<u>Definition of AUTOMATH</u>

as given by N G de Bruyn in a course in April/May 1970
Notes by L.S. v.B. Jutting.

I

I.1 <u>Syntax</u>.

Basic symbols are

a countable set of constants: $A = \{a, b, c, a1, b1, c1, a2, b2, c2, \dots\}$

a countable set of variables: $X = \{x, y, z, x1, y1, z1, x2, y2, z2, \dots\}$

a countable set of dummies $S = \{s, t, u, s1, t1, u1, s2, t2, u2, \dots\}$

and moreover

† | * | := | PN | EB | <u>type</u> | , | (|) | { | } | [|]

```
<book> ::=          | <book> † <line>
<line> ::= <indicator string> * <definitional part> * <category>
<indicator string> ::=          | <indicator string> , <variable>
<variable> ::= x | y | z | x1 | y1 | z1 | x2 | y2 | z2 | ...
<definitional part> ::= <constant> := <expression> | <constant> := PN | <variable> := EB
<constant> ::= a | b | c | a1 | b1 | c1 | a2 | b2 | c2 | ...
<category> ::= type | <expression>
<expression string> ::= <expression> | <expression string> , <expression>
<expression> ::= <constant> | <variable> | <constant> ( <expression string> )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        | <dummy> | {<expression>} <expression> | [ <dummy> , <expression> ]
        <expression>
<dummy> ::= s | t | u | s1 | t1 | u1 | s2 | t2 | u2 | ...
```

<u>Remark</u> The part of the syntax above the dotted line is the syntax of PAL (cf [2])

The constant resp. variable occurring before := in the definitional part of a line is called the <u>identifier</u> of that line.

I.2 **Free and bound dummies.**

Let Σ be an expression string

$A(\Sigma)$ is the set of constants occurring in Σ

$X(\Sigma)$ is the set of variables occurring in Σ

$S(\Sigma)$ is the set of dummies occurring in Σ

Let Σ be an expression string and s a dummy.
We define a function $f(\Sigma, s)$ with values in $\{$free, bound, fraudulent, absent$\}$ by recursion as follows

If Σ is an expression then

if $\Sigma \in A$ then $f(\Sigma, s) =$ absent

if $\Sigma \in X$ then $f(\Sigma, s) =$ absent

if $\Sigma = a(\Sigma_1)$ where Σ_1 is an expression string then $f(\Sigma, s) = f(\Sigma_1, s)$

if $\Sigma \in S$, $\Sigma \neq s$ then $f(\Sigma, s) =$ absent

if $\Sigma = s$ then $f(\Sigma, s) =$ free

if $\Sigma = \{\Sigma_1\}\Sigma_2$, Σ_1 and Σ_2 expression then $f(\Sigma, s)$ is as given in Table I

if $\Sigma = [u, \Sigma_1]\Sigma_2$, Σ_1 and Σ_2 expressions and $u \neq s$ then $f(\Sigma, s)$ is as given in Table I

if $\Sigma = [s, \Sigma_1]\Sigma_2$, Σ_1 and Σ_2 expressions then $f(\Sigma, s)$ is as given in Table II

If Σ is not an expression,

$\Sigma = \Sigma_1, \Sigma_2$ where Σ_1 an expression string, Σ_2 an expression then $f(\Sigma, s)$ is as given in Table I

Table I

$f(\Sigma_1, s)$ \ $f(\Sigma_2, s)$	free	bound	fraudulent	absent
free	free	fraudulent	fraudulent	free
bound	fraudulent	fraudulent	fraudulent	bound
fraudulent	fraudulent	fraudulent	fraudulent	fraudulent
absent	free	bound	fraudulent	absent

Table II

$f(\Sigma_1, s)$ \ $f(\Sigma_2, s)$	free	bound	fraudulent	absent
free	fraudulent	fraudulent	fraudulent	fraudulent
bound	fraudulent	fraudulent	fraudulent	fraudulent
fraudulent	fraudulent	fraudulent	fraudulent	fraudulent
absent	bound	fraudulent	fraudulent	bound

Let Σ be an expression string

$F(\Sigma)$ is the collection of free dummies occurring in Σ

$B(\Sigma)$ is the collection of bound dummies occurring in Σ.

I_3 Substitution

Let Λ be an expression string, $n \geq 0$, $\Sigma_1, \dots \Sigma_n$ expressions, $\xi_i \in X \cup S$ $(i = 1 \dots n)$, $\xi_i \neq \xi_j$ $(i \neq j,\ i, j = 1 \dots n)$

We will define $\mathrm{Subst}_{\xi_1 \to \Sigma_1, \dots \xi_n \to \Sigma_n} \Lambda$ (or briefly $\mathrm{Subst}\ \Lambda$) by recursion

If Λ is an expression then

- if $\Lambda \in A$ then $\mathrm{Subst}\ \Lambda = \Lambda$
- if $\Lambda = \xi_i$ for a certain i then $\mathrm{Subst}\ \Lambda = \Sigma_i$
- if $\Lambda \in X$, $\lambda \neq \xi_i$ $(i = 1 \dots n)$ then $\mathrm{Subst}\ \Lambda = \Lambda$
- if $\Lambda = a(\Lambda_1)$, Λ_1 an expression string then $\mathrm{Subst}\ \Lambda = a(\mathrm{Subst}\ \Lambda_1)$
- if $\Lambda \in S$, $\lambda \neq \xi_i$ $(i = 1 \dots n)$ then $\mathrm{Subst}\ \Lambda = \Lambda$
- if $\Lambda = \{\Lambda_1\}\Lambda_2$, Λ_1 and Λ_2 expressions, then $\mathrm{Subst}\ \Lambda = \{\mathrm{Subst}\ \Lambda_1\}\mathrm{Subst}\ \Lambda_2$
- if $\Lambda = [s, \Lambda_1]\Lambda_2$, Λ_1 and Λ_2 expressions, then $\mathrm{Subst}\ \Lambda = [\mathrm{Subst}\ s, \mathrm{Subst}\ \Lambda_1]\ \mathrm{Subst}\ \Lambda_2$

If Λ is not an expression then

$\Lambda = \Lambda_1, \Lambda_2$ where Λ_1 an expression string, Λ_2 an expression and $\mathrm{Subst}\ \Lambda = \mathrm{Subst}\ \Lambda_1, \mathrm{Subst}\ \Lambda_2$

Two expressions Σ and Σ^* are said to be congruent if there exist a 1-1 mapping $\varphi : B(\Sigma) \to B(\Sigma^*)$ such that if $B(\Sigma) = \{\xi_1, \dots \xi_n\}$

$$\mathrm{Subst}_{\xi_1 \to \varphi(\xi_1), \dots \xi_n \to \varphi(\xi_n)} \Sigma = \Sigma^*$$

For expression strings we define congruence by recursion:
Let Σ and Σ^* be expression strings and not expressions.
Then $\Sigma = \Sigma_1, \Sigma_2$ and $\Sigma^* = \Sigma_1^*, \Sigma_2^*$ where Σ_1 and Σ_1^* are expression strings, Σ_2 and Σ_2^* are expressions
Σ and Σ^* are called congruent if Σ_1 and Σ_1^* are congruent and Σ_2 and Σ_2^* are congruent.

Lemma 1 Congruence is an equivalence relation

proof By recursion.

Lemma 2 Let Σ be an expression string, $T \subset S$, T finite.
Then there exists an expression string Σ^* congruent Σ such that $B(\Sigma^*) \cap T = \emptyset$
Moreover, if $S(\Sigma) = B(\Sigma) \cup F(\Sigma)$, i.e. if Σ does not contain Fraudulent elements, then the same is true for Σ^*

proof Let $T \cap B(\Sigma) = \{t_1, \dots t_n\}$ Choose n different elements $u_1, \dots u_n$ from $S \setminus (S(\Sigma) \cup T)$. Then the expression $\Sigma^* = \mathrm{Subst}_{t_1 \to u_1, \dots t_n \to u_n} \Sigma$ has the required properties.