

Deliverable 3.4:
Semantic Refinement

Mathematical Knowledge Management Network
MKMnet, IST-2001-37057

Summary

The present document is the result of **Task 3.4: *A notion of meaning refinement***, of the **Work Package 3: Varied Presentations of Mathematics and Varied User Needs**.

The fourth objective of this work package was to explore different ways to handle different presentation levels of a single meaning while remaining consistent between the different semantic views.

The sections have been compiled by various partners in the project, using information from other groups when necessary.

The complete report has been collected and integrated by Fairouz Kamareddine, Heriot-Watt University, Edinburgh.

1 Semantic refinements within the project

As well as considering content related issues such as notation and subject-specific variations, it is also necessary to consider how to present mathematical knowledge to make it accessible to the user. There are many types of people who require mathematical knowledge: students and school children; mathematics researchers; researchers from other disciplines; professionals looking for “off the shelf” solutions to their specific problems; and so on. And a single person may have different needs at different times. For example, a programmer looking for a mathematical algorithm to solve a problem may initially just look at a description of what an algorithm does to see if it is what they need. Later they will need the formal description of the algorithm itself and possibly even later, the proof of the algorithm’s correctness in order to adapt the algorithm to their particular situation. Throughout, examples may also be required covering different levels of complexity of the algorithm.

Many languages to encode mathematics exist already. We sort them in categories based on what they were invented for:

1. Languages for printing mathematical symbols on paper/screen (e.g. \LaTeX). Since they follow a *rendering* aim, they encode only the shape of a document and not its meaning.
2. Languages for theorem provers and computer algebra systems [Wie03] which try to formalize mathematics. These systems claim to assist the mathematician to prove theorems by verifying them.
3. Languages which store the semantical structure of mathematical texts without checking it (e.g. OMDoc [Koh03]). They combine natural language with notions like formulae or text structures (theorems, examples, etc.).

Different levels are needed during the mathematician’s work and the present need to encode this work on the computer: these levels vary from the mathematician’s informal level to more and more complete levels until we reach the fully formalised stage. The problem with the above mentioned encodings is that they do not allow the various levels:

- Languages for rendering do not capture the semantics of a text.
- Languages of theorem provers are too strict to encode incomplete or partly-formalized mathematical contents.

- Languages which store the semantical structure of mathematical texts without checking it do not allow enough automation to benefit from computers at each level.

Under this project, we set out to reflect on what can be done to allow for a framework in which different representations are possible and these can then be refined semantically to add extra information depending on the user's needs. We summarize our findings below.

1.1 Semantic refinement through formal logic

In [3, 4] it is argued that the semantics of mathematical knowledge is best explained in the frame of some formal logic. Practically, (versions of) predicate logic have proven to be a sound theoretical and versatile practical choice. In predicate logic, by a cultivated use of definitions, well structured hierarchies of notions and their properties (theories) can be built up. With some additional constructs like functors, which can be viewed as abbreviating mechanisms for constructs in pure predicate logic, the build-up of all of mathematics in a well-structured, reliable, efficient, and re-usable way is possible. This approach of "mathematics internal" - i.e. predicate logic based - allows a future build-up of mathematics which is more promising than a "mathematics external" mark-up of mathematical formulae by the semantic description mechanisms that prevail in other areas of knowledge management: A mathematics-internal semantics of mathematical knowledge makes the mathematics accessible to being processed by provers and other meta-mathematical tools whereas mathematics-external techniques consider only the semantics of mathematical expressions in a textual context.

1.2 Semantic refinement through weak type theory

In [5] it is argued that it is important to create bridges between the different methods that exist already so that the extensive research and findings in these different frameworks can be more widely used. If we can have a certain unifying framework or language that can be used at every level of mathematical work, we may be able to take the benefits of each of levels and languages of encoding and representing mathematics, to all other levels. The idea of the article is to use type theory as a first level of refinement and then to build further levels by using well studied technologies such as the proof construction style of OMEGA and later on further techniques which result finally in a fully formalised style.

1.3 Semantic refinement through proof construction

In [1], the authors work around the idea that research on automated and interactive theorem proving aims at the mechanization of logical reasoning. Aside from the development of logic calculi it became rapidly apparent that the organization of proof search on top of the calculi is an essential task in the design of powerful theorem proving systems. Different paradigms of how to organize proof search have emerged in that area of research, the most prominent representatives are generally described by the buzzwords: automated theorem proving, tactical theorem proving and proof planning. Despite their paradigmatic differences, all approaches share a common goal: to find a proof for a given conjecture. In this paper the authors start with a rational reconstruction of proof search paradigms in the area of proof planning and tactical theorem proving. Guided by similarities between software engineering and proof construction they develop a uniform view that accommodates the various proof search methodologies and eases their comparison. Based on this view, the authors propose a unified framework that enables the combination of different methodologies for proof construction to take advantage of their individual virtues within specific phases of a proof construction.

1.4 Semantic Refinement through representation of mathematical Knowledge

Representation of mathematical knowledge is one of the main concerns of the FoC project. The aim of FoC is to build a development environment well-suited to mathematical algorithms encoding. But computer algebra algorithms can be very intricate. Thus, in order to ensure code correctness, it is mandatory to decrease the distance between the mathematical specification and the implementation of the algorithms. To do that, FoC provides a language to write down properties, which is a sub-language of polymorphic typed lambda-calculus with dependent types. Algorithms can then be described at a rather abstract level, by their properties or in a more effective manner, using atomic operations which are known only by their name and their required properties. To obtain runnable code, the developer further needs to choose a data representation and implement atomic operations and prove that they meet their required properties.

The user who is developing elements of the library expects syntactical facilities helping her/him to embed mathematical knowledge into code. He also wants powerful programming tools which ease code writing, they must furthermore prevent misuses of already defined functions. The FoC end user has to write only a simple (read-eval-print) program which call library ele-

ments to perform his/her computations. Thus, FoC developers were faced with the different "faces" a user can have, depending on what she/he wants to do within FoC. The solution adopted by FoC is to offer three levels of notions in the language: species, collections, functional kernel. Species are compilation units. We can see species as record-like structures which fields can be either typed declarations/definitions or statements/proofs. They were formally specified (in Coq or in a categorical model) by records with dependent types (see [2]).

New species can be built by adding new fields or by a multiple inheritance mechanism allowing redefinitions. Since redefinitions of functions can invalidate some proofs, some restrictions on species have to be made (see [8, 9]). A species is accepted by the compiler if it can receive an interface, which records declarations and statements and the types of definitions and proofs (with some consistency checks). Now, a species can be parameterized by interfaces or by values "belonging" to known species, a mechanism which offers a true dependent language to developers (see [7]).

Collections are built by an encapsulation mechanism applied to complete species, that is, species where all declarations have received definitions and all statements have been proved. They are needed to ensure for example that data representation invariants cannot be broken by a misunderstanding use. A species parameter can only be instantiated by a collection. A value parameter is just an element of a collection, given by its representation (an element of a data type) together with a bunch of proved properties and operations.

The functional kernel is just intended to allow a read-eval-print program and provides no proof possibility. Indeed, if some proof is needed, we consider that the user ceases to be an end user and propose him to define a new species.

An example of development with FoC is shortly described under task 4 report, which also contains a presentation of the FoC compiler.

2 Making semantic refinement accessible to user

A user may start from an abstract, compact presentation of mathematical objects and may want to receive a more detailed presentation later. For instance, start with a proof idea presentation and on demand obtain more information about subgoals, methods applied, etc. Given a structured presentation, it is easily possible to fold and unfold problem solutions including proofs.

It is not just important to have different levels of detail but also different presentation forms, e.g., an explanation with a function graph, more symbolic, or more textual. This has been one didactical requirement in a national project in which ActiveMath is involved. The same meaning is still underlying that presentation. Different views on the same meaning can also be important for different notations (depending on the local culture).

In ActiveMath, the dictionary provides access to related mathematical and instructional items. By clicking on a presented mathematical object the dictionary is called for that object and presents links to *related* items. The basis for this is the underlying semantic representation is OpenMath or MathML. This semantics (in form of a URI) is used to invisibly enrich the presentation. The actual search in most current tools, including the ActiveMath dictionary, is text-based. The semantic encoding provides the potential for searching formulae and semantic expressions, and this will be investigated in future projects.

The definition of the notion of 'semantic approximation' has not been investigated yet.

3 A rationale on a common proposal for a notion of meaning refinement

The situation with different MKM systems can be comparable to the situation of different natural languages. Do we choose one natural language to act as a common language to all (just like English seems to be at the moment or just like Esperanto was intended in the past), or do we find efficient translation and communication mechanisms between the different languages. Apart from the issue of communication amongst different systems, there is also the issue that usually a system is designed to deal with certain domains or applications in mind. Hence, a system may be tailored to suit its application domains and tailoring it to suit other applications may reduce its efficiency. It is reasonable to deduce that since centuries of research on "what is the best logical formalism" still have not produced the ultimate logical system, it is a fact that one system does not suit all. Different applications require different systems. Witness the huge amount of work by the like of Russell, Whitehead, Hilbert, Weil, Ackermann, Frege, etc on formalising mathematics a century ago and the many arguments on why (im)predicativity is/is not needed in this/that domain (just to give one example). So, what have we achieved on the issue of a common proposal for a notion of meaning refinement in our net? As we discussed in Section 1., different partners have been active

on research on semantic refinement. Our description may give the impression that each of the partners' proposal is independent of, or incompatible with the others. This is not the case. For example, Buchberger explicitly aims to build the so-called mathematics-internal semantics of mathematical knowledge which makes mathematics accessible to other tools and provers. The MathLang approach at Heriot-Watt (through *weak* type theory) fits very well as an intermediate step between the mathematicians original text and its semantically refined version à la Buchberger. OMEGA on the other hand stands between the MathLang approach and the Buchberger approach. In fact, OMEGA texts have at some stages more semantically refined concepts and at other stages, the concepts are less semantically refined. This works very well in some environments (like teaching where OMEGA has showed specific advantages). Desiring different layers of semantic refinement leads inevitably to the issue of proof construction where incomplete proofs and information is filled with more and more details so that eventually the proofs are logically complete. This again has been successfully investigated by the OMEGA team and the MathLang team is currently developing details together with the OMEGA team on how to best carry out semantic refinements from the original mathematical texts through to semi-refined texts as in MathLang to more semantically refined texts (and proofs) as in OMEGA, to even more refined texts as in fully formalised theorem provers. Especially, we would like to integrate the work of Buchberger in this framework and to move towards more specific computer algebra environments like FoC. Our future collaboration will also investigate how to place the OMEGA parts that remain semantically unrefined.

Perhaps the best lessons we learned together as a net on this subject is how to divide tasks between our different approaches, so that for example, an approach like MathLang plays as a first bridge between mathematics and the computer, OMEGA would fill the gaps in proofs and concepts of MathLang (hence in the original mathematical text) and then, Theorema together with OMEGA may allow the presentation of MathLang documents as a higher form of formalism so that they can be taken further into systems that make the task of code checking easier (as in FoC). We also have identified important bridges between other MKM styles. For example, OpenMath finds its precise position on the refinement path, as does OMDoc, and fully formalised theorem provers like Coq.

In summary, through our collaboration under this net, we have identified that it is sensible to build semantic refinement as a sequence of steps which start from the original mathematical text (the least refined) and move step by step (identifying where each of the existing systems lie) until full formalisation is reached. Of course, it is up to the user how far he/she wants to

move in this path. Teachers and students of Mathematics, are not interested at all in formalised mathematics, but they are interested in the meaning of their concepts (so OMEGA may be more suitable for the teachers and students than Coq). Exactly how long or how refined this path from original mathematical texts to fully formalised ones remains to be studied. The net has allowed us to identify the need for such a path and to establish ways of refining it.

4 Conclusions

As can be seen from this report, members of the net have been very productive in the area of semantic refinement and have provided various products which have been well tested by the various groups. Work on mathematical knowledge management, as well as work on its semantic refinement, remains a very young subject which needs a lot of investment and research. The net has already provided impressive results in this area, but we are at the start of a very useful and fruitful direction from which the large community will greatly benefit. The support of the European Union has been greatly appreciated. Without that support, we would not have been able to make these useful contributions.

References

- [1] Serge Autexier and Christoph Benzmüller and Dieter Hutter. Towards a Framework to Integrate Proof Search Paradigms. Technical report number SR-03-02. Fachrichtung Informatik, Universität des Saarlandes, Saarbrücken, Germany. 2003.
- [2] Sylvain Boulmé. *Spécification d'un environnement dédié à la programmation certifiée de bibliothèques de Calcul Formel*. PhD thesis, Université Paris 6, 2000.
- [3] B. Buchberger. Theory Exploration with Theorema. Analele Universitatii Din Timisoara, Ser. Matematica-Informatica, Vol. XXXVIII, Fasc.2, 2000, (Proceedings of SYNASC 2000, 2nd International Workshop on Symbolic and Numeric Algorithms in Scientific Computing, Oct. 4-6, 2000, Timisoara, Rumania, (T. Jebelean, V. Negru, A. Popovici eds.), pp. 9-32. (ISSN 1124-970X.)
- [4] B. Buchberger. Algorithm Retrieval: Concept Clarification and Case Study in Theorema. Technical Report SFB 2003-44 of the Special Re-

- search Area "Scientific Computing", Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, October 2003.
- [5] F. Kamareddine, M. Maarek, and J.B. Wells. MathLang: Experience-driven new mathematical language. In [10]
- [Koh03] Michael Kohlhase. *OMDOC: An Open Markup Format for Mathematical Documents (Version 1.1)*. <http://www.mathweb.org/omdoc>, 2003.
- [6] Herman Geuvers and Fairouz Kamareddine, editors. *Special Issue on Mathematics, Logic and Computation, Electronic Notes in Theoretical Computer Science*, volume 85(7). Elsevier, July 2003.
- [7] Virgile Prévosto. *Conception et implantation du langage FoC pour le développement de logiciels certifiés*. PhD thesis, Université Paris 6, 2003.
- [8] Virgile Prévosto and Damien Doligez. Algorithms and proof inheritance in the foc language. *Journal of Automated Reasoning*, 29(3-4):337–363, December 2002.
- [9] Virgile Prévosto and Mathieu Jaume. Making proofs in a hierarchy of mathematical structures. In *Calculemus 2003 Proceedings*, September 2003.
- [10] Mathematical Knowledge Management Symposium. <http://www.macs.hw.ac.uk/~fairouz/mkm-symposium03/>
- [Wie03] Freek Wiedijk. Comparing mathematical provers. In *MKM'03*, 2003.