Преглед НЦД 10 (2007), 1–8

F. Kamareddine, M. Maarek,
K. Retel and J. B. Wells
(ULTRA group, Heriot-Watt University)

DIGITISED MATHEMATICS: COMPUTERISATION VS. FORMALISATION

Abstract. In only few decades, computers have changed the way we approach documents. Throughout history, mathematicians and philosophers had clarified the relationship between mathematical thoughts and their textual and symbolic representations. We discuss here the consequences of computer-based formalisation for mathematical authoring habits and we present an overview of our approach for computerising mathematical texts.

1. Introduction

The mathematics we are taught from primary school to university level is almost always written in a distinctive language style. This style, used since the ancient times differs, from natural language by its extensive use of abstraction and its rigor. This very same language can designate first order equations, geometric problems or demonstrations in general topology. Of course the background requested to understand what is expressed depends on the knowledge of the reader but what remains is a certain uniformity of the language. This linguistic style has been called [**18**] the Common Mathematical Language (CML). Its ingredients are natural language and symbols composed into formulae. Recognisable text constructions, familiar labels and fonts constitute the visible substance of CML text. However, an important part of what makes the consistence of a text – such as the connections between pieces of text, the justification of an obvious deduction or the origin of a variable – is left to the reader's understanding.

A number of mathematicians and philosophers have advocated the use of a formal language (instead of CML) in order to exclude any ambiguity for the reader. The philosophical discussions over the essence of mathematical deductions and demonstrations led to the elaboration of logics. The search for a foundation of mathematics where proofs could be expressed in a formal language is ongoing and has already led to accepted and time-honored foundations (such as Gödel set theory or Zermelo/Fraenkel set theory). Nevertheless, despite these influential efforts, CML remained the communication medium for mathematicians. To accommodate the use of this informal but expressive language it is common to assume the feasibility of transcribing "formal" CML text into formal expressions.

> By analysis of the mechanism of proofs in suitably chosen mathematical texts, it has been possible to discern the structure underlying both vocabulary and syntax. This analysis has led to the conclusion that a sufficiently explicit mathematical text could be expressed in a conventional language containing only a small number of fixed "words", assembled according to a syntax consisting of a small number of unbreakable rules: such a text is said to be *formalized*.
>
> *N. Bourbaki*[1]*, Elements Of Mathematics* [**7**]

With their computational capabilities, computers offer the possibility to put formalisation into practice. Many systems (see a comprehensive comparison in [**31, 32**]) offer computational tools to automatically check the correctness of formal proofs. These systematic validations are time-wise impracticable by human. Automath [**27, 14, 1**] and Mizar [**29**] are among these precursor computer-based languages and checking systems. Computers can also in a way "invent" mathematical knowledge. The field of automatic deduction aims at assisting the mathematician or even replacing the mathematician by searching for a valid proof of a given property. But this revolutionary approach to mathematics has its skeptics. They are mathematicians who regards formal logic of philosophical interest but who consider that mathematics should not be restricted only to formal proofs. These mathematicians have not found their interest in using formal computer-based systems. This opposition between computer-based formal mathematics and more intuition-based mathematics raises the following question: *Does all mathematics need to be formalised?*

- If the answer to this question is positive then we end up with a more practical questions. How should we consider branches of mathematics that have not been formalised? What is the role in the universal body of mathematics for non-formalised material or material known to contain logical mistakes (therefore non-fully-formalisable)? How do we treat draft mathematics and unaccomplished theories that can not yet fed into a formal system?

  We are at a period of mathematics where computer-checked proofs are gaining respect and importance in the wide mathematical community – the four colour theorem is the first major theorem computer-proved [**13**] (using Coq proof assistant [**23**]). Nevertheless, the dreams of a universal library for formal mathematics has not come true [**4**]; it may never come true.
- If the answer is negative then we need to question the usability by mathematicians of proof assistant software. For most such software, the only possible outcome for a document, is to be a logically valid one. This orientation creates early choices for authors. In such framework, the design of a theory and the elaboration of proofs are oriented towards achieving full-formalisation. Mathematicians can only think of using a formal system if they are sure a formalisation will be carried out to its final close.

We explore in this paper an intermediate solution for putting mathematics on the computer. In Section 2 we discuss N.G. de Bruijn's views on computerisation of mathematics. In Section 3 we give a flavour of our approach in encoding mathematics

[1]N. Bourbaki is the pseudonym chosen by a group of French mathematicians (André Weil, Jean Dieudonné, Szolem Mandelbrot, Claude Chevalley, Henri Cartan were some of the founding members) in the 1930's. They wrote under this name a full treatise of modern mathematics: *Elements of Mathematics* [**7**].

on computer with the Core Grammatical aspect (CGa) and Text & Symbol aspect (TSa) of the MathLang framework. In Section 4 we review other research.

2. On the computerisation of Mathematics

The Automath experience over two decades has played two important roles. One in the history of mathematics, as the first attempt to create a formal language for mathematics on the computer. And the other in computer sciences, as the first computer-based language for automatic proof checking. Professor N.G. de Bruijn published in 1991 an article [**12**] which is a mixture of a philosophical summary of the Automath experience and a list of perspectives opened by this experience. In this article, he presents what he calls *justification systems* (which he considered to include Automath). He introduced this notion of justification when putting face to face automated checking and automated proving. The first one being the computer task of systematic logical checking of proofs and the other one is the ability of asking computers to *invent* a valid proof for a given theorem. He considered justification to be an automated checking that does not only deal with proofs but with mathematics in general. Formalisation has two means: correctness of proofs and better *understanding* of mathematics. N.G. de Bruijn proposed to clarify the actions one is doing when formalising mathematics. One should distinguish a formalism for mathematics and a computer based language for checking mathematical knowledge.

> The work may be subdivided. One can think of a first stage where a person with some mathematical training inserts a number of intermediate steps whenever he feels that further workers along the belt might have trouble, and a second stage where the logical inference rules are supplied and the actual coding is carried out. For the latter piece of work one might think of a person with just some elemenary mathematics training, or of a computer provided with some artificial intelligence. But we should not be too optimistic about that: programming such jobs is by no means trivial.
>
> *N.G. de Bruijn,* [**12**]

N.G. de Bruijn clearly differentiates two tasks in the work of formalising mathematics. The first one is close to traditional mathematical demonstration which identifies the steps that compose a proof. The second one is a systematic justification of every single logical step. This subdivision assumes that the foundation used in the formalisation and the chosen representation of mathematical objects are adequate for both steps. In practice, the systematic verification requires to adapt an early formalism and even sometimes change it profoundly to facilitate a result or even, in the worst case, to obtain a working proof.

Here lies the difficulty of choosing a suitable logical foundation when formalising mathematics. N.G. de Bruijn takes the party of the "mathematicians' approach" which is keen on protecting liberties and choices even if this would discard a full formalisation. This approach is opposed to a more "logicians' approach" for which mathematics should be sooner or later formalised in a universal logical framework.

> The notion of correctness is not formulated in terms of a mathematical reality, but involves rules about how a statement should be related to material that has been said before.
>
> *N.G. de Bruijn,* [**12**]

N.G. de Bruijn being a mathematician inclined to make good use of computers aims to reconcile the working-mathematician with the working-logician by raising the intrinsic question of the definition of a mathematical vernacular.

> I think that in formalizing mathematics, and in particular in preparing mathematics for justification, it is usually elegant as well as efficient to do everything in the *natural* way. That word of course does not mean "like in nature"; it can at most mean "like normally in our culture".
>
> [...]
>
> But of course, since the word "natural" means "cultural", it is subject to change. [...]
>
> *N.G. de Bruijn,* [**12**]

The way mathematicians represent and therefore think of mathematical objects is highly dependent on their unconscious social and cultural context. No formalisation is fully suitable – unless your cultural background is logical. A mathematician is the most likely to be able to computerise his mathematics. According to de Bruijn, refinements could slowly move this computerised knowledge into a formalised one.

3. MathLang-CGa-TSa: the lower level of MathLang, an overview

MathLang's philosophy [**19, 20**] is to bridge the gap between full-formalisation and common mathematical writings. Our starting point (at the lower level of MathLang) is to give a solution for putting mathematics on the computer. This solution:

- Leaves the author free to edit any mathematical texts, it should not be restricted to specific branches of mathematics,
- Gives a framework in which CML is the medium for the human reader but where the reasoning, expressed in CML, is computerised in a comprehensive manner for computer-based analysis,
- Opens the possibility of semantical and logical refinements and even for formalisation if achievable or needed.

The easiest way to get the intuition of how MathLang puts mathematics on the computer and manipulates it at this lower level, is through examples. Let us see how common mathematical constructions are identified in the lower level MathLang-CGa-TSa. See [**16, 15, 17, 24**] on MathLang-CGa-TSa for a broader explanation.

When considering only the aspect of a mathematical text concerning justifications, a mathematical text is typically a succession of deductions derived from facts. These deductions are brought forward by rational arguments which are composed by assumed facts or standing results from earlier parts of the text. The material of such deductions are notions and mathematical objects, concrete or abstract, that could often be defined within the mathematical text. This understanding of the composition of mathematical discourse is shared by N.G. de Bruijn's Mathematical Vernacular [**11**], Weak Type Theory [**18**] and MathLang-CGa.

Examples. Let us consider the following expressions.

(1) "$A \subset B$"
(2) "For every natural number n, $n + 1$ is also a natural number"

Which information is hosted in these sentences? Let us investigate both sentences.

(1) In the sentence "$A \subset B$", A and B are two identifiers. They have surely been introduced before the sentence was stated. If we assume that A and B are sets, the sentence introduces a new fact: one set is subset of the other.

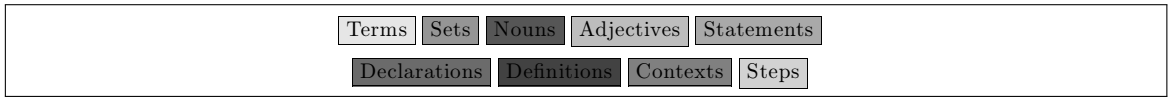


FIGURE 1. Grey scale coding of MathLang-CGa's grammatical categories

Here $\subset$ is a relation between sets. In MathLang-CGa-TSa, we *annotate* these three sub-expressions (A, B and $A \subset B$) with their grammatical role (respectively term, term and statement). Here is a view of this sentence with one coloured box per annotation. Figure 1 gives our colour coding.

$A \subset B$

Each annotation comes with an attribute input by the user. This attribute provides the identifier used to construct the expression contained in the annotation box. Here is a version of this sentence with these annotations' attributes printed in between angle brackets `< >`. An interpretation that only concerns these attributes would give the symbolic expression: `subset(A,B)`.

<subset> <A> A $\subset$ <B> B

These two views are automatically obtained from the same TEXMACS document (see Section 4 for information about TEXMACS). This document contains MathLang-CGa materials input using the MathLang-CGa-TSa plugin we developed for TEXMACS. In addition to offering customisable views, our plugin can also communicate the content of a document to our MathLang-CGa checker. This checker analyses the well-grammatical formation of the text. In the case of our example, the checker makes sure that the three identifiers `A`, `B` and `subset` have been properly introduced and are used in accordance to their definition. See [**17, 24**] for a complete description of MathLang-CGa's type system.

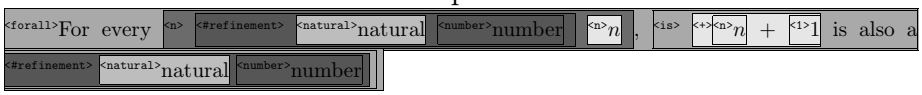
(2) The sentence "For every natural number n, $n+1$ is also a natural number" is an expression which states that the result of adding one to a natural number is a natural number. The implicit meaning of "For every" is the universal quantification. It introduces the variable n. This variable represents any natural number. The operator + applied to n and 1 creates a new object $n + 1$ which is said to belong to the same type of objects as n . The abstract object *a natural number* refers to one or two notions *natural* and *number* presumably defined earlier or simply assumed to be known by the reader. If we understand *number* as a type of object (we call them nouns in MathLang-CGa), then *natural* could be seen as a refiner for this type (that we call adjectives). It is important to notice the two slightly different uses of "being a natural number". In the first part of the sentence it is stating that n belongs to the type "natural number" but also participates into the declaration of n. As in the second part of the sentence, $n + 1$ is an object stated to be of the same type "natural number" but there is no declaration of identifier here. A symbolic version of this sentence would be like:

$$\forall n : \textit{natural number},\ n + 1 : \textit{natural number}$$

Here is the annotated MathLang-CGa-TSa version of this sentence.

For every natural number n, $n+1$ is also a natural number

And its annotation version with printed annotation attributes.



The symbolic interpretation (derived from the MathLang-CGa-TSa annotations) of this sentence would therefore be as follow.

```
forall(n :  natural number, in( +(n,1), natural number))
```

These two sentences cause no difficulty of being understood by any human reader with basic mathematical knowledge. Even taken out of their context, their meaning is easily inferred. It is exactly this inference (e.g. the mind action to extract the implicit meaning from such sentences) that we would like to conduct in MathLang-CGa-TSa. Such sentences left as they were earlier presented are meaningless for a computer software. The extra information we added in our explanation about variables' scoping, belonging of objects to a kind or a set, arity of symbols, and actions – new fact or object introduction – performed by a piece of text, are relevant for a comprehensible encoding of mathematical text.

We shall make it clear here that this extra information is not meant to be purposeful in the sense that they do not make explicit a calculation or a deduction, they simply express some knowledge already contained in the CML version of the text. The difference lies in the fact that they are expressed in a computerised way and therefore in an explicit manner. This extra information is simply grammatical and does not have for now a purpose for a specific computation or proof validation. What we are interested in here is the tangible grammatical information we could clarify when encoding mathematics.

4. Overview of systems for digitised mathematics

Research in computerised mathematics has gained interest in recent years as shown by the quality and scope of the Mathematical Knowledge Management conferences. Its community focuses on the handling of mathematics on the computer. This computerised management of mathematics requires a wide range of expertise.

Digitisation. At first, one could think of making use of existing CML documents. This is effectively the larger body of mathematical documents. Digitisation of mathematical documents requires some automatic recognition of natural language texts and mathematical formulae [**21, 28**].

Representation. Digitisation and authoring raise the question of documents' format. The output format depends mainly on the users' choices and requirements. LaTeX (`http://www.latex-project.org/`), used to typeset this paper, is a typesetting system which, focusing on the structural aspect of the document, provides an elegant visual representation. Some alternatives have been proposed to rectify some of LaTeX' representation inadequacies. TeXmacs (`http://www.texmacs.org/`) is being developed to offer a more intuitive WYSIWYG[2] authoring system. MathML (`http://www.w3.org/TR/REC-MathML/`) developed by the W3C[3] is intended to be a universal representation for mathematical symbols and formulae. OpenMath [**10**] and OMDoc [**22**] are two open markup formats which focuses on capturing the document's semantic meaning. OpenMath concentrates on the formula level, OMDoc on the theory and document level. This list of LaTeX alternatives is not meant to

[2]What You See Is What You Get.

[3]World Wide Web Consortium.

be exhaustive. MathLang-TSa focuses on providing an uncropped representation. It differs from MathML/OpenMath/OMDoc which disregard most of the portion of text written in natural language in their computerisation.

Automation. Following the development of theorem provers, a number of researchers have investigated the possibility of assisting the working mathematician in his formalisation experience. These researchers have led to more comprehensive access to *formal libraries* (see for example the HEΛM project [**2**]) and more suitable authoring software. We have already mentioned Mizar [**29**] whose syntax mimics CML. Similarly, Isar [**30**] is an alternative proof language interface for Isabelle. There exists several interfaces for formal system, let us mention three here: Theorema [**9**], a computer algebra system which is interfaced with Mathematica, Proof General [**3**] which is a generic IDE[4] for theorem provers and the Ωmega project [**5**] which aims at offering mathematicians with support for formalisation.

5. Conclusion

We believe that mathematicians and therefore mathematics would gain from a broader use of computers. The gap between mathematics expressed in natural language form and formal mathematics repulses the mathematicians not interested in full-formalisation. Bridging this gap will open the access to formalisation software.

In an attempt to bridge this gap we propose to computerise mathematical text prior to any formalisation. MathLang aims to support non-fully-formalised mathematics and to support automated processing of mathematical knowledge. Annotating books of mathematics written in English, as we are doing, can also be performed for books written in any other natural language.

References

[1] Automath archive. `http://automath.webhop.net/` Brouwer Institute in Nijmegen and the Formal Methods section of Eindhoven University of Technology.

[2] A. Asperti, L. Padovani, C. Sacerdoti Coen, and I. Schena. HELM and the semantic math-web. In *Theorem Proving in Higher Order Logics: 14th Int'l Conf., Proceedings*, volume 2152 of *Lecture Notes in Computer Science*, pages 59–74. Springer-Verlag, 2001.

[3] D. Aspinall, C. Lüth, and B. Wolff. Assisted proof document authoring. In MKM '05 [**25**], pages 65–80.

[4] H. Barendregt and F. Wiedijk. The challenge of computer mathematics. *Royal Society of London Transactions Series A*, 363(1835):2351–2375, Oct. 2005.

[5] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, A. Meier, E. Melis, W. Schaarschmidt, J. H. Siekmann, and V. Sorge. Omega: Towards a mathematical assistant. In *The Fourteenth International Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Computer Science*, pages 252–255, Townsville, North Queensland, Australia, July 1997. Springer-Verlag.

[6] N. Bourbaki. *Théorie des ensembles*, volume XVII (Livre I) of *Eléments de mathématique* [**8**]. Actualités Scientifiques et Industrielles, no. 1212. Hermann & Cie, Paris, 1954. Première partie: Les structures fondamentales de l'analyse. Chapitre I: Description de la mathématique formelle. Chapitre II: Théorie des ensembles.

[7] N. Bourbaki. *Theory of Sets*, volume I of *Elements of Mathematics*. Addison-Wesley Publishing Company, 1968. Chapters I and II are translations of [**6**].

[8] N. Bourbaki. Eléments de mathématique, Since 1939.

[9] B. Buchberger, A. Crăciun, T. Jebelean, L. Kovács, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, pages 470–504, 2006.

[4]Integrated Development Environment.

[10] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaëtano, and M. Kohlhase. *The OpenMath Standard*. The OpenMath Society, `http://www.openmath.org`, June 2004. Version 2.0.

[11] N. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In *Workshop on Programming Logic*, 1987. Reprinted in [**27**, F.3].

[12] N. G. de Bruijn. Checking mathematics with computer assistance. *Notices of the American Mathematical Society*, 38(1):8–15, 1991. Available at [**1**].

[13] G. Gonthier. A computer-checked proof of the four colour theorem. Available at `http://research.microsoft.com/~gonthier/4colproof.pdf`.

[14] F. Kamareddine, editor. *Thirty Five Years of Automating Mathematics*, volume 28 of *Kluwer Applied Logic series*. Kluwer Academic Publishers, Nov. 2003.

[15] F. Kamareddine, M. Maarek, and J. B. Wells. Flexible encoding of mathematics on the computer. In *Mathematical Knowledge Management, 3rd Int'l Conf., Proceedings*, volume 3119 of *Lecture Notes in Computer Science*, pages 160–174. Springer-Verlag, 2004.

[16] F. Kamareddine, M. Maarek, and J. B. Wells. Mathlang: Experience-driven development of a new mathematical language. In *Proc. [MKMNET] Mathematical Knowledge Management Symposium*, volume 93 of *ENTCS*, pages 138–160, Edinburgh, UK (2003-11-25/---29), Feb. 2004. Elsevier Science.

[17] F. Kamareddine, M. Maarek, and J. B. Wells. Toward an object-oriented structure for mathematical text. In MKM '05 [**25**], pages 217–233.

[18] F. Kamareddine and R. Nederpelt. A refinement of de Bruijn's formal language of mathematics. *J. Logic Lang. Inform.*, 13(3):287–340, 2004.

[19] F. Kamareddine and J. B. Wells. ProMath presenting, proving, and programming mathematical books (case for support). 9 pages, acknowledgement G583082, Dec. 2001.

[20] F. Kamareddine and J. B. Wells. MathLang: A new language for mathematics, logic, and proof checking (case for support). 9 pages, acknowledgement 2002102111273723614163, Sept. 2002.

[21] T. Kanahori, A. Sexton, V. Sorge, and M. Suzuki. Capturing abstract matrices from paper. In MKM '06 [**26**], pages 124–138.

[22] M. Kohlhase. *An open markup format for mathematical documents, OMDoc (Version 1.2)*, volume 4180 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2006.

[23] LogiCal Project, INRIA, Rocquencourt, France. *The Coq Proof Assistant Reference Manual – Version 8.0*, June 2004. Available at `ftp://ftp.inria.fr/INRIA/coq/V8.0/doc/`.

[24] M. Maarek. *Mathematical documents faithfully computerised: the grammatical and text & symbol aspects of the MathLang framework*. PhD thesis, Heriot Watt University, Edinburgh, June 2007. Supervised by Professor Fairouz Kamareddine and Doctor Joe Wells.

[25] *Mathematical Knowledge Management, 4th Int'l Conf., Proceedings*, volume 3863 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2006.

[26] *Mathematical Knowledge Management, 5th Int'l Conf., Proceedings*, volume 4108 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin / Heidelberg, 2006.

[27] R. Nederpelt, J. H. Geuvers, and R. C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1994.

[28] A. Raja, M. Rayner, A. Sexton, and V. Sorge. Towards a parser for mathematical formula recognition. In MKM '06 [**26**], pages 139–151.

[29] P. Rudnicki. An overview of the Mizar project. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, 1992.

[30] M. Wenzel. Isar – a generic interpretative approach to readable formal proof documents. In *Theorem Proving in Higher Order Logics: 12th Int'l Conf., Proceedings*, volume 1690 of *Lecture Notes in Computer Science*, pages 167–184. Springer-Verlag, 1999.

[31] F. Wiedijk. Comparing mathematical provers. In *Mathematical Knowledge Management, 2nd Int'l Conf., Proceedings*, volume 2594 of *Lecture Notes in Computer Science*, pages 188–202. Springer-Verlag, 2003.

[32] F. Wiedijk, editor. *The Seventeen Provers of the World, foreword by Dana S. Scott*, volume 3600 of *LNCS*. Springer-Verlag Berlin, Heidelberg, 2006.