

## On the use of bound variables in Automath

***Citation for published version (APA):***

de Bruijn, N. G. (1970). *On the use of bound variables in Automath*. Technische Hogeschool Eindhoven.

***Document status and date:***

Published: 01/01/1970

***Document Version:***

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

***Please check the document version of this publication:***

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

***General rights***

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

***Take down policy***

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

On the use of bound variables in AUTOMATH.

by N.G. de Bruijn.

The basic syntax of AUTOMATH was described in a little note [2]. The syntax of its extension AUT-QE was not described there, but it can be obtained by just replacing the rule

$$\langle \text{category} \rangle ::= \underline{\text{type}} \mid \langle \text{expression} \rangle$$

by

$$\langle \text{category} \rangle ::= \langle \text{expression} \rangle \mid \underline{\text{type}} \mid [ \langle \text{dummy variable} \rangle, \langle \text{expression} \rangle ] \langle \text{category} \rangle$$

We shall now explain how to distinguish free and bound occurrences of dummy variables. To facilitate our discussion, we define

$$\langle \text{symbol} \rangle ::= \langle \text{constant} \rangle \mid \langle \text{variable} \rangle \mid \langle \text{dummy variable} \rangle \mid ( \mid ) \mid [ \mid ] \mid \{ \mid \}$$

and we note that each expression can be described as a string, where each entry in the string is a copy of one of the symbols. We speak of "copy of x" rather than "x" in order to be able to distinguish between different occurrences. We shall also refer to these copies as occurrences of that dummy in that expression.

Moreover we shall use capital Greek letters (like  $\Sigma$ ,  $\Lambda$ , ...) as metalingual symbols. They stand for expressions we do not wish to write down themselves.

An occurrence of a dummy variable is called binding in the expression  $\Sigma$ , if it immediately follows a copy of [ .

To each expression  $\Sigma$ , and to each binding occurrence of any dummy, we shall define a set of occurrences which will be called bound (in  $\Sigma$ ) by that binding occurrence. This will be done in such a way that each copy of a dummy is bound by at most one binding copy of that dummy; if it is not binding and not bound by any binding copy, it will be called free in  $\Sigma$ .

To start with an example (where we use letters s, t for dummies, x, y for variables, a, b for constants):

$$[s, a(s,t)] \{ [s,t] a(t, [s, b(s,x,y)]s, [t,y]s) \} a(t)$$

is an expression. There are seven occurrences of s (to be called the first, ..., seventh, reading from left to right). The first, third and fourth are binding. The first s binds nothing, the third s binds the fifth and the seventh, the fourth s binds the sixth. Only the second s is free.

As far as t is concerned: the first three are free, the fourth is binding but binds nothing, the fifth is free.

We shall define the relation between binding and bound dummies by recursion, taking the rule

$$\begin{aligned} \langle \text{expression} \rangle ::= & \langle \text{constant} \rangle \mid \langle \text{variable} \rangle \mid \langle \text{constant} \rangle ( \langle \text{expression string} \rangle ) \\ & \mid \langle \text{dummy variable} \rangle \mid \{ \langle \text{expression} \rangle \} \langle \text{expression} \rangle \mid \\ & [ \langle \text{dummy variable} \rangle , \langle \text{expression} \rangle ] \langle \text{expression} \rangle \end{aligned}$$

as a basis. This rule gives 6 ways to make an expression, and for each way we have to describe the relation.

If the expression is of the form  $\langle \text{constant} \rangle$  or  $\langle \text{variable} \rangle$ , we have nothing to define. If  $\Sigma$  is of the form  $\langle \text{constant} \rangle ( \langle \text{expression string} \rangle )$  we note that each binding occurrence of a dummy is situated in one of the expressions  $\Sigma_1, \dots, \Sigma_k$  this expression string consists of, and we agree that such an occurrence binds in  $\Sigma$  exactly those copies of dummies it was already binding in the  $\Sigma_i$  it belonged to.

A similar arrangement is made for the case  $\{ \langle \text{expression} \rangle \} \langle \text{expression} \rangle$ . Here we have just two constituents  $\Sigma_1, \Sigma_2$ , which form  $\{ \Sigma_1 \} \Sigma_2$ . A binding copy of a dummy in  $\Sigma_1$  binds just the copies of dummies it did bind already in  $\Sigma_1$ , and a binding copy of a dummy in  $\Sigma_2$  binds just the copies of dummies it did bind already in  $\Sigma_2$ .

If the expression is of the form  $\langle \text{dummy variable} \rangle$ , we consider that single occurrence as free.

If the expression  $\Sigma$  is of the form  $[ \langle \text{dummy variable} \rangle , \langle \text{expression} \rangle ] \langle \text{expression} \rangle$ , we agree on the following.

Write it as  $[x, \Sigma_1] \Sigma_2$ . The first  $x$  is binding, and we agree that in  $\Sigma$  it binds all occurrences of  $x$  in  $\Sigma_2$  that were free in  $\Sigma_2$ , and no others. For the rest, a binding dummy in  $\Sigma_1$  binds (in  $\Sigma$ ) just those occurrences of dummies it did bind already in  $\Sigma_1$ , and a binding dummy in  $\Sigma_2$  binds (in  $\Sigma$ ) just those occurrences of dummies it did bind already in  $\Sigma_2$ .

The above definition of binding is not the one of the report [1], but there is no objection to change the system of [1] into the present one. With the present definition it is easier to handle expressions: we are not so often under the obligation to change names for dummy variables.

#### References.

1. N.G. de Bruijn, Automath, a language for mathematics. THE-report 68-WSK-05 (1968) Technological University, Eindhoven.
2. ————— The syntax of PAL and AUTOMATH, Techn. Univ. Eindhoven, Internal Report, Notitie 32, 9 April 1970.