Telescopic mappings in typed lambda calculus

INFORMATION AND COMPUTATION 91, 189–204 (1991)

Telescopic Mappings in Typed Lambda Calculus

N. G. DE BRUIJN

Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513, 5600MB Eindhoven, The Netherlands

The paper develops notation for strings of abstractors in typed lambda calculus, and shows how to treat them more or less as single abstractors.

1. INTRODUCTION

1.1. Lambda calculus is not intended to be just a formal game: it was created with the intent to represent operations with functions in mathematics. This had become necessary ever since the time (in the first half of the 19th century) that mathematicians began to see functions as objects, rather than as constructions in the metalanguage. As we see it today, lambda calculus is indispensable in this connection, and it is very remarkable that for such a long time mathematicians have been trying to live without it.

Now that lambda calculus is a well-delveloped subject, it is able to cover many other things than the standard idea of functions in mathematics. It can handle other kinds of mapping situations, where the domain and range consist of things which are quite different from the usual mathematical objects. To mention a few, they can be mathematical proofs (cf. Bruijn, 1970, 1980) but also things like geometrical constructions and computer programs (cf. de Bruijn, 1984; 1990; Coquand and Huet, 1988).

1.2. Much of this trade can be treated by means of untyped lambda calculus, but it has several advantages to take *typed* lambda calculus instead. The typing machinery is not just a powerful tool that provides satisfactory results in language theory, like strong normalization, but is also useful for interpretations where typing somehow corresponds to what we intuitively express by the words 'is a': this *is a* construction, that *is a* real number, this *is a* proof of Pythagoras' theorem, that *is a* program, etc.

In typed lambda calculus, all variables have a type, and such types can

0890-5401/91 $3.00

in general be lambda expressions themselves. This typing of the variables generates typing of the full lambda expressions. An essential feature is the development of a notion of *validity* (or *correctness*), which is connected to requiring everywhere that the argument we have for a function has the type that was given as the type of the domain variable at the occasion of the introduction of the function.

1.3. But although the lambda calculus is able to deal with more situations than just the ordinary functional relationships of mathematics, there still is a kind of deficiency, even when dealing with those ordinary functions. The simplest case is the one of functions of two variables x and y. We can describe them as functions of x whose values are functions of y. In the formulas that gives rise to pairs like $\lambda x \lambda y$. But that is not always what we want: we often want to think of those functions as a function of a single variable, i.e., the variable point in the xy-plane. In lambda theories extended with particular facilities for handling cartesian products and definitions the trouble may be overcome, but it gets more difficult when the type of the y depends on x. Moreover, if we want to apply lambda calculus for constructions which are no longer ordinary mathematical functions, the notion of cartesian products has to be revised too.

These remarks lead to the question of whether it is possible to treat sequences of λ's, like the $\lambda x \lambda y$, as a whole, without condensing them into a single λ. This paper will present a framework for organizing this.

1.4. The application of type theory to the representation of formal knowledge such as mathematics runs the danger of getting much harder than we expect at first, at least if we insist on having a system in which it is feasible to express everything that we intuitively believe to be formal already. In order to get a kind of feasibility, system designers have attached extra features here and there, sometimes quite different from the basic ideas of typed lambda calculus. An example is the feature of *type inclusion* in AUT-68 (see de Bruijn, 1970, 1980). More or less equivalent is the usage of Π's in many other systems (like Zucker, 1975; Martin-Löf, 1984). In all such cases these attachments introduce new reductions, which lead to more definitional equivalences, and that means that the human user can leave more details to the computer.

For a recent survey of a number of typed lambda calculi we refer to (Barendregt, to appear).

1.5. It is difficult to compare all the different systems of typed lambda calculus, since they have many ideas in common without having been developed on a common basis. An attempt to such a common basis for various systems was given in the language AUT-QE-NTI (see de Bruijn,

1978) (NTI stands for 'no type inclusion'). With some slight adjustments AUT-QE-NTI is representable in the general typed lambda calculus $\Delta\Lambda$ of (Bruijn, 1987). Compared to the system of (de Bruijn, 1971; Nederpelt, 1973), $\Delta\Lambda$ is more liberal about the application of a function to an argument, and that liberalism is just what we need for treating definitions and abbreviations as a part of the language. This is important since processing large amounts of mathematical material would be unfeasible without such a feature.

In Section 2 we shall describe the notations and some of the basic properties for the kind of lambda calculus to be referred to in this paper, without fixing exactly what calculus we have in mind.

Since it seems that there is no absolute standard for typed lambda calculus yet, and since the ideas on telescopic mapping are not tied to one particular kind of theory, the paper will not attempt to be absolutely formal.

1.6. A few remarks here about the word "telescope" for an abstractor string. The word was inspired, of course, by the old-fashioned instrument consisting of segments that slide one into another. In Automath the words "telescope" and "context" were both used, whereas others (cf. Martin-Löf, 1984) used the word "context" only. One can say that a context is a closed telescope, i.e., a telescope without free variables. But in Automath the word "context" was used for closed telescopes only as far as they were used to indicate the collection of assumptions and typed variables for a particular set of lines in a book. In that way the word is close to its meaning in science and literature in general. One can say that contexts are used to describe the book structure, and that telescopes appear in expressions in the book.

Properly speaking one might say that in its general usage "context" refers not just to assumptions and variables, but also to the defined notions that can be freely used. This comes near to what was called a "knowledge frame" in (de Bruijn, 1987). In Automath that notion was not used, since there was no need for contexts that include definitions: the instantiation machinery of Automath enables us to refer to all previously defined notions in the same as well as in other contexts.

Another reason for not using the word "context" for strings of abstractors in general, not even if they are closed, is that for technical notions one should not use names that have a meaning already in one's own metalanguage.

1.7. Automath was the first case of a type theory that was used very extensively for the writing of mathematics, and it was there that in the early 1970s the need to think in terms of telescopes and telescopic mappings was felt and expressed for the first time.

In his work on AUT-II J. Zucker (1975) made extensive use of abbreviations for telescopes in his introduction into modern mathematics. As an example we mention that the notion "group" is described by a telescope (and, in the terminology of this paper, a particular group is represented as a vector fitting into that telescope). The sequence of abstractors involved in such a telescope provide names for the type and the set of the group elements, for the unit element, and the assumptions that take care of the group rules. But on a simpler level, also "set" requires a telescope.

The telescopic mappings come in if we have two telescopes, such as "set" and "group," and if we wish to talk about mappings that attach a group to every set. In one sense the idea of telescopic mappings can be considered as a very special case of what happens in category theory in mathematics, but in another sense it is more general since the expressions in type theory can denote things quite different from the objects in mathematics.

Nevertheless it has to be said that Zucker never felt that in his setup of classical analysis he needed explicit use of notation for telescopic mappings (the material of this paper was available in 1973). He always found himself in relatively simple situations that could be handled *ad hoc*.

1.8. In this paper there is no attempt to present a formal theory of telescopes. We just treat telescopes on the level of metalanguage (and that is why the paper can be relatively informal). For a formal treatment of a kind of generalized lambda calculus that deals with strings of abstractors and applicators ("segments") as if those segments were just objects, with segment variables and quantification over such variables, we refer to (Balsters, 1987). That calculus can provide a full lambda calculus for the telescopic mappings of this paper. But it is by no means easy.

2. Notation and Rules for Typed Lambda Calculus

2.1. The kind of calculus to be considered is what one might call *minimal* lambda-typed lambda calculus, having the minimal set of rules for making it work. It is essentially the one of (de Bruijn, 1971; Nederpelt, 1973). The calculus of (de Bruijn, 1987) is roughly the same, the difference being that a slight change in the rules for application made the latter system useful for feasible implementation of languages that handle abbreviations and definitions. Moreover we mention that (de Bruijn, 1987) has a novelty on the metalevel: it describes the expressions of the calculus in the form of trees with reference arrows, thus avoiding the necessity to bother about the names of variables as well as avoiding the depth references of (de Bruijn, 1972). But here we shall take the usual line of slightly informal use of names of variables.

2.2. The set Λ of typed lambda expressions is given recursively by

(i) $\tau \in \Lambda$

(ii) If x is a variable, then $x \in \Lambda$.

(iii) If x is a variable, and if $T \in \Lambda$, $B \in \Lambda$, then $[x : T]B \in \Lambda$.

(iv) If $A \in \Lambda$ and $B \in \Lambda$ then $\langle A \rangle B \in \Lambda$.

2.3. We use the term *abstractor* for a fragment $[x : A]$, and the word *applicator* for a fragment $\langle A \rangle$.

The notation is the one of Automath. Writing $[x : T]$ instead of a subscript notation like $\lambda_{x:A}$ is merely a matter of typographical convenience since the A itself might contain things with subscripts too. The other thing that deviates from standard lambda calculus tradition is that that applicators are written on the left instead of on the right. This is reasonable because of the prominent role of β-reduction: pairs $\langle A \rangle [x : T]$ which are ready for β-reduction are kept together in the notation.

The set Λ_c is defined as the set of all *closed* expressions in Λ, i.e., expressions without free variables.

2.4. The crucial notions are *validity* and *typing*. The set Λ_v of all valid expressions is a certain subset of Λ_c that can be defined either by a set of generic rules or by an algorithm for checking validity. We do not display such rules here, but refer to (Nederpelt, 1973; van Daalen, 1980; de Bruijn, 1987).

There are the extra notions of *degree*, *typing*, *β-reduction*, and *definitional equivalence*.

An expression is said to have *degree* 1 if its rightmost symbol is τ. The simplest thing of degree 1 is τ itself.

If $P \in \Lambda_c$ and if the rightmost symbol of P is not τ, then that symbol is a variable x that has been introduced in an abstractor $[x : T]$ inside P. We say that T is the *type* of x. We now define $\mathrm{typ}(P)$ as the expression we get by replacing that rightmost x by its type.

If $\mathrm{typ}(P)$ has degree 1, we say that P has degree 2, if $\mathrm{typ}(P)$ has degree 2, we say that P has degree 3, etc. It is not hard to show that every $P \in \Lambda_c$ has a finite degree.

2.5. The usual notion of *β-reduction* can be slightly extended by allowing *local* β-reduction (de Bruijn, 1987). *Definitional equivalence* (for which we use the symbol $\equiv$) is the usual reflexive, transitive, and symmetric closure of the (local)β-reduction.

We get a slightly stronger form of definitional equivalence if we allow both β- and η-reductions instead of just β-reductions. From Section 8 onwards it will be assumed that equivalence is this $\beta\eta$-equivalence.

We can now say that $P : Q$ just means $\mathrm{typ}(P) \equiv Q$.

2.6. The set Λ_c is nicely closed with respect to the operations of typing and reduction. We mention some of the properties here (P, Q, F, G, A, T stand for elements of Λ_c):

(i) If $P \in \Lambda_v$ and $\mathrm{degree}(P) > 0$ then $\mathrm{typ}(P) \in \Lambda_v$.

(ii) If $P \in \Lambda_v$, and if P reduces to Q in a local β-reduction step, then also $Q \in \Lambda_v$; if moreover $\mathrm{degree}(P) > 1$ then $\mathrm{typ}(P) \equiv \mathrm{typ}(Q)$.

(iii) If C is a sequence of abstractors and applicators, and if $CT \in \Lambda_v$ then $C[x : T]x \in \Lambda_v$ and $C[x : T]\tau \in \Lambda_v$.

(iv) If C is a sequence of abstractors and applicators, if $CF \in \Lambda_v$, $CA \in \Lambda_v$, and if $\mathrm{typ}^j(CF) \equiv C[x : T]G$ for some $j \geqslant 0$, with $\mathrm{typ}(CA) \equiv CT$, then $C\langle A \rangle F \in \Lambda_v$.

2.7. The syntax of Λ can easily be enriched by adding instantiation expressions of the type $f(A_1, ..., A_n)$ as used in Automath. Although it would not cause any difficulty, we shall not enter into this in the present paper.

In this connection it can be mentioned that the λ's and Π's written in front of abstractors in several systems of typed lambda calculus can be seen as unary operators acting on expressions starting with an abstractor, and can be described syntactically by means of instantiations with a single subexpression.

2.8. There is also a notion of *local* typing. If C is a sequence of abstractors and applicators, and if $CA \in \Lambda_v$, $CT \in \Lambda_v$, $\mathrm{typ}(CA) \equiv CT$ we may write $C \vdash A : T$. If in particular C is a string of abstractors only this is usually expressed by saying that A has the type T *in the context* C.

3. Telescopes and Vectors Fitting into Them

3.1. A telescope is an abstractor string

$$[x_1 : A_1][x_2 : A_2(x_1)] \cdots [x_k : A_k(x_1, ..., x_{k-1})].$$

The number k is called its length. The $A_i(x_1, ..., x_{i-1})$ stands for an expression in Λ that we allow to contain, possibly on top of a set of free variables, the variables $x_1, ..., x_{i-1}$. Note that the symbols A_i have no separate meaning, it is only the combination $A_i(x_1, ..., x_{i-1})$ that makes sense.

We use column vector notation

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_k \end{pmatrix}, \qquad \mathbf{A}(\mathbf{x}) = \begin{pmatrix} A_1(x_1, ..., x_k) \\ \vdots \\ A_k(x_1, ..., x_k) \end{pmatrix}.$$

On the right we have written the column vector $\mathbf{x}$ as a row $x_1, ..., x_k$, just for typographical reasons. And for simplicity we have extended the strings of variables to length k for every A_i, although we know that A_i does not contain $x_i, x_{i+1}, ...$

If $v_1, ..., v_k$ are expressions in Λ then $\mathbf{v}$ will be called a Λ-vector.

The above telescope will be abbreviated to

$$[\mathbf{x} : \mathbf{A}(\mathbf{x})].$$

3.2. We say that the Λ-vector $\mathbf{v}$ *fits into* the telescope, notation

$$\mathbf{v} \in\in [\mathbf{x} : \mathbf{A}(\mathbf{x})],$$

if the following typings are typings of $\Lambda_{\mathbf{v}}$:

$$v_1 : A_1$$
$$v_2 : A_2(v_1)$$
$$\vdots$$
$$v_k : A_k(v_1, ..., v_{k-1}).$$

It has to be pointed out that what is meant here is not the instantiation referrred to in Section 2.7. $A_k(v_1, ..., v_{k-1})$ stands for the expression we get if we start from the expression that was denoted by $A_k(x_1, ..., x_{k-1})$, and replace all occurrences of x_1 by the expression v_1, etc.

We might have chosen any other symbol instead of $\in\in$. The only reason for $\in\in$ is that the fitting of a vector into a telescope of length 2 sometimes implements the belonging of an element to a set.

3.3. For the fitting of $\mathbf{v}$ into the telescope $[\mathbf{x} : \mathbf{A}(\mathbf{x})]$ we shall also use the notation

$$\mathbf{v} : \mathbf{A}(\mathbf{v}).$$

Note that on the right of $\in\in$ we have a telescope, on the right of the colon we have a vector. If

$$Q = [\mathbf{x} : \mathbf{A}(\mathbf{x})]$$

then $\mathbf{v} \in\in Q$ and $\mathbf{v} : \mathbf{A}(\mathbf{v})$ are synonymous. Accordingly, we might even write $[\mathbf{x} \in\in Q]$ instead of Q itself. If $\mathbf{v}$ has length 1, the notation $\mathbf{v} : \mathbf{A}(\mathbf{v})$ describes just ordinary typing.

3.4. If the vector $\mathbf{v}$ fits into the telescope Q, and if the length of $\mathbf{v}$ is >1, then we do not have the right to speak of Q as being *the* telescope of $\mathbf{v}$. If the length is 1, the simple relation between type and telescope guarantees that Q is uniquely determined by $\mathbf{v}$ in the sense that if $\mathbf{v}$ fits both into Q and into R, then Q and R are definitionally equal. If the length is >1, this is not longer the case. J. Zucker gave the following simple example. If

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \in\in [x_1 : A_1][x_2 : A_2(x_1)]$$

then we also have

$$\mathbf{v} \in\in [x_1 : A_1][x_2 : A_2(v_1)],$$

and these two telescopes are not difinitionally equal.

4. Further Notation

4.1. If $\mathbf{v}$ and $\mathbf{w}$ are vectors, possibly of different length,

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_k \end{pmatrix}, \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_h \end{pmatrix},$$

then $\langle \mathbf{v} \rangle \mathbf{w}$ denotes the vector

$$\begin{pmatrix} \langle v_k \rangle \cdots \langle v_2 \rangle \langle v_1 \rangle w_1 \\ \vdots \\ \langle v_k \rangle \cdots \langle v_2 \rangle \langle v_1 \rangle w_h \end{pmatrix}$$

and $[\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{w}$ denotes the vector

$$\begin{pmatrix} [\mathbf{x} : \mathbf{A}(\mathbf{x})] w_1 \\ \vdots \\ [\mathbf{x} : \mathbf{A}(\mathbf{x})] w_h \end{pmatrix}.$$

4.2. Concatenation of the vectors $\mathbf{v}$ and $\mathbf{w}$ is denoted by $\mathbf{v} \circ \mathbf{w}$:

$$\mathbf{v} \circ \mathbf{w} = \begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} v_1 \\ \vdots \\ v_k \\ w_1 \\ \vdots \\ w_h \end{pmatrix}$$

4.3. We can also concatenate two telescopes. The second one may depend on variables introduced in the first one. If

$$Q = [\mathbf{x} : \mathbf{A}(\mathbf{x})],$$

$$R(\mathbf{x}) = [\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})],$$

then the concatenation can be written as

$$[\mathbf{x} : \mathbf{A}(\mathbf{x})][\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})].$$

If we introduce $\mathbf{A}^*(\mathbf{x} \circ \mathbf{y}) = \mathbf{A}(\mathbf{x})$, $\mathbf{B}^*(\mathbf{x} \circ \mathbf{y}) = \mathbf{B}(\mathbf{x}, \mathbf{y})$ then the concatenation can be written as a single telescope

$$[\mathbf{x} \circ \mathbf{y} : \mathbf{A}^*(\mathbf{x} \circ \mathbf{y}) \circ \mathbf{B}^*(\mathbf{x} \circ \mathbf{y})].$$

4.4. Quite often we have to deal with the case that a concatenated vector $\mathbf{z} \circ \mathbf{v}(\mathbf{z})$ (the first entries are variables, the last ones are expressions containing these variables) fits into the concatenated telescope $[\mathbf{x} : \mathbf{A}(\mathbf{x})][\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})]$. In those cases we say (in accordance with Section 2.8) that in the context $[\mathbf{x} : \mathbf{A}(\mathbf{x})]$ the vector $\mathbf{v}(\mathbf{x})$ fits into $[\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})]$.

5. Telescopic Mappings

5.1. The semantics of a *telescopic mapping* is a function that attaches a vector $\mathbf{v}$ to every vector that fits into a telescope Q (to be called the it domain telescope). If $Q = [\mathbf{x} : \mathbf{A}(\mathbf{x})]$, the syntax is

$$[\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{v}(\mathbf{x}),$$

which will also be written as $\lambda_{\mathbf{x} \in\in Q} \mathbf{v}(\mathbf{x})$. Note that there is some danger of confusion in the notation $[\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{v}(\mathbf{x})$. It does not reveal what the domain telescope is, since $\mathbf{v}(\mathbf{x})$ itself may start with abstractors.

5.2. We shall be concerned with mappings where the values $\mathbf{v}(\mathbf{x})$ fit into a second telescope. In general, the second one can depend on the variables of the first:

$$Q = [\mathbf{x} : \mathbf{A}(\mathbf{x})]$$

$$R(\mathbf{x}) = [\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})].$$

So if we say that the values of the mapping $[\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{v}(\mathbf{x})$ fit into the second telescope, we mean that in the context $[\mathbf{x} : \mathbf{A}(\mathbf{x})]$ we have $\mathbf{v}(\mathbf{x}) \in\in \mathbf{R}(\mathbf{x})$, which means

$$\mathbf{x} \circ \mathbf{v}(\mathbf{x}) \in\in [\mathbf{x} : \mathbf{A}(\mathbf{x})][\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})].$$

We shall build a new telescope into which all these mappings $[\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{v}(\mathbf{x})$, and nothing but these mappings, fit. We denote it by $\mu_{\mathbf{x}\in\in Q}R(\mathbf{x})$. We shall refer to it as a *functional telescope*. It is defined as

$$\mu_{\mathbf{x}\in\in Q}R(\mathbf{x}) = [\mathbf{s} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{B}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{s})].$$

In Section 8 it will be established to have the required properties.

If Q has length k, R length m, then we have the following lengths:

Q	$R(\mathbf{x})$	$\mathbf{s}$	$\mathbf{x}$	$\mathbf{A}(\mathbf{x})$	$\mathbf{B}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{s})$	$\mu_{\mathbf{x}\in\in Q}R(\mathbf{x})$
k	m	m	k	k	m	m

6. An Example

6.1. Let us describe mappings from the interior of the unit circle in the complex plane into the set of all real numbers y with $0 \leqslant y < 1$.

The interior of the unit circle can be related to the telescope

$$Q = [z : \text{complex}][u : P(z)]$$

(if z is a complex number then $P(z)$ represents the class of all proofs for the statement that the absolute value of z is less than 1). The range set is related to the telescope

$$R = [y : \text{real}][v : W(y)]$$

(if y is a real number, then $W(y)$ is the class of all proofs for the statement that $0 \leqslant y < 1$). The functional telescope becomes (note that $z \circ u$ is a column vector of length 2, with entries z and u)

$$\mu_{z\circ u\in\in Q} = [p : [z : \text{complex}][u : P(z)]\,\text{real}]$$

$$[q : [z : \text{complex}][u : P(z)]\, W(\langle u \rangle \langle z \rangle p)].$$

6.2. Let the vector $\mathbf{g} = f \circ w$ fit into this functional telescope. So

$$f : [z : \text{complex}][u : P(z)]\ \text{real},$$

$$w : [z : \text{complex}][u : P(z)]\ W(\langle u \rangle \langle z \rangle f).$$

Furthermore, let $a \circ m$ fit into the domain telescope Q, whence a: complex, $m : P(a)$. Now

$$\langle a \circ m \rangle \mathbf{g} = \begin{pmatrix} \langle m \rangle \langle a \rangle f \\ \langle m \rangle \langle a \rangle w \end{pmatrix}, \qquad \text{typed by} \quad \begin{pmatrix} \text{real} \\ W(\langle m \rangle \langle a \rangle f) \end{pmatrix}.$$

If we put $\langle m \rangle \langle a \rangle f = b$, $\langle m \rangle \langle a \rangle w = r$, we have $\langle a \circ m \rangle f = b \circ r$, and we infer that b : real and $r : Q(b)$, i.e., r is a proof for the statement that b satisfies $0 \leqslant b < 1$. In other words, $\langle a \circ m \rangle f$ fits into R.

We can now check that

$$[z : \text{complex}][u : P(z)]\langle z \circ u \rangle \mathbf{g}$$

is a telescopic mapping (defined on Q) whose values fit into R. It reduces to $\mathbf{g}$ by η-reduction (see Section 7).

7. Beta and Eta Reduction

7.1. If $\mathbf{v} \in\in [\mathbf{x} : \mathbf{A}(\mathbf{x})]$ then

$$\langle \mathbf{v} \rangle [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{w}(\mathbf{x}) >_\beta \mathbf{w}(\mathbf{v}).$$

The number of β-reduction steps is equal to the length of $\mathbf{v}$.

7.2. If the vector $\mathbf{f}$ does not contain the variable $\mathbf{x}$ then

$$[\mathbf{x} : \mathbf{A}(\mathbf{x})]\langle \mathbf{x} \rangle \mathbf{f} >_\eta \mathbf{f}.$$

The number of η-reduction steps is equal to the length of $\mathbf{x}$.

In the next sections we shall take definitional equivalence in the β–η sense (cf. Section 2.5).

8. Derived Rules for Telescopic Mappings

8.1. Throughout this section, Q and $R(\mathbf{x})$ will be as in Section 5.2:

$$Q = [\mathbf{x} : \mathbf{A}(\mathbf{x})], \qquad R(\mathbf{x}) = [\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})].$$

We shall formulate the derived rules I, II, III. The latter two have exactly the form of introduction and elimination rules for ordinary lambda calculus.

In order to keep things simples, these derived rules are formulated here for the empty context, but they hold similarly in an arbitrary context (cf. Section 2.8).

8.2. As a warning we first mention that $\mathbf{f}(\mathbf{x})$ is not the value of a function $\mathbf{f}$ at a point $\mathbf{x}$. Like $\mathbf{A}(\mathbf{x}), \ldots$ in Section 3, $\mathbf{f}(\mathbf{x})$ stands for a vector of expressions containing the variables $x_1, \ldots$. If in $\mathbf{f}(\mathbf{x})$ we replace all x_i by corresponding v_i's, the result will be denoted by $\mathbf{f}(\mathbf{v})$. It will be a consequence of Rule I that $\mathbf{f}(\mathbf{v})$ can be interpreted as a function value, but not as a function value of $\mathbf{f}$. The function it can be interpreted as a value of, is $\mathbf{g}$, where $\mathbf{g} = [\mathbf{x} \in\in \mathbf{Q}]\mathbf{f}(\mathbf{x})$.

8.3. RULE I. *If* $\mathbf{f} \in\in \mu_{\mathbf{x}\in\in Q} R(\mathbf{x})$ *then (by* η*-reduction)*

$$\lambda_{\mathbf{x}\in\in Q}\langle \mathbf{x}\rangle \mathbf{f} = \mathbf{f}.$$

If moreover $\mathbf{v} \in\in Q$ *then we have as a consequence*

$$\langle \mathbf{v}\rangle \lambda_{\mathbf{x}\in\in Q}\langle \mathbf{x}\rangle \mathbf{f} = \langle \mathbf{v}\rangle \mathbf{f},$$

but here β*-reduction will do, we do not need* η.

Proof. Trivial.

8.4. RULE II (introduction rule).

$$\frac{\mathbf{f}(\mathbf{x}) \in\in R(\mathbf{x}) \text{ in the context } [\mathbf{x} \in\in Q]}{\lambda_{\mathbf{x}\in\in Q}\mathbf{f}(\mathbf{x}) \in\in \mu_{\mathbf{x}\in\in Q} R(\mathbf{x})}$$

Proof. In the context $[\mathbf{x} : \mathbf{A}(\mathbf{x})]$ we have $\mathbf{f}(\mathbf{x}) : \mathbf{B}(\mathbf{x}, \mathbf{f}(\mathbf{x}))$. If $\mathbf{g} = \lambda_{\mathbf{x}\in\in Q}\mathbf{f}(\mathbf{x})$ then $\langle \mathbf{x}\rangle \mathbf{g}$ is definitionally equal to $\mathbf{f}(\mathbf{x})$, so $\mathbf{f}(\mathbf{x}) : \mathbf{B}(\mathbf{x}, \langle \mathbf{x}\rangle \mathbf{g})$ in the context $[\mathbf{x} : \mathbf{A}(\mathbf{x})]$. Therefore

$$[\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{f}(\mathbf{x}) : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{B}(\mathbf{x}, \langle \mathbf{x}\rangle \mathbf{g}),$$

and so

$$\mathbf{g} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{B}(\mathbf{x}, \langle \mathbf{x}\rangle \mathbf{g}).$$

As $\mu_{\mathbf{x}\in\in Q} R(\mathbf{x}) = [\mathbf{s} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{B}(\mathbf{x}, \langle \mathbf{x}\rangle \mathbf{s})]$ we now conclude $\mathbf{g} \in\in \mu_{\mathbf{x}\in\in Q} R(\mathbf{x})$.

8.5. RULE III (elimination rule).

$$\frac{\mathbf{g} \in\in \mu_{\mathbf{x}\in\in Q} R(\mathbf{x}) \qquad \mathbf{v} \in\in Q}{\langle \mathbf{v} \rangle \mathbf{g} \in\in R(\mathbf{v})}$$

Proof. We have

$$\mathbf{g} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{B}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{g}),$$

$$\mathbf{v} : \mathbf{A}(\mathbf{v}),$$

and therefore $\langle \mathbf{v} \rangle \mathbf{g} : \mathbf{B}(\mathbf{v}, \langle \mathbf{v} \rangle \mathbf{g})$.

Now note that $R(\mathbf{v}) = [\mathbf{y} : \mathbf{B}(\mathbf{v}, \mathbf{y})]$, so $\langle \mathbf{v} \rangle \mathbf{g} : \mathbf{B}(\mathbf{v}, \langle \mathbf{v} \rangle \mathbf{g})$ can be interpreted as $\langle \mathbf{v} \rangle \mathbf{g} \in\in R(\mathbf{v})$.

9. COMPOSITION OF FUNCTIONS

9.1. We consider telescopes Q, R, S. For simplicity we shall take them as independent, i.e., we do not have one of them depending on the variables of another one, like the $R(\mathbf{x})$ of Sections 4.3 and 5.2.

THEOREM. *Let Q, R, S be telescopes, and let* $\mathbf{f}$ *and* $\mathbf{g}$ *be vectors with*

$$\mathbf{f} \in\in \mu_{\mathbf{x}\in\in Q} R, \qquad \mathbf{g} \in\in \mu_{\mathbf{y}\in\in R} S.$$

Then we have

$$\lambda_{\mathbf{x}\in\in Q} \langle \langle \mathbf{x} \rangle \mathbf{f} \rangle \mathbf{g} \in\in \mu_{\mathbf{x}\in\in Q} S.$$

Proof. In the context $[\mathbf{x} \in\in Q]$ (cf. Section 4.4) we have $\mathbf{x} \in\in Q$, and therefore by Rule III $\langle \mathbf{x} \rangle \mathbf{f} \in\in R$. Again by Rule III $\langle \langle \mathbf{x} \rangle \mathbf{f} \rangle \mathbf{g} \in\in S$. Finally we apply Rule II.

10. MAPPINGS INTO A PRODUCT

10.1. Under this heading we generalize the idea of mappings of a set A into the cartesian product of two sets B and C. The set $A \to (B \times C)$ of all mappings of A into $B \times C$ can be seen as the cartesian product $(A \to B) \times (A \to C)$. We shall generalize this to telescopes.

THEOREM. *If*

$$Q = [\mathbf{x} : \mathbf{A}(\mathbf{x})]$$
$$R_1(\mathbf{x}) = [\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})]$$
$$R_2(\mathbf{x}, \mathbf{y}) = [\mathbf{z} : \mathbf{C}(\mathbf{x}, \mathbf{y}, \mathbf{z})],$$

then we have

$$\mu_{\mathbf{x}\in\in Q}([\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})][\mathbf{z} : \mathbf{C}(\mathbf{x}, \mathbf{y}, \mathbf{z})]$$
$$= [\mathbf{f} \in\in \mu_{\mathbf{x}\in\in Q} R_1(\mathbf{x})[\mathbf{g} \in\in \mu_{\mathbf{x}\in\in Q} R_2(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{f})].$$

Proof. Introduce $\mathbf{B}^*$ and $\mathbf{C}^*$ by

$$\mathbf{B}^*(\mathbf{x}, \mathbf{y} \circ \mathbf{z}) = \mathbf{B}(\mathbf{x}, \mathbf{y}), \qquad \mathbf{C}^*(\mathbf{x}, \mathbf{y} \circ \mathbf{z}) = \mathbf{C}(\mathbf{x}, \mathbf{y}, \mathbf{z}).$$

Then the concatenation $[\mathbf{y} : \mathbf{B}(\mathbf{x}, \mathbf{y})][\mathbf{z} : \mathbf{C}(\mathbf{x}, \mathbf{y}, \mathbf{z})]$ can be written as a single telescope

$$[\mathbf{y} \circ \mathbf{z} : \mathbf{H}(\mathbf{x}, \mathbf{y} \circ \mathbf{z})], \qquad \text{where} \quad \mathbf{H} = \mathbf{B}^* \circ \mathbf{C}^*.$$

Now

$$\mu_{\mathbf{x}\in\in Q}[\mathbf{y} \circ \mathbf{z} : \mathbf{H}(\mathbf{x}, \mathbf{y} \circ \mathbf{z})] = [\mathbf{s} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{H}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{s})].$$

Writing $\mathbf{s} = \mathbf{f} \circ \mathbf{g}$ (with appropriate lengths of $\mathbf{f}$ and $\mathbf{g}$) we get

$$\mathbf{H}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{s}) = \mathbf{H}(\mathbf{x}, (\langle \mathbf{x} \rangle \mathbf{f}) \circ (\langle \mathbf{x} \rangle \mathbf{g}))$$
$$= \mathbf{B}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{f}) \circ \mathbf{C}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{f}, \langle \mathbf{x} \rangle \mathbf{g}).$$

Therefore

$$[\mathbf{s} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{H}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{s})]$$
$$= [\mathbf{f} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{B}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{f})][\mathbf{g} : [\mathbf{x} : \mathbf{A}(\mathbf{x})]\mathbf{C}(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{f}, \langle \mathbf{x} \rangle \mathbf{g})]$$
$$= [\mathbf{f} \in\in \mu_{\mathbf{x}\in\in Q} R_1(\mathbf{x}))][\mathbf{g} \in\in R_2(\mathbf{x}, \langle \mathbf{x} \rangle \mathbf{f})].$$

11. MAPPINGS WHERE THE DOMAIN IS A PRODUCT

11.1. Under this heading we generalize the idea of mappings of a cartesian product of two sets A and B into a set C. The set of all those mappings, $(A \times B) \to C$, can be interpreted as the set $A \to (B \to C)$ of all mappings of A into the set of all mappings of B into C. We shall generalize this to telescopes.

THEOREM. *Considering the telescope*

$$[\mathbf{y} : \mathbf{B}(\mathbf{y})][\mathbf{z} : \mathbf{C}(\mathbf{y}, \mathbf{z})][\mathbf{w} : \mathbf{D}(\mathbf{y}, \mathbf{z}, \mathbf{w})]$$

we put $R_1 = [\mathbf{y} : \mathbf{B}(\mathbf{y})]$, $R_2(\mathbf{y}) = [\mathbf{z} : \mathbf{C}(\mathbf{y}, \mathbf{z})]$, $T(\mathbf{y}, \mathbf{z}) = [\mathbf{w} : \mathbf{D}(\mathbf{y}, \mathbf{z}, \mathbf{w})]$, $S = R_1 \circ R_2(\mathbf{y})$. *Then we have*

$$\mu_{\mathbf{y} \circ \mathbf{z} \in S} T(\mathbf{y}, \mathbf{z}) = \mu_{\mathbf{y} \in R}(\mu_{\mathbf{z} \in R_2(\mathbf{y})}(T(\mathbf{y}, \mathbf{z})).$$

Proof.

$$\mu_{\mathbf{y} \circ \mathbf{z} \in S} T(\mathbf{y}, \mathbf{z}) = [\mathbf{s} : [\mathbf{y} : \mathbf{B}(\mathbf{y})][\mathbf{z} : \mathbf{C}(\mathbf{y}, \mathbf{z})]\mathbf{D}(\mathbf{y}, \mathbf{z}, \langle \mathbf{y} \circ \mathbf{z} \rangle \mathbf{s})].$$

On the other hand

$$\mu_{\mathbf{z} \in R_2(\mathbf{y})} T(\mathbf{y}, \mathbf{z}) = [\mathbf{t} : [\mathbf{z} : \mathbf{C}(\mathbf{y}, \mathbf{z})]\mathbf{D}(\mathbf{y}, \mathbf{z}, \langle \mathbf{z} \rangle \mathbf{t})],$$

so

$$\mu_{\mathbf{y} \in R_1}(\mu_{\mathbf{z} \in R_2(\mathbf{y})} T(\mathbf{y}, \mathbf{z})) = [\mathbf{s} : [\mathbf{y} : \mathbf{B}(\mathbf{y})][\mathbf{z} : \mathbf{C}(\mathbf{y}, \mathbf{z})]\mathbf{D}(\mathbf{y}, \mathbf{z}, \langle \mathbf{y} \rangle \langle \mathbf{z} \rangle \mathbf{s})].$$

It now suffices to remark that $\langle \mathbf{y} \circ \mathbf{z} \rangle \mathbf{s} = \langle \mathbf{z} \rangle \langle \mathbf{y} \rangle \mathbf{s}$.

ACKNOWLEDGMENTS

The author is indebted to L. S. van Benthem Jutting and R. Nederpelt for their remarks.

RECEIVED June 17, 1988; FINAL MANUSCRIPT RECEIVED November 9, 1989

REFERENCES

BALSTERS, H. (1987), Lambda calculus extended with segments, *in* "Mathematical Logic and Theoretical Computer Science" (D. W. Kueker, E. G. K. Lopez-Escobar, and C. H. Smith, Eds.), pp. 15–28, Lecture Notes in Pure and Applied Mathematics, Vol. 106, Dekker, New York/Basel.

BARENDREGT, H. (to appear), Introduction to generalised type systems, "Proceedings of the Third Italian Converence on Theoretical Computer Science (Mantova, 2–4 November 1989)" (U. Moscati, Ed.), World Scientific, Singapore.

BRUIJN, N. G. DE (1970), The mathematical language AUTOMATH, its usage, and some of its extensions, *in* "Symposium on Automatic Demonstration (Versailles, December 1968)," pp. 29–61, Lecture Notes in Mathematics, Vol. 125, Springer Verlag, Berlin/New York.

BRUIJN, N. G. DE (1971), "AUT-SL, a Single Line Version of AUTOMATH," Memorandum 71-17, Eindhoven University of Technology, Department of Mathematics.

BRUIJN, N. G. DE (1972), Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem, *Nederl. Akad. Wetensch. Indag. Math.* **34** 381–392.

BRUIJN, N. G. DE (1978), "AUT-QE without Type Inclusion," Memorandum 78–04, Eindhoven University of Technology, Department of Mathematics.

BRUIJN, N. G. DE (1980), A survey of the project Automath, *In* "To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism" (J. P. Seldin and J. R. Hindley, Eds.), pp. 579–606, Academic Press, New York/Basel.

BRUIJN, N. G. DE (1984), Formalization of constructivity in Automath, *In* "Papers Dedicated to J. J. Seidel," pp. 76–101, EUT Report 84-WSK-03, Department of Mathematics and Computing Science, Eindhoven University of Technology.

BRUIJN, N. G. DE (1987), Generalizing Automath by means of a lambda-typed lambda calculus, *in* "Mathematical Logic and Theoretical Computer Science," (D. W. Kueker, E. G. K. Lopez-Escobar, and C. H. Smith, Eds.), pp. 71–92, Lecture Notes in Pure and Applied Mathematics, Vol. 106, Dekker, New York.

BRUIJN, N. G. DE (1990), The use of justification systems for integrated semantics, *in* "Proceedings of Colog-88, Tallinn USSR, December 1988," pp. 9–24, Lecture Notes in Computer Science, Vol. 417, Springer-Verlag, Berlin/New York.

COQUAND, T., AND HUET, G. (1988), The calculus of constructions, *Inform. and Computat.* **76**, 95–120.

DAALEN, D. T. VAN (1980). "The language theory of AUTOMATH," Doctoral Thesis, Dept. of Mathematics, Eindhoven University of Technology.

MARTIN-LÖF, P. (1984) "Intuitionistic Type Theory," Bibliopolis, Napoli.

NEDERPELT, R. P. (1973) "Strong Normalization in a Typed Lambda Calculus with Lambda Structured Types," Doctoral Dissertation, Eindhoven University of Technology.

ZUCKER, J. (1975) Formalization of classical mathematics in AUTOMATH, in "Colloque International de Logique, Clermont-Ferrand 1975" (Guillaume, Ed.), pp. 135–145, Editions du Centre National de Recherche, Paris.