EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics

Memorandum 78-04.
Issued May 1978.

# AUT-QE WITHOUT TYPE INCLUSION.

by

N.G. de Bruijn.

Eindhoven University of Technology
Department of Mathematics
P.O.Box 513
5600 MB EINDHOVEN
The Netherlands.

# AUT-QE WITHOUT TYPE INCLUSION.

by

N.G. de Bruijn.

1. Introduction. We consider a language AUT-QE-NTI. Its definition is identical to the one of AUT-QE in [3] but for the fact that the type inclusion rule (rule 6 of section 5.5.4 of [3]) is omitted. The letters NTI stand for "no type inclusion".

The power of this language can be increased in various ways; e.g. (i) by admitting "mock typing" for writing axiom schemes (cf. [1]), (ii) adding the type inclusion of AUT-QE with about the same effect, (iii) taking the $\Pi$-operators of AUT-$\Pi$ (cf. [2]). The latter language has some features that seem peculiar from the point of view of language structure, e.g. if compared to the simplicity of AUT-QE-NTI, but, on the other hand, AUT-$\Pi$ seems to be quite natural from the user's point of view.

In this note we shall try to show that in AUT-QE-NTI, a set of axioms on universal quantifications can lead to a set of theorems which can take over the role of type inclusion. Application of these theorems is a kind of automatic affair. It is to be expected that AUT-QE-NTI, enriched with such automatic devices (which means that some "book theorems" are shifted to the language definition) has about the same expressive power as AUT-QE and AUT-$\Pi$.

2. The rules of AUT-QE-NTI. In [2] we described AUT-$\Pi$ by a set of basic rules, omitting everything the AUTOMATH family has in common (like structure of lines, books, contexts, instantiation). We describe AUT-QE-NTI in the same fashion ($\tau$ stands for either type or prop):

(i) $\vdash^1 \tau$

(ii) $$\frac{\vdash^2 \alpha : \tau \qquad (x{:}\alpha) \vdash^1 P}{\vdash^1 [x{:}\alpha]P}$$

(iii) $$\frac{\vdash^2 \alpha:\tau \qquad (x:\alpha) \vdash^2 Q : P}{\vdash^2 [x:\alpha]Q : [x:\alpha]P}$$

(iv) $$\frac{\vdash^2 \alpha:\tau \qquad (x:\alpha) \vdash^3 R : Q}{\vdash^3 [x:\alpha]R : [x:\alpha]Q}$$

(v) $$\frac{\vdash^3 A : \alpha : \tau \qquad \vdash^2 Q : [x:\alpha]P}{\vdash^2 \{A\}\ Q : [\![x/A]\!]\ P}$$

(vi) $$\frac{\vdash^3 A : \alpha : \tau \qquad \vdash^3 R : Q : [x : \alpha]P}{\vdash^3 \{A\}\ R : \{A\}\ Q}$$

3. The axioms for universal quantification

| | | | |
|---|---|---|---|
| | $\alpha$ := | ——— | : $\tau$ |
| \| | $\mathcal{Q}$ := | ——— | : $[x:\alpha]\tau$ |
| \| | All := | PN | : $\tau$ |
| \| \| | x := | ——— | : All(Q) |
| \| | Ax 1 := | PN | : Q |
| \| \| | y := | ——— | : Q |
| \| | Ax 2 := | PN | : All(Q) |

Properly speaking, we have to consider various sets of axioms: the $\tau$ in the first line may be either type or prop, and the $\tau$'s of the second and third lines may be either both type or both prop.

We do not go into these difficulties, and we shall behave as if $\tau$ were the only basic expression of degree 1.

4. Generalization of All to multiple quantification. We consider cases like $Q : [x_1 : \alpha_1] \ldots [x_m : \alpha_m]\ \tau$. We cannot introduce $\alpha_1,\ldots,\alpha_m$ independently: $\alpha_2$ may depend on $x_1$, etc. But in order to give a preliminary idea, we show what can be done if the $\alpha$'s are independent. We take m=3.

| | |
|---|---|
| Q | := ——— : $[x_1 : \alpha_1]\ [x_2 : \alpha_2]\ [x_3 : \alpha_3]\ \tau$ |
| $k_{31}$ | := $[x_1 : \alpha_1][\ x_2 : \alpha_2]$ All$(\alpha_3,\{x_2\}\{x_1\}Q)$ : $[x_1 : \alpha_1][x_2 : \alpha_2]\tau$ |
| $k_{32}$ | := $[x_1 : \alpha_1]$ All$(\alpha_2,[x_2 : \alpha_2]$ All$(\alpha_3, \{x_2\}\{x_1\}Q))$ : $[x_1 : \alpha_1]\tau$ |
| $k_{33}$ | := All$(\alpha_1, [x_1 : \alpha_1]$ All $(\alpha_2, [x_2 : \alpha_2]$ All$(\alpha_3, \{x_2\}\{x_1\}Q)))$ : $\tau$ |

These formulas are the same as those presented in [2] : in AUT-Π the role of All is played by Π, which is not introduced by means of axioms, but by a language rule).

Note how the $k_{mi}$'s imitate type inclusion. If

$$Q : [x_1 : \alpha_1]\,[x_2 : \alpha_2]\,[x_3 : \alpha_3]\tau$$

then a language with type inclusion permits us to write $Q:[x_1:\alpha_1]\tau$. Here in AUT-QE-NTI we just have to write $k_{32}(Q) : [x_1 : \alpha_1]\tau$.

We now turn to the general case. Let m be an integer > 1. We introduce $\phi_1,\ldots,\phi_m$ as follows (the example shows the case m=4)

$$\begin{aligned}
&\phi_1 := \text{———} : \tau\\
&\phi_2 := \text{———} : [x_1 : \phi_1]\tau\\
&\phi_3 := \text{———} : [x_1 : \phi_1]\,[x_2 : \{x_1\}\,\phi_2]\,\tau\\
&\phi_4 := \text{———} : [x_1 : \phi_1]\,[x_2 : \{x_1\}\,\phi_2][x_3 : \{x_2\}\{x_1\}\phi_3]\tau
\end{aligned}$$

Instead of the independent types $\alpha_1,\ldots,\alpha_4$ we now have, in the context of $\phi_4$, the types $\phi_1,\{x_1\}\phi_2$, $\{x_2\}\{x_1\}\phi_3,\{x_3\}\{x_2\}\{x_1\}\phi_4$. When writing in this context, we shall introduce as "typographical" abbreviations:

$$\langle 1\rangle = \{x_1\}, \qquad \langle 2\rangle = \{x_2\}\{x_1\}, \ \ldots\ , \ \langle j\rangle = \{x_j\}\ \ldots\ \{x_1\}$$

if j is any integer > 0; <0> will stand for an empty string. Similarly

$$[j] = [x_1 : \phi_1]\,[x_2 : \{x_1\}\phi_2]\ \ldots\ [x_j : \{x_{j-1}\}\ \ldots\ \{x_1\}\phi_j]$$

and [0] stands for the empty string.

In the context of $\phi_m$ we now write as correct lines

$$\begin{aligned}
&k_{m\,0} := Q && : [m]\tau\\
&k_{m\,1} := [m-1]\ \mathrm{All}(\langle m-1\rangle\phi_m,\ \langle m-1\rangle\ k_{m\,0}) && : [m-1]\tau\\
&k_{m\,2} := [m-2]\ \mathrm{All}(\langle m-2\rangle\phi_{m-1},\ \langle m-2\rangle\ k_{m\,1}) && : [m-2]\tau\\
&\qquad\qquad \ldots\ldots\ldots\\
&k_{m\,m} := \mathrm{All}(\phi_1, k_{m\,m-1}) && : \tau
\end{aligned}$$

The recursion is produced by

$$k_{m\,i} := [m-i]\ \mathrm{All}(\langle m-i\rangle\ \phi_{m-i+1},\ \langle m-i\rangle\ k_{m\,i-1}) : [m-i]\tau$$

for $0 < i \leq m$.

5. Generalization of Ax1 and Ax2. Ax1 expresses something for the case m=1. There it says: if we have something of type $k_{11}$(=A11(Q)) then we have something of type $k_{10}$(=Q). Ax2 does it the other way round. We can generalize this to all $m \geq 1$: if $0 \leq i \leq m$, $0 \leq j \leq m$, and if we have something of type $k_{m\,i}$ then we have something of type $k_{m\,j}$. It suffices to deal with the cases $j=i\pm1$.

We start in the context of $\phi_m$ (of section 4). If $1 \leq i \leq m$ we can write (by virtue of $\eta$-reduction rules)

$$\left|\begin{array}{l} u := \text{————————} : k_{m\,i} \\ s_{m-i} := [m-i]\ \mathrm{Ax1}\ (\langle m-i\rangle\ \phi_{m-i+1},\ \langle m-i\rangle\ k_{m\,i-1},\ \langle m-i\rangle u) \quad : k_{m\,i-1} \end{array}\right.$$

In particular if i=m this takes the form

$$\left|\begin{array}{l} u := \text{————————} : k_{m\,m} \\ s_o := \mathrm{Ax1}(\phi_1, k_{m,m-1}, u) : k_{m\,m-1}. \end{array}\right.$$

In the other direction we have, if $1 \leq i < m-1$,

$$\left|\begin{array}{l} u := \text{————————} : k_{m\,i-1} \\ t_{m-i} := [m-i]\ \mathrm{Ax2}(\langle m-i\rangle \phi_{m-i+1},\ \langle m-i\rangle\ k_{m\,i-1}, \langle m-i\rangle u) : k_{m\,i} \end{array}\right.$$

In particular if i=m this becomes

$$\left|\begin{array}{l} u := \text{————————} : k_{m\,m-1} \\ t_o := \mathrm{Ax2}(\phi_1, k_{m\,m-1}, u) : k_{m\,m} \end{array}\right.$$

By means of composition of these operations we can pass indeed from any $k_{m\,i}$ to any $k_{m\,j}$:

$$\left|\begin{array}{l} u := \text{————————} : k_{m\,i} \\ t_{m\,i\,j} := \quad \ldots \quad : k_{m\,j} \end{array}\right.$$

where for each set of integers m,i,j ($0 \leq i \leq m$, $0 \leq j \leq m$) the dots stand for a particular expression. The larger m and $|i-j|$ are, the longer this expression will become.

In partcular, we can pass from

R : Q : [m]τ

to

$$t_{m\ 0\ j}(R) : k_{m\ j}(Q) : [m-j]\tau.$$

6. Equality. It is attractive to consider the Ax1 and Ax2 of section 3 as inverse operations. This means that in the context of Q the expressions

$$x \quad \text{and} \quad Ax2(Ax1(x))$$

refer to the same object (of type A11(Q)), and

$$y \quad \text{and} \quad Ax1(Ax2(y))$$

refer to the same object (of type Q). We can express this by equality axioms for which we need further PN's).

The applications we have in mind, suggest that identification of x and Ax2(Ax1(x)), and of y and Ax1(Ax2(y)) is of a linguistic rather than of a mathematical nature. So it is attractive to accept the above equalities as definitional equalities (thus extending the notion of definitional equality). If we accept this, we conclude that the $s_i$ and $t_i$ (of section 5) are each other's inverses in the same definitional equality sense, and similarly $t_{m\ i\ j}$ and $t_{m\ j\ i}$ are each other's inverses.

7. Omitting the $k_{mi}$'s. The category of $k_{m\ i}(Q)$ (section 4) is $[m-i]\tau$. On the basis of what we know on uniqueness of types we can remark that once the category of $k_{m\ i}(Q)$ is given, then i is known. This can lead us to the following convention:

$$k_{m\ i}(Q) : [m-i]\tau$$

is to be abbreviated as

$$Q : [m-i]\tau.$$

From the right-hand side we see that the Q on the left is intended to denote $k_{m\ i}(Q)$.

Another convention is that we omit the $t_{m\ i\ j}$'s (see the last line of section 5): from the typing it is obvious which ones they should have been (provided we have the definitional equalities of section 6).

These funny conventions are exactly equivalent to the rules of type inclusion! That is, they turn AUT-QE-NTI into AUT-QE.

It should we remarked that AUT-QE is very convenient for writing and checking. Compared to AUT-QE-NTI it must be said that AUT-QE saves a large number of $\beta$- and $\eta$-reductions and similar definitional equivalences.

8. Abstraction index. In order to facilitate the discussion on the relation between the various languages we coin the term "abstraction index" of an expression in an AUT-QE-NTI book.

In a context containing the context of the $\phi_1,\ldots,\phi_n$ of section 4, and with the abbreviation of section 4, we say that the abstraction index of $[m]\tau$ is m. Also, if $Q : [m]\tau$ and if $R : Q : [m]\tau$ we say that the abstraction index of Q and R is m. The abstraction index increases by 1 if we put a single abstractor $[x : \alpha]$ in front, and decreases by 1 if we put a single applicator $\{A\}$ in front (the latter only applies to Q and R, not to $[m]\tau$ itself).

9. Comparing AUT-QE-NTI to AUT-$\Pi$. Let us take AUT-QE-NTI with extended definitional equality as described in section 6. A book in that language my contain expressions of degrees 1,2,3 and arbitrary abstraction index. It is not hard to see that we can rewrite such a book in an equivalent form in which all identifiers of degree 3 have abtraction index 0. We replace any block opener line x := —— : Q (where Q has degree 2 and abstraction index m) by a line x := —— $k_{m\,m}(Q)$. Furthermore we replace definitional lines c := R : Q by c := $t_{m\,o\,m}(R) : k_{m\,m}(Q)$. Because of these changes there have to be obvious adaptions in all places where such an x or such a c is referred to.

Let us call a book with this property (all identifiers of degree 3 have abstraction index 0) a "zero abstraction index book". Such a book may very well contain subexpressions of degree 3 and abstraction index > 0, just becasue the index of an expression increases if we put an abstractor in front of it.

A zero abstraction index book is essentially an AUT-$\Pi$ book. There is a peculiarity with abstraction and application in AUT-$\Pi$, however. In AUT-$\Pi$ all expressions of degree 3 have abstraction index 0 : if $R : Q$ then $Q : \tau$. If we translate AUT-$\Pi$ into AUT-QE-NTI we have to do the following: every abstractor $[x : \alpha]$ of AUT-$\Pi$ has to be replaced by $t_{1\,0\,1}\,[x : \alpha]$, and every applicator $\{A\}$ of AUT-$\Pi$ has to be replaced by $\{A\}\,t_{1\,1\,0}$.

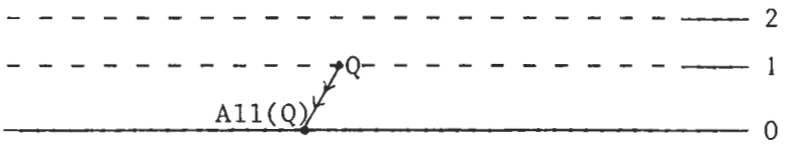
One might naturally ask whether, conversely, AUT-QE-NTI can be interpreted in terms of AUT-$\Pi$. A convenient way to do this seems to be the following one. First rewrite all AUT-$\Pi$-expressions of degree 3, replacing { } by $\{\ \}_\pi$ and [ ] by $[\ ]_\pi$. Next extend the collection of all expressions of degree 3 by putting

[ ]'s (without $\pi$) in front of other expressions and provide them with a type according to a rule "if R : Q then [] R : []Q". In this way the typing of expressions of higher index are interpreted as typings of expressions of lower index. Needless to say, quite some work has yet to be done!

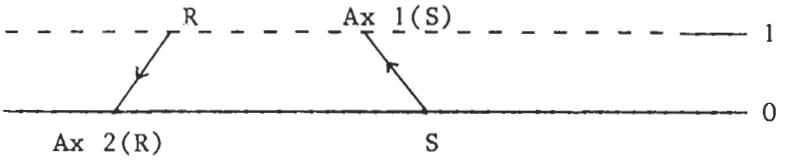
10. Schematic presentation. In order to get a quick survey of the various operations discussed thus far, we present expressions as points in a diagram. In such a diagram we have horizontal levels 0,1,2,..., according to the abstraction index.

If A is an expression then [x : $\alpha$]A is drawn one level higher, with a vertical arrow connecting the two, and similarly an application is drawn one level lower:

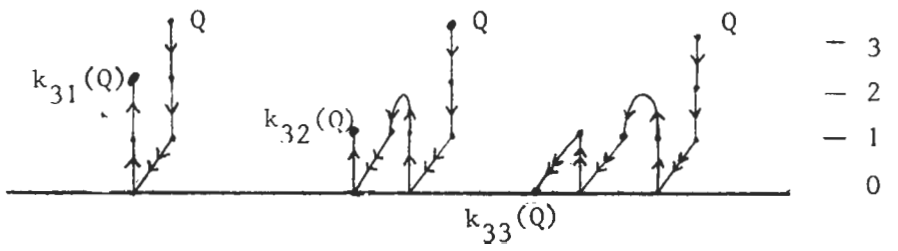

If Q has degree 2 and abstraction index 1 we can apply All, and this is indicated as follows



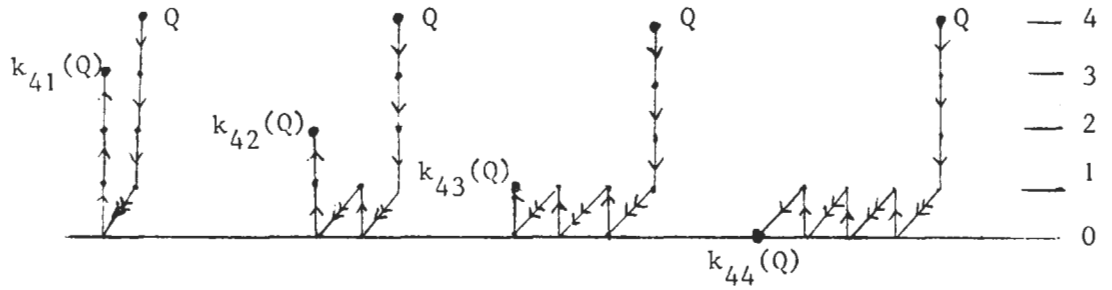

If R has degree 3 and abstraction index 1 we have Ax 2(R) with index 0; if S has degree 3 and abstraction index 0 we have Ax 1(R) with index 0, provided that S : All(Q) with some Q of index 1. These operations are indicated as follows



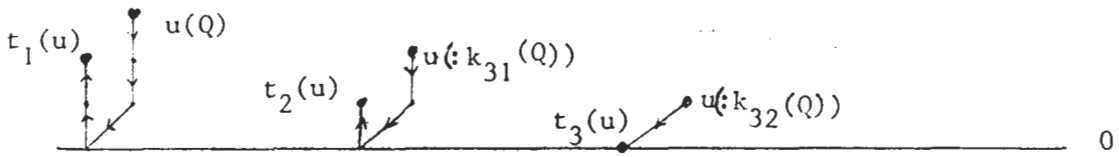

Presenting composite operations we proceed from right to left. Let Q have index 3, then $k_{3\,1}(Q)$, $k_{3\,2}(Q)$ and $k_{3\,3}(Q)$ are depicted like this

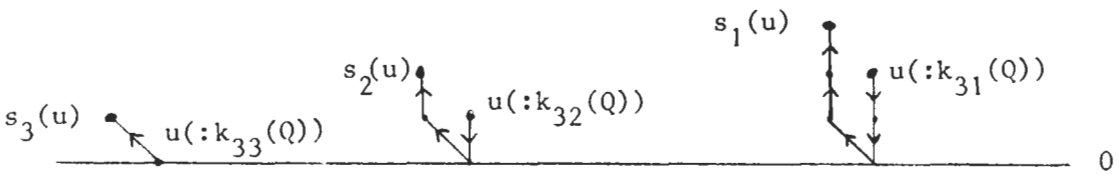


Note that in these cases the detours depicting { } [ ] can be dissolved by β-reduction, whence we get (now taking m=4, for a change)

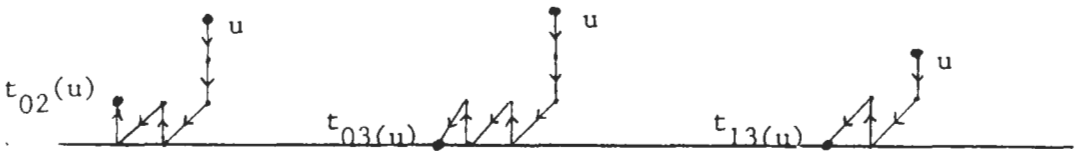


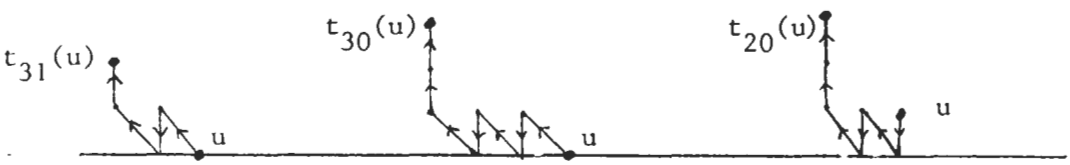

Next we take the case if degree 3. Let Q have index 3, and let R:Q. The operations of section 5 can be depicted as follows




The composite operations $t_{m\,i\,j}$ can be drawn accordingly. Note that some detours { } [ ] can be eliminated by β-reduction and detours [ ] { } by η-reduction. Some examples are




At the end of section 6 we mentioned definitional equivalences. In the pictures they mean that the detours

can be eliminated.

At the end of section 9 we mentioned that for expressions of degree 3 the abstractors and applicators in AUT-Π have to be adjusted when translating into AUT-QE-NTI. Denoting them by $[\ ]_\pi$ and $\{\ \}_\pi$ we have


In this way AUT-Π manages to keep the abstraction index zero for all expressions of degree 3.

References.

[1] N.G. de Bruijn: A framework for the description of a number of members of the AUTOMATH family. Memorandum 74-08. Issued June 1974. Eindhoven University of Technology, Dept. of Mathematics, Eindhoven, The Netherlands.

[2] N.G. de Bruijn: Some auxiliary operators in AUT-Π. Memorandum 1977-15. Issued November 1977. Eindhoven University of Technology, Department of Mathematics, Eindhoven, The Netherlands.

[3] D.T.v.Daalen: A description of AUTOMATH and some aspects of its language theory.Proceedings of the Symposium APLASM. Vol. I, ed. P. Braffort, Orsay, France (Dec. 1973) reprinted in: L.S.v. Benthem Jutting: Checking "Landau's "Grundlagen" in the AUTOMATH system. Thesis, 1977, 120 pages. Also to appear in Mathematical Centre Tracts, Mathematical Centre, Amsterdam.