The use of justification systems
for integrated semantics

N.G. de Bruijn
Department of Mathematics and Computing Science
Eindhoven University of Technology
PO Box 513, 5600MB Eindhoven, The Netherlands

1. Justification systems

1.1. We use the term justification system for any system that can be used for the complete verification of large coherent parts of mathematics and similar formal material (like logic, computer science, and formal constructions in general). Fundamental characteristics of such systems are (i) that what is written in the system has a clear interpretation in the outside world that is to be described, and (ii) that there is a notion of correctness in the system that corresponds to correctness in the outside world without formally depending on it. Correctness in the justification system is easily checked by a non-intelligent agent (who knows nothing of the interpretation in the outside world). That agent might be a computer without artificial intelligence.

1.2. Checking correctness in a justification system of course does not lead to the philosophical notion of absolute certainty. There always remains the question of the dependability of the non-intelligent agent, and the matter of the interpretation of the outside world.

The matter of dependability of the agent may be a philosophical problem, but it is hardly a practical problem. We are able to satisfy ourselves that the work is done all right, even if we find it hard to express that formally. But as long as the outside world is not a justification system itself, the relation with the outside world will always need some kind of intuition, supported by the vast experience in modeling and formalizing built up over the centuries.

1.3. We may take it that the material presented to the justification system has the form of what we shall call a book. It is a sequence of lines, and the correctness of the book is the matter of every line being correct with respect to the book consisting of all previous lines. The lines themselves are written according to a language that remains constant over the book.

1.4. One of the first justification systems to be tried out and used extensively in various domains was Automath, started around 1967 (see [2], [6]). The system can act on the basis of various languages (like AUT68, AUT-QE), but for the present discussion some language-independent aspects may be more important, and more likely to be common to a larger class of justification systems. We first mention block structure (section 1.5) and gap-leaving (section 1.6).

1.5. Block structure is connected with the fact that whatever is to be said, is to be said under a number of assumptions which are assumed to be locally valid. It does not seem to be adequate to recall all these assumptions in every line: many assumptions are accepted in a large number of lines. Therefore we write the book in the form of nested blocks of lines. For every block we have an assumption that is taken as valid upon entering the block and discarded upon leaving it.

In the Automath system the block structure is mixed. Some blocks are initiated by assumptions, others by the introduction of variables. But that may be a language dependent feature already, since there can be languages that use no variables at all, or use a constant large pool of variables available before the book starts.

From a theoretical point of view one can say that block structure and in general the subdivision of a book into lines, is just a matter of syntactic sugar, presenting a layout that is close to our habits of presenting mathematics. In [9] it is described how a complete Automath book can be seen as a single tree, and how it suffices to formulate correctness notions for that tree.
1.6. Gap-leaving is not so much a matter of the structure of books. The notion rather refers to a writing strategy or writing style. It is connected to the way human knowledge, in particular mathematical knowledge, has been structured over the centuries. Many gaps are accepted temporarily in the hope or the certainty that they can be filled later. In mathematics we quite often leave gaps even without knowing it: it is recognized only later that a gap had been left.

The gap-leaving method can be used in justification systems as well. But since we want to have the books in a justification system to be correct at every moment during its construction, we will not have the situation of gaps we are not aware of.

Let us take the case that the gap consists of the fact that a proof for some theorem is lacking. Then we just do not call it a theorem, but an axiom, and the question whether that axiom can be derived from more basic material may remain open for some time.

Connected to the gap-leaving strategy are other ways to modify correct books. Even if we have a detailed text on something, it is often quite easy to adapt it to another basis. It can be like restructuring the lower floors and even the foundations of a building while in the upper floors business is as usual.

Sometimes we wish to consider a substantial piece of a book just as a black box. A paradigm for this is the use of a set of basic properties for the real number system with the effect that we never need to dig again into the definition of reals. Socially this has the effect that different people who have learned different definitions of the real number system can talk to one another about real numbers without ever noticing that they talk about different things! We just agree socially never to refer to details of our own definition of the reals. With some of the definitions a thing like $2 \sqrt{5}$ might make no sense, in others it might mean a well-formed but incorrect proposition, in others it might be just correct.

1.7. It can be considered as the miracle of mathematics that the process of leaving and filling gaps can terminate. In the large, it led eventually (in this century) to the possibility of a complete formal description of almost all mathematical knowledge.

But over and over again we have to stress that being formal is not everything; it is an end-stage of a process that misses the interesting, stimulating, intriguing, challenging stages of the development of mathematics, with its trial and error approach, learning from experience on various levels. Formalization of mathematics not just fails to describe mathematics as a social phenomenon, it also fails to describe its own development. In a way we can say that "Formalization is the Graveyard of Mathematics". But it is not as simple as that. In the past we have seen quite often that formalization can give rise to new mathematical developments. It may bring mathematical thinking on a new level. Without a certain amount of formalization many subjects (like algebra) just would not exist.

1.8. In designing justification systems there are two goals which are sometimes in conflict. Our first concern is to create facilities for standard material to be written and processed as fast as possible, the second one is to keep the system so general that it can handle many kinds of material we have not been thinking of at the start. Classical predicate logic and Zermelo-Fraenkel set theory may be adequate for describing most of modern mathematics, but on the other hand it does not seem to be wise to tie our hands completely to them.

What seems to be most appropriate is not to build things like logic and set theory in the basis of the system, but to develop them as material in the books. If that appears to lead to slowing down our processing of mathematics, we can still think of speeding it up by kinds of "sugaring". This way we can get to something that is both practical and general.

1.9. In order to enable justification systems to cope with the need for generality, it seems to be adequate to admit a flexible notion of typing. The things we can talk about are to have types we can talk about, and we should be able to create new types, possibly depending on parameters. The feature of typing can be used to extend our discourse, and to enable us to write in our books about things like propositions, proofs, constructions, programs, etc., without having to code them