. 1.
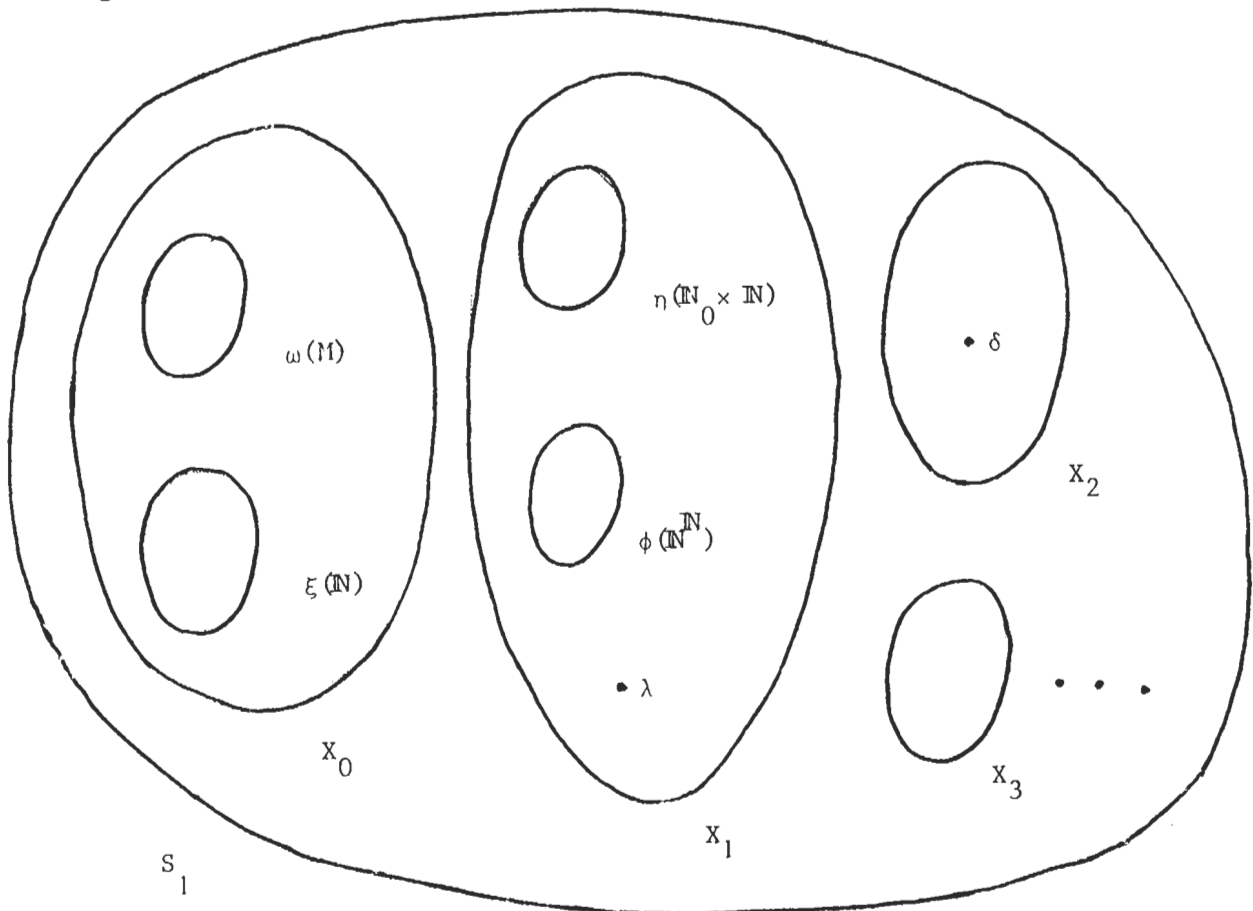
fig.1.

We define the subset T of S as the minimal solution of

$$T = X_0 \cup \boxed{X_1 \mid T} \cup \boxed{X_2 \mid T \mid T} \cup \boxed{X_3 \mid T \mid T \mid T} \cup \ldots \qquad (3.1)$$

By recursion we define mappings $\sigma : T \to \{0,1,2\}$ and $\rho : T \to \mathbb{N}_0$. If $x_0 \in X_0$, $x_1 \in X_1$, $x_2 \in X_2, \ldots$, $t \in T$, $t_1 \in T, \ldots$ we agree that

$$\sigma(x_0) = 1 \text{ if } x_0 \in \omega(M), \text{ and } \sigma(x_0) = 0 \text{ if } x_0 \notin \omega(M),$$

$$\sigma(\boxed{x_1 \mid t}) = \begin{cases} 2 \text{ if } x_1 \in \phi(\mathbb{N}^{\mathbb{N}}) \wedge \sigma(t) = 1 \\ \sigma(t) \text{ otherwise} \end{cases}$$

$$\sigma(\boxed{x_2 \mid t_1 \mid t_2}) = \sigma(t_2),$$

$$\sigma(\boxed{x_3 \mid t_1 \mid t_2 \mid t_3}) = \sigma(t_3),$$

etc.

As to ρ we agree that

$$\rho(x_0) = 0,$$

$$\rho(\boxed{\lambda \mid t}) = \rho(t) + 1,$$

$$\rho(\boxed{\eta(k,m) \mid t}) = \rho(t) + k \text{ if } k \in \mathbb{N}_0, \ m \in \mathbb{N},$$

$$\rho(\boxed{x_1 \mid t}) = \rho(t) \text{ if } x_1 \in X_1 \setminus \{\lambda\} \setminus \eta(\mathbb{N}_0 \times \mathbb{N}),$$

$$\rho(\boxed{x_2 \mid t_1 \mid t_2}) = \rho(t_2),$$

$$\rho(\boxed{x_3 \mid t_1 \mid t_2 \mid t_3}) = \rho(t_3),$$

etc.

For interpretation we refer to section 6.16.

4. Reductions. We define a relation $>_1$ on T. The relation is called "single-step reduction" and will simply be defined by a list of cases.

In this list, w_1 and w_2 stand for arbitrary elements of S, which are such that the thing on the left of $>_1$ is an element of T. The right-hand side will be in T as a consequence.

(A1) $\boxed{w_1\,|\,\phi(\theta)\,|\,x_0\,|\,w_2}\ >_1\ \boxed{w_1\,|\,x_0\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $x_0 \in X_0 \setminus \xi(\mathbb{N}) \setminus \omega(M)$.

(A2) $\boxed{w_1\,|\,\phi(\theta)\,|\,\xi(n)\,|\,w_2}\ >_1\ \boxed{w_1\,|\,\xi(\theta(n))\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $n \in \mathbb{N}$.

(A3) $\boxed{w_1\,|\,\phi(\theta)\,|\,\omega(k,\theta_1)\,|\,w_2}\ >_1\ \boxed{w_1\,|\,\omega(k,\theta\theta_1)\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $(k,\theta_1) \in M$; here $\theta\theta_1$ is the composition of θ and θ_1.

(A4) $\boxed{w_1\,|\,\phi(\theta)\,|\,x_1\,|\,w_2}\ >_1\ \boxed{w_1\,|\,x_1\,|\,\phi(\theta)\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $x_1 \in X_1 \setminus \eta(\mathbb{N}_0 \times \mathbb{N}) \setminus \phi(\mathbb{N}^\mathbb{N}) \setminus \{\lambda\}$.

(A5) $\boxed{w_1\,|\,\phi(\theta)\,|\,\eta(k,m)\,|\,w_2}\ >_1\ \boxed{w_1\,|\,\eta(k,\theta(m))\,|\,\phi(\theta^*)\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $k \in \mathbb{N}_0$, $m \in \mathbb{N}$; here θ^* is defined by $\theta^*(j)=j$ if $1 \le j \le k$, $\theta^*(j) = k + \theta(j-k)$ if $j > k$.

(A6) $\boxed{w_1\,|\,\phi(\theta)\,|\,\phi(\theta_1)\,|\,w_2}\ >_1\ \boxed{w_1\,|\,\phi(\theta\theta_1)\,|\,w_2}$

if $\theta,\theta_1 \in \mathbb{N}^\mathbb{N}$; here $\theta\theta_1$ is the composition of θ and θ_1 .

(A7) $\boxed{w_1\,|\,\phi(\theta)\,|\,\lambda\,|\,w_2}\ >_1\ \boxed{w_1\,|\,\lambda\,|\,\phi(\theta^*)\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$; here θ^* is defined by $\theta^*(1)=1$, $\theta^*(j) = 1 + \theta(j-1)$ if $j > 1$.

(A8) $\boxed{w_1\,|\,\phi(\theta)\,|\,x_2\,|\,t\,|\,w_2}\ >_1\ \boxed{w_1\,|\,x_2\,|\,\phi(\theta)\,|\,t\,|\,\phi(\theta)\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $x_2 \in X_2$, $t \in T$.

(A9) $\boxed{w_1\,|\,\phi(\theta)\,|\,x_3\,|\,t_1\,|\,t_2\,|\,w_2}\ >_1\ \boxed{w_1\,|\,x_3\,|\,\phi(\theta)\,|\,t_1\,|\,\phi(\theta)\,|\,t_2\,|\,\phi(\theta)\,|\,w_2}$

if $\theta \in \mathbb{N}^\mathbb{N}$, $x_3 \in X_3$, $t_1,t_2 \in T$; similarly for $X_4, X_5, \ldots$

(B1) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid x_0 \mid w_2\,} >_1 \boxed{\,w_1 \mid x_0 \mid w_2\,}$

if $x_0 \in X_0 \setminus \xi(\mathbb{N}) \setminus \omega(M)$, $t \in T$.

(B2) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \xi(n) \mid w_2\,} >_1 \boxed{\,w_1 \mid \xi(n-1) \mid w_2\,}$

if $t \in T$, $n \in \mathbb{N}$, $n > 1$.

(B3) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \xi(1) \mid w_2\,} >_1 \boxed{\,w_1 \mid t \mid w_2\,}$

if $t \in T$, $\sigma(t)=0$.

(B4) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \phi(\theta) \mid w_2\,} >_1 \boxed{\,w_1 \mid \phi(\theta) \mid w_2\,}$

if $\theta \in \mathbb{N}^{\mathbb{N}}$ and $1 \notin \theta(\mathbb{N})$; here θ_1 is defined by $\theta_1(j) = \theta(j)-1$ for all $j \in \mathbb{N}$.

(B5) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \omega(k,\theta) \mid w_2\,} >_1 \boxed{\,w_1 \mid \omega(k,\theta_1) \mid w_2\,}$

if $(k,\theta) \in M$ and $1 \notin \theta(\{1,\dots,k\})$; here θ_1 is defined by $\theta_1(j) = \theta(j) - 1$ for $j = 1,\dots,k$.

(B6) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid x_1 \mid w_2\,} >_1 \boxed{\,w_1 \mid x_1 \mid \delta \mid t \mid \lambda \mid w_2\,}$

if $t \in T$, $x_1 \in X_1 \setminus \eta(\mathbb{N}_0 \times \mathbb{N}) \setminus \phi(\mathbb{N}^{\mathbb{N}}) \setminus \{\lambda\}$.

(B7) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \lambda \mid w_2\,} >_1 \boxed{\,w_1 \mid \lambda \mid \delta \mid \phi(\theta_1) \mid t \mid \lambda \mid \phi(\theta_2) \mid w_2\,}$

if $t \in T$, where θ_1 is defined by $\theta_1(j) = j+1$ for all $j \in \mathbb{N}$, and θ_2 by $\theta_2(1) = 2$, $\theta_2(2) = 1$, $\theta_2(j) = j$ if $j > 2$.

(B8) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \eta(k,m) \mid w_2\,} >_1 \boxed{\,w_1 \mid \eta(k,m-1) \mid \delta \mid \phi(\theta_1) \mid t \mid \lambda \mid \phi(\theta_2) \mid w_2\,}$

if $t \in T$, $k \in \mathbb{N}_0$, $m \in \mathbb{N}$, $m > 1$; here θ_1 is defined by $\theta_1(j) = j+k$ for all $j \in \mathbb{N}$, and θ_2 by $\theta_2(j) = j+1$ ($1 \le j \le k$), $\theta_2(k+1) = 1$, $\theta_2(j) = j$ if $j > k + 1$.

(B9) $\boxed{\,w_1 \mid \delta \mid t \mid \lambda \mid \eta(k,1) \mid w_2\,} >_1 \boxed{\,w_1 \mid s \mid \phi(\theta_3) \mid \delta \mid \phi(\theta_1) \mid t \mid \lambda \mid \phi(\theta_2) \mid w_2\,}$

if $t \in T$, $k \in \mathbb{N}_0$, $\sigma(t) = 1$, $t = \boxed{\,s \mid \omega(k,\theta)\,}$; here θ_1 and θ_2 are defined as in (B8), and $\theta_3(j) = \theta(j)$ if $j \le k$, $\theta_3(j) = j + \rho(t) - k$ if $j > k$.

(B10) $\boxed{w_1 \mid \delta \mid t \mid \lambda \mid x_2 \mid t_1 \mid w_2} \; >_1 \; \boxed{w_1 \mid x_2 \mid \delta \mid t \mid \lambda \mid t_1 \mid \delta \mid t \mid \lambda \mid w_2}$

if $t, t_1 \in T$, $x_2 \in X_2$.

(B11) $\boxed{w_1 \mid \delta \mid t \mid \lambda \mid x_3 \mid t_1 \mid t_2 \mid w_2} \; >_1 \; \boxed{w_1 \mid x_3 \mid \delta \mid t \mid \lambda \mid t_1 \mid \delta \mid t \mid \lambda \mid t_2 \mid \delta \mid t \mid \lambda \mid w_2}$

if $t, t_1, t_2 \in T$, $x_3 \in X_3$; similarly for $X_4, X_5, \ldots$

5. The reduced form. We define the set R as a subset of T, consisting of all those words of T which do not contain any $\phi(\theta)$'s. In other words it is the minimal solution of

$$R = X_0 \cup \boxed{X_1^* \mid R} \cup \boxed{X_2 \mid R \mid R} \cup \boxed{X_3 \mid R \mid R \mid R} \cup \ldots$$

where $X_1^* = X \setminus \phi(\mathbb{N}^{\mathbb{N}})$.

We define a mapping $rf : T \to R$. If $t \in T$ then $rf(t)$ is called the reduced form of t. It is defined by recursion:

(i)   If $m \in \mathbb{N}$, $x_m \in X_m$ if $m \neq 1$, $x_n \in X_m^*$ if $m = 1$, and if $t_1 \in T, \ldots, t_n \in T$, then

$$rf(\boxed{x_n \mid t_1 \mid \ldots \mid t_n}) = \boxed{x_n \mid rf(t_1) \mid \ldots \mid rf(t_n)}.$$

(ii) If t has the form of one of the left-hand members of (A1),...,(A9) with $w_1 = \varepsilon$, then $rf(t)$ is defined as the rf of the right-hand member.

Theorem 5.1. If $t \in T$ then $rf(t) \in R$.
            If $t \in R$ then $rf(t) = t$.

Theorem 5.2. If $t \in T$, $\theta \in \mathbb{N}^{\mathbb{N}}$ then

$$rf(\boxed{\phi(\theta) \mid t}) = rf(\boxed{\phi(\theta) \mid rf(t)}).$$

Theorem 5.3. If $t \in T$ then there exists a natural number m such that every sequence

$$t = t_1 \; >_1 \; t_2 \; >_1 \; t_3 \; >_1 \; \ldots \; >_1 \; t_n$$

with reductions taken from (A1),...,(A9) satisfies $n \leq m$, moreover $rf(t_1) = rf(t_2) = \ldots = rf(t_n)$, and either $t_n = rf(t)$ or the reduction sequence can be continued until $rf(t)$.

These theorems are easily proved by induction with respect to the length of t. At a certain point of the proof of theorem 5.2 it plays a rôle that the operations occurring in (A5) and (A7) satisfy $(\theta_1 \theta_2)^* = = \theta_1^* \theta_2^*$.

6. Explanations and comments. In the previous sections we described a system (or, as one might prefer to say, a language) which we shall now denote by $C\lambda\xi\phi\omega\eta$. We shall try to explain the purpose of all this, and to connect it with related systems and their notations. We start with a simple formula language, to be called CV.

6.1 The language CV. The system CV has formulas like this:

$$p(f(g(x,y), h(y,c,g(x,c)))). \qquad (6.1.1)$$

Here p,f,g,h,c are constants, and x,y are variables. There is no difference between constants and variables until we introduce the notion of substitution, where variables are replaced by formulas (all x's by one formula, all y's by a second formula, etc).

In case of long formulas it helps to represent them in tree form; (6.1.1) becomes

(6.1.2)
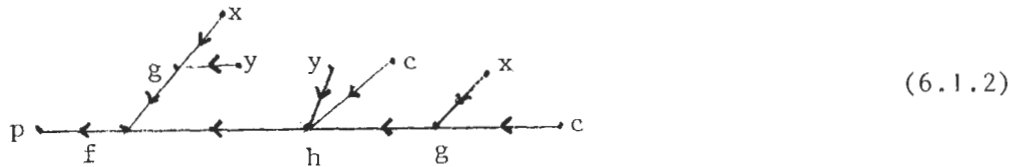
It is what combinatorialists might call a "planted planar tree", oriented towards the "root" p. In further pictures we shall omit the arrows: we agree that the arrows always have to run from right to left.

Every node in such a tree has an "indegree", i.e. the number of arrows entering from the right. Quite often one requires that one and the same constant has the same indegree at all its occurrences, like the g in (6.1.2). (This is not required in AUTOMATH, but that is irrelevant at this moment). The variables x,y,... all have indegree 0.

The correspondence between formulas like (6.1.1) and trees like (6.1.2) is one-to-one. However, if the indegrees are known (either known from some list, or indicated in the formula itself) one can represent (6.1.2) without parentheses and commas:

$$p \quad f \quad g \quad x \quad y \quad h \quad y \quad c \quad g \quad x \quad c \qquad\qquad (6.1.3)$$
$$1 \quad 2 \quad 2 \quad 0 \quad 0 \quad 3 \quad 0 \quad 0 \quad 2 \quad 0 \quad 0$$

Parsing is easy. Read p, write p(...) and try to interpret the remainder of
(6.1.3) as a formula to be placed on the dots. Next read f, write f(...,...),
and try to interpret the remainder of (6.1.3) as two formulas, to be placed
in succession on the dots, etc. For such transformations of various tree
representations we refer to Knuth [8].

It would not be a very serious restriction to require that the indegrees
are at most 2. We can always simulate the general case by means of indegrees
0,1 and 2, e.g. replacing
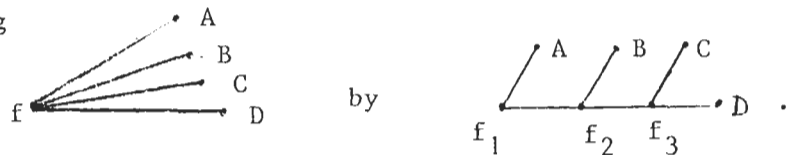
The system CV is a subsystem of $C\lambda\phi\omega\eta$, at least as long as we do not
think of reductions, and provided we represent the formulas of CV in the
form (6.1.3), and as long as we agree that all identifiers of indegree 0 are
taken from a set X_0, indegree 1 from X_1, indegree 2 from X_2, etc.. The sets
$X_0, X_1, \ldots$ are disjoint, and therefore it is no longer necessary to indicate
the indegrees in the formula (like in (6.1.3)). Now (3.1) just defines the
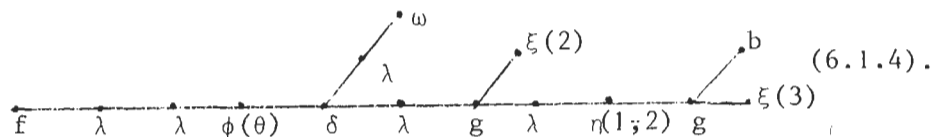set T of all formulas of CV. The $\xi, \omega, \eta, \phi, \lambda, \delta$ of section 3 play no rôle.

The relation between CV-formulas and trees can obviously be extended
to formulas of $C\lambda\xi\phi\omega\eta$. Example:

$$(6.1.4).$$

where $b \in X_0$, $f \in X_1$, $g \in X_2$, $\theta \in \mathbb{N}^{\mathbb{N}}$. It certainly helps to think of the elements
of T (of section 3) as such trees, and in particular to see the reductions of
section 4 as reductions of such trees.

The above embedding of CV into $C\lambda\xi\phi\omega\eta$ is a matter of the syntactic struc-
ture of the formulas only. If we bear in mind what our formulas will be used
for, it is better to distinguish between the variables and the other constants
of indegree 0. Let us order the variables: first x, then y, etc. Instead of x
we now write $\xi(1)$, etc. So (6.1.2) becomes
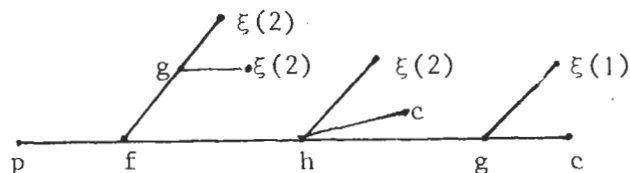
(6.1.5)

The idea that $\xi(k)$ always refers to the k-th variable will be given up later, when intermediate lambdas have their effect on the references.

6.2 <u>Substitution</u>. Let the CV-formula F have variables x,y,z. Let P_1,P_2,P_3 be CV-formulas (they may also contain x,y,z). We now construct a formula G as follows: Mark all x,y,z occurring in F (let us say colour them red), in order to distinguish them from the x,y,z's in P_1,P_2,P_3. Now replace each red x in F by P_1, each red y by P_2, each red z by P_3. In this way F turns into what we call G. This G will be called the result of the substitution $(x,y,z) \rightarrow (P_1,P_2,P_3)$ on F. Example:
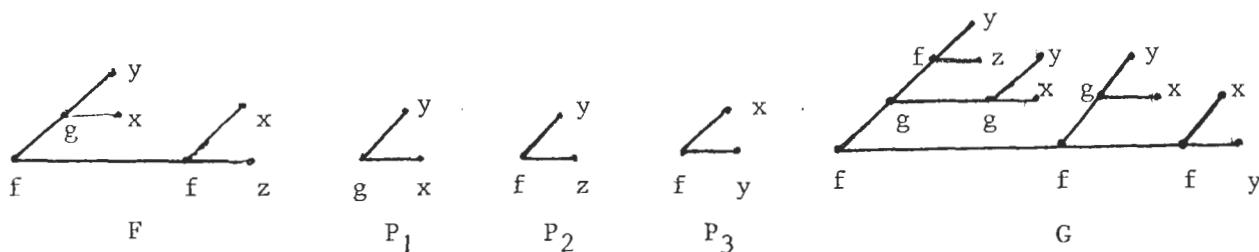
Quite a different thing is what we get from F if we carry out the three substitutions consecutively: first replace x by P_1 (leaving y,z unaltered), then replace y by P_2 (leaving x,z unaltered), then replace z by P_3 (leaving x,y unaltered). We obtain
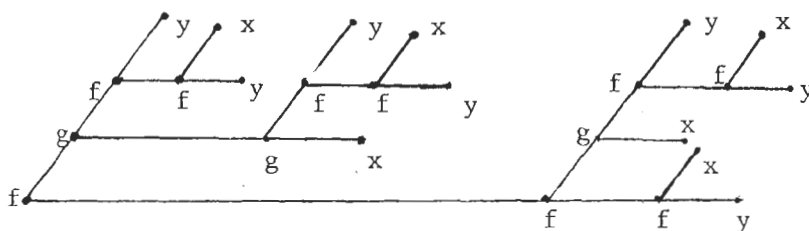
6.3 <u>Abbreviation</u>. Formulas are often presented by means of a phrase like this: F is the formula
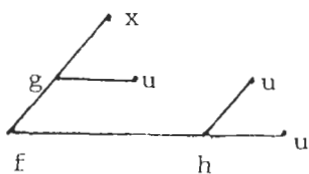
where u is an abbreviation for

What is meant by this, is the formula

(6.3.2)
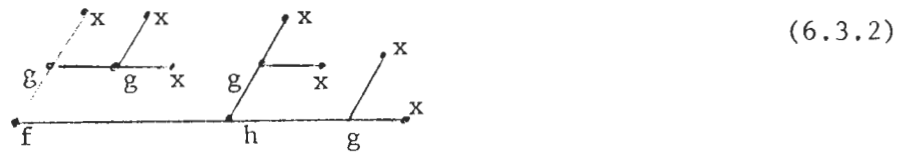
One might also say that it is the effect of substitution: then we say that u is a variable, and u is replaced by $g\overset{x}{\underline{}}x$. A difference with general substitution, however, is that we are not supposed to let u abbreviate a formula containing another u.

Abbreviation goes beyond this, however. The symbol used for abbreviation may be something of non-zero indegree. Let us take the case of indegree 2. We may speak of the formula

(6.3.3)
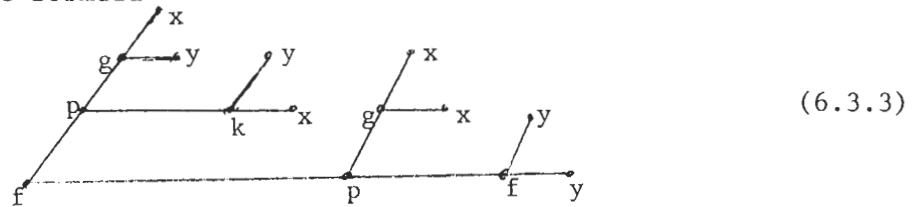
and say that p(u,v) is an abbreviation for

(6.3.4)

The abbreviating symbol p can not be considered as a variable in the sense of the syntax of CV, for it has indegree 2. What we mean by the above phrase ("(6.3.3), where p(u,v) stands for (6.3.4)") is of course the formula

(6.3.5)
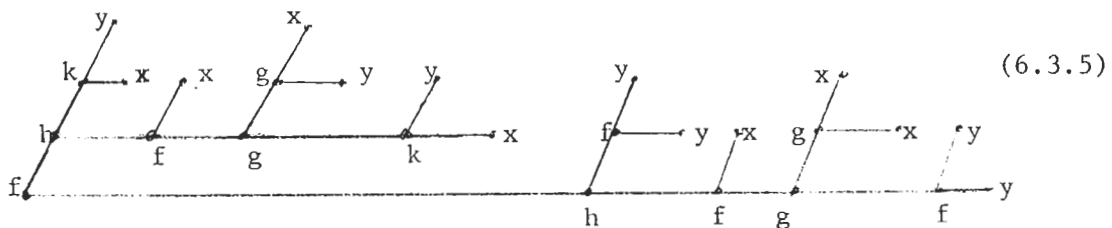
Let us introduce the term "indegree of an abbreviation". The abbreviation of (6.3.4) by p(u,v) is said to have indegree 2, the one in (6.3.1) is said to have indegree 0.

In Cλφωη there is something that plays the rôle of abbreviations of indegree 0 and there is something that plays the rôle of a particular case of indegree 1. The particular case is that p(u) abbreviates a formula that contains u only on the extreme right, like

$$x \!\!\! \diagdown$$

(6.3.6)

The use of more general cases (including indegree > 1) might lead to extensions of $C\lambda\phi\omega\eta$, some of which are not essentially more difficult than $C\lambda\phi\omega\eta$ itself, but they require a rather unpleaseant system of notation. They will not be considered in this paper.

An LSP-book (cf. [2]) is nothing but a game of abbreviations of various indegrees. We give an example, where two constants p,q (of indegree 2 and 1, respectively) are considered to be <u>primitive</u>, i.e. p and q are no abbreviations for anything else. Next we write a list of abbreviations:

$$
\begin{aligned}
r(u,v) &:= q(p(q(u),q(v))) \\
s(u,v) &:= r(q(u),v) \\
t(u,v) &:= r(s(u,v),s(v,u))
\end{aligned}
\qquad (6.3.7)
$$

(there happens to be an interpretation which is interesting, but irrelevant at the moment: u,v are propositional variables, p stands for conjunction, q for negation, r for disjunction, s for implication, t for equivalence).

The abbreviations in a "book" like (6.3.7) can be eliminated in various ways and lead to "normal forms" (which contain no other constants than primitive ones). The question of the existence and uniqueness of these normal forms is something most people take for granted! A theory for this will be one of the topics to be treated in a forthcoming Ph.D. thesis by D. van Daalen.

6.4 <u>Shifting substitution and abbreviation from metalanguage to language</u>. If CV is considered to be the "language" then substitution and abbreviation are expressed in terms of a metalanguage, i.e. a language that speaks <u>about</u> CV. One might try to extend a language like CV with a symbolism that produces some of the effects of substitution and abbreviation.

Something in this direction is done if the language is extended in such a way that a complete LSP-book (like (6.3.7)) is considered as an expression in the language. (Actually LSP does a bit more: it also expresses that some constants are primitive; in the case of (6.3.7) the book has to start with the lines $p(u,v) := PN$, $q(u) := PN$).

Quite a different point of view is taken in lambda calculus. We introduce it informally, connecting it to CV.

If we put λ_x in front of the formula (6.1.1), containing the variable x, we intend to describe the "function" that maps x into (6.1.1). After that, we no longer admit anything to be substituted for x, which is now called a <u>bound</u> variable; the other variables are called <u>free</u>.

Connected with the addition of λ's, we enrich our syntax with symbolism for "application". It tries to describe what we get if we substitute something, let us say k(c,y), for x in (6.1.1). In λ-calculus we write what can be considered as the <u>instruction</u> for substitution. The instruction says:

"apply k(c,y)   to $\lambda_x p(f(g(x,y),h(y,c,g(x,c))))$".

This has to be incorporated in the syntax. Let us take a constant δ with indegree 2 (δ has to be "new": it did not occur earlier in the language). We write the instruction as

$$\delta(k(c,y),\quad \lambda_x p(f(g(x,y),h(y,c,g(x,c))))). \qquad (6.4.1)$$

The next step is that we also admit $\delta(F,G)$ in cases where the formula G does not start with λ. This extension makes two things possible: (i) to use abbreviations, to the effect that G turns into a formula starting with a lambda if we carry out the abbreviation instructions, and (ii) to incorporate things which are, in ordinary mathematical language, expressed by phrases like: "let f be any function; its value at c is denoted by f(c)". Here we write $\delta(c,f)$ instead of f(c).

We remark that in [4] the notation for application was A(f,c). Most presentations of lambda calculus (cf. [1]) write f(c). In AUTOMATH (cf. [2]) the notation for λ_x is [x] (or rather [x : S] where S represents a type) and for application curly brackets are used: {c}f instead of f(c). This AUTOMATH notation corresponds to what we get if we draw the tree of (6.4.1): we have to put

$$(6.4.2)$$

in front of the tree (6.1.2).

Needless to say, the use of λ's and δ's is cumulative in the sense that several λ's and δ's may occur in one and the same tree.

6.5 <u>Namefree lambda calculus</u>. One of the troubles of lambda calculus is that we have to make it clear which bound variables refer to which lambdas. When we carry out substitutions we get duplication of bound variables and we might fear ambiguities. A second thing is that the tradition in mathematics is that the name of the bound variable is unessential: a formula is not considered to change if such a name is altered, provided this does not conflict with names we already have in the formula. In [4] the names of variables are eliminated. We just write λ instead of $\lambda_x, \lambda_y, \dots$, and every further bound variable is replaced by a positive integer indicating the "reference depth". This number tells us where to find the corresponding λ. If the reference depth is k, we run down the tree and take the k-th λ we encounter. If we get beyond the root of the tree we count free variables (which have to be arranged in some order) instead of λ's.

Let us call this namefree lambda calculus $C\lambda\xi$. Its syntax is the one of section 3 if we just omit the ϕ's, ω's and η's. The rôle of the letter ξ is just a formality: for every $n \in \mathbb{N}$ the $\xi(n)$ is a constant of indegree 0. "Essentially" $\xi(n)$ is the same thing as n.

6.6 <u>Beta reduction</u>. In lambda calculus there is no automatic identification between the instruction for substitution and the effect of the substitution. The <u>effect</u> of the instruction (6.4.1) is

$$p(f(g(k(c,y),y),h(y,c,g(k(c,y),c)))))\,. \qquad (6.6.1)$$

We say that (6.6.1) is obtained from (6.4.1) by <u>beta reduction</u>.

6.7 <u>Lambda calculus with reference transforming mappings</u>. If we think of lambda calculus formulas as strings that are handled by a machine, the process of substitution is less simple than the intuitive idea of replacing all x's by some other expression. What we have to do is connected with recursive definition of substitution. This recursivity is explained by an example. If we have to replace all x's by P in the formula

(where Q and R stand for trees) then we replace all x's by P in Q (effect Q*), all x's by P in R (effect R*) and we build
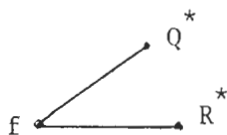
A similar thing has to be described in order to replace all x's by P in the formula $\lambda_y Q$.

In practice it is not always desirable to carry out the substitution in all branches of the tree. We may do it in some of the branches, and leave it at the mere instruction in the other branches. This may involve that instructions like (6.4.2), which carry a lambda, have to jump over other lambdas. In combination with the namefree notation this creates the need for <u>reference transforming mappings</u> (see [5]). The idea is that we attach, at various points of a tree, mappings θ of $\mathbb{N}$ into $\mathbb{N}$ (syntactically we do this by attaching a constant of indegree 1 to every θ; this constant is written as $\phi(\theta)$). The rôle of these ϕ's is in helping to indicate to what λ's endpoints of the tree refer. It works like this. We read at an end-point the number n. We run down the tree. If we encounter a λ we subtract 1, but if we encounter a θ we apply that θ . If we have, e.g.,
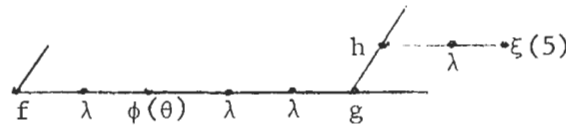

and if $\theta(2)=4$ then the $\xi(5)$ refers to the 3rd free variable. This calculus can be called $C\lambda\xi\phi$. We can describe it as in section 3, just omitting the ω's and η's. And we can provide it with the reduction operations A1,2,4,6,7,8,9 and B 1,2,3,6,7,10,11 of section 4 (i.e. all those reductions which do not contain ω's or η's). Note that repeated use of the A-reductions shifts the ϕ's towards the end-points of the tree, where they vanish on behalf of A1,A2,A3.

The formulas of $C\lambda\xi\phi$ that do not contain any ϕ's, form the subsystem $C\lambda\xi$ (see section 6.5). We can provide $C\lambda\xi$ with a set of reduction rules, viz.

$$B1, \quad B2, \quad B3, \quad B6, \quad B^*7, \quad B10, \quad B11, \qquad\qquad (6.7.1)$$

where B^*7 is obtained from B7 if we replace the right-hand side by the reduced form rf (where all ϕ's have been pushed to the end-points of the tree, see section 5). There is no equivalent of B4, where a ϕ occurs on the left. We shall comment on this B4 in sections 6.8, 6.19 and 6.20.

6.8 <u>Beta reduction in $C\lambda\xi$</u>. The beta reduction described in 6.6 can be carried out in $C\lambda\xi$ by means of a sequence of smaller steps. In order to show how this works, we start informally with an example that is not namefree. Starting from

$$(6.8.1)$$
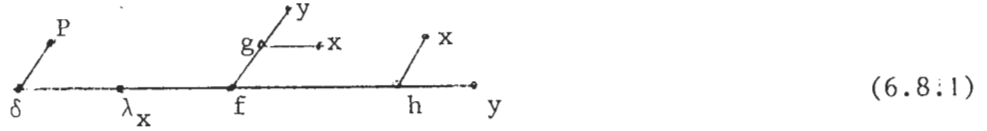
we proceed by single-step operations:

$$(6.8.2)$$

$$(6.8.3)$$
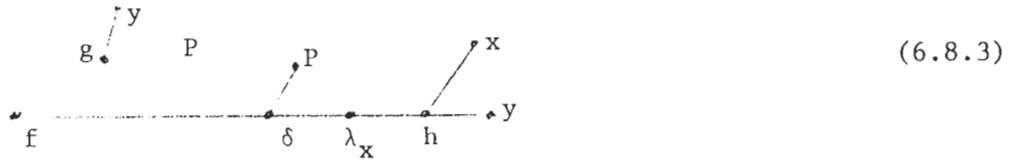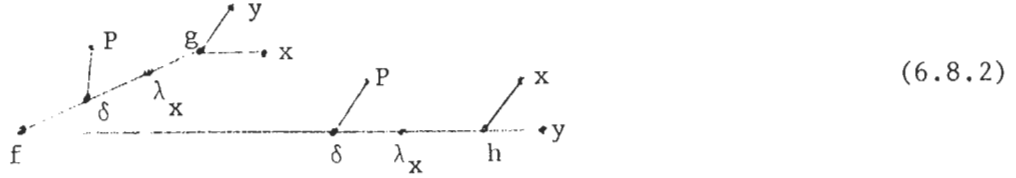
and this means that some of the x's have been replaced by P and some others have not. The single-step operations have the effect that the instructions

 is shifted to the endpoints of the tree.
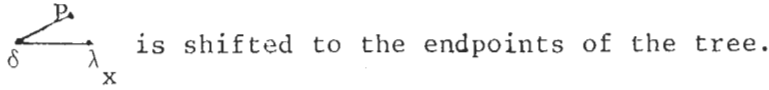
Actually we have cheated when passing from (6.8.1) to (6.8.2), taking several steps at a time. If we stick to our single-step list of section 4, we have to proceed like this:
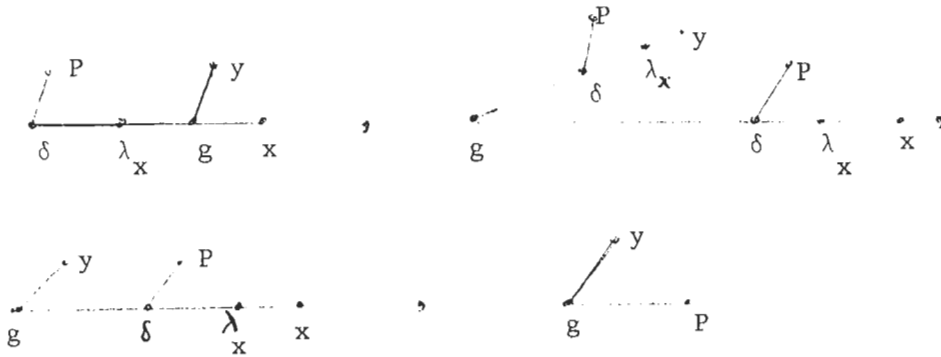
It will be clear how beta reduction is effectuated by means of a sequence of these single-step operations, shifting all instructions (δ-λ pairs) towards the end-points.

In the name-free version (with $\phi(\theta)$'s) of sections 3 and 4, this shift of substitution instructions is described by the B-reductions (except B5,B8,B9 which contain ω and η). It is not possible to shift these instructions (δ-λ pairs) over a $\phi(\theta)$, so occasionally these $\phi(\theta)$'s have to be shifted first.

The reduction B4 does not have its equivalent in $C\lambda\xi$. Stated in popular terms, B4 says that a definition (a $\delta-\lambda$ pair) may be cancelled if there is an indication (directly following that $\delta-\lambda$ pair) that no reference to that definition will ever be made. If we start shifting the ϕ's to the end-points (i.e. if we start evaluating the reduced form) this advantage gets lost. The $\delta-\lambda$ pair can still be cancelled, but not without a long sequence of B-reductions, shifting $\delta-\lambda$ pairs to the end-points where they vanish on behalf of B1 and B2. For further comments on B4 see section 6.19.

Substitution was described in [4] and [5] by means of a recursive process with things corresponding to our reduction steps A1-B11 as basic ingredients. Yet there are differences. In [4] and [5] substitution is described in the metalanguage, and not by means of substitution instructions in the formulas themselves. Another differerence is that in [4] and [5] substitutionis described for a number of variables, and not just for one at a time.

6.9   The normal form problem. A formula in $C\lambda\xi$ is said to be normal if it admits no reductions of our list of section 4. The normal form problem asks: starting from a formula, it is possible to arrive at a normal form by means of a number of reduction steps? The answer is not always affirmative. Church's well-known example, in our present notation, is

(6.9.1)
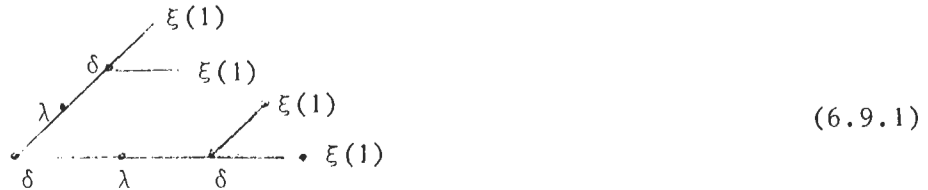
The only possible reduction is B10, leading to

(6.9.2)
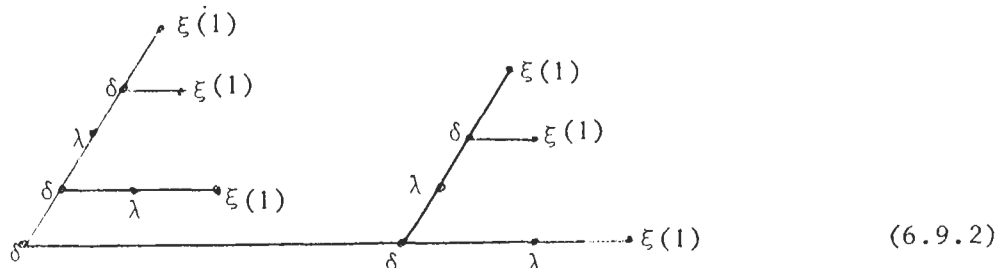
By means of two applications of B3 we get back to (6.9.1) again, and so there is an infinite reduction sequence.

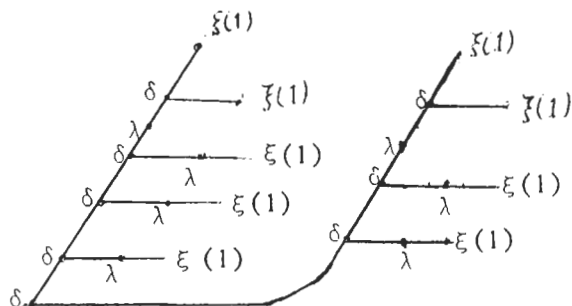It is a bit more troublesome to show that every reduction sequence is infinite. (This bit of trouble is the price we have to pay for our single-step policy!). We do it for the sake of curiosity. The formulas we can get are characterized by pairs (m,n) of nonnegative integers. As an example we draw formula (3,2):

In the general case there are m branches $\delta-\lambda-\xi(1)$ on the left and m on the right. Possible reductions are:

$$(m,0) \rightarrow (m+1,m+1) \qquad \text{by} \qquad \text{B10}$$

$$(m,n) \rightarrow (m-1,n) \qquad \text{by} \qquad \text{B3 if } m > 0,$$

$$(m,n) \rightarrow (m,n-1) \qquad \text{by} \qquad \text{B3 if } n > 0.$$

Formula (6.9.1) is (0,0). It is obvious that no reduction sequence ever terminates.

6.10 <u>Diamond properties and the Church-Rosser theorem</u>. Two formulas F,G of $C\lambda\xi\phi$ are called equivalent if there is a chain $F=F_0,F_1,\ldots,F_n=G$ such that for each i ($1 \leq i \leq n$) either F_{i-1} reduces to F_i (by one of the reductions of our list) or F_i reduces to F_{i-1}. We say that F has a reduction chain to G if we have, with the same notation, that (for all i) F_{i-1} reduces to F_i. We also say that F has a reduction chain to G if that chain is empty, i.e. if F=G.

The Church-Rosser proposition for this system says that if F and G are equivalent then H exists such that F and G have a reduction chain to H. It is very close to the Church-Rosser theorem in ordinary lambda calculus. We shall prove it in section 6.21 (theorem 6.21.2). That proof depends on having the Church-Rosser property for ordinary lambda calculus. Needless to say, it would be nicer to have a proof that sticks to the small reduction steps of our calculus, in particular since that would look promising for generalization to the more complicated system $C\lambda\xi\phi\omega\eta$.

A particular case of the Church-Rosser theorem is the <u>diamond property</u> for reduction chains. It says: if there are reduction chains from F to F_1 and from F to F_2 then there is an H with reduction chains from F_1 to H and from F_2 to H. The Church-Rosser theorem easily follows from this special case.

We also consider a <u>weak</u> diamond property: If there are single-step reductions from F to F_1 and from F to F_2 then H exists with reduction chains from F_1 to H and from F_2 to H. This weak diamond property is easily established for our system $C\lambda\xi\phi$, but unfortunately the Church-Rosser theorem does not follow from it (cf. [7]). For the weak diamond property of $C\lambda\xi\phi\omega\eta$ see section 6.22.

6.11 <u>Substitution and abbreviation in the metalanguage</u>. Substitution and abbreviation have, to a certain extent, been built in by the passage from CV to $C\lambda\xi\phi$ (cf. section 6.4). Nevertheless we can again discuss in the metalanguage what substitution (for the free variables) does to the formulas of $C\lambda\xi\phi$, and we can use abbreviations when discussing these formulas. As to substitution, this was actually done in [5].

6.12 <u>Typed lambda calculus</u>. In typed lambda calculus every bound variable is introduced as having a certain formula as its <u>type</u>. Along with it, there is a procedure for adhering a type to any formula, and for some reductions restrictions concerning the types are made. We do not go into this here. We only mention how the typing of bound variables can be described in the syntax. Instead of just a lambda we write, with a new symbol τ of indegree 2:

$$\underset{\tau \quad \lambda}{\underbrace{}^{\,.\,P}} \quad , \text{ where P is some formula.}$$

In AUTOMATH this would be read as [x : P] ... .

6.13 <u>What things can be abbreviated in $C\lambda\xi\phi$?</u> In 6.4 we explained how instructions for abbreviation can be written inside the formulas as soon as we have the lambda notation (e.g. in $C\lambda\xi$ or in $C\lambda\xi\phi$). A combination like (6.4.2) can be considered to define x as an abbreviation (in this case it abbreviates $k\overset{c}{\longleftarrow}y$; the fact that this example is <u>not</u> in the namfree calculus, does not matter). The kind of abbreviation is the one mentioned in (6.3.1) (the case of zero indegree). The case of non-zero indegree can also be dealt with, but in an indirect way, reduced to zero indegree. Instead of abbreviating, e.g.

$$p(u,v) := h(v,f(x,g(u,v))), \tag{6.13.1}$$

(i.e. the example of (6.3.4)) we define, as an abbreviation of zero indegree.
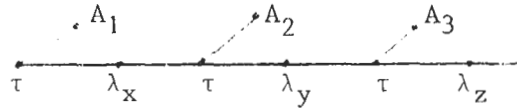
$$P := \lambda_u \lambda_v \, h(v, f(x, g(u,v))). \tag{6.13.2}$$

Instead of later use of $p(R,S)$ (where R and S are formulas) we have $\delta(S, \delta(R,P))$.

In AUTOMATH notation, (see section 6.4) this reads $\{S\}\{R\}P$.
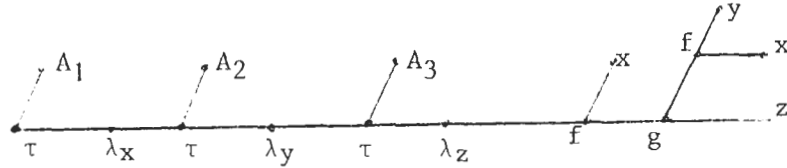
6.14 <u>Segments</u>. In addition to what was indicated in section 6.13, there are, however, still quite different things we want to abbreviate. One thing is a <u>string</u> like $\{S_1\}\{S_2\}\{S_3\}\{S_4\}$. If it occurs more than once, we wish to abbreviate it. Yet it is not a formula but a <u>part</u> of a formula, actually a beginning segment. In AUTOMATH we have many cases where we would like to abbreviate <u>strings of abstractors</u> like $[x : A_1] \, [y : A_2] \, [z : A_3]$ (also called <u>telescopes</u>).

In tree notation (cf. section (6.12)) this is the segment

$$\tag{6.14.1}$$

If we abbreviate (6.14.1) by η, then

is to be abbreviated by

$$\tag{6.14.2}$$

This is strange: the x,y,z refer to $\lambda_x \, \lambda_y \, \lambda_z$ which are hidden in the symbol η. Each time the η is used, we have to create new dummy variables, so this is an obvious case for namefree calculus.

We should certainly not confuse things like (6.14.2) with abbreviations of indegree 3 (then there is no question of built-in lambda').

Roughly speaking, segments are trees which are open-ended on the extreme right, like (6.14.1). Another example is the tree representation of the string $\{S_1\}\{S_2\}\{S_3\}\{S_4\}$:
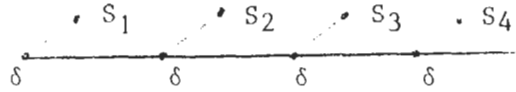
We shall use the symbol ω to represent the open end on the right. So
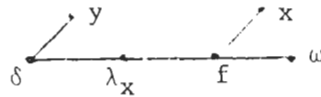
is a segment. These ω's may also occur in other branches, like
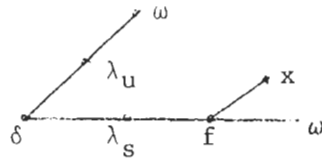
(6.14.3)

The notation of the ω's is a preliminary one; in section 6.17 we shall explain the way we actually use the ω's in our syntax.

Abbreviation and reference to abbreviation takes place as follows. Let us give the segment (6.14.3) the name η(or rather, η refers to (6.14.3) without the ω on the extreme right), then η has become a symbol of indegree 1. In tree form (not yet namefree) we can have
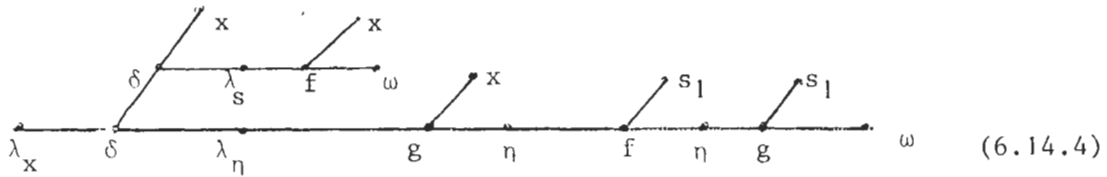
(6.14.4)

and we indicate what we get if the two η-references are replaced by their definition:
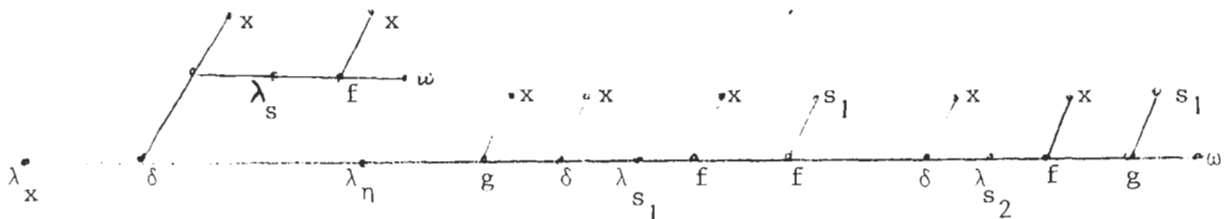
Note that we get two new dummies s_1 and s_2. The s_1's in (6.14.4) refer, in a clumsy manner, to one of them. It is clumsy since (6.14.4) does not

reveal what names are to be given to the various copies of s that arise when the η's are replaced by their definition.

The reader may wonder why we have not removed the instruction (i.e the part) after having carried it out. The answer is that (6.14.4) itself is a segment, and possible later insertions of that segment into some tree may involve further references to η.

6.15 Towards $C\lambda\xi\phi\omega\eta$. The extension of $C\lambda\xi\phi$ to $C\lambda\xi\phi\omega\eta$ is just this addition of ω's and η's. The references to ordinary variables are different from those to segments, and yet in the namefree calculus they are treated as equals: an integer n > 0 still refers to the n-th λ we meet when running down the tree. Only, when passing an η (reference to a segment) we have to realize that it stands for something having number of lambdas. So to every η we adhere (apart from the integer that tells us to which λ this η refers to) a nonnegative integer k : its weight. Taking a fragment of a tree as an example, we show by means of dotted arrows what the integers refer to

(6.15.1)
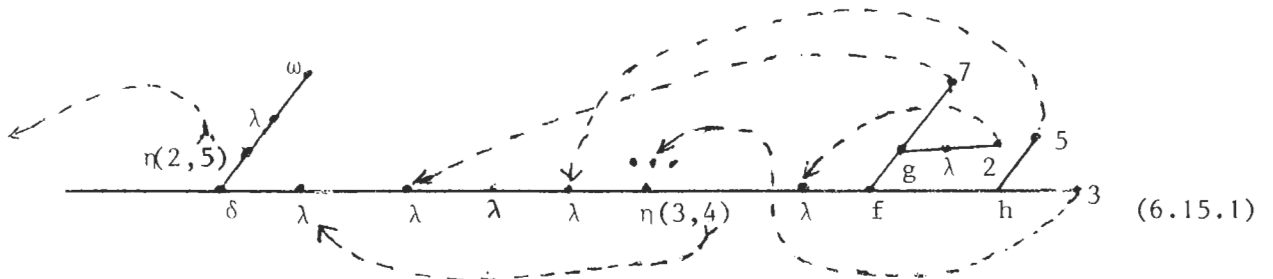
Over the η we have drawn three dots (representing what happens if the substitution is carried out: then we get three lambdas). Let us call these dots the "reference dots". The 3 on the extreme right refers to the middle dot.

6.16 The root distance of an ω. In the formulas we shall deal with, each ω corresponds to a certain delta to be called its root. If we start at an ω and run down the tree, then we get, sooner or later, to a δ into which we enter through the upper one of the two branches that enter from the right.
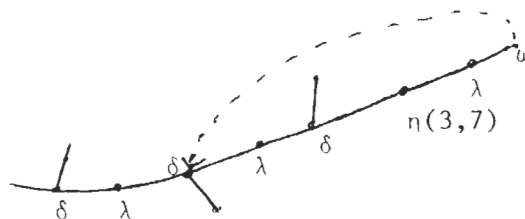
(6.16.1)

This δ is called the root of ω; here it is indicated by means of a dotted arrow. The number of lambdas we meet when travelling down from ω to its root (counting an $\eta(k,m)$ for k lambdas) is called the <u>root distance</u> of ω; here it is 5. The root distance of ω is easily seen to be equal to $\rho(t)$ (see section 3), where t is the tree growing on that same upper branch that enters the root δ from the right, viz.

$$\lambda \qquad \delta \qquad \eta(3,7) \qquad \lambda$$
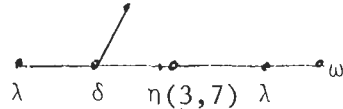
The root of an ω is invariant under the operations of our list of section 4, but the root distance is not! In the example (6.16.1), the $\delta-\lambda$ on the left might shift into the branches of the next δ (i.e. the root of ω), and increase the root distance.

In the example (6.15.1) the weight of the $\eta(3,4)$ (i.e. the number 3) is equal to the root distance of the ω in the $\delta-\lambda$ branch we are supposed to substitute for the η. So upon substitution (which can be effected by means of a number of single-step operations, cf. section 6.8) we get three λ's on the places of the three dots. Unfortunately this cannot be maintained. As we saw above, local operations on the $\delta-\omega$ branch can alter the root distance, and far away "customers" like our $\eta(3,4)$ are in no way informed about this. They are not even interested. If a $\delta-\lambda$ pair enters into the $\delta-\omega-$ branch (as indicated above), then the customers have no interest in referring to it: they keep referring to the $\delta-\lambda$ pair that shifts into the other branch of the δ. The system of reference transforming $\phi(\theta)$'s will see to it that all references (like the 3 pointing at the middle dot in (6.15.1) keep referring to the proper lambda's, even if the root distance has changed.

6.17 <u>The $\omega(k,\theta)$'s</u>: We shall now explain how we actually write our ω's syntactically, and why we do it that way.

For various reasons it is desirable to write the weight of the customers of a $\delta-\omega$ branch at the spot where we write the ω. One advantage is that this makes it easier to express a correctness condition that formulates that all customers for one and the same $\delta-\omega$ branch have the same weight. Another reason is the fact that this number plays a rôle in the θ's to be introduced presently.

As stated before (section 6.7) the $\phi(\theta)$'s can be pushed towards the end-points of the tree by means of the A-reductions. But we can no longer say that the $\phi(\theta)$'s vanish there (as we said in section 6.7 concerning $C\lambda\xi\phi$). If they

get to an ω there is no reduction rule by which they disappear. And actually they should not: they have to stand by for the possible customers, and we cannot pass the θ's to the customers as long as we do not know the customers. But we have fixed the weight k of the customers (other customers will not be acceptable) and we know that these customers will only refer to the k reference dots (cf. the three reference dots in (6.15.1)). This means that if we have shifted the φ(θ) next to an ω of weight k, then no customer will ever be interested in the value of θ(j) if j > k.

For these reasons we have decided to write ω(k,θ) (where θ has domain {1,...k}) in order to keep the essential part of θ and to discard the rest. From now on we can say again that all φ(θ) disappear at all end-points. For the case of an endpoint ω this is expressed in A3.

We now phrase the "internal reference condition" (IRC). The potential "customers" of ω(k,θ) indicate one of the references θ(1),...,θ(k). We can trace where these references refer to, just by reading the λ's, η's and φ(θ)'s on the way down. They may refer to λ's, possibly to λ's hidden in an η(the "reference dots" of the η), lying between the ω and its root δ: in that case we speak of _internal references_. They may also refer to λ's beyond that δ (even to free variables): then we speak of _external references_.

We remark that there is not much use for external references. The customers do not need them: they can refer ro these lambdas directly, without pointing at the dots.

We say that a formula t ∈ T satisfies IRC if it contains no ω(k,θ) where any of θ(1),...,θ(k) is _external._ (Note that we do not forbid the possibility that some ω has no root at all).

It is easy to prove the following theorem by checking A1-B11:

Theorem 6.17.1. If $t \in T$, $t >_1 t_1$, and if t satisfies IRC, then t_1 satisfies IRC.

The IRC will play a rôle in the diamond property (section 6.22).

6.18 _Further comments on reductions._ In Cλξ we already had the reduction B3, which expresses that a definition (i.e. a δ-λ pair) can be discarded if it is obvious from the neighbouring θ that no reference will ever be made to that definition. Now that we have ω(k,θ)'s, we can also say this for a δ-λ pair that gets in front of ω(k,θ), in cases where none of the values θ(1),...,θ(k)

refers to the λ of that pair. In other words, the $\delta-\omega$ pair can be discarded if none of these values equals 1. That is expressed in reduction rule B5.

The replacement of an η by its definition is expressed in B9. This rule gets into effect only if the $\delta-\lambda$ pair is right in front of that η. At that moment the root distance of the $\rho(k,\theta)$ is $\rho(t)$. Now requests coming from the right have to be treated as follows. If they are references j with $j \leq k$, they have to be replaced by $\theta(j)$. If they are $>k$ they were not supposed to point at one of the k dots, but jump over them. Since the number of λ's we actually substitute is $\rho(t)$, and not necessarily equal to the weight k, we have to correct these references by addition of $\rho(t)-k$. The θ_1 and θ_2 are an obvious matter of honest administration of references.

Rules A5 and B8 express what happens if a $\phi(\theta)$ or a $\delta-\lambda$ pair jumps over an $\eta(k,m)$. There we see that this η just behaves like k lambdas (cf. A7 and B7).

The effect of B9 is that $\eta(k,1)$ is replaced by a copy of the $\delta-\omega$ branch, i.e. by a copy of the segment it refers to. At this moment we explain a terminology which corresponds to a similar situation in ALGOL procedures. We discern amongst global, local and formal variables. We refer to the picture (6.15.1). The global variables are those whose λ's are situated to the left of the δ on the left. From points in the $\delta-\omega$ branch (including side branches) there can be references to these global variables. If we replace the η by the $\delta-\omega$ branch these references are just taken over: we do not make copies of these global variables. Local variables are those whose λ's lie in side-branches of the $\delta-\omega$ branch (they may also be formal variables in these side branches), and variables with λ's in the main branch as far as they cannot be referred to from the $\omega(k,\theta)$ at the end of the branch (note that this is controlled by θ). Formal variables are those in the main branch which correspond with the dots over the η. Some of these appear as lambdas, others are hidden in η's (like the $\eta(2,5)$ in (6.15.1)). Both local and formal variables are copied (i.e. replaced by new ones) if η is replaced by the segment. The difference is that from points to the right of η no references to local variables are possible; references to global variables are possible indeed: they are just the referneces to the dots. From points to the right of η it is quite possible to refer to the global variables, but this is not done by means of references to the dots. These references to global variables are made in exactly the same way as if the η had never been there.

We have to take a precaution with the $\phi(\theta)$'s before replacing the η by the segment: we have to require that there are no ϕ's in the main branch (i.e. on the line from δ to ω); this is expressed by saying in B9 that $\sigma(t)$ should

be 1 (see section 3 for the definition of σ). Such $\phi(\theta)$'s in the main branch
make sense for the side branches, but we do not intend them to act on the
customer's j's if j > k (those which have to be corrected by addition of
$\rho(t) - k$ (see above). Therefore we first have to push these $\phi(\theta)$'s into
the side branches (by A-reductions), and the only remaining one in the
main branch vanishes into the ω by A3. Once this has been achieved,
application of B9 gives just what we want.

One of the conditions of B9 is that the k in $\omega(k,\theta)$ equals the k in
$\eta(k,1)$. Actually we hope that, if we start with "sensible" formulas, we
get into situations where these two k's are different. And also we hope
we never get into a situation as described in the left-hand side of B9
if $\sigma(t) \neq 1$, just as we hope that B3 will not occur with $\sigma(t) \neq 0$. We
shall return to this in 6.22.

6.19 <u>Derivable and semiderivable rules</u>. A reduction is called <u>derivable</u> if its
effect can be obtained by a finite chain of reductions $>_1$ from our list
A1-B11. A simple example of such a rule is

$$\boxed{w_1 \mid \phi(\theta) \mid w_2} > \boxed{w_1 \mid w_2} \qquad\qquad (6.19.1)$$

if θ is the identity ($\theta(j)=j$ for all j). We can shift this $\phi(\theta)$ towards the
end-points where it vanishes according to A1,A2,A3,A4, unless it vanished
under way according to A6.

A rule is called <u>semiderivable</u> if its effect can be obtained by equi-
valence with respect to the reductions $>_1$ from our list A1-B11. In other
words, if for all reductions F > G produced by the rule there is a chain
$F=F_0,F_1,\ldots,F_n=G$ such that for each i ($1 \leq i \leq n$) either $F_{i-1} >_1 F_i$ or
$F_i >_1 F_{i-1}$ (cf. section 6.10).

An example of a semiderivable rule that looks quite useful, is

$$\boxed{w_1 \mid \delta \mid \xi(m) \mid \lambda \mid w_2} > \boxed{w_1 \mid \phi(\theta) \mid w_2} \qquad\qquad (6.19.2)$$

if $m \in \mathbb{N}$, where θ is defined by $\theta(1) = m$, $\theta(j) = j - 1$ if $j > 1$. A clear
reason why it cannot be derived is that, if the $\delta-\xi(m) - \lambda$ occurs in a $\delta-\omega$
branch, it may happen that the left-hand side of (6.19.2) satisfies IRC
(see section 6.17) whereas the right-hand side does not.

Another semiderivable rule that may turn out to be quite useful is the long distance variant of B3 (and there is a similar variant of B9). It is not a rule that can be evaluated locally like the rules Al-B11, and therefore we describe it in a different style. Let $\xi(k)$ be a point that refers to the λ of some combination $\delta \overset{t}{\angle} \lambda$. Assume that on the path down from the $\xi(k)$ to λ we have no $\phi(\theta)$'s. Then the long distance rule says: replace this $\xi(k)$ by $\phi(\theta_k)$——t where θ_k is defined by $\theta_k(j) = j+k$ for all $j \geq 1$.

We can also phrase a slightly stronger long distance rule by admitting $\phi(\theta)$'s between the λ and the $\xi(k)$, provided that it is possible to find a θ^* such that the replacement of $\xi(k)$ by $\phi(\theta^*)$——t still gives the proper corrections to all the references occurring in t.

These long distance variants of B3 can be very practical: they enable us to apply a definition at a place where we want it, without the obligation to apply it simultaneously at all other places of the tree.

A further example of a non-local semiderivable rule is an extension of B4. The rule says that we may discard a δ-λ pair (just replacing it by $\phi(\theta_1)$ with $\theta_1(j) = j - 1$ for all $j > 1$, $\theta_1(1)$ arbitrary if there is no demand (see 6.20).

As a last example of a semiderivable rule we mention that if we cross out B4 from our list Al-B11, then B4 becomes semiderivable.

6.20 <u>The system $C\lambda\xi\omega\eta$</u>. The set T of section 3 has a subset R (section 5) consisting of all ϕ-free formulas. In R there is no question of reductions Al-A9, nor of B4. The reductions B1,B2,B3,B5,B6,B10,B11 can be carried out in R, but B7,B8,B9 take us outside R. Let us define the reduction B^*7 (and similarly B^*8, B^*9) as follows: first apply B7, then apply rf (the operator of section 5) to the right-hand side. Now B^*7, B^*8, B^*9 transform formulas of R into formulas of R.

Let us call the system $C\lambda\xi\omega\eta$, i.e. R with reduction rules

$$B1,B2,B3,B5,B6,B^*7,B^*8,B^*9,B10,B11. \qquad (6.20.1)$$

Something like B4 can be formulated in $C\lambda\xi\omega\eta$ as a derivable rule. It says that a δ-λ pair can be cancelled if there is no demand for it (provided that all references occurring in the formula are adjusted because of the disappearance of a lambda). "No demand" means: (i) none of the end-points refer to this lambda, and (ii) for no $\omega(k,\theta)$ in the formula there is a j (with $1 \leq j \leq k$) such that $\theta(j)$ refers to this lambda. Let us call this derived rule B^*4.

In the theorems of section 5 it was stated that from every formula t in T there is a sequence of reductions leading to rf(t), and that this is the only

element of R we can ever reduce to. For many questions about T it suffices
to study R. The following theorem is useful in this respect.

Theorem 6.20.1. If $t_1, t_2 \in T$, and $t_1 >_1 t_2$ in $C\lambda\xi\phi\omega\eta$, then there is a
reduction chain from $rf(t_1)$ to $rf(t_2)$ in $C\lambda\xi\omega\eta$ (i.e. R with reductions
(6.20.2)).

We do not present a formal proof. The only case that requires some
attention is the one where t_1 reduces to t_2 by B4. Then we have a $\delta-\lambda$ pair
followed by $\phi(\theta)$ with $1 \notin \theta(\mathbb{N})$. Now $rf(t_1)$ is no longer followed by such a $\phi(\theta)$
In order to get from $rf(t_1)$ to $rf(t_2)$ we have to apply the operations of
(6.20.1), shifting the $\delta-\lambda$ pair to the end-points or towards the ω's. The
$\delta-\lambda$ pair is annihilated there because the $\phi(\theta)$ arrived there first.

From theorem 6.20.1 it can be seen that once we have the Church-Rosser
theorem for $C\lambda\xi\omega\eta$, we have it for $C\lambda\xi\phi\omega\eta$. In 6.21 it is shown how this works
for $C\lambda\xi$ and $C\lambda\xi\phi$.

6.21 The system $C\lambda\xi$. Just as we got from $C\lambda\xi\phi\omega\eta$ to $C\lambda\xi\omega\eta$ in 6.20, we can get from
$C\lambda\xi\phi$ to $C\lambda\xi$. The formulas of $C\lambda\xi$ form the set R_1, consisting of all formulas
of R which contain no ω's or η's. In the set R_1 we can consider the reductions

$$B1, B2, B3, B6, B^*7, B10, B11. \qquad (6.21.1)$$

The difference of $C\lambda\xi$ with ordinary lambda calculus (e.g. in the form
of [4] is very slight: in [4] the reductions are just β-reductions and not
our present single-step β-reductions. A full β-reduction (notation $>_\beta$) is
carried out by taking some $\delta-\lambda$ pair and pushing it to the end-points of the
tree. So if $t_1 >_\beta t_2$ then we can get from t_1 to t_2 by means of a chain of
reductions $>_1$. The converse is not true, but the following is easy to see:
if $t_1 >_1 t_2$ then t_3 exists with β-reduction chains from t_1 to t_3 and from t_2
to t_3. It follows that if $t_1 >_1 t_2$ then t_1 and t_2 are β-equivalent (i.e.
connected by a chain of $>_\beta$'s and $<_\beta$'s).

Theorem 6.21.1. (Church-Rosser theorem for $C\lambda\xi$). If $t_1 \sim t_2$ (equivalence
in the sense of $>_1$) then there is a t_3 with reduction chains (in the sense
of $>_1$) from t_1 to t_3 and from t_2 to t_3

Proof. We know that if $t_1 >_1 t_2$ then t_1 and t_2 are β-equivalent (see above),

so if $t_1 \sim t_2$ then t_1 and t_2 are β-equivalent. We know the Church-Rosser theorem for R_1 with $>_\beta$ (see [4] for the namefree formulation). So if $t_1 \sim t_2$ there is a t_3 with reduction chains (in the sense of $>_\beta$) from t_1 to t_3 and from t_2 to t_3. These reduction chains can be refined to reduction chains in the sense of $>_1$. This proves the theorem.

Theorem 6.21.2. (Church-Rosser theorem for $C\lambda\xi\phi$). If $t_1 \sim t_2$ in $C\lambda\xi\phi$ (equivalence in the sense of $>_1$) then there is a t_3 with reduction chains (in the sense of $>_1$) from t_1 to t_3 and from t_2 to t_3.

Proof. Let $t_1 \sim t_2$. By theorem 6.20.1 we have $rf(t_1) \sim rf(t_2)$ in $C\lambda\xi$ (note that $rf(t_1)$ and $rf(t_2)$ contain no ω's and η's, and that the reduction chain does not produce such ω's and η's from nowhere: our reduction rules have no ω's or η's on the right if they do not occur on the left). By theorem 6.21.1 we have t_3 with reduction chains from $rf(t_1)$ to t_3 and from $rf(t_2)$ to t_3. Since we have also chains from t_1 to $rf(t_1)$ and from t_2 to $rf(t_2)$ this finishes the proof.

6.22 Weak diamond property for $C\lambda\xi\phi\omega\eta$. We are still quite far from proving the Church-Rosser property for $C\lambda\xi\phi\omega\eta$, but proving the weak diamond property (see section 6.10) seems to be the least thing we can do.

The reduction rules A1-B11 were just designed to satisfy this weak diamond property in the first place. The many cases of the weak diamond property that actually arise, were checked independently by R.Wieringa [10] and   L.S. van Benthem Jutting. They showed the weak diamond property for the sub-system consisting of all formulas of $C\lambda\xi\phi\omega\eta$ that do not lead to a deadlock (as defined in 6.23). In particular this means that IRC is required.

A case where IRC does not hold and that gives trouble is

$$| \delta \; |t \; | \lambda \; | \delta \; \omega(1,\theta_1) \; | \lambda \; |\eta(1,1) \; |\omega(2,\theta_2) \; |$$

where $\theta_1(1)=1$, $\theta_2(1)=1$, $\theta_2(2)=2$. If we start applying B9 we get into a situation where $\theta_2(1)$ and $\theta_2(2)$ refer to one and the same lambda (and this will never change by further reductions). If we start applying B10 we never get in such a situation by any further reductions. For other cases where deadlocks give trouble see [10].

6.23 Correctness of formulas in $C\lambda\xi\phi\omega\eta$. We shall say that a formula is at a deadlock in each one of the following circumstances:

(i)   it is the left-hand side of B3 with $\sigma(t)\neq 0$,

(ii) it is the left-hand side of B9 with $\sigma(t)\neq 0$ or $t\neq \lceil s\rceil\omega(k,0)\rceil$,

(iii) it does not satisfy the condition IRC (section 6.17).

A formula which is at a deadlock may still admit reductions.

We say that a formula t_1 leads to a deadlock if there is a reduction sequence $t_1 >_1 t_2 >_1 \ldots >_1 t_n$ such that t_n is at a deadlock.

This definition is not very practical. We of course like to have a decision procedure that shows in a finite number of steps whether a given formula leads to a deadlock or not.

Actually we want more. We want to be sure we are dealing with formulas that do not lead to deadlocks if we apply operations which are not reductions but yet reasonable things in formula manipulations. We think of substitution of other formulas or segments for the free variables of t, or substitution of t into other formulas, also in cases where t is a segment.

We feel that a general notion of "correctness" should be formulated in such a way that if we act "correctly" with "correct" formulas, we never run into a deadlock. Some indications of what might be done are to be found in the next sections.

6.24 Frames. In ordinary lambda calculus (let us say in $C\lambda\xi$) it is a fundamental question whether a formula is normalizable, i.e. whether it can be reduced to a form that does not admit any further reductions. Together with the Church-Rosser property, normalizability gives unique normal forms. In order to study normalization in typed lambda calculi, R.P.Nederpelt [9] developed a system in which a certain tree-shaped norm was attached to every formula, somehow indicating a frame of the expression serving as the type of that formula. He showed that the possibility of attaching such frames ("normability") guarantees normalizability.

In $C\lambda\xi\phi\omega\eta$ (or in $C\lambda\xi\omega\eta$) there is the problem of finding guarantees against deadlocks. The idea of attaching a kind of frame trees presents itself naturally: the frames have to indicate whether a variable refers to a segment or not, and if it does, in what manner that segment is composed of segments and other formulas. Digging deeper into this, we observe   that the matter cannot be separated from a kind of normability, and it seems to be appropriate to study deadlock-prevention and normalization simultaneously.

In the system we are presenting here, the frames will have a red root in case of a segment, and a green root in case of other formulas. In the red-rooted frames the edges pointing to the root are divided into three groups, with i,j,k edges, respectively. A vague indication of what we intend, follows here. The first   i

edges are intended to play the same rôle as the root-pointing edges in case of a green root: they indicate the nature of the λ's that remain in front of the formula after certain reductions have been carried out. The next j edges indicate the structure of the δ-branches we have not been able to dissolve by β-reduction. The last k correspond to the k reference dots we have pictured (cf. (6.15.1)) over an $\eta(k,m)$: they display the frames of the k formal variables the "customers" have in mind. These frames will be called <u>reference frames</u>.

We now describe the syntax of the frames formally. We start with a set of one-letter words consisting of the (distinct) elements $g_h (h \in \mathbb{N}_0)$ and $r_{i\,j\,k}$ $(i,j,k \in \mathbb{N}_0)$. The set of all frames is defined as the minimal solution of

$$F = \{g_0\} \cup \boxed{g_1 \boxed{F}} \cup \boxed{g_2 \boxed{F} \boxed{F}} \cup \ldots \cup \{r_{0\,0\,0}\} \cup$$

$$\cup \bigcup_{i,j,k \in \mathbb{N}_0 | i+j+k > 0} \boxed{r_{i\,j\,k} \boxed{F} \ldots \boxed{F}}$$

(the $r_{i\,j\,k}$ is followed by i+j+k F's). In case of a red root we can write

$$\boxed{r_{i\,j\,k} \,\Big|\, u_1 \,\Big|\, \ldots \,\Big|\, u_i \,\Big|\, v_1 \,\Big|\, \ldots \,\Big|\, v_j \,\Big|\, w_1 \,\Big|\, \ldots \,\Big|\, w_k} \ . \qquad (6.24.1)$$

The frames $w_1,\ldots,w_k$ are the reference frames of the frame (6.24.1).

In notations like (6.24.1) we admit the possibilities like i=0. In that case the u's have to be omitted entirely.

6.25 <u>Frame products</u>. For use in section 6.26 we define, under some conditions, a "product" of frames f_1 and f_2. If these conditions are satisfied we shall say that the product is "valid". It will be denoted by $f_1 * f_2$, and it will become a frame with the same root colour as f_2. First take the case where f_2 has a red root:

$$f_1 = \boxed{r_{i\,j\,k} \,\Big|\, u_1 \,\Big|\, \ldots \,\Big|\, u_i \,\Big|\, v_1 \,\Big|\, \ldots \,\Big|\, v_j \,\Big|\, w_1 \,\Big|\, \ldots \,\Big|\, w_k} , \qquad (6.25.1)$$

$$f_2 = \boxed{r_{m\,n\,p} \,\Big|\, x_1 \,\Big|\, \ldots \,\Big|\, x_m \,\Big|\, y_1 \,\Big|\, \ldots \,\Big|\, y_n \,\Big|\, z_1 \,\Big|\, \ldots \,\Big|\, z_p} . \qquad (6.25.2)$$

(The u's, v's, w's, x's, y's, z's are frames). The idea in forming the product is that v_j will cancel x_1, that v_{j-1} will cancel x_2, etc. If $j \leq m$ the validity condition is

$$v_j = x_1, \quad v_{j-1} = x_2, \quad \ldots \quad , \quad v_1 = x_j$$

(if $j = 0$ this condition is void) and if this is satisfied,

$$f_1 * f_2 = \boxed{r_{i+m-j\ n\ p} \mid u_1 \mid \cdots \mid u_i \mid x_{j+1} \mid \cdots \mid x_m \mid y_1 \mid \cdots \mid y_n \mid z_1 \mid \cdots \mid z_p}.$$

If $j > m$ the validity condition is

$$v_j = x_1, \quad v_{j-1} = x_2, \quad \cdots \quad , \quad v_{j-m+1} = x_m,$$

and if this is satisfied,

$$f_1 * f_2 = \boxed{r_{i\ j-m+n\ p} \mid u_1 \mid \cdots \mid u_i \mid v_1 \mid \cdots \mid v_{j-m} \mid y_1 \mid \cdots \mid y_n \mid z_1 \mid \cdots \mid z_p}.$$

If the root of f_2 is green, we have, instead of (6.25.2),

$$f_2 = \boxed{g_m \mid x_1 \mid \cdots \mid x_m}.$$

Now we require for validity of the product that

$$j \leq m, \quad v_j = x_1, \quad \cdots \quad , \quad v_1 = x_j$$

and if this is satisfied we define

$$f_1 * f_2 = \boxed{g_{i+m-j} \mid u_1 \mid \cdots \mid u_i \mid x_{j+1} \mid \cdots \mid x_m}.$$

We note that the product operation is associative. That is, if f_1, f_2, f_3 are frames, if $f_2 * f_3$ is valid, and if $f_1 * (f_2 * f_3)$ is valid, then $f_1 * f_2$ and $(f_1 * f_2) * f_3$ are valid, and

$$f_1 * (f_2 * f_3) = (f_1 * f_2) * f_3. \tag{6.25.3}$$

It also works the other way: if the products on the right are valid, the products on the left are valid too.

6.26 <u>Framed formulas</u>. The set T^* (the set of all "framed formulas") can be defined in the same way as T in section 3, with the difference that the point λ (see fig. 1) is replcaed by a set $\lambda^*(F)$, where λ^* is an injection of F into X_1 (with $\lambda^*(F)$ disjoint from $\phi(\mathbb{N})$ and $\eta(\mathbb{N}_0 \times \mathbb{N})$). This takes care of attaching a frame to every instance of λ in our formulas. We also adhere frames to the free variables, and therefore we consider formulas t (with $t \in T^*$) with respect

t_0 a sequence of frames $f_1, f_2, \ldots$ (f_i is intended to describe the frame of the i-th free variable).

We shall now describe an algorithm that evaluates (if $t \in T^*$, $f_1 \in F$, $f_2 \in F, \ldots$)

$$\text{frame}(t, f_1, f_2, \ldots) \tag{6.26.1}$$

as an element of the set $F \cup \{\underline{\text{incorrect}}\}$. That is, a formula t is (in the context of $f_1, f_2, \ldots$) considered either as incorrect or as correct, and in the latter case it has a frame.

For the time being we shall restrict ourselves to $\omega(M)$, $\xi(\mathbb{N})$, $\eta(\mathbb{N}_0 \times \mathbb{N})$, $\phi(\mathbb{N}^{\mathbb{N}})$, $\lambda^*(F)$, δ, forget about all other elements of X_0, X_1, X_2, and forget about $X_3, X_4, \ldots$.

We shall define (6.26.1) recursively according to (3.1). In all cases we assume $f_1, f_2, \ldots$ to be frames.

(i) If $m \in \mathbb{N}$ then

$$\text{frame}(\xi(m), f_1, f_2, \ldots) = \begin{cases} f_m & \text{if } f_m \text{ has a green root,} \\[2mm] \underline{\text{incorrect}} & \text{if } f_m \text{ has a red root.} \end{cases}$$

(ii) If $(k, \theta) \in M$ then

$$\text{frame}(\omega(k, \theta), f_1, f_2, \ldots) = \boxed{r_{0\,0\,k} \mid f_{\theta(1)} \mid \cdots \mid f_{\theta(k)}} .$$

(iii) If $\theta \in \mathbb{N}^{\mathbb{N}}$, $t \in T^*$ then

$$\text{frame}(\boxed{\phi(\theta) \mid t}, f_1, f_2, \ldots) = \text{frame}(t, f_{\theta(1)}, f_{\theta(2)}, \ldots).$$

(iv) If $k \in \mathbb{N}_0$, $m \in \mathbb{N}$, $t \in T^*$ then for correctness of

$$\text{frame}(\boxed{\eta(k, m) \mid t}, f_1, f_2, \ldots) \tag{6.26.2}$$

we require that the root of f_m is red and that the number of reference parameters of f_m equals k. Denoting these parameters by $w_1, \ldots, w_k$ we moreover require that

$$f^1 = \text{frame}(t, w_1, \ldots, w_k, f_1, f_2, \ldots)$$

is correct, and that the product $f_m * f^1$ is valid. If all these conditions are satisfied, (6.26.2) is defined as $f_m * f^1$.

(v) If $f \in F$, $t \in T^*$ then for correctness of

$$\text{frame}(\overline{|\lambda^*(f)|\,t|}, f_1, f_2, \ldots)$$
(6.26.3)

we require that

$$f^1 = \text{frame}(t, f, f_1, f_2, \ldots)$$

is correct, and that the product

$$\overline{|r_{1\ 0\ 1}|f|f|} * f^1$$
(6.26.4)

is valid. If these conditions are satisfied, (6.26.3) is defined as (6.26.4).

(vi) If $t_1 \in T^*$, $t_2 \in T^*$ then for correctness of

$$\text{frame}(\overline{|\delta|t_1|t_2|}, f_1, f_2, \ldots)$$
(6.26.5)

we require the correctness of f^1 and f^2, where

$$f^1 = \text{frame}(t_1, f_1, f_2, \ldots),$$

$$f^2 = \text{frame}(t_2, f_1, f_2, \ldots),$$

and the validity of

$$\overline{|r_{0\ 1\ 0}|f^1|} * f^2 .$$
(6.26.6)

If these conditions are satisfied, we define (6.26.5) as (6.26.6).

The following theorem shows that the definition of the frame of a framed formula makes sense:

<u>Theorem 6.26.1.</u> Let $f_1, f_2, \ldots \in F$, $t_1 \in T^*$ and let

$$\text{frame}(t_1, f_1, f_2, \ldots)$$
(6.26.7)

be correct. Let $t_1 >_1 t_2$ (the $>_1$ refers to the reductions of section 4). Then

$$\text{frame}(t_2, f_1, f_2, \ldots)$$

is again correct, and has the same value as (6.26.7).

It is not very difficult to prove the theorem by direct verification of all cases. The hardest case is B9, where we have to carry out induction with respect to the length of the main branch of s.

If we start with a framed formula, we can never run into a deadlock of the type (i) or (ii) of section 6.23. The deadlock of the type (iii) is not excluded, but if we start with a formula satisfying the condition IRC, that condition will not be violated by reductions.

The above theorem paves the way to a theory of normal forms of framed formulas, although it must be said that the normal forms will not be very simple. The following two cases seem to be manageable:

(i) The "ordinary" case: the frames of all free and bound variables have green roots, and there are neither ω's nor η's. Such formulas are very close to the normable formulas as considered by Nederpelt [9], and normalizability can be proved in the same fashion.

(ii) The case where the free variables have no red roots and where every red-rooted lambda is immediately preceded by a delta. In this case all η's can be removed by reductions, and the formula can "almost" be reduced to the ordinary case.

6.27 <u>AUTOMATH books</u>. An AUTOMATH book (see [2] for an informal introduction) contains a number of formulas arranged in lines. Every line contains a context indicator, a new identifier, and two formulas, of which the second is the "type" of the first. It is possible to condense the complete contents of such an AUTOMATH book into a single line (AUT-SL see [3]). This line contains an amount of duplication, which can be reduced considerably by internal definitions (see [9]), but yet we keep quite some duplication in strings and telescopes. With $C\lambda\xi\omega\eta$ these can be abbreviated as well, and we get to a formula which bears all the information of the AUTOMATH book (including the way things are abbreviated in that book) and which is possibly shorter than the original book. In this way the layout of the book is replaced by a much simpler one: a book gets the same structure as a formula. The book is open-ended: it can always be extended by adding further lines. Acoordingly, the representation of a book in $C\lambda\xi\omega\eta$ becomes a segment (i.e. it ends with an ω).

One of the advantages of this approach is that the reductions of the language theory may come very close to the elementary steps a computer has to take for checking correctness. That is, language definition and checking programs get closely together. In this respect it seems to be efficient to incorporate things like long distance beta reduction (see section 6.19), for this is what the AUTOMATH checkers have essentially used thus far.

The tree that represents an AUTOMATH book has usually a very long main branch (the length corresponds to the number of lines in the book, and each line is one of the very many elementary steps we take when building some substantial portion of mathematics) ending in an ω. The side branches are relatively short: each one of them represents the information contained in a line, as shown in the example of section 6.28.

Any set of consecutive lines in an AUTOMATH book is a segment of the tree, and therefore it can be abbreviated by a single symbol. This makes it clear that the abbreviation facilities of $C\lambda\xi\omega\eta$ are stronger than those of AUTOMATH itself.

A minor abbreviation facility available in AUTOMATH and not in $C\lambda\xi\omega\eta$ is the following one. If an identifier f was originally introduced in the context x,y,z,w, say, then later use of f(P,Q,R,S), with expressions P,Q,R,S. is possible. But it is also possible to use fewer subexpressions, f(R,S), say, and then we mean f(x,y,R,S) (add the original variables on the left). This kind of abbreviation is not in $C\lambda\xi\omega\eta$, but it can of course be incorporated in various ways with some extra trouble.

Above we mentioned $C\lambda\xi\omega\eta$ and not $C\lambda\xi\phi\omega\eta$. The latter system plays an intermediate rôle. The ϕ's are nor required in order to write an AUTOMATH book, but they come to life in the process of efficient checking, including postponement of substitution.

6.28 Examples. We show two examples of AUTOMATH books and the way they can be written as a formula in $C\lambda\xi\omega\eta$. In particular we show what can be done about the PN's (the axioms of the book). The first example will be slightly stylized: expressions have been replaced by single letters.

|			
x : =	———	: A
y : =	———	: B
p : =	PN	: C
g : =	D	: E
z : =	———	: F
r : =	G	: H
s : =	PN	: K
t : =	L	: M

First we replace all identifiers defined on the empty context. Thus we replace
p by a new identifier ②p, g by ①g, r by ②r, s by ①s (the integer inside the
circle indicates the number of abstractions); and we write, in the empty context

②p :   [x : A][y : B]C
①g := [x : A]D   : [x : A]E
②r :   [x : A][z : F]G : [x : A][z : F]H
②s :   [x : A][z : F]K
①t := [x : A]L   : [x : A]M

For the type of a variable we shall use a system that is slightly different
from the one suggested in section 6.14. A variable appears as a lambda in the tree,
and we shall now indicate it as a node with indegree 2. Instead of ——•—— we
take $\underset{\lambda}{\overset{A}{\cdot}}$ if A is the type of the variable. We have to stipulate that referen-
ces in A are interpreted by skipping this lambda when descending the tree (without
the usual subtraction of 1). For λ's which will be used as segment variables we
shall not do this, and just keep indegree 1.

With these conventions, the book is translated into the following formula:

The system will be clear. The three kinds of lines, block opener lines, PN
lines, and definition lines become, respectively,

The reference arrows leading from an η to the λ of a block opener just
correspond to the context indicator of the line.

In the example we have not presented any details about the A,B,... .
These may contain variables, and in the tree interpretation these variables
are indicated by references to some lambda. It should be pointed out that these
are not the block opening lambdas on the main branch, but lambdas hidden in η's
on the side branches, i.e. reference dots of η's. Our second example will show
this clearly.

Our second example is a PAL book (i.e. there are no abstractions and applications in expressions inside the lines).

				type
–	⌐	α :=	———	type
α		x :=	———	α
x		y :=	———	α
y		EQ :=	PN	prop
x		a :=	PN	EQ(x,x)
y		u :=	———	EΩ(x,y)
u		z :=	———	α
z		v :=	———	EQ(z,y)
v		b :=	PN	EQ(z,x)
u		c :=	b(y,a(y))	EQ(y,x)
z		w :=	———	EQ(y,z)
w		k :=	c(y,z,w)	EQ(x,z)

Writing type as τ and prop as π,
abbreviating ξ(n) to n, and ω(k,θ₀)
(θ₀ is the unit mapping) as ω(k),
the formula becomes:
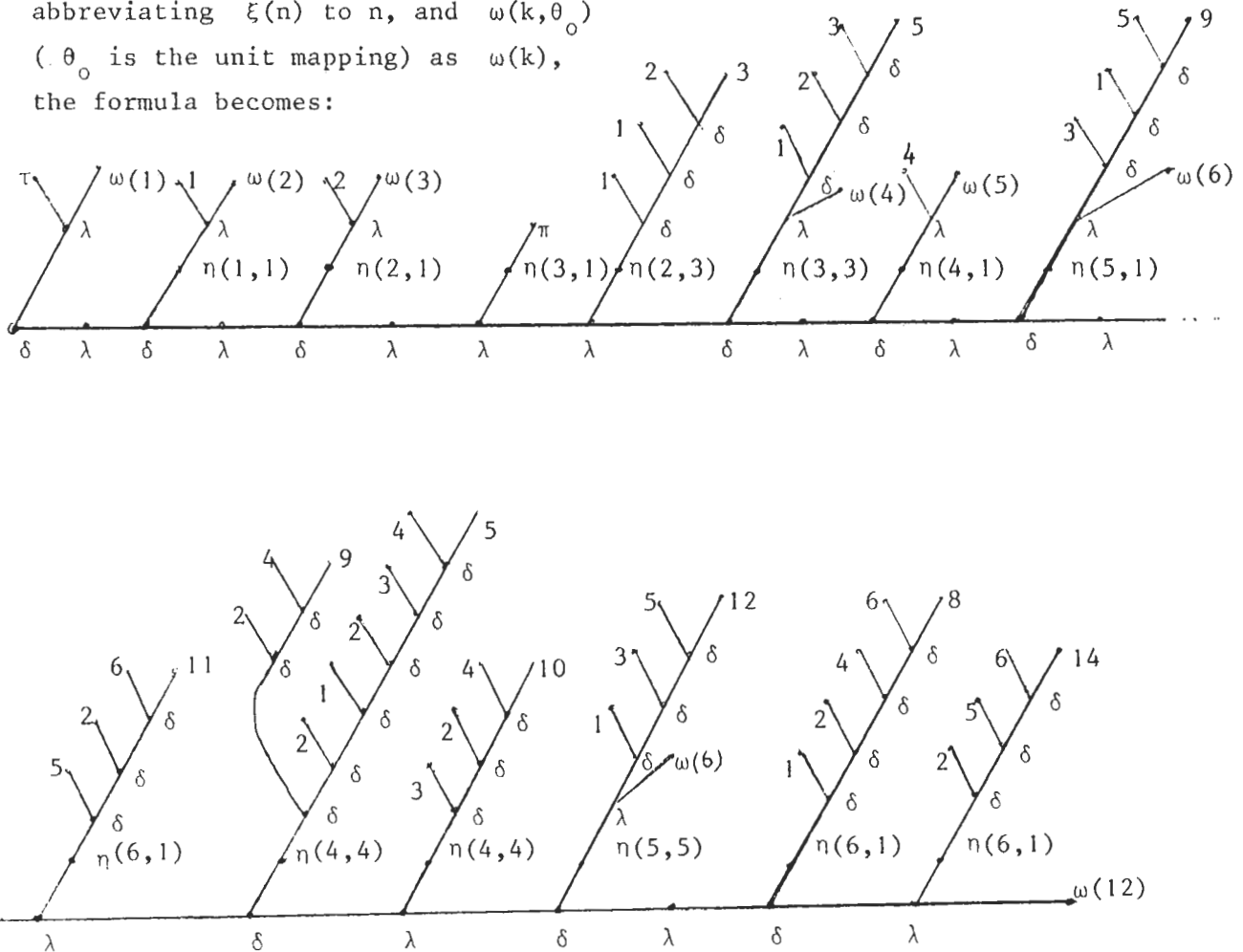
It is easy to see how the reference numbers are calculated. For example, in (k,m), the k is the length of the indicator string, and m the distance (in number of lines) to the last block opener.

References:

[1] Barendregt, H.P.: The type free lambda calculus, ch. D7 in Handbook of Mathematical Logic, ed. J. Barwise, North-Holland Publ.Comp., Amsterdam-New York-Oxford 1977.

[2] de Bruijn, N.G.: AUTOMATH, a language for mathematics. Séminaire de Mathématiques Supérieures - été 1971, Les Presses de l'Université de Montréal, 1973. Lecture Notes prepared by B.Fawcett.

[3] de Bruijn, N.G.: AUT-SL, a single line version of AUTOMATH. Notitie 22, Department of Mathematics, Technological University Eindhoven, May 1971.

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

[5] de Bruijn, N.G.: Lambda calculus with namefree formulas involving symbols that represent reference transforming mappings. Kon.Ned. Akad.Wetensch.Proc.Ser.A 81 (=Indag.Math. 40) 348-356 (1978).

[6] de Bruijn, N.G.: Notation for concatenation. Memorandum 1977-09 July 1977, Eindhoven University of Technology, Department of Mathematics.

[7] de Bruijn, N.G.: A note on weak diamond properties. Memorandum 78-08 August 1978, Eindhoven University of Technology, Dept. of Math.

[8] Knuth, D.E.: The Art of Computer Programming, Vol. 1. Addison-Wesley, Reading, Mass. (1968).

[9] Nederpelt, R.P.: Strong normalization in a typed lambda calculus with
                    lambda structured types. Dissertation June 1973 - Eindhoven
                    University of Technology, Dept. of Math.

[10] Wieringa, R.M.A.: Een notatiesysteem voor lambda-calculus met definities,
                    Master's Thesis, Dept. of Math. Eindhoven University of
                    Technology, The Netherlands.