Set theory with type restrictions

Set Theory with Type Restrictions

N.G. de Bruijn

1. It has been stated and it has been believed throughout this century that set theory is the basis of all mathematics. Usually (but not always) people think of the Cantor set theory, with some formalization like the one of Zermelo-Fraenkel. It describes a universe of things called sets, and everything discussed in mathematics is somehow interpreted in this universe.

2. It seems, however, that there is a revolt. Some people have begun to dislike the doctrine "everything is a set", just at the moment that educational modernizers have pushed set theory even into kindergarten. It is not unthinkable that this educational innovation is one of the things that make others try to get rid of set theory's grip for absolute power.

Mostowski is reputed to have claimed a counterexample by declaring "I am not a set". At the present state of science it seems to be impossible to find out whether this statement is true or false. Anyway, there is no safe ground for saying that everything is a set. Let us try to be more modest and say: "very many things can be *coded* as sets". For example, Beethoven's 9-th symphony can be coded as a set. But the coding is quite arbitrary, and we are not sure that nothing gets lost in the coding. To quote a more mathematical example: Gauss' construction of the regular 17-gon may be interpretable as a set, but again such an interpretation is quite arbitrary and does not seem to be illuminating. An expression like "the intersection of the set of even integers with the set of all constructions of the 17-gon" makes sense only *after* the codings have been stated.

Sets have become a very important part of our language. Until 1950 many rigorous texts on mathematical analysis were written with little or no use of the language and notation of sets. This has changed considerably, but quite often the change is very superficial. It is superficial as long as it is nothing but a translation from predicates to sets. One of the reasons for this translation may be that there is a vague opinion that a set is a mathematical object and a predicate is not. Accordingly, it is felt that someone who makes assumptions and proves theorems about predicates is a logician and not a mathematician.

Nevertheless, there still remains a tremendous use for sets in mathematics. Sets are here to stay, and we have to ask what kind of set theory we should adhere to. The question which set theory is the true set theory, is not a true question, of course. It is all a matter of taste: relevant things are whether a theory is beautiful, economic, powerful, easy to manipulate, natural, easy to explain, etc. The fact that the Cantor-Zermelo-Fraenkel theory is interesting, correct, rich and deep, does not imply that it is necessarily the tool that should be available for every mathematician's use. It has some disadvantages too. One is that it makes the foundation of mathematics rather hard for the non-specialist. We have the sad situation that late in the 20th century the average ordinary mathematician has rather vague ideas about the foundation of his science. Another unpopular feature in Cantor set theory is the admission of $x \in x$, which seems to be rather far away from possible interpretations.

3. The natural, intuitive way to think of a set, is to collect things that belong to a class or type given beforehand. In this way one can try to get theories that stay quite close to their interpretations, that exclude $x \in x$ and are yet rich enough for everyday mathematics. Some of these theories may exclude large parts of the interesting, funny paradise of Cantor's set theory which has been explored by so many expert mathematicians. For a survey of various type theories we refer to [Fraenkel et al. 58].

4. In this paper we shall try to make a plea for a kind of type theory where the use of types is very similar to the rôle of types in cases where the objects to be discussed are not sets.

Let us first note that natural languages are confusing when dealing with types. The word "is" is used for too many things. We say "5 is a number", "5 is the sum of 1 and 5", "5 is the sum of two squares". It is only in the first sentence that "is" can refer to a type. We shall use the symbol E for this: 5 E number. We shall call such a formula a *typing*.

5. We think of a type theory where the type of an object is unique. If A E B then B is completely determined by A. This seems to drift us away from the idea that B is something like a set and that A is a member of B, and we have to be careful not to confuse the typing symbol E with the membership symbol $\in$, although there is a conceptual similarity.

We of course run into circumstances where we want to say that our number 5 is also a complex number: 5 E complex number. We have to make the distinction between the real number 5 and the complex number 5 in order to maintain that in A E B the B is completely determined by A. It is a bit awkward, we have to talk a great deal about identification and embedding (but in de Cantor-

Zermelo-Fraenkel theory this is not any better). Yet it should be done; let us not forget that most mathematicians would hesitate to identify the real number 5 with the 2×2 matrix $\begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}$, and the latter situation is really not very much different from the one with the complex numbers.

6. Let us first explain some other cases where E plays a rôle. If B is a theorem and if A is a proof for B, then we can write $A \mathrel{\mathtt{E}} B$. The theorem can have several proofs, but a proof proves just one theorem. Another example: Let B be a statement of constructibility of some geometrical figure that can be constructed by means of ruler and compass, and let A be a description of one of the constructions. We again write $A \mathrel{\mathtt{E}} B$. There can be several different A's to a given B, but if the construction A is given, there is no doubt about what it constructs. A third example: Let A be a computer program and let B be a description of what the execution of the program achieves (i.e., A describes the syntax and B the semantics).

In all these cases the A's and B's may depend on several variables (of certain types), and the results $A \mathrel{\mathtt{E}} B$ may be transformed into other results $A' \mathrel{\mathtt{E}} B'$ by means of substitution.

Moreover, in all these cases there is a possibility to introduce a name for a thing in B if we do not actually have that thing. There are two ways for this.

(i) The thing can be introduced as something primitive and fixed. For example Peano's first axiom says that there is a natural number to be denoted by 1, and nothing is assumed about it at that stage. An example with a different interpretation is that B is a proposition and we say that its truth is assumed, i.e. that B is an axiom. From then on, B plays the same rôle as a theorem: we act as if we have a proof, i.e. we have something $\mathrel{\mathtt{E}} B$ that we do not wish to describe. Let us now look at the case of geometrical constructions. We want to express that the possibility to connect the distinct points P and Q by a straight line is a primitive construction, i.e. a construction that cannot be described in terms of simpler constructions. That is, we act as if we have a fixed thing $\mathrel{\mathtt{E}} B$, where B is a statement of constructibility.

(ii) The thing can be introduced as a variable. Its validity is restricted to a piece of text (a "block") that is opened by the introduction of the variable; that is why we call it a block opener. The variable is introduced by stating its type: if its name is x, we write something to the effect of "let $x \mathrel{\mathtt{E}} B$". If B is a type like "number", "point", then this phrase "let $x \mathrel{\mathtt{E}} B$" sounds quite familiar. If B is a statement, however, we may interpret the phrase "let $x \mathrel{\mathtt{E}} B$" as "assume that B is true". That is, we act as if we have a proof for B. (This is not the same thing as the introduction of B as an axiom:

"let x E B" does not reach beyond the block opened by x, and secondly, we can substitute A for x if we later get any proof A for B.) There is a slightly unfamiliar feature: most mathematicians have not got used to giving names to proofs, and here we give names even to would-be-proofs.

7. The parallels between the various interpretations of typings are very strong indeed. The mechanism of substitution is the same for the various interpretations, and actually the various interpretations are happily intermingled. Everything that is said in mathematics is said in a certain context. That context consists of a string of variables (block openers), each one having been introduced as a thing of a certain type. The type of the second variable may depend on the first variable, etc. In such a string some of the variables have to be interpreted as conventional mathematical objects (like numbers, points), others as would-be-proofs for assumptions. The linguistic treatment makes no difference as to the interpretations.

8. The above characteristics are the common root of the mathematical languages of the Automath family [*de Bruijn 70a (A.2)*]. The definitions of these languages hardly contain anything on logic or on the foundation of mathematics. Notions like "truth", "theorem", "proof", "set", "definition", "and", "implies", "inference rule" are either things that can be explained by means of the language (like any other piece of mathematical material) or else they are only meant to emphasize pieces of text to a reader who likes to have a feeling for motivation. To mention an example: a definition, an abbreviation and a theorem have the same linguistic form. It would not be necessary to distinguish between these three, if it were not for the fact that "readability" has something to do with the relation to conventional modes of expressing mathematics.

The languages of the Automath family have the property that books written in these languages can be checked for syntactic correctness by means of a computer. We emphasize that syntactic correctness guarantees that the interpretations of the text are correct mathematics. Note that various sets of the typing symbol E can occur on one and the same piece of text, and therefore we can pursue a kind of unification of mathematical theories.

It is not the right place to go into a complete exposition of these languages, but one thing should be made clear; just as they admit to introduce objects of a given type, and to build new objects by means of old ones, it is equally possible to introduce new types (by way of variables or of primitive notions) and to build new types in terms of old. For this purpose we create the extra symbol `type` and we write things like "number E `type`", "let B E `type`", etc.

9. Having such type languages available as relatively simple tools, we are in-

duced to base mathematics on a type theory where types can be constructed as abundantly as other mathematical objects, i.e., where types may depend on parameters, are defined under certain assumptions only, where types can be introduced as variables or as primitive notions.

10. There are various ways to do set theory in such a system.

One possibility is that we take a primitive type called *SET*, and from then on, we write A E SET for every A which we want to consider as a set. We can write the complete Cantor-Zermelo-Fraenkel theory this way. The relations $A \in B$, $A \subset B$ are relations that have a meaning whenever A E SET and B E SET. There is not the slightest danger to confuse $\in$ and E. The $\in$ is a relational symbol just like any other; it does not occur in the language definition.

There is a second, entirely different way, that implements set theory with types, in the sense of the "5 E number" mentioned before. Now the symbol E means something like $\in$. If B is a type, and if P is a predicate on B, we form the set S of all A with A E B for which $P(A)$ is true. So sets in B correspond to predicates on B. We write $S \sqsubset B$, and we define $\in$ by saying that $A \in S$ means $P(A)$. Quite often we like to consider S as a new temporary universe, i.e. we wish to have $A \in S$ in the form of a formula with an E. To that end we create a type called $OWNTYPE(B, P)$ and a one-to-one mapping of that type onto S. Some of this work can be simplified by special notation we shall not develop here; such notation can be used both for ordinary and for automated reading.

11. In order to work with the predicates mentioned in the previous section, we want some kind of typed lambda calculus. It is roughly this. If B is a type, and if for every x E B we have a formula of the form $A(x)$ E $C(x)$, then we want to write

$$[x : B]\, A(x) \text{ E } [x : B]\, C(x) \ .$$

The left-hand side is the function that sends x into $A(x)$, defined for all x E B; the conventional notation in non-typed lambda calculus is $\lambda_x A(x)$. The right-hand side is slightly unconventional; in the case that $C(x)$ does not depend on x one may think of the class of all mappings that send B into C.

This kind of lambda calculus is part of the language definition, independently of the mathematical axioms we are going to write in our books. So there is a primitive idea of mapping available before sets are discussed. In particular, predicates are such mappings, so if sets are introduced by means of predicates, they already require the lambda calculus. Later, one can show that the concept of a mapping as a subset in a cartesian product is equivalent to the notion of mapping provided by the lambda calculus.

12. Cantor produced his paradise by means of linguistic constructions. (This created considerable controversies in his time, since he did not specify his language.) Now let us see what we get by linguistic constructions in our typed set theory. Assume we introduce (by means of an axiom) the type N of all natural numbers (and we take a set of axioms like Peano's). Then we have, whether we want it or not, subscribed to N^N, to N^{N^N}, etc., since the lambda calculus prescribes that we accept the type of all mappings of N into N, etc. However, it seems (we use the phrase "it seems" since no formal proof has been given thus far) that we cannot form something of the strength of the union

$$(1) \qquad N \cup N^N \cup N^{N^N} \cup \dots .$$

The reason is not that we would not allow ourselves to form the union of a countable number of types. That will be provided, anyway, by an axiom we would not like to live without. The reason is that we are unable to index the sets of the sequence (1) in our language. The indexing we want is $N_1 = N, N_2 = N^{N_1}, N_3 = N^{N_2}, \dots$, and this is in terms of our metalanguage, since it requires a discussion of something like the length of a formula. This is a little detail Cantor never made any trouble about.

The fact that the union (1) is "inaccessible" does not mean that bigger types are forbidden. After all, we can just start saying "let B be any type (i.e. B E type)" and we can make assumptions about B that cannot be satisfied by the types $N, N^N, \dots$.

The world where we have $N, N^N, \dots$, but where (1) is "inaccessible", is a world most mathematicians will doubtlessly find big enough to live in. For those who want to have a bigger world, where they cannot be troubled by people asking for interpretations, there is a simple way out: they just take a type *SET* and provide it with Zermelo-Fraenkel axioms. If they want to have the picture complete, they will not find it hard to embed the types $N, N^N, \dots$ into a small portion of their paradise.

13. Having discarded the idea that every mathematical object is a set, we should be careful not to fall into the next trap. We might like to say that a mathematical object is either some B with B E type or an A with A E B (where B E type). However, the situation can be more complex than this. Let us consider the notion "group" that occurs in the sentence "let G be a group". What we want to say is something like this: assume we have a type A, that we have in A a set B, that in B we have a multiplication rule, that the multiplication is associative, etc. The object we want to handle can be denoted by a string of identifiers $x_1, \dots, x_k$, where x_1 E A_1, x_2 E $A_2, \dots, x_k$ E A_k, but where A_2 may depend on x_1, A_3 on x_1 and x_2, etc. It is not as if the string $A_1, \dots, A_k$ were something type-like, and $x_1, \dots, x_k$ were something chosen in it. Accordingly, we

cannot write let "G be a group" as a single typing "G E group". We can of course create, by means of a set of axioms, a new type "group", but that is a poor remedy: we cannot afford to adopt axioms for every new notion we like to introduce.

14. In Section 10 we compared two different ways to talk about sets by means of typings. The choice between the two has a more general aspect; viz. the question whether we shall or shall not aim at minimal use of typings. The word "minimal" refers to the number of different uses of the typing symbol. In order to say what we mean, we describe a kind of minimal system that seems to be in the spirit of basing mathematics on Zermelo-Fraenkel set theory. In the first place we use typings ... E *SET* (as in Section 10). Secondly we create a type called *BOOL*, and we use "A E *BOOL*" in order to express that A is a proposition. Finally, for every X with X E *BOOL* we create a type called *TRUE*(X), and we use the typing P E *TRUE*(X) for expressing that P is a proof for the truth of X. In this minimal system, the use of typings of the form ... E **type** is restricted to the above-mentioned three instances right at the beginning of the book of mathematics. The author thinks that talking mathematics in such a minimal system is not always the natural thing to do. There is much to be said for a more liberal use of typings, where typings of the form ... E **type** are used throughout the book. Let us consider the geometrical constructions mentioned in Section 6. It seems natural to use A E B for saying that A describes a construction and that B says what has been constructed. Let us say that we have created, for every point P, a type *CONSTR*(P). Hence statement A E B has the form A E *CONSTR*(P). If we want to phrase this in our minimal system, we get something as follows. The point is a set (P E *SET*), and so is some coded form A^* of A (A^* E *SET*). We form a proposition $q(P, A^*)$ (so $q(P, A^*)$ E *BOOL*) that says that A^* is a construction for P. Finally we need a proof S for this proposition, whence we write

$$S \text{ E } TRUE(q(P, A^*))$$

for what was A E *CONSTR*(P) in the liberal system. In the latter case it is not necessary to provide a proof corresponding to S, since the type of A can be determined by a simple algorithm.

This example shows two advantages of liberal use of typings: one is that many unnatural codings can be suppressed, the other one is that a higher degree of automation can be achieved. Yet there are many other advantages of which we mention two:

(i) We are neither forced nor forbidden to introduce the types *SET*, *BOOL* and *TRUE*(X);

(ii) There is a possibility that one and the same piece of text gives rise to various pieces of standard mathematics, just by the use of different interpretations.