Invited lecture at the
Mathematics Knowledge Management Symposium
25-29 November 2003
Heriot-Watt University, Edinburgh, Scotland

Memories of the AUTOMATH project

by N.G. de Bruijn

N.G. de Bruijn, emeritus professor
email: n.g.d.bruijn@tue.nl
Homepage: http://www.win.tue.nl/~wsdwnb/
Address:
  Eindhoven University of Technology,
  Department of Mathematics and Computer Science
  PO Box 513, 5600 MB EINDHOVEN, The Netherlands
THE AUTOMATH PROJECT

1972-1977

EINDHOVEN, THE NETHERLANDS

about 30 man-years

Support: ZWO + THE

ZWO: The Netherlands national science foundation
THE: Technological University Eindhoven

THE LANGUAGE AUTOMATH HAD BEEN DEFINED IN 1968
AND NEVER CHANGED ANY MORE.
ARISTOTLE (384-322 B.C.)

DISCOVERING MATHEMATICS,
FINDING PROOFS
require ingenuity

but

CHECKING PROOFS
can be mechanical (algorithmical)
CLAIM

COMPLETE FORMALIZATION
AND VERIFICATION
OF MATHEMATICS
IS NOT JUST
THEORETICALLY POSSIBLE,
IT IS FEASIBLE!

IT IS EVEN ATTRACTIVE,
IN PARTICULAR IN CASES WHERE
METICULOUS MATHEMATICAL PRECISION
GENERATES ELEGANCE.
CHECKING CORRECTNESS OF A COMPLETE NETWORK OF definitions references theories books

is finally feasible in the computer age.

The system should be SIMPLE and VERY GENERAL

For $\Delta \Lambda$ (the essence of AUTOMATH) the core of the checker is less than a page of Pascal code.
Such a tree (with reference arrows) might represent a full mathematics encyclopedia.
DREAM OF THE SIXTIES
INTERACTIVE
PRODUCTION AND VERIFICATION
OF MATHEMATICS

NOW BEGINNING
TO COME TRUE

ALL BY THEMSELVES, THE CREATIVE POWER
OF COMPUTERS IS LIMITED

(NEARLY LIKE OURS)
THE AUTOMATH PROJECT
1972 - 1977

L.S. van Benthem Jutting
(translated Landau’s Introduction to Analysis)

R.P. Nederpelt
(first proofs of strong normalisation)

D.T. van Daalen
(full theory of Automath)

I. Zandleven
(Chief programmer)

J. Zucker
(intro analysis in AUT-Π, using telescopes)

A. Kornaat
(general assistant)

R. de Vrijer
(some theory)

Assistant programmers, typists, students.
The Automath project, 1972-1977

IMPORTANT DECISIONS

• Don’t try Automatic Theorem Proving.

• Stay as close as possible to standard mathematical presentation.

• Prefer the use of TYPED SETS, but don’t forbid untyped sets.

• Try to keep the LOSS FACTOR constant, using the mechanism for definitions and abbreviations that made mathematics so successful. (AUTOMATH expands definitions and abbreviations only when strictly necessary).
• Don’t put logic in the system; let the user start his books with it.

• Don’t put induction and recursion in the system; consider it as book material, even when that might be slightly clumsier.

• Never prefer speed and efficiency over simplicity and generality.

• Test the feasibility and get experience by translating a sizable amount of mathematical material (the choice fell on Landau’s Introduction to Analysis). Keep strictly to the text. Never replace it by a text that would suit Automath better.
WHAT PRECEDED AUTOMATH

The language of math used to be acquired by OSMOSIS (sponge).

The rules of the game were NEVER explained explicitly.

Yet there had been perfect authors.

Authors and readers had NO support from philosophy, logic, foundations.

Nevertheless:

Mathematical literature offered an almost perfect vehicle for mathematical communication in the twentieth century.
TEACHING AND LEARNING
MATHEMATICS

If you can't explain your mathematics to a machine, it is an illusion to think I can explain it to a student.
Logic teaching never explained the rules of the Mathematics game. Logic or Mathematics? The family (finite automaton).
A mathematician’s personal history and interests before starting Automath

Born 1918, The Hague.
Finished middle school 1934.
Depression, no future.
Studying mathematics (old-fashioned) in isolation 1934-1936.
Leiden University 1936-1939.
Assistant at Techn. Univ. Delft 1939-1944.

analytic number theory

Ph.D. 1943

algebraic number theory, modular functions

Professor at Delft 1946-1952.

combinatorics, collaboration with Erdős

Professor at Amsterdam 1952-1960

asymptotics
Polya theory
functional analysis

Professor at Eindhoven 1960-

functional analysis
1962: programming combinatorics on an IBM 1820
Fourier theory for generalized functions
Algol
THINGS THAT PUSHED ME INTO THE DIRECTION OF AUTOMATH

1. Using $\lambda$-calculus notation in analysis courses.

2. A long tedious proof that I did not trust anymore.


4. I was looking for original interesting programming tasks.
PREVIOUS EXPERIENCES RELEVANT FOR IDEAS ON AUTOMATH

• 1937 Learning Naive Set Theory (Kamke)
  "Do these large cardinals ”exist”? The power of language, talking coherently on things that make no sense at all.

• 1952-1960 Amsterdam, the world of Brouwer-Heyting-Beth. Heyting: proof of \( A \rightarrow B \).

• 1958 New Math!! I felt that it offered no understanding for the rules of the mathematics game.

• 1962- ... Playing with early computers.
  Combinatorics programming.
  Learning Algol in short courses by Van Wijngaarden and Dijkstra.
  Inspirations from block structure and declarations in Algol ’60
  (Mathematicians had always opened blocks but had never bothered to close them explicitly).

• 1964 Teaching analysis with \( \lambda \)’s (the notation only, no theory).

• Central problem for Van der Meiden’s thesis on Banach algebras, solved mechanically by natural deduction techniques (with flags).
Using $\lambda$-notation in courses on Functional Analysis

**Notation suggested by Freudenthal:**

$\vdash x \in S$ instead of $\lambda x \in S$.

**Example:**

**Defining the Fourier operator:**

$$F := \vdash_{f \in L^2} \vdash_{x \in \mathbb{R}} \int_{-\infty}^{\infty} e^{-2\pi ixt} f(t) \, dt$$

**Bourbaki should have done it.**
GELFAND’S ELEGANT BANACH ALGEBRA SOLUTION OF WIENER’S TAUBERIAN THEOREM.

IT USED THE AXIOM OF CHOICE!!

The maximal ideals of the Banach algebra were the points of a topological space. My attempt to get rid of the axiom of choice: replace the topology by point-free topology.

There remained a hard set-theoretical problem that I could not do until I managed to solve it 'mechanically by flag-style natural deduction.'
In 1966 I had to check a long proof in combinatorics, full of repetitions of a small number of simple ideas, and I felt the need to build a computer program to do the checking.
In Eindhoven I had a privileged position, being completely free to start any subject I liked. Accordingly, I began, at the end of 1966, to work on computer verification of mathematics. The first attempts already showed some of the AUTOMATH's characteristics of AUTOMATH.

For these first attempts see: Selected papers on Automath, p. 57-72.

Developing AUTOMATH was a very emotional affair.

It was a combination of natural deduction, type theory and lambda calculus. In particular, by the idea of 'proofs as objects', everything fell into its place. The whole structure of mathematics became visible in a quite simple system.

One is always happy with a discovery, of course, but here it was so much more than discovering theorems and facts. AUTOMATH was more than a system or a theory. It was a LIFE STYLE.

Its philosophy was definitely anti-platonistic. There was no longer any question of the existence of mathematical objects. The only thing that could claim existence was mathematical language, being completely representable in the physical world.
MATHEMATICAL DEDUCTION

CONTEXT INDICATED BY
NESTED FLAGS AND FLAG POLES
INTRO VARIABLE | ASSUMPTION

$\exists x \in S$

$\exists x > 5$

BOOK STRUCTURE

LINES:
DEFINITIONS
OR
FACTS
(PROVED
PROPOSITIONS
AND
AXIOMS)
A THEOREM

\[ x : \text{T} \]
\[ K(x) \]
\[ y : S(x) \]
\[ f = \ldots : \ldots \]

AN APPLICATION

MACHINE WANTS

- A VALUE FOR \( x \)
- A PROOF FOR \( K(a\ldots) \)
- A VALUE FOR \( y \)

AND DOES:
- A TYPE CHECK
- A PROOF CHECK
- A TYPE CHECK
ANOTHER PARALLEL

\[ x : \text{TREATMENT} \]
\[ f : = \ldots : Z \]
\[ \forall x : \text{TREATMENT} \quad f(x) \]

\[ x : \text{PROOF OF} (a) \]
\[ \quad \vdots = \ldots : \text{PROOF OF} (b) \]
\[ \quad \vdots : \text{PROOF OF} (a \rightarrow b) \]

HEYTING'S IDEA
THIS LED TO THE CONCEPTS

• dependent types
• proofs as objects ('propositions as types')

\[ p(c(x, \ldots), \ldots) : \text{PROOFOF}(\kappa > 5) \]

(The \( p(c(x, \ldots), \ldots) \) is an instantiated reference to a previous proof.)

All this happens in PAL (lambda-free Automath) already. PAL’s mathematics is roughly the mathematics from before about 1800. It took the whole 19-th century to let the idea of a FUNCTION develop from a metamathematical notion to a mathematical object.

For this, lambda calculus was an essential tool, but it is taking a very long time to let lambda calculus conquer the whole field of mathematics.
Mathematical Deduction

Books with the following kinds of lines:

**Information-carrying lines** (indicated below as ···):
- C: constructing objects, propositions or types
- A: claiming a statement without proof (axiom)
- P: declaring objects, propositions or types as primitives
- T: giving a statement with proof

**Context-changing lines:**
- assum: introducing an assumption
- intro: Introducing a (typed) variable or a variable type

(1) ...
(2) ...
(3) ...
(4) intro
(5) intro
(6) ...
(7) intro
(8) assum
(9) ...
(10) ...
(11) intro
(12) ...
(13) assum
(14) intro
(15) ...
(16) ...
(17) ...
Long Semipal

(3) impl(x,y) := PN
(4) con := PN
(5) non(x) := impl(x,con)
(6) or(x,y) := impl(non(x),y)
(7) and(x,y) := non(or(non(x),non(y)))
(8) equiv(x,y) := or(and(x,y),and(non(x),non(y)))

Semipal, presented with flags

(1) x
(2) y
(3) impl := PN
(4) con := PN
(5) non := impl(con)
(6) or := impl(non, y)
(7) and := non(or(non, non(y)))
(8) equiv := or(and, and((non, non(y))))

Semipal, presented with context indicators

(1) ◦x := —
(2) x ◦ y := —
(3) y ◦ impl := PN
(4) ◦con := PN
(5) x ◦ non := impl(con)
(6) y ◦ or := impl(non, y)
(7) y ◦ and := non(or(non, non(y)))
(8) y ◦ equiv := or(and, and(non, non(y)))
Survey of kinds of lines

Kinds of lines in AUTOMATH (as well as in PAL), with interpretation. They can occur in any context.

On the left is a code for the kind of line; it is not a part of the line. The letters x, c, f stand for identifiers; p, q stand for expressions.

(2tEB) x := —– : type introducing a type variable
(2tPN) c := PN : type introducing primitive type
(2tDF) f := p : type defining a new type

(2pEB) x := —– : prop introducing a prop variable
(2pPN) c := PN : prop introducing primitive prop
(2pDF) f := p : prop defining a new prop

(3tEB) x := —– : q introducing an object variable
(3tPN) c := PN : q introducing a primitive object
(3tDF) f := p : q defining a new object

(3pEB) x := —– : q making an assumption
(3pPN) c := PN : q proclaiming an axiom
(3pDF) f := p : q proving a theorem

Note: A prop is a proof class. It should not be seen as a proposition, but as the class of all proofs of a particular proposition.

In group 3p the letters x, c, f do not correspond to things that occur in formulas in ordinary mathematical texts. There they are replaced by references used in explanations.
Example of a book written in PAL

(1) \( \alpha : \text{type} \)

(2) \( x : \alpha \)

(3) \( y : \alpha \)

(4) \( \text{EQ} := \text{PN} : \text{prop} \)

(5) \( a := \text{PN} : \text{EQ}(x,x) \)

(6) \( u : \text{EQ}(x,y) \)

(7) \( z : \alpha \)

(8) \( v : \text{EQ}(z,u) \)

(9) \( b := \text{PN} : \text{EQ}(z,x) \)

(10) \( m := a(y) : \text{EQ}(y,y) \)

(11) \( c := b(y,m) : \text{EQ}(y,x) \)

(12) \( w : \text{EQ}(y,z) \)

(13) \( d := c(y,z,w) : \text{EQ}(y,z) \)

(14) \( e := b(d) : \text{EQ}(z,x) \)

(15) \( f := c(z,x,e) : \text{EQ}(x,z) \)

(16) \( g := b(z,y,d,x,u) : \text{EQ}(x,z) \)

(17) \( h := c(z,x,b(c(y,z,w))) : \text{EQ}(x,z) \)

(18) \( k := b(z,y,c(y,z,w),x,u) : \text{EQ}(x,z) \)

(19) \( l := b(z,x,b(b(y,z,w,z,a(z))),x,a) : \text{EQ}(x,z) \)

(20) \( m := b(z,y,b(y,z,w,z,a(z)),x,u) : \text{EQ}(x,z) \)

**Interpretation.** If \( \equiv \) satisfies (i) \( x \equiv x \) and (ii) \( \frac{x \equiv y}{\equiv z} \) then it satisfies (iii) \( \frac{\equiv y}{\equiv z} \) and (iv) \( \frac{\equiv z}{\equiv y} \).

**Proof.** Replacing \( x \) by \( y \) in (i) we get \( x \equiv y \) (This is expressed by \( m \)). Replacing \( z \) by \( y \) in (ii) we get \( \frac{x \equiv y}{\equiv z} \) (this is expressed by \( c \), which uses \( b \) and \( m \)).

In order to prove (iv) we assume \( x \equiv y \) and \( y \equiv z \), and our goal is \( x \equiv z \). We have two different proofs. In the first one we start by turning \( y \equiv z \) into \( z \equiv y \) by application of (iii) (updating \( c \)), then we apply (iv) (updating now just means fulfilling the assumption \( v \) by the proof \( d \)). So we have reached \( z \equiv x \), and we have to apply an updated version of (ii) in order to get to the goal \( x \equiv z \).

The second proof starts by writing (ii) in the form \( \frac{x \equiv y}{\equiv z} \) (just interchanging \( x \) and \( z \)) and this time we have to appeal to \( d \) only once.

The proofs \( f \) and \( g \) are essentially different. We evaluate their normal forms. First, by elimination of \( d \) and \( e \), \( h \) and \( k \) are obtained from \( f \) and \( g \), respectively. Eliminating \( c \) we get \( t \) and \( m \) as normal forms of \( f \) and \( g \), respectively.
NATURAL DEDUCTION

was NOT a new idea.

On the contrary:

it was the way people reasoned

before it was tried to explain reasoning

by means of the

ALGEBRA of TRUTH VALUES.

Boolean logic is

metatheory of classical reasoning.

NOT reasoning itself.
<table>
<thead>
<tr>
<th>Landau &amp; Jutting</th>
<th>Grundlagen der Analysis 1927</th>
</tr>
</thead>
<tbody>
<tr>
<td>L. S. v. Benthem Jutting</td>
<td>Translation into Automath</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number Char.</th>
<th>10^5</th>
<th>10^6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing Time</td>
<td>60 d.</td>
<td>600 d.</td>
</tr>
<tr>
<td>Reading Time</td>
<td>2 d.</td>
<td>1/10 d.</td>
</tr>
</tbody>
</table>

- 1180 KB
- 2 sec.

F. Wiedijk in 21st Century

Burrough's Algol Computer 1970-1975

As powerful as 1990 PCs.
<table>
<thead>
<tr>
<th>Name</th>
<th>Years</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Udding</td>
<td>1979</td>
<td>620 KB, 300 d. All in direct definition of reals.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural Numbers</td>
<td>- ± ×</td>
</tr>
<tr>
<td>Integers</td>
<td>- ± x</td>
</tr>
<tr>
<td>Rationals</td>
<td>- ± x: Real numbers</td>
</tr>
<tr>
<td>Complex Numbers</td>
<td>- ± x: Complex numbers</td>
</tr>
</tbody>
</table>

**Mathematician**: Mathematician

**Logician**: Logician

**Computer Scientist**: Computer Scientist
GENIUS

BRILLIANT MATHEMATICIAN

PUBLICATION

COMPETENT MATHEMATICIAN

DETAILS

BEGINNER

MATH. VERNAC.

BEGINNER + COMPUTER

AUT. BOOK

COMPUTER

DEAD, BUT ABSOLUTELY CORRECT

HERE LIES MATH
The AUTOMATH RESTAURANT

In one and the same restaurant
one can eat fish or meat,
eat vegetarian or kosher
or just have a few drinks,
smoking or non-smoking.

In one and the same Automath book:
classical or intuitionistic mathematics,
constructive or non-constructive,
with or without Cantor’s paradise.

Protected against clash of
conflicting axioms.
THE RESTAURANT’S FLEXIBLE UNIVERSE

Once we admit the type \( Z \) we have admitted

\[ Z, Z^2, Z^{2^2}, \ldots \]

but not their union.

We can always create types, saying: ”Let \( X \) be a type” (without demanding it to fit in the above universe), so we can, if we insist, introduce BIG types by means of axioms.
SOME LATER DEVELOPMENTS


1978   AUT-QE-NTI (weaker than AUT, leaving as much as possible to the user).

1978   The segment calculus \( \Lambda \Sigma \).

1980   Formalized mathematical vernacular (MV).


1983   Program semantics in space and time.

1984   Geometric constructions.

1986   H. Balsters’ theory on \( \Lambda \Sigma \).

1987   \( \Lambda \Delta \) (a very simple vehicle for all systems with internalized definitions).


The Automath project generated a large number of reports, electronic copies of which are being made now (January 2004).