On the use of bound variables in Automath

Technische Hogeschool, Eindhoven.
Onderafdeling der Wiskunde.
Notitie 9 , d.d. 26 november 1970.

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

<category> ::= type | <expression>

by

<category> ::= <expression> | type | [<dummy variable>, <expression>] <category>

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

<symbol> ::= <constant> | <variable> | <dummy variable> |(|)|,|[|]|{|}

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 occurences. We shall also refer to these copies as occurrences of that dummy in that expression.

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

An occurence of a dummy variable is called binding in the expression Σ, if it immediately follows a copy of [.

To each expression Σ, and to each binding occurence of any dummy, we shall define a set of occurences which will be called bound (in Σ) by that binding occurence. 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 Σ.

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 occurences 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

<expression> ::= <constant> | <variable> | <constant> (<expression string>)

| <dummy variable> | {<expression>} <expression> |

[<dummy variable> , <expression>] <expression>

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 <constant> or <variable>, we have nothing to define. If Σ is of the form <constant> (<expression string>) we note that eacht binding occurrence of a dummy is situated in one of the expressions $\Sigma_1, \ldots, \Sigma_k$ this expression string consists of, and we agree that such an occurrence binds in Σ exactly those copies of dummies it was already binding in the Σ_i it belonged to.

A similar arrangement is made for the case {<expression>} <expression>. Here we have just two constituents Σ_1, Σ_2, which form $\{\Sigma_1\}\Sigma_2$. A binding copy of a dummy in Σ_1 binds just the copies of dummies it did bind already in Σ_1, and a binding copy of a dummy in Σ_2 binds just the copies of dummies it did bind already in Σ_2.

If the expression is of the form <dummy variable>, we consider that single occurrence as free.

If the expression Σ is of the form [<dummy variable> , <expression>] <expression>, we agree on the following.

Write it as $[x, \Sigma_1] \Sigma_2$. The first x is binding, and we agree that in Σ it binds all occurrences of x in Σ_2 that were free in Σ_2, and no others. For the rest, a binding dummy in Σ_1 binds (in Σ) just those occurrences of dummies it did bind already in Σ_1, and a binding dummy in Σ_2 binds (in Σ) just those occurrences of dummies it did bind already in Σ_2.

The above definition of binding is <u>not</u> 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.