Automath : a language for mathematics

AUTOMATH, a language for mathematics.

by

N.G. de Bruijn

1. Introduction.

AUTOMATH is a language intended for expressing detailed mathematical thoughts. It is <u>not</u> a programming language, although it has several features in common with existing programming languages. It is defined by a grammar, and every text written according to its rules is claimed to correspond to correct mathematics. It can be used to express a large part (see 1.6) of mathematics, and admits many ways for laying the foundations. The rules are such that a computer can be instructed to check whether texts written in the language are correct. These texts are not restricted to proofs of single theorems; they can contain entire mathematical theories, including the rules of inference used in such theories.

AUTOMATH was developed in 1967-1968 at the Technological University, Eindhoven, The Netherlands. The author is indebted to Mr. L.S. van Benthem Jutting for very valuable help in trying out the language in several parts of mathematics, and both to him and to Mr. L.G.F.C. van Bree for their assistance with the programming (in ALGOL) of processors by means of which books written in AUTOMATH can be checked.

1.1   Texts written in AUTOMATH can be step - by - step translations of ordinary mathematics. In order to obtain this translation, the organization of a given piece of mathematics can be left intact, but the details have to be presented meticuously; after that, the coding into AUTOMATH is a matter of routine. One of the features of the language is that the coding does not require more effort if we proceed further into mathematics. This is achieved by means of an abbreviating system which is essentially taken from existing mathematical habits.

1.2   Properly speaking, the rules of AUTOMATH involve little more than the art of substitution. A text written in AUTOMATH consists of a sequence of lines.

In each line a new name is introduced, and (if it is not declared to indicate a primitive notion or a new variable) is expressed in terms of names introduced previously (this expression is called the "definition" of the name). The names can correspond to various things in ordinary mathematical presentation: they can correspond to mathematical objects (and can be considered to be the name of the object), to variables, to propositions, to assertions, axioms, assumptions, definitions, theorems, formula numbers, etc.

1.2.1   In every line there is also an indication as to the nature of the thing denoted by the name. This we shall call the "category". This category is usually a previously introduced name or an expression in terms of previously introduced names, but it can also be the symbol "type". In the latter case the name introduced in the line is the name of a category, and can be used later as category of new names.

1.3   The grammar of AUTOMATH contains facilities for expressing functional relation. If two categories are available, it admits to build a third category consisting of the mappings of the one category into the other. It also contains rules for the art of substitution concerning these mappings, and for expressing mappings by means of bound variables.

One thing seems to be unusual but nevertheless useful: the grammar contains the possibility to speak of a mapping that attaches, to every object x of a given category α, an object of a category $\beta(x)$, that is, a category depending on the object x.

1.4   The language can be used in different ways, especially since it is not tied to any particular logical system. Not even notions like "theorem", "proof", "proposition", "definition" occur in the grammar of the language; they can be introduced by the user of the language in the way he prefers.

1.4.1   Quoting an out - of - the - way example, we have the possibility to talk

about points in a plane, to fix a number of points as "given", and to talk

about other points as "constructible by ruler and compass, using the given

points to start from". To this end we can introduce, for any point P, a cate-

gory "CONSTR(P)", which we may visualize as the class of all constructions

for P, if any. Now if we have a construction for P, we can state that fact in

a single line. The new name introduced can be used later as a reference to the

fact that P is constructible, its definition in terms of previous names indi-

cates a particular contruction; and finally, the category is CONSTR(P).

1.4.2   In complete analogy to this, we have the possibility to talk about propo-

sitions instead of points, and proofs instead of constructions. If we call the

corresponding category TRUE(P), then any line having TRUE(P) as its category

states that we have a proof for P. That is to say, it asserts P if the name

of that line has a proper definition. If, on the other hand, the name indicates

a new variable, then the line expresses the assumption that P is true, and if

the name is a new primitive notion, then the line expresses the axiom that P

is true. We refer to 2.4 for an example.

1.5   For every line it is vital to know the context in which the line has its

meaning. This context is exposed by stating the name of the last introduced

variable relevant for that line. This name is called the "indicator" of the

line. The way how this is done, opens the possibility to write, in some chap-

ter, lines belonging to the context of a previous chapter, in spite of the

fact that the line contains material developed in the later chapter.

1.6   As to the question what part of mathematics can be written in AUTOMATH,

it should first be remarked that we do not possess a workable definition of

the word "mathematics". Quite often a mathematician jumps from his mathematic-

al language into a kind of metalanguage, obtains results there, and uses these

results in his original context. It seems to be very hard to create a single

language in which such things can be done without any restriction. Of course it is possible to have a language in which discussions about the language itself can be expressed, but that is not the difficulty. The problem is to extend a given text T_1 by means of a metalingual discussion T_2 (T_2 talks about T_1), and to put T_2 in the context of T_1, instead of creating a new context where both T_1 and T_2 can take place. For, if T_1 is placed in a new context, it is not the same text any more; anyway, it is not available at the places where the old T_1 was available.

In AUTOMATH it is not strictly impossible to mix language and metalanguage, but it seems that such possibilities have to be "frozen" somehow. It seems impossible to write one big book containing all mathematics such that we never regret the way how we began. If at a later stage we divise, and want to legitimate, new ways to inject metalingual results into the existing text, we may have to rewrite the book or even to redefine the language.

1.7   The author feels that very little is essentially new in AUTOMATH, and that it just expresses the way how mathematicians have always been writing and talking, at least as long as they were presenting things step-by-step. Mathematical inventive thinking, however, usually does not proceed in the same fashion, driven as it is by things like intuition, inspiration, insight, wishful thinking, tradition, taste, ambition.

1.8   In one respect AUTOMATH may seem to deviate from existing mathematical habits. To every name it attaches a single category. So if we say that "5 is an integer" and that "5 is a rational number" the words "integer", "rational number" cannot both refer to a category to which 5 belongs. We can escape by saying that "5 is an integer", that "5* is a rational number", and that there is an embedding of the integers into the rational numbers that sends 5 into 5*. This is safe but troublesome. An entirely different way is to translate "5 is

an integer" into a line expressing that we have a proof for the proposition

$5 \in Z$, where Z stands for the set of integers. If we take this point of view

we can translate both "5 is an integer" and "5 is a rational number" that way, for

the word "is" does no longer refer to inclusion of something in a given cate-

gory. The categories involved here are, for example, "element" (for "5"), "set"

(for "Z"), "proposition" (for "∈"), "TRUE" (for the proof of $5 \in Z$). Note that

the set-theoretical interpretation of things like "let n be an integer" requires

two lines instead of one, viz. the two lines describing the sentences "let n be

an element","let p be a proof for $n \in Z$".

1.8.1   Present-day mathematicians seem to prefer set-theoretical terminology,

in the conviction that everything is a set. That is, they claim that almost

everything in mathematics belongs to one and the same category, viz. the cate-

gory "set". In spite of the simplicity of this point of view it must be said

that it often gives a quite unnatural presentation. It is certainly very un-

natural to consider things like propositions, classes, constructions and proofs

as sets. AUTOMATH leaves its users free to introduce the categories they prefer.

1.8.2   As we remarked in the beginning of 1.8, categories are unique. Actually

there is an algorithm that derives, by repeated substitution, the category

of any expression occurring in the text. There seems to be little use in

trying to say in AUTOMATH that an expression Σ is <u>not</u> of category σ, since

whenever we talk about Σ there is not any doubt as to its category.

2.   Informal introduction into AUTOMATH.

2.1   Before we give a formal description we shall sketch some of the aspects

of AUTOMATH in an informal way.

Assume that, in the text prior to the lines we are going to discuss

presently, the category "nat" (for natural number) was introduced. We now

want to introduce the notion of product, although we do not bother about the

properties of the product. We write

```
0  | a     :=        ——      nat

a  | | b    :=        ——      nat

b  | | prod  :=      PN       nat
```

The indicators are 0,a,b; the names a,b,prod, the categories are nat,nat,nat.

The indicator 0 says that nothing is assumed, no variables are valid at that

point. In the second line, the variable a is considered a known quantity, in

the third line both a and b are known. The structure is indicated by the ver-

tical bars, describing the validity interval for a and b, respectively. These

bars are drawn in order to make it easier to get a quick survey of the text.

They do <u>not</u> belong to the language.

The name prod is not defined in this text; it is introduced as a <u>primitive</u>

<u>notion</u> (PN). It would not make a difference for our present discussion if it

were defined somehow.

We now want to define the square of a number. We write

```
0  | c       :=        ——        nat

c  | square   :=   prod(c,c)       nat
```

Since we went back to indicator 0, the word prod has lost its meaning. But if

we have two valid expressions for natural numbers, u and v, say, then prod(u,v) is a legitimate expression, and its category is nat. Hence prod(c,c) is legitimate if c is legitimate. Note that square stands for the square of c as long as c is "alive" as a variable. We proceed by writing

c	cube	:=	prod(c,square)	nat
c	d	:=	square(square)	nat
0	e	:=	———	nat
e	f	:=	d(cube(square(e)))	nat

In the definition of cube we do not have to say that square is the square of c. In the definition of f, however, it would have been unacceptable to write cube(square), since square itself has no meaning at that point. Note that d indicates the square of the square of c, and that f indicates the 24-th power of e.

2.2   Let us be a little bit more precise about our abbreviation habits. Suppose that $p_1, \ldots, p_n$ have been introduced consecutively as variables, that is, p_2 has p_1 as indicator,..., p_n has p_{n-1} as indicator. Moreover let q be introduced by some expression Σ (which might be PN) at indicator p_n. If at a later stage we use q, we can only do this by providing n expressions, $\Sigma_1, \ldots, \Sigma_n$, say, for $p_1, \ldots, p_n$, which have to be of the right category. But quite often it happens that a number of the p_i are still valid variables, and that some of the first Σ's are just the corresponding p's, like

$$q(p_1, \ldots, p_k \ , \ \Sigma_{k+1}, \ldots, \Sigma_n).$$

In that case we may omit a number of p's on the front. That is, we may write $q(p_2, \ldots, p_k \ , \ \Sigma_{k+1}, \ldots, \Sigma_n)$ or $q(p_3, \ldots, p_k \ , \ \Sigma_{k+1}, \ldots, \Sigma_n), \ldots,$ or $q(\Sigma_{k+1}, \ldots, \Sigma_n)$. Accordingly, if $p_1, \ldots, p_n$ are all valid, we may abbreviate $q(p_1, \ldots, p_n)$ as $q(p_2, \ldots, p_n), \ldots, q(p_n)$, or even as q. There cannot be any confusion, for q needs n arguments, and if some are lacking, we supply extra p's, starting with $p_1, p_2, \ldots$ at the front.

2.3   Although our description is far from complete, and calls for a feeling for existing habits in mathematical notation, the reader may see from it how functional relationship can be handled. That is, as long as a functional relationship is explicitly exhibited. If we want to assume the existence of a functional relationship, or if we want to prove something about all mappings of a given type, then we have a metalingual problem: "assuming we have a piece of text looking like this, can we add a line looking like that?".

At this stage we decide to build extra facilities to describe functional relationship. We introduce the right to create bound variables, and we agree to write the mapping $c \to c^2$ as

$$[x, nat\,]\; square(x)$$

which indicates, in this order, the variable's name, the domain, the value. We prefer this notation over Church's lambda sybolism since we have to lay so much stress on the categories. The category of the above expression will be written as

$$[x, nat\,]\; nat.$$

The x here seems superfluous, but the notation is divised for situations where the range depends on the variable.

Let us call the mapping "squaring" and write

$$0 \quad\quad squaring \quad := [x, nat]\; square(x) \quad\quad\quad [x, nat]\; nat.$$

We can use the category $[x, nat]nat$ in order to introduce new mappings. For example, we can write things like

$$
\begin{array}{c|c}
0 & g \; := \; \text{———} \quad\quad\quad nat \\
\hline
g & |h \; := \; \text{———} \quad\quad\quad [x, nat]\; nat
\end{array}
$$

in order to express: let g be a natural number, and let h be a mapping of the

natural numbers into the natural numers. We now want to express the effect that h has on g, i.e. what is customarily written as h(g). For reasons to be explained just now, this is an ambiguous notation, and we agree to write {g}h instead.

So we may write

h	i	:=	{g}h	nat
h	j	:=	——	nat
j	k	:=	[x,nat] prod(square(x),prod(i,j))	[x,nat]nat
j	l	:=	{i}k	nat
h	m	:=	k(i)	[x,nat]nat

We elaborate this to this extent in order to display the difference between {i}k and k(i). The first one means the product of j and the third power of the image of g under h, the second one is the mapping that sends each x into the product of the square of x and the square of the image of g under the mapping h.

2.4   As an example we describe how the axiom of induction can be presented in AUTOMATH. For simplicity, it is detached from the other axioms for the natural numbers.

We open our book by saying that we shall speak about certain things called "propositions" or "booleans":

O   bool   :=   PN   type

Next we say that to any boolean there belongs a class of proofs for that boolean, possibly empty. If b is a boolean, then this class is called TRUE(b). It seems slightly mystical what TRUE(b) represents, but we can take a pragmatic point of view: our way of asserting a boolean b is saying that there is a something in its truth class TRUE(b). So we write

```
O  | b    := ———        bool
b  | TRUE := PN          type
```

Next we introduce the category of natural numbers, and the number 1 as a natural number. Both are primitive notions:

```
O    nat  := PN          type
O    1    := PN          nat
```

We need, for any natural number, the successor of that number. This can be done in two ways which are entirely equivalent. We can define SUCC as a primitive notion of category [x,nat]nat, but we may also write

```
O  | n    := ———         nat
n  | succ := PN          nat
```

We can denote the successor of m by succ(m); in terms of SUCC it would be {m}SUCC.

We want to talk about a predicate, that is a mapping of the natural numbers into the booleans. So we write "let P be a predicate", and assume that it is true for the natural number 1:

```
O  | P    := ———         [x,nat]bool
P  | |if  := ———         TRUE({1}P)
```

Note that the predicate attaches to 1 the boolean {1}P, and "if" denotes the assumption that this boolean is true. We next want to say that if P is true for m then it is true for its successor. We first abbreviate:

```
if | | |m    := ———                              nat
m  | | |IMP  := [u,TRUE({m}P)] TRUE({succ(m)}P)   type
```

This says that once the number m has been fixed, IMP is the category of mappings which attach, to any proof of $\{m\}P$, a proof of $\{succ(m)\}P$. Indicating something of category IMP amounts to the same thing as asserting the implication $(\{m\}P) \Rightarrow \Rightarrow (\{succ(m)\})P$.

Let us now assume that this implication is true for all m. That is, we assume to have a mapping of the natural numbers into the corresponding implication category. Going back to level "if" we write

| if | | | | assume | := | ———— | | $[x,nat]$ IMP(x) |
| assume | | | | induction | := | PN | | $[x,nat]$ TRUE$(\{x\}P)$ |

The last line contains the induction axiom. It says that assuming P is a predicate, assuming $\{1\}P$ is true, and assuming the truth of $\{m\}P \Rightarrow \{succ(m)\}P$ for all m, then $\{x\}P$ is true for all x.

Let us now consider an application of this axiom. Assume we have, somewhere further on in the book, a piece of text like the one below. It is not written in full; if we write ....., we mean that the text contains some legitimate expression which we do not wish to discuss:

0	h	:=	————	nat
h	Q	:=	.....	bool
h	when	:=	————	TRUE(Q)
when	then	:=	.....	TRUE$(Q(succ(h)))$
0	also	:=	.....	TRUE$(Q(1))$
0	r	:=	,....	nat

Under these circumstances we can prove that $Q(r)$ is true. We can write it in a single line (which might be simplified by devoting other lines to definition of abbreviations):

0   now   := $\{r\}$ induction$([x,nat]$ $Q(x)$,also,

$[x,nat]$ $[y,TRUE(Q(x))]$ then $(x,y))$ TRUE$(Q(r))$

The fact that the definition of "now" is so complicated does not disturb us seriously, since we do not expect to use it. It only matters that we do have something in $TRUE(Q(r))$. This amounts to saying that it is the result that counts, not the method. That is, when quoting a theorem we do not have to quote the proof. Actually this is what theorems are for: short statements have to be remembered, long arguments can be forgotten.

3.   Formal description of AUTOMATH.

3.1   In this chapter we shall describe four languages, to be called LONGPAL,
PAL, LONGAL and AUTOMATH. The discussion of the first three is intended to
be a preparation for the discussion of AUTOMATH. PAL (short for "primitive
Automath language") is a sublanguage of AUTOMATH, in the sense that every
book written in the first language can be read as if it were a book written
in the second one. Similarly, LONGPAL is a sublanguage of PAL, LONGAL is a
sublanguage of AUTOMATH, and LONGPAL is a sublanguage of LONGAL.

        PAL is an abbreviated form of LONGPAL. The latter has the simpler rules,
but has the disadvantage of very long expressions. Similarly, LONGAL has sim-
pler rules than AUTOMATH, but LONGAL is very inpractical.

        The description in sections 3.2 and 3.3 will apply to all four languages
simultaneously.

3.2   Notation concerning strings.

        The nomenclature and notation of this section do not appear in the books
written in our languages, but in the discussions about the rules of these lan-
guages.

        A string is any finite sequence of things, possibly empty. We can talk
about a string of letters, a string of words, etc. If the string is not empty,
and if it contains, in this order, the objects $a_1, \ldots, a_n$, then we denote it by

$$\langle a_1 \rangle + \ldots + \langle a_n \rangle ,$$

as long as we do not prefer to abbreviate it by a single greek letter. In par-
ticular, if the string consists of the single element a_1, then it can be denot-
ed by $\langle a_1 \rangle$.

        If α stands for the string $\langle a_1 \rangle + \ldots + \langle a_n \rangle$ and if β stands for the
string $\langle b_1 \rangle + \ldots + \langle b_m \rangle$, then $\alpha + \beta$ stands for the juxtaposition of α and
β , viz.

$$\langle a_1 \rangle + \dots + \langle a_n \rangle + \langle b_1 \rangle + \dots + \langle b_m \rangle .$$

If α stands for the string $\langle a_1 \rangle + \dots + \langle a_n \rangle$, then length (α) is defined to be n; moreover, if $1 \leqslant k \leqslant n$, we write $elt_k(\alpha)$ in order to indicate the element a_k. The empty string is denoted by $\emptyset$ and is said to have length zero.

We use the notation $front_k(\alpha)$ in order to indicate the initial segment of length k:

$$front_k(\langle a_1 \rangle + \dots + \langle a_n \rangle) = \langle a_1 \rangle + \dots + \langle a_k \rangle \quad (1 \leqslant k \leqslant n),$$
$$front_0(\langle a_1 \rangle + \dots + \langle a_n \rangle) = \emptyset .$$

We write $\alpha \subset \beta$ if α is an initial segment of β, i.e. if there is a number k with $0 \leqslant k \leqslant length(\beta)$ and $\alpha = front_k(\beta)$.

3.3   A <u>book</u> is a string of <u>lines</u>. A <u>line</u> consists of four parts: an <u>indicator</u>, an <u>identifier</u>, a <u>definition</u> and a <u>category</u>.

3.3.1   The symbols of which the parts of a line are composed, are

(i)   The seven separation marks, listed here:

,   (   )   {   }   [   ]

The last four of these do not occur in PAL or LONGPAL.

(ii)   Four other basic symbols, listed here:

—   0   PN   <u>type</u>

(iii)   Arbitrarily many other symbols to be called identifiers, mutually distinct, and distinct from the 11 symbols listed under (i) and (ii).

3.3.2   The <u>identifier</u> <u>part</u> of a line consists of a single identifier. It has to be different from the identifier part of any previous line. There would be no objection against systematic use of positive integers in such a way that the number n is the identifier part of the n-th line. However, in order to make books easier to read, and easier to compare with existing ways to express mathematics, one may prefer to choose more suggestive symbols like words, or words with numbers added to them. Note that an identifier is to be considered as a single symbol. It has already been stipulated that identifiers have to be distinct from the other basic symbols. In a printed text an identifier may be represented by a string of letters, digits or other signs, containing no separation marks.

3.3.3   The <u>definition</u> <u>part</u> of a line can be one of the folowing things:

(i)   The symbol —— (short for "variable").

(ii)   The symbol PN (short for "primitive notion").

(iii)   An <u>expression</u>. This is a certain string of symbols, consisting of separation marks and identifiers. We shall explain later how expressions should be built.

3.3.4   The <u>category</u> <u>part</u> of a line can be one of the following things:

(i)   The symbol <u>type</u>.

(ii)   An <u>expression</u> (see 3.3.3).

3.3.5   A <u>block</u> <u>heading</u> is a line whose definition part is ——. The identifier part of such a line is called a <u>block</u> <u>opener</u>.

3.3.6   The <u>indicator</u> <u>part</u> of a line is either the symbol 0 or an identifier.

In a book we always require that whenever the indicator part of a line is an identifier, it is the identifier of a block heading occurring earlier in the book.

Our discussions will be made easier by the introduction of the notion indicator string of a line in a book. We define it recursively. If the indicator is 0, then the indicator string is empty. If the indicator of line λ is α, if α is the identifier of line μ, and if σ is the indicator string of μ, then $\sigma + \langle \alpha \rangle$ is the indicator string of line λ. Thus the indicator strings are strings of blockopeners.

3.3.7   <u>Notation.</u> We shall use indic(λ) for the indicator of line λ, indstr(λ) for the indicator string, ident(λ) for the identifier of λ, def(λ) for the definition of λ, cat(λ) for the category of λ.

3.4   <u>Description of LONGPAL.</u>

3.4.1   <u>Parentheses expressions.</u> The expressions mentioned in 3.3.3 and 3.3.4 are, as far as PAL and LONGPAL are concerned, parentheses expressions, to be defined presently.

A parentheses expression is a non-empty string of symbols; the symbols consists of are either identifiers, or comma's, or opening parentheses "(", or closing parentheses ")". They have to be built in a certain way, which we describe by recursion:

(i)   If β is an identifier, then the string $\langle \beta \rangle$ is a parentheses expression.

(ii)   If β is an identifier, if n is a positive integer, and if $\varepsilon_1, \ldots, \varepsilon_n$ are parentheses expressions, then the string

$$\langle \beta \rangle + \langle (\rangle + \varepsilon_1 + \langle , \rangle + \varepsilon_2 + \langle , \rangle + \ldots + \langle , \rangle + \varepsilon_n + \langle) \rangle .$$

is again a parentheses expression.

3.4.1.1   In examples we usually omit the $\langle$'s, the $\rangle$'s, and the +'s.

3.4.1.2   Example: if a,b,c,d,e are identifiers, then

$$a(b(c,d), \ a(e), \ e(a(a,b)))$$

is a parentheses expression.

3.4.2   <u>Well-formed LONGPAL books.</u>

This notion is defined recursively:

(i)   The empty book is well-formed.

(ii)   Let Λ be a well-formed LONGPAL book, and let λ be a line. Then the book $\Lambda + \langle \lambda \rangle$ is called well-formed if the following conditions are satisfied simultaneoulsy:

1. The identifier of λ is different from the identifiers of the lines of Λ.

2. The idicator of λ is either 0 or one of the block openers of Λ.

3. The definition of λ is either — or PN or a parentheses expressio

4. The category of λ is either <u>type</u> or a parentheses expression.

A well-formed book is not necessarily correct. "Well-formed" is an intermediate notion that we need in order to formulate the further conditions for correctness.

3.4.3   <u>The set $S(\Lambda)$.</u>

Let Λ be a well-formed LONGPAL book. Then $S(\Lambda)$ is a set of strings of block openers of Λ, defined recursively as follows:

(i)   The empty string belongs to $S(\Lambda)$.

(ii)   If β is a block opener of Λ, the one-element string $\langle \beta \rangle$ belongs to $S(\Lambda)$.

(iii) If $\langle \beta_1 \rangle + \ldots + \langle \beta_n \rangle$ belongs to $S(\Lambda)$, and if β_{n+1} is the identifier of a block heading whose indicator is β_n, then $\langle \beta_1 \rangle + \ldots + \langle \beta_n \rangle + \langle \beta_{n+1} \rangle$ belongs to $S(\Lambda)$.

3.4.3.1   In other words, if the elements of $S(\Lambda)$ are not indicator strings of lines of Λ then they still have the form $\alpha + \langle \beta \rangle$, where α is the indicator of a line of Λ whose identifier is β.

It is easy to see that if $\Lambda + \langle \lambda \rangle$ is well-formed (see 3.4.2), then the indicator string of the last line of $\Lambda + \langle \lambda \rangle$ lies in $S(\Lambda)$.

3.4.4   A substitution operator.

Let Λ be a well-formed LONGPAL book, and let $\sigma \in S(\Lambda)$. Assume that σ is non-empty, and put length$(\sigma) = k$, $\sigma = \langle \beta_1 \rangle + \ldots + \langle \beta_k \rangle$ (so $\beta_1, \ldots, \beta_k$ are block openers of Λ). Let $\Sigma_1, \ldots, \Sigma_k$ denote parentheses expressions. We shall define an operator $\Omega_\sigma(\Sigma_1, \ldots, \Sigma_k)$ on a sub-class of the class of all parentheses expressions.

This sub-class is denoted by $C(\Lambda)$. It consists of all parentheses expressions that do not contain any block opener of Λ followed by an opening parentheses. For example, if a, b are block openers of Λ, and c, d are no block openers of Λ, then $c(a,d(b,d(a))$ belongs to $C(\Lambda)$, but $d(c,a(b))$ does not.

On $C(\Lambda)$ we define the operator recursively.

(i)   If ξ is one of the β's, $\xi = \beta_j$, say, then the effect of the operator on the expression $\langle \xi \rangle$ is described by

$$(\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k))\langle\xi\rangle = \Sigma_j.$$

(ii)   If ξ is any other identifier, then

$$(\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k))\langle\xi\rangle = \langle\xi\rangle.$$

(iii)   Moreover, if ξ is an identifier, if $\Gamma_1,\ldots,\Gamma_m$ are elements of $C(\Lambda)$, and if Θ is the expression $\langle\xi\rangle + \langle(\rangle + \Gamma_1 + \langle,\rangle + \ldots + \langle,\rangle + + \Gamma_m + \langle)\rangle$, then $(\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k))\Theta$ is the expression

$$\langle\xi\rangle + \langle(\rangle + (\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k))\Gamma_1 + \langle,\rangle + \ldots +$$

$$+ \langle,\rangle + (\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k))\Gamma_m + \langle)\rangle.$$

3.4.4.1   Another way to describe the effect of the operator on some $\Gamma \in C(\Lambda)$, is the following one: for each j $(1 \leqslant j \leqslant n)$, replace, in the expression Γ, each β_j by the corresponding symbol Σ_j. Next replace each Σ_j by the expression it denotes.

Note that $\Sigma_1,\ldots,\Sigma_k$ themselves may contain $\beta_1,\ldots,\beta_n$, and that the seemingly simpler formulation: "replace β_1 by Σ_1 everywhere, replace β_2 by Σ_2 everywhere,..." would be quite ambiguous.

We also define the effect of the operator on the symbol type, by

$$(\Omega_\sigma(\Sigma_1,\ldots,\Sigma_n)) \text{ type } = \text{ type } .$$

3.4.5   Completed expressions.

Let Λ be a well-formed LONGPAL book. Let λ be a line of Λ, ξ be the identifier of λ, and let τ be the indicator string of λ. We build an expression $\text{compl}(\xi)$ as follows.

If $\tau = \emptyset$ then $\text{compl}(\xi) = \langle\xi\rangle$.

If $\text{def}(\lambda) = \text{---}$ then $\text{compl}(\xi) = \langle\xi\rangle$.

If $\tau = \langle \beta_1 \rangle + \ldots + \langle \beta_n \rangle$, $n > 0$, then

$$\text{comp}(\xi) = \langle \xi \rangle + \langle (\rangle + \langle \beta_1 \rangle + \langle , \rangle + \ldots + \langle , \rangle +$$

$$+ \langle \beta_n \rangle + \langle) \rangle .$$

For example, in the book of 3.4.9, the identifier b produces as completed expression $b(x,y,u,z,v)$.

3.4.6 Admissible triples.

Let Λ be a well-formed LONGPAL book. We consider all triples (σ, Θ, Π), where $\sigma \in S(\Lambda)$ (see 3.4.3), Θ is a parentheses expression (see 3.4.1), Π is either a parentheses expression or the symbol type. Some of these triples will be called admissible. We define admissibility recursively by means of (i), (ii), (iii):

(i)   If $\sigma \in S(\Lambda)$, if λ is a line of Λ, if $\text{ident}(\lambda)$ is one of the elements of the string σ, then

$$(\sigma, \langle \text{ident}(\lambda) \rangle, \text{cat}(\lambda))$$

is an admissible triple (for notation see 3.3.7).

(ii)   If $\sigma \in S(\Lambda)$, if λ is a line of Λ, if $\text{def}(\lambda) = \text{PN}$, and if $\text{indstr}(\lambda) \subset \sigma$ (for definition of "$\subset$" see 3.2), then

$$(\sigma, \text{compl}(\text{ident}(\lambda)), \text{cat}(\lambda))$$

is an admissible triple.

(iii)   Let $\sigma \in S(\Lambda)$, $\tau \in S(\Lambda)$. Assume $n = \text{length}(\tau) > 0$. Let $\lambda_1, \ldots, \lambda_n$ be the lines of Λ with the property that $\text{ident}(\lambda_j) = \text{elt}_j(\tau)$ $(j = 1, \ldots, n)$.

Let $\Theta, \Pi, \Sigma_1, \ldots, \Sigma_n$ be such that

$$(\tau, \Theta, \Pi), (\sigma, \Sigma_1, \Gamma_1), \ldots, (\sigma, \Sigma_n, \Gamma_n),$$

are admissible triples, and where $\Gamma_1, \ldots, \Gamma_n$ are defined by

$$\Gamma_j = (\Omega_\tau(\Sigma_1, \ldots, \Sigma_n)) \, \mathrm{cat}(\lambda_j).$$

Then the triple

$$(\sigma, (\Omega_\tau(\Sigma_1, \ldots, \Sigma_n))\Theta, \, (\Omega_\tau(\Sigma_1, \ldots, \Sigma_k))\Pi)$$

is admissible.

3.4.6.1.   Note that it follows from the definition that there are no admissible triples if Λ is empty. Another consequence is that, if (τ, Θ, Π) is admissible, and if $\sigma \in S(\Lambda)$, $\tau \subset \sigma$, then (σ, Θ, Π) is admissible.

3.4.7   <u>Acceptable lines.</u>

We consider a well-formed LONGPAL book Λ and a line μ (not necessarily one of the lines of Λ). We say that μ is <u>acceptable</u> with respect to Λ if the following conditions (i), (ii), (iii) hold simultaneously.

(i)   Either $\mathrm{indic}(\mu) = 0$ or there is a $\lambda \in \Lambda$ with $\mathrm{indic}(\mu) = \mathrm{ident}(\lambda)$.

Before phrasing the other conditions we define the string σ as follows: $\sigma = \emptyset$ if $\mathrm{indic}(\mu) = 0$, and $\sigma = \mathrm{indstr}(\lambda) + \langle \mathrm{indic}(\mu) \rangle$ if $\mathrm{indic}(\mu) = \mathrm{ident}(\lambda), \lambda \in \Lambda$.

(ii)   If $\mathrm{def}(\mu) = -$ or $\mathrm{def}(\mu) = \mathrm{PN}$ then either $\mathrm{cat}(\mu) = $ <u>type</u> or $(\sigma, \mathrm{cat}(\mu), \underline{\text{type}})$ is an admissible triple (with respect to Λ).

(iii)   If PN $\neq$ def(μ) $\neq$ —, then (σ,def(μ),cat(μ)) is an admissible triple (with respect to Λ).

3.4.7.1   Note that, if Λ is the empty book, then the only acceptable lines are the lines μ with indic(μ) = $\emptyset$, def(μ) = — or PN, cat(μ) = type.

We also remark that it follows from 3.4.7 and from 3.4.2 that, if the line μ is acceptable with respect to the well-formed LONGPAL book Λ, then $\Lambda + \langle \mu \rangle$ is a well-formed LONGPAL book.

3.4.8   Correct LONGPAL book.

A correct LONGPAL book is a well-formed LONGPAL book Λ satisfying the following condition: for each n ($1 \leqslant$ n $\leqslant$ length(Λ)) the line $\text{elt}_n(\Lambda)$ is acceptable with respect to the book $\text{front}_{n-1}(\Lambda)$ (i.e. the book consisting of the first n-1 lines of Λ). Note that it follows from 3.4.2 that $\text{front}_{n-1}(\Lambda)$ is well-formed.

3.4.8.1   According to the above definition, the empty book is correct.

By virtue of the remark made in section 3.4.7.1 we may also give a recursive definition: The empty book is correct, and if the line μ is acceptable with respect to the correct book Λ, then $\Lambda + \langle \mu \rangle$ is correct.

3.4.9   Example of a correct LONGPAL book.

The example is printed on the top of page 23.

The columns "line" and "indstr" do not belong to the lines themselves. We have added them in order to facilitate the discussion.

The question whether a book like this corresponds to a piece of mathematics is not under discussion at the moment. Nevertheless the reader may be interested to know that it does have a certain significance. It tells the following story: if we axiomatize equality by saying that (i) x = x and (ii) x = y and z = y imply z = x, then we also have that x = y implies y = x and that x = y and y = z imply x = z. We can represent the argument

line	ind	indstr	ident	def	cat
λ_1	0		elt	PN	<u>type</u>
λ_2	0		x	—	elt
λ_3	x	x	y	—	elt
λ_4	y	x,y	is	PN	<u>type</u>
λ_5	x	x	a	PN	is(x,x)
λ_6	y	x,y	u	—	is(x,y)
λ_7	u	x,y,u	z	—	elt
λ_8	z	x,y,u,z	v	—	is(z,y)
λ_9	v	x,y,u,z,v	b	PN	is(z,x)
λ_{10}	u	x,y,u,	c	b(x,y,u,y,a(y))	is(y,x)
λ_{11}	z	x,y,u,z	w	—	is(y,z)
λ_{12}	w	x,y,u,z,w	d	b(y,z,w,z,a(z))	is(z,y)
λ_{13}	w	x,y,u,z,w	e	b(x,y,u,z,b(y,z,w,z,a(z)))	is(z,x)
λ_{14}	w	x,y,u,z,w	f	b(z,x,b(x,y,u,z,b(y,z,w,z,a(z))),x,a(x))	is(x,z)

in steps (1) - (14) which can be considered as translations of λ_1 - λ_{14} into common language.

(1) Assume there is a category of things called elements.

(2) If x is an element, and

(3) if y is an element,

(4) then there is a sort of things called "is(x,y)".

(This is a bit strange : we use this category "is(x,y)" in order to assert equality. Instead of asserting a = b we say that there is something of the category "is(a,b)").

(5) For all x we assume x = x (without proof; it is an axiom: we do not specify a particular defined object in the class is(x,x), but introduce it as a primitive notion).

(6) Again consider x,y as elements. Assume x = y

(7) Let z be the third element.

(8) Assume z = y

(9) We assume (without proof; it is an axiom) that under these circumstances z = x

(10) Let x and y be elements, and assume x = y. Since y = y (by virtue of (5)) we may replace, in (9), z by y, and we infer that y = x.

(11) Still x and y are elements with x = y. Again introduce the element z, Assume y = z.

(12) If in (9) we replace x by y, y by z (which is allowed because of y = z (see (11)) and z = z (see (5)), then we infer z = y.

(13) We may now apply (9) (by (12) we have satisfied (8)). It results that z = x.

(14) Apply (10), replacing x by z, z by x (by (13) we have satisfied the assumption of (10) for that case). It results that x = z.

The reader will notice awkward repetitions towards the end of the book (lines $\lambda_{10}, \lambda_{12}, \lambda_{13}, \lambda_{14}$), and this makes it pratically impossible to proceed in this way. This difficulty is overcome in the abbreviated version of LONG-PAL, to be treated in 3.5.

3.4.10   Without proof we mention a number of properties of a correct LONGPAL book $\Lambda = \langle \lambda_1 \rangle + \dots + \langle \lambda_n \rangle$.

(i)   For each j ($1 \leqslant j \leqslant n$) the expressions def($\lambda_j$) (if this is not — or PN) and cat(λ_j) (if this is not type) are entirely composed of identifiers taken from the sequence ident(λ_1),...,ident(λ_{j-1}) (apart from parentheses and comma's).

(ii)   In such expressions, the following is true. Those identifiers which are followed by an opening parenthesis are identifiers of lines $\lambda \in \Lambda$ satisfying both $\text{indstr}(\lambda) \neq \emptyset$ and $\text{def}(\lambda) = \text{PN}$. Those identifiers which are not followed by an opening parenthesis are either block openers or indicators of lines $\lambda \in \Lambda$ satisfying both $\text{indstr}(\lambda) = \emptyset$ and $\text{def}(\lambda) = \text{PN}$.

For example, in $\text{def}(\lambda_{10})$ of 3.4.8, viz. the expression $b(x,y,u,a(y))$, the letters x,y,u are block openers (since $\text{def}(\lambda_2) = \text{def}(\lambda_3) = \text{def}(\lambda_6) = \text{---}$, whereas a and b are the identifiers of lines (λ_5 and λ_9) with definition PN and indicator $\neq 0$.

(iii)   If (σ, Ω, Π) is an admissible triple with respect to Λ, and if $\Pi \neq \underline{\text{type}}$, then $(\sigma, \Pi, \underline{\text{type}})$ is admissible.

(iv)   If (σ, Ω, Π) and (σ, Ω, Π') are admissible with respect to Λ, then $\Pi = \Pi'$.

(v)   There exists an algorithm that achieves the following: If we start from a correct LONGPAL book and if we erase the category of each line apart from those whose definition is --- or PN, then the algorithm enables us to reconstruct the erased entries with the aid of the non-erased ones.

3.5   Description of PAL.

Having described LONGPAL completely, it is quite easy to say what PAL is. The difference lies only in the fact that the PAL-expressions are abbreviated notations for the LONGPAL-expressions. Actually a correct book written in PAL can be translated into a correct book in LONGPAL by the simple procedure of replacing every definition and every category (if they are not ---, PN or $\underline{\text{type}}$) by the LONGPAL-expressions they are abbreviations for. On the other hand, every

correct book written in LONGPAL is also a correct book in PAL. When we pass from a correct LONGPAL book to a PAL book, we may abbreviate several of its expressions, often in several ways, but is is by no means an obligation to do so.

3.5.1   We start with an example of a correct PAL book, that abbreviates the LONGPAL book of 3.4.9. This new book consists of the lines $\lambda_1, \ldots, \lambda_9$ of the one of 3.4.9, followed by $\lambda'_{10}, \ldots, \lambda'_{14}$:

line	ind	indstr	ident	def	cat
λ'_{10}	u	x,y,u	c	$b(y,a(y))$	$is(y,x)$
λ'_{11}	u	x,y,u,z	w	—	$is(y,z)$
λ'_{12}	w	x,y,u,z,w	d	$c(y,z,w)$	$is(z,y)$
λ'_{13}	w	x,y,u,z,w	e	$b(d)$	$is(z,x)$
λ'_{14}	w	x,y,u,z,w	f	$c(z,x,e)$	$is(x,z)$

3.5.2   Our present description of PAL is given by means of LONGPAL. However, part of the practical value of PAL lies in the fact that it is possible to manipulate with the abbreviated expressions themselves, rather then translating into LONGPAL at every stage.

3.5.3   The expressions occurring in PAL are still of the form described in 3.4.1, and a book written in PAL still satisfies (i) of 3.4.10, but it does not necessarily satisfy (ii) of 3.4.10.

3.5.4   Translation operator.

Let Λ be a correct LONGPAL book and let $\sigma \in S(\Lambda)$. We shall define, by recursive definition, a class Z_σ of parentheses expressions, and an operator

T_σ, mapping Z_σ into another class of parentheses expressions.

Let λ be one of the lines of Λ, and put $\mathrm{indstr}(\lambda) = \tau$. Assume that the integer h has the property that

$$\mathrm{front}_h(\tau) = \mathrm{front}_h(\sigma),$$

$$0 \leqslant h \leqslant \mathrm{length}(\tau), \quad 0 \leqslant h \leqslant \mathrm{length}(\sigma).$$

Put $k = \mathrm{length}(\tau) - h$. We shall build an expression Σ as follows. Let $\Theta_1, \ldots, \Theta_k$ be expressions for which $T_\sigma(\Theta_1), \ldots, T_\sigma(\Theta_k)$ have already been defined, i.e. $\Theta_1 \in Z_\sigma, \ldots, \Theta_k \in Z_\sigma$. We then take as a new element of Z_σ:

$$\Sigma = \langle \mathrm{ident}(\lambda) \rangle + \langle (\rangle + \Theta_1 + \langle , \rangle + \ldots + \langle , \rangle + \Theta_k + \langle) \rangle.$$

(If $k = 0$ we just take $\Sigma = \langle \mathrm{ident}(\lambda) \rangle$). Putting $\tau = \langle \beta_1 \rangle + \ldots + \langle \beta_{h+k} \rangle$, we define the effect of T_σ on the expression Σ as

$$(\Omega_\tau (\beta_1, \ldots, \beta_h, T_\sigma(\Theta_1), \ldots, T_\sigma(\Theta_k))) \quad \mathrm{def}(\lambda)$$

if $\mathrm{PN} \neq \mathrm{def}(\lambda) \neq -$, and by

$$T_\sigma(\Sigma) = (\Omega_\tau (\beta_1, \ldots, \beta_h, T_\sigma(\Theta_1), \ldots, T_\sigma(\Theta_k))) \quad \mathrm{compl}(\mathrm{ident}(\lambda))$$

if $\mathrm{def}(\lambda) = \mathrm{PN}$ or $-$ (for the definition of "compl" see 3.4.5).

3.5.4.1   In the above notation, the operator should, of course, be written as $\Omega_\tau(\beta_1, \ldots, \beta_h)$ if $k = 0$, $h > 0$, as $\Omega_\tau(T_\sigma(\Theta_1), \ldots, T_\sigma(\Theta_k))$ if $h = 0$, $k > 0$, and it is just the identity operator if $h = k = 0$.

In particular it follows from the above definition that if ξ is one of the elements of the string σ, then $T_\sigma(\langle \xi \rangle) = \langle \xi \rangle$.

Example. Consider the LONGPAL book of 3.4.9. We take $\sigma = \langle x \rangle + \langle y \rangle + \langle u \rangle + \langle z \rangle + \langle w \rangle$, and show that T_σ applied to the expression $b(d)$

produces $b(x,y,u,z,b(y,z,w,z,a(z)))$. We start from $b(d)$. We notice that b is

the identifier of line λ_9, with indicator string $\tau = \langle x \rangle + \langle y \rangle + \langle u \rangle + \langle z \rangle +$

$+ \langle v \rangle$. So we may take $h = 4$, $k = 1$, $\Theta_1 = d$. Assuming it has already been

shown that $T_\sigma(d)$ is defined, we infer that $T_\sigma(b(d)) = b(x,y,u,z,T_\sigma(d))$ (since

$\text{compl}(\text{ident}(\lambda_9)) = b(x,y,u,z,v))$.

Next we apply our definition in order to find $T_\sigma(d)$. We now have to con-

sider line λ_{12}, and $\tau = \langle x \rangle + \langle y \rangle + \langle u \rangle + \langle z \rangle + \langle w \rangle$. So we can take $h = 5$,

$k = 0$, and we get

$$T_\sigma(d) = (\Omega_\tau(x,y,u,z,w)) \; b(y,z,w,z,a(z)) \; = b(y,z,w,z,a(z))).$$

Hence $T_\sigma(b(d)) = b(x,y,u,z,b(y,z,w,z,a(z)))$.

3.5.5   Correct PAL book.

A book is called a correct PAL book if it can be obtained from a LONGPAL

book Λ in the following way: For every line $\lambda \in \Lambda$ we consider $\text{def}(\lambda)$ and $\text{cat}(\lambda)$.

If $\text{def}(\lambda)$ is an expression, and if we can find an expression Σ for which

$T_{\text{indstr}(\lambda)}(\Sigma)$ is defined and equal to $\text{def}(\lambda)$ then we replace $\text{def}(\lambda)$ by Σ. We

carry out the same procedure with $\text{cat}(\lambda)$. The procedure is not always unique:

a given LONGPAL book can give rise to several correct PAL books.

A correct PAL book can be translated into a correct LONGPAL book line

by line (in order to carry out the translations we only need the initial seg-

ment of the book that has already been translated). This reverse translation

is unique.

3.5.6   Definitional equality.

Let Λ' be a correct PAL book, and let Λ be its translation into LONGPAL.

If $\sigma \in S(\Lambda)$, if Σ is an expression for which $T_\sigma(\Sigma)$ is defined, then we say that

Σ is a _valid expression at_ σ. If both Σ_1 and Σ_2 are valid at σ, and if $T_\sigma(\Sigma_1) =$

$= T_\sigma(\Sigma_2)$, we say that Σ_1 and Σ_2 are _definitionally equal_ at σ.

3.5.7   Every expression valid at σ has a category which is either <u>type</u> or an expression valid at σ. Here we mean by "category of Σ" a Π such that $(\sigma, T_\sigma(\Sigma), \Pi))$ is an acceptable triple with respect to Λ.

3.5.8   As we remarked before, the expressions occurring in a correct PAL book do not always satisfy (ii) of 3.4.10. One thing remains true: a block opener occurring in an expression in a PAL book is never followed by an opening parenthesis.

3.6   <u>Description of LONGAL.</u>

As said in 3.1, LONGPAL is a sublanguage of LONGAL. In LONGAL we have a more complicated kind of expressions, using the full set of separation marks 3.3.1 (i), and using bound variables.

Although it is not strictly necessary, we shall take it as a principle not to use one and the same letter as a bound variable in different subexpressions of an expression.

3.6.1   <u>Expressions and bound variables.</u>

The expressions to be considered are certain strings of symbols; the symbols admitted in these strings are identifiers, and the seven separation marks listed in 3.3.1 (i). The parentheses expressions of 3.4.1 are special cases of the expression to be defined here. To each expression we shall attach a set of identifiers as well as a subset of that set. If an expression is denoted by Σ , then the sets attached to it are denoted by U_Σ and V_Σ . The elements of V_Σ are called the <u>bound variables</u> of Σ. We define these notions by recursion:

(i)   If β is an identifier, then ⟨ β ⟩ is an expression, and

$$U_{\langle \beta \rangle} = \{\beta\}, \qquad V_{\langle \beta \rangle} = \emptyset.$$

(ii)   If $n \geq 1$, if β is an identifier, and if $\Sigma_1, \ldots, \Sigma_n$ are expressions, such that, for i, j $(1 \leq i \leq n,\ 1 \leq j \leq n,\ i \neq j)$

$$U_{\Sigma_i} \cap V_{\Sigma_j} = \emptyset, \qquad \beta \notin V_{\Sigma_j}$$

then Σ, defined as

$$\langle \beta \rangle + \langle (\rangle + \Sigma_1 + \langle , \rangle + \ldots + \langle , \rangle + \Sigma_n + \langle) \rangle$$

is an expression, and

$$U_\Sigma = U_{\Sigma_1} \cup \ldots \cup U_{\Sigma_n} \cup \{\beta\}, \qquad V_\Sigma = V_{\Sigma_1} \cup \ldots \cup V_{\Sigma_n}.$$

(iii)   If Σ_1, Σ_2 are expressions, and if

$$U_{\Sigma_1} \cap V_{\Sigma_2} = \emptyset, \qquad U_{\Sigma_2} \cap V_{\Sigma_1} = \emptyset,$$

then Σ, defined as

$$\langle \{ \rangle + \Sigma_1 + \langle \} \rangle + \Sigma_2$$

is an expression, and

$$U_\Sigma = U_{\Sigma_1} \cup U_{\Sigma_2}, \quad V_\Sigma = V_{\Sigma_1} \cup V_{\Sigma_2}.$$

(iv)   If Σ_1 and Σ_2 are expressions, if β is an identifier, and if

$$U_{\Sigma_1} \cap V_{\Sigma_2} = \emptyset, \quad U_{\Sigma_2} \cap V_{\Sigma_1} = \emptyset, \beta \notin U_{\Sigma_1}, \beta \notin V_{\Sigma_2},$$

then Σ, defined as

$$\langle [\rangle + \langle \beta \rangle + \langle , \rangle + \Sigma_1 + \langle] \rangle + \Sigma_2$$

is an expression, and

$$U_\Sigma = U_{\Sigma_1} \cup U_{\Sigma_2} \cup \{\beta\}, \qquad V_\Sigma = V_{\Sigma_1} \cup V_{\Sigma_2} \cup \{\beta\}.$$

3.6.1.1 Examples.

In the examples a,b,c,x,y,z are identifiers.

(1) $\Sigma = [x,\{a\}b(a,w)]\,[y,c(x)]\,\{b(x,z)\}x;$ $U_\Sigma = \{a,b,c,x,y,z,w\}$, $V_\Sigma = \{x,y\}$.

(2) $\Sigma = \{a\}[b,a]\,a(b);$ $\qquad$ $U_\Sigma = \{a,b\}$, $\qquad$ $V_\Sigma = \{b\}$.

(3) Every parentheses expression Σ is an expression with $V_\Sigma = \emptyset$.

3.6.1.2

It is easy to see that for any given expression Σ the sets U_Σ and V_Σ are uniquely determined, and that $V_\Sigma \subset U_\Sigma$.

3.6.2 Congruent expressions.

Let Σ_1, Σ_2 be two expressions (in the sense of 3.6.1) and let $U_{\Sigma_1}, V_{\Sigma_1}$, $U_{\Sigma_2}, V_{\Sigma_2}$ be the sets attached to them. We say that Σ_1 and Σ_2 are congruent if there is a one-to-one mapping φ of V_{Σ_1} onto V_{Σ_2} such that

(i) the string Σ_1 transforms into Σ_2 if we replace those identifiers β of Σ_1 that are elements of V_{Σ_1} by the corresponding $\varphi(\beta)$'s, and if, moreover,

(ii) no $\varphi(\beta)$ belongs to $U_{\Sigma_1} \setminus V_{\Sigma_2}$.

3.6.2.1 Examples.

(1) $[x,y]\,a(x)$ and $[z,y]\,a(z)$ are congruent.

(2) $[x,y]\,w(x,z)$ and $[z,y]\,w(z,z)$ are not congruent, since they do not satisfy (ii). What amounts to the same thing is that they do not satisfy (i) if we interchange the expressions.

3.6.2.2   The notion of congruence is reflexive, symmetric and associative. The equivalence classes induced by it will be called congruence classes. If Σ belongs to the congruence class E, we shall also say that Σ is a representative of E.

3.6.3   Well-formed LONGAL books.

The notion is identical to the one of 3.4.2 apart from two modifications; both in 3.6.2 (ii) 3 and 3.6.2 (ii) 4 we replace "parentheses expression" by "expression".

3.6.4   The set S(Λ).

If Λ is a well-formed LONGAL book, we define S(Λ) exactly as in 3.4.3.

3.6.5   Substitution operator.

Let k be a positive number, and let σ be a string of k different identifiers: $\sigma = \langle \beta_1 \rangle + \ldots + \langle \beta_k \rangle$. We do not require that $\sigma \in S(\Lambda)$. Let $\Sigma_1, \ldots, \Sigma_k$ be expressions. We shall define an operator $\Omega_\sigma(\Sigma_1, \ldots, \Sigma_k)$. Its domain is the set $'(\Lambda)$ of all those expressions which have the property that an opening parentheses "(" is never preceded by one of the symbols $\beta_1, \ldots, \beta_k$. The range of the operator does not consist of expressions, but of congruence classes. We define the effect of the operator by recursion.

(i)   If ξ is an identifier, not occurring in the string σ, then the expression $\langle \xi \rangle$ is transformed into the congruence class containing the expression $\langle \xi \rangle$; if however, ξ is an element of σ, $\xi = \beta_j$, say, then the expression $\langle \xi \rangle$ is transformed into the congruence class containing the expression Σ_j.

(ii)   If ξ is an identifier, if $\Gamma_1,\dots,\Gamma_m$ are expressions, belonging to $C'(\Lambda)$, and if Θ is the expression

$$\langle \xi \rangle + \langle (\rangle + \Gamma_1 + \langle , \rangle + \dots + \langle , \rangle + \Gamma_m + \langle) \rangle \, ,$$

then $(\Omega_\sigma(\Sigma_1,\dots,\Sigma_k))\Theta$ is the congruence class containing

$$\langle \xi \rangle + \langle (\rangle + \Delta_1 + \langle , \rangle + \dots + \langle , \rangle + \Delta_m + \langle) \rangle \, .$$

Here we have chosen Δ_j, for each j $(1 \leqslant j \leqslant m)$, from the congruence class $(\Omega_\sigma(\Sigma_1,\dots,\Sigma_k))\Gamma_j$, in such a way that for all i,j $(1 \leqslant i \leqslant m, 1 \leqslant j \leqslant m, i \neq j)$,

$$U_{\Delta_i} \cap V_{\Delta_j} = \emptyset, \; \xi \notin V_{\Delta_j}$$

(since we have infinitely many identifiers available, we can always replace the bound variables by letters that are not used for any other purpose).

(iii)   If Γ_1 and Γ_2 are expressions belonging to $C'(\Lambda)$, and if Θ is the expression $\langle \{ \rangle + \Gamma_1 + \langle \} \rangle + \Gamma_2$, then $(\Omega_\sigma(\Sigma_1,\dots,\Sigma_k))\Theta$ is the congruence class containing $\langle \{ \rangle + \Delta_1 + \langle \} \rangle + \Delta_2$, where the Δ_1, Δ_2 are chosen as described under (ii) (in this case $m = 2$).

(iv)   If ξ is an identifier, not belonging to the string σ, and $\xi \notin U_{\Sigma_1},\dots,\xi \notin U_{\Sigma_k}$, if Γ_1,Γ_2 are expressions belonging to $C'(\Lambda)$, with $\xi \notin U_{\Gamma_1}$, $\xi \notin V_{\Gamma_2}$ and if Θ is the expression

$$\langle [\rangle + \langle \xi \rangle + \langle , \rangle + \Gamma_1 + \langle] \rangle + \Gamma_2,$$

then $(\Omega_\sigma(\Sigma_1,\dots,\Sigma_k))\Theta$ is the congruence class containing

$$\langle [\rangle + \langle \xi \rangle + \langle , \rangle + \Delta_1 + \langle] \rangle + \Delta_2,$$

where Δ_1 and Δ_2 are chosen as under (ii) (again with n = 2) with the extra precaution that $\xi \notin V_{\Sigma_1}$, $\xi \notin V_{\Sigma_2}$.

If, on the other hand, ξ belongs to σ, we first replace ξ by a new variable.

3.6.5.1   We also define $\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k)$ type = {type}, i.e. the set consisting of the symbol type only. In order to unify our terminology, we shall also say that type is a representative of the congruence class {type}.

3.6.6   Completed expressions.

Are defined as in 3.4.5.

3.6.7   Admissible quadruples.

We shall give something similar to the recursive definition of admissible triples in 3.4.6. The present situation is more complicated. Along with the recursive definition of triple (σ, Θ, Π) we wish to give a recursive definition of definitional equality. Roughly speaking, by admitting $(\sigma, \Theta_1, \Theta_2, \Pi)$ we say that, (in the context σ) Θ_1 and Θ_2 are definitionally equivalent expressions of category Π.

Another complication is that the recursion does not deal with a fixed book. In some cases the question whether a quadruple is admissible is reduced to the question whether a simpler quadruple is admissible with respect to a slightly longer book.

3.6.7.1   Shorter notation.

From now on we shall use an abbreviated notation. We shall quite often omit the $\langle$'s, $\rangle$'s and +'s of the string notation, and simply write things like $[\beta,\Sigma]\{\Theta_1\}\Theta_2$ instead of $\langle$ [$\rangle$ + $\langle \beta \rangle$ + $\langle , \rangle$ + Σ + $\langle \{ \rangle$ + Θ_1 + $\langle \} \rangle$ + Θ_2.

Furthermore we shall be a bit careless about congruence classes, simply

(but incorrectly) saying "the expression E" instead of "the congruence class containing the expression E". And we shall not always repeat the condition that in building new expressions, the bound variables of the constituents have to be replaced by new ones in order to avoid that a bound variable occurs at any other part of the expression in a different sense. We shall write $\Sigma \neq \Omega$ if we want to indicate that $U_\Sigma \cap V_\Omega = U_\Omega \cap V_\Sigma = \emptyset$.

3.6.7.2 Recursive definition of admissible quadruples.

Let Λ be a well-formed LONGAL book. We consider all quadruples $(\sigma, \Theta_1, \Theta_2, \Pi)$, where $\sigma \in S(\Lambda)$, Θ_1, Θ_2 are expressions, Π is either an expression or the symbol type. Some quadruples are called admissible, according to the following conditions.

(i)   If $(\sigma, \Theta_1, \Theta_2, \Pi)$ is admissible, then $(\sigma, \Theta_2, \Theta_1, \Pi)$ is admissible.

(ii)   If $(\sigma, \Theta_1, \Theta_2, \Pi)$ and $(\sigma, \Theta_2, \Theta_3, \Pi)$ are admissible, then $(\sigma, \Theta_1, \Theta_3, \Pi)$ is admissible.

(iii)   If $(\sigma, \Theta_1, \Theta_2, \Pi_1)$ and $(\sigma, \Pi_1, \Pi_2, \underline{type})$ are admissible, then $(\sigma, \Theta_1, \Theta_2, \Pi_2)$ is admissible.

(iv)   (cf. 3.4.6 (i)). If $\sigma \in S(\Lambda)$, if λ is a line of Λ, and if ident(λ) is one of the elements of the string σ, then

$$(\sigma, \langle \text{ident}(\lambda) \rangle, \langle \text{ident}(\lambda) \rangle, \text{cat}(\lambda))$$

is admissible.

(v)   (cf. 3.4.6 (ii). If $\sigma \in S(\Lambda)$, if λ is a line of Λ, if def(λ) = PN, and if indstr(λ) $\subset \sigma$, then

$$(\sigma, \text{compl}(\text{ident}(\lambda)), \text{compl}(\text{ident}(\lambda)), \text{cat}(\lambda))$$

is admissible.

(vi)   (cf. 3.4.6 (iii)). Let $\sigma \in S(\Lambda)$, $\tau \in S(\Lambda)$. Assume $n = \text{length}(\tau) > 0$.

Let $\lambda_1,\ldots,\lambda_n$ be the lines of Λ with the property that $\mathrm{ident}(\lambda_j) = \mathrm{elt}_j(\tau)$ $(j = 1,\ldots,n)$.

Let $\Theta,\Theta',\Pi,\Sigma_1,\ldots,\Sigma_n,\Sigma_1',\ldots,\Sigma_n'$ be such that the following quadruples are admissible:

$$(\tau,\Theta,\ \Theta',\Pi),$$

$$(\sigma,\Sigma_j,\Sigma_j',(\Omega_\tau(\Sigma_1,\ldots,\Sigma_n))\ \mathrm{cat}(\lambda_j)) \qquad (j = 1,\ldots,n),$$

then

$$(\sigma,(\Omega_\tau(\Sigma_1,\ldots,\Sigma_n))\Theta,\quad (\Omega_\tau(\Sigma_1',\ldots,\Sigma_n'))\Theta',\quad (\Omega_\tau(\Sigma_1,\ldots,\Sigma_k))\Pi)$$

is admissible. Note that Π and the $\mathrm{cat}(\lambda_j)$'s are not necessarily expressions; they can be the symbol <u>type</u>.

(vii)   Let $\sigma \in S(\Lambda)$, let $\Sigma_1,\Sigma_2,\Sigma_1',\Sigma_2'$ be expressions, let β be an identifier different from $\mathrm{ident}(\mu)$ for all $\mu \in \Lambda$. Assume (cf. 3.6.1 (iv)) that

$$\Sigma_1 \neq \Sigma_2,\ \Sigma_1' \neq \Sigma_2',$$

$$\beta \notin U_{\Sigma_1},\ \beta \notin V_{\Sigma_2},\ \beta \notin U_{\Sigma_1'},\ \beta \notin V_{\Sigma_2'}.$$

We shall extend the book Λ by an extra line λ. Its indicator string is σ, its identifier is β, its definition is ⸺, its category is Σ_1. The extended book is $\Lambda' = \Lambda + \langle\lambda\rangle$, and we also consider an extended string $\sigma' = \sigma + \langle\beta\rangle$ (thus $\sigma' \in S(\Lambda')$). We assume that

$$(\sigma,\Sigma_1,\Sigma_1',\ \underline{\mathrm{type}})$$

is admissible with respect to Λ, and that

$(\sigma', \Sigma_2, \Sigma'_2, \underline{type})$

is admissible with respect to Λ'. Then

$(\sigma, [\beta,\Sigma_1] \Sigma_2, [\beta,\Sigma'_1]\Sigma'_2, \underline{type})$

is admissible with respect to Λ.

(viii)   Let $\sigma \in S(\Lambda)$, let $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma'_1, \Sigma'_2$ be expressions; let β be an identifier different from $ident(\mu)$ for all $\mu \in \Lambda$. Assume that

$$\Sigma_1 \not\equiv \Sigma_2, \; \Sigma'_1 \not\equiv \Sigma'_2, \; \Sigma_1 \not\equiv \Sigma_3,$$

$$\beta \notin U_{\Sigma_1}, \; \beta \notin U_{\Sigma'_1}, \; \beta \notin V_{\Sigma_2}, \; \beta \notin V_{\Sigma'_2}, \; \beta \notin V_{\Sigma_3}.$$

We define Λ' and σ' as under (vii). Assume that

$(\sigma, \Sigma_1, \Sigma'_1, \underline{type})$

is admissible with respect to Λ, and that

$(\sigma', \Sigma_2, \Sigma'_2, \Sigma_3)$

is admissible with respect to Λ'. Then

$(\sigma, \; [\beta,\Sigma_1] \Sigma_2, \; [\beta,\Sigma'_1] \Sigma'_2, \; [\beta,\Sigma_1] \Sigma_3)$

is admissible with respect to Λ.

(ix)   Let $\sigma \in S(\Lambda)$, let $\Sigma_1, \Sigma'_1, \Sigma_3, \Sigma'_3, \Theta, \Theta'$ be expressions; let β be an identifier different from (μ) for all $\mu \in \Lambda$ Let

$$\Sigma_1 \not\equiv \Sigma_3, \; \Sigma'_1 \not\equiv \Sigma_3, \; \Theta \not\equiv \Sigma_3, \; \Theta' \not\equiv \Sigma'_3,$$

$$\beta \notin U_{\Sigma_1}, \quad \beta \notin U_{\Sigma'_1}, \quad \beta \notin V_{\Sigma_3}, \quad \beta \notin V_{\Sigma'_3}.$$

Assume that

$$(\sigma, \Sigma_1, \Sigma'_1, \underline{type})$$

$$(\sigma, [\beta, \Sigma_1] \Sigma_3, [\beta, \Sigma'_1] \Sigma'_3, \underline{type})$$

$$(\sigma, \Theta, \Theta', \Sigma_1)$$

are admissible with respect to Λ. Then

$$(\sigma, \Omega_{\langle \beta \rangle} (\Theta) \Sigma_3, \quad \Omega_{\langle \beta \rangle} (\Theta') \Sigma'_3, \underline{type})$$

is admissible with respect to Λ. Note that the effect of $\Omega_{\langle \beta \rangle}(\Theta)$ is that every occurrence of β is replaced by Θ.

(x)   Let $\sigma \in S(\Lambda)$, let $\Sigma_1, \Sigma_3, \Xi, \Xi', \Theta, \Theta'$ be expressions; let β be an idenifier different from $\text{ident}(\mu)$ for all $\mu \in \Lambda$.

$$\Sigma_1 \neq \Sigma_3, \quad \Theta \neq \Sigma_3, \quad \Theta \neq \Xi, \quad \Theta' \neq \Xi',$$

$$\beta \notin U_{\Sigma_1}, \quad \beta \notin V_{\Sigma_3}.$$

Assume that both

$$(\sigma, \Xi, \Xi', [\beta, \Sigma_1] \Sigma_3)$$

$$(\sigma, \Theta, \Theta', \Sigma_1)$$

are admissible with respect to Λ. Then

$$(\sigma, \{\Theta\}\Xi, \{\Theta'\}\Xi', (\Omega_{\langle \beta \rangle} (\Theta)) \Sigma_3)$$

is admissible with respect to Λ.

(xi)   Let $\sigma \in S(\Lambda)$, let $\Sigma_1, \Sigma_2, \Sigma_3, \Theta$ be expressions; let β be an identifier different from $\text{ident}(\mu)$ for all $\mu \in \Lambda$. Let

$$\Sigma_1 \neq \Sigma_2, \ \Sigma_1 \neq \Sigma_3, \ \Theta \neq \Sigma_1, \ \Theta \neq \Sigma_2, \ \Theta \neq \Sigma_3,$$

$$\beta \notin U_{\Sigma_1}, \ \beta \notin V_{\Sigma_2}, \ \beta \notin V_{\Sigma_3}.$$

Assume that both

$$(\sigma, \Theta, \Theta, \Sigma_1),$$

$$(\sigma, \ [\beta, \Sigma_1]\Sigma_2, \ [\beta, \Sigma_1]\Sigma_2, \ [\beta, \Sigma_1]\Sigma_3)$$

are admissible with respect to Λ. Then

$$(\sigma, \ \{\Theta\} [\beta, \Sigma_1]\Sigma_2, \ (\Omega_{\langle \beta \rangle}(\Theta)) \Sigma_2, \ (\Omega_{\langle \beta \rangle}(\Theta)) \Sigma_3)$$

is admissible with respect to Λ.

(xii)   Let $\sigma \in S(\Lambda)$, let $\Sigma_1, \Sigma_3, \Sigma_4$ be expressions, and let β be an identifier different from $\text{ident}(\mu)$ for all $\mu \in \Lambda$. Let

$$\Sigma_1 \neq \Sigma_3, \ \Sigma_1 \neq \Sigma_4,$$

$$\beta \notin U_{\Sigma_1}, \ \beta \notin U_{\Sigma_4}.$$

$$(\sigma, [\beta, \Sigma_1] \{\beta\} \Sigma_4, \ [\beta, \Sigma_1] \{\beta\} \Sigma_4, \ [\beta, \Sigma_1] \Sigma_3)$$

be admissible with respect to Λ (so in comparison with (viii) we have

replaced Σ_2 by $\{\beta\}\Sigma_4$, where Σ_4 does not contain β). Then

$$(\sigma, \ [\beta,\Sigma_1] \ \{\beta\} \ \Sigma_4, \ \Sigma_4, \ [\beta,\Sigma_1] \ \Sigma_3)$$

is admissible with respect to Λ.

3.6.8 Admissible triples.

We say that (σ,Θ,Π) is an admissible triple whenever $(\sigma,\Theta,\Theta,\Pi)$ is an admissible quadruple.

3.6.9 Acceptable lines.

Let Λ be a well-formed LONGAL book, and let μ be a line, not necessarily one of the lines of Λ. The definition of acceptability of μ with respect to Λ is verbally the same as the one for the LONGPAL case in 3.4.7.

3.6.10 Correct LONGAL book.

The definition of correctness of a well-formed LONGAL book is exactly the same as in the LONGPAL case of 3.4.8.

3.6.11 Further properties of a correct LONGAL book.

Without proof we mention a number of properties of a correct LONGAL book. Partly ((i) - (v)) they are very close to the corresponding properties of LONGPAL books (see 3.4.10). The second part of (iv) is merely a conjecture, and so is (v).

(i)   As 3.4.10(i), with the modification that the expressions do not exclusively contain $\text{ident}(\lambda_1),\ldots,\text{ident}(\lambda_{j-1})$, but possibly also bound variables.

(ii)   As 3.4.10 (ii), with the modification that identifiers which are not followed by an opening parenthesis can also be bound variables.

(iii)   As 3.4.10 (iii).

(iv)   If (σ, Ω, Π) and (σ, Ω, Π') are both admissible, then we have:

If $\Pi = \underline{type}$, then $\Pi' = type$.
Probably we have moreover:

If $\Pi \neq \underline{type}$, then $(\sigma, \Pi, \Pi', \underline{type})$ is an admissible quadruple.

(v)   There probably exists an algorithm that does the following. If we start from a correct LONGPAL book, and if we erase the category of each column apart from those whose definition is — or PN, then the algorithm enables us to construct the missing categories. Although the new categories are not necessarily the same expressions as the old ones, they are interchangable with the old ones in the following sense: if σ is the indicator string of a line, and if Π is the old category and if Π' is the new one, then $(\sigma, \Pi, \Pi', \underline{type})$ is an admissible quadruple.

(vi)   If $(\sigma, \Sigma, \underline{type})$ is admissible, then Σ has the form

$$[\beta_1, \Sigma_1] \ldots [\beta_k, \Sigma_k] \beta(\Theta_1, \ldots, \Theta_m),$$

where $\beta_1, \ldots, \beta_k$, are bound variables, $\Theta_1, \ldots, \Theta_n$ are expressions, and β is the identifier of a line with definition PN and category $\underline{type}$. (If $k = 0$, $m > 0$ we of course mean $\beta(\Theta_1, \ldots, \Theta_k)$, if $k > 0$, $m = 0$ we mean $[\beta_1, \Sigma_1] \ldots [\beta_k, \Sigma_k]\beta$; if $k = m = 0$ we just mean β).

3.6.12   <u>Conjecture on normal forms.</u>

We say that an expression is _normal_ if it does not contain any } followed by a [. This means that it has the form

$$[\beta_1, \Sigma_1] \cdots [\beta_k, \Sigma_k] \{\Gamma_1\} \cdots \{\Gamma_h\} \beta(\Theta_1, \ldots, \Theta_m)$$

where $\Sigma_1, \ldots, \Sigma_k$, $\Gamma_1, \ldots, \Gamma_h$, $\Theta_1, \ldots, \Theta_m$ are normal expressions, and β is an identifier (if one or more of the integers k,h,m are 0, the corresponding parts of the formula have to be omitted).

The following statements about a correct LONGAL book are conjectured.

Conjecture 1.

If (σ, Θ, Π) is an admissible triple (where either Π is an expression or $\Pi = \underline{type}$), then there is a normal expression Θ' such that $(\sigma, \Theta, \Theta', \Pi)$ is an admissible quadruple.

If Θ' is such that there is no shorter normal expression Θ' for which $(\sigma, \Theta', \Theta'', \Pi)$ is admissible, then Θ' is said to have minimal normal form.

Conjecture 2.

If (σ, Θ, Π) is an admissible triple, then there is a Θ' in minimal normal form for which $(\sigma, \Theta, \Theta', \Pi)$ is admissible. This Θ' is unique up to congruence. There is an algorithm enabling us to compute Θ' (assuming that Λ is correct, $\sigma \in S(\Lambda)$, (σ, Θ, Π) admissible).

3.6.12.1   If the above conjectures are true, then the rules for admissibility (see 3.6.7) can be replaced by a much simpler, equivalent set of rules. In particular we might restrict ourselves to expressions in minimal normal form, and we might reduce the discussion to triples instead of quadruples. Note that if the conjectures are true, and if $(\sigma, \Theta_1, \Theta_2, \Pi)$ is admissible, then Θ_1 and Θ_2 have the same minimal normal form.

3.7   Description of AUTOMATH.

AUTOMATH can be considered as an abbreviated form of LONGAL, just like PAL is an abbreviated form of LONGPAL.

An AUTOMATH book can be obtained from a LONGAL book Λ by abbreviating the expressions $def(\lambda)$ and $cat(\lambda)$ (for each $\lambda \in \Lambda$). The translation from abbreviated expressions into the full expression, i.e. the translation from AUTOMATH into LONGAL, is obtained by a translation operator S_σ. We remark that every correct PAL book is also a correct AUTOMATH book, and that the effect of S_σ on the PAL book is the smae as the one described in 3.5.4. Thus, in the terminology of 3.5.4 and 3.7.1, T_σ is the restriction of S_σ to Z_σ.

Actually, the extension of T_σ to S_σ hardly involves new ideas.

3.7.1   Translation operator.

Let Λ be a correct LONGAL book and let $\sigma \in S(\Lambda)$. We shall define a class Y_σ of expressions, and an operator S_σ mapping the elements of Y_σ into expressions not necessarily in Y_σ.

The class Y_σ and the operator S_σ are defined recursively by means of (i), (ii), (iii):

(i)   This is what has been said in 3.5.4 (only replace Z_σ by Y_σ, T_σ by S_σ).

(ii)   If $\Sigma_1 \in Y_\sigma$, $\Sigma_2 \in Y_\sigma$ and if β is an identifier not occurring as a $def(\mu)$ with $\mu \in \Lambda$, and if

$$\Sigma_1 \not\equiv \Sigma_2, \; \beta \notin U_{\Sigma_1}, \; \beta \notin V_{\Sigma_2},$$

then

$$[\beta, \Sigma_1] \Sigma_2 \in Y_\sigma$$

and

$$S_\sigma([\beta,\Sigma_1]\ \Sigma_2) = [\beta,\ S_\sigma(\Sigma_1)]\ S_\sigma(\Sigma_k).$$

(iii)   If $\Sigma_1 \in Y_\sigma$, $\Sigma_2 \in Y_\sigma$, $\Sigma_1 \neq \Sigma_2$, then

$$\{\Sigma_1\}\ \Sigma_2 \in Y_\sigma$$

and

$$S_\sigma(\{\Sigma_1\}\Sigma_2) = \{S_\sigma(\Sigma_1)\}\ S_\sigma(\Sigma_2).$$

3.2.7   Correct AUTOMATH book.

The definition of a correct AUTOMATH book can be copied from the one of a correct PAL book in 3.5.5. Throughout that section 3.5.5 replace "PAL" by "AUTOMATH", "LONGPAL" by "LONGAL", T by S.

The AUTOMATH book obtained from a correct PAL book need not be in normal form.

3.7.3   Direct definition of AUTOMATH.

We have defined AUTOMATH by means of LONGAL; in this definition LONGPAL and PAL served a heuristic purpose only.

Needless to say, it is possible to define the rules for correctness of an AUTOMATH book without discussing LONGPAL first.

We shall not attempt a complete description of such rules in this report. Actually any mathematician will apply them more or less intuitively, since it is so close to the way he uses to handle mathematical material.

3.7.4   Processors for AUTOMATH.

A _processor_ is a computer program enabling a computer to check line by line whether a certain amount of input represents a correct AUTOMATH book.

If the conjectures of 3.6.12 are true then there exists, in theory, an ideal processor. The normal forms of the expressions in mathematics are, however, very long. If we translate everything into normal forms then the amount of work can be expected to depend exponentially on the length of the book, and that is, of course, prohibitive.

The practical problem is to train a computer to do his work in the relatively simple cases the human author deals with, taking only a small number of steps at a time. If the computer does not see in a reasonable amount of time that a line is correct, he asks for help. In cases where the line was correct but the poor processor was unable to grasp, the human author might assist the computer with a hint, or make it easier for him by writing an extra line. Therefore, the man-machine cooperation will be close to a teacher-pupil relation.

Needless to say, the computer should never believe an incorrect line.

The processor in operation at the time this report was written (november 1968) seems to be reasonably effective, in the sense that it did not do much superfluous work, and did not need any hints.

But is must be said that experiece has been very limited thus far.

Thus far we have hardly tackled the problem of storing the large amounts of material necessary to read an AUTOMATH text of an ordinary mathematical book, or of a research paper. It seems reasonable that we have to devise systematic rules for forgetting things, and certainly for forgetting names of identifiers so that we can use them agian. There is not too much reason for pessimism; the development and use of languages like AUTOMATH will take place in a period where computer memories will grow tremendously; at the same time they can be expected to become cheaper and cheaper.

INDEX.

Lingual symbols

, [ ] ( ) { } — o	3.3.1
PN	2.1; 3.3.1
<u>type</u>	1.2.1; 3.3.1

Metalingual symbols

cat(λ)	3.3.7
def(λ)	3.3.7
elt$_k$	3.2
front$_k$	3.2
ident(λ)	3.3.7
indic(λ)	3.3.7
indstr(λ)	3.3.7
$S(\Lambda)$	3.4.3; 3.6.4
S_σ	3.7.1
T_σ	3.5.4
U_Σ	3.6.1
V_Σ	3.6.1
Y_σ	3.7.1
Z_σ	3.5.4
$\Omega_\sigma(\Sigma_1,\ldots,\Sigma_k)$	3.4.4
$\emptyset$	3.2
$\subset$	3.2
$\langle\ \rangle$	3.2
$+$	3.2
$\neq$	3.6.7.1