The FMathL Project ⋆

Short Communication for the Ninth International Conference on Mathematical Knowledge Management

Arnold Neumaier, Hermann Schichl, and Peter Schodl

Universität Wien, Nordbergstraße 15, 1090 Wien, Austria

1 Introduction

FMathL (= Formal Mathematical Language) is the working title for a modeling and documentation language for mathematics, suited to the habits of mathematicians. This project aims at the development of a flexible modeling system for the specification of models for large-scale numerical work in optimization, data analysis, and partial differential equations. Its input will be provided in a form natural for the working mathematician, while the choice of the numerical solvers and the transformation to the format required by the solvers is done by the interface system. The input format will combine the simplicity of LaTeX source code with the semantic conciseness and modularity of current modeling languages such as AMPL, and it will be as close as possible to the mathematical language people use to explain and communicate their models in publications and lectures. In order that the system is useful for the intended applications, interfaces translating the model formulated in the proposed system into the input required for current state of the art solvers, and into the dominant current modeling languages are needed and shall be provided. Moreover, certain shortcomings of the current generation of modeling languages, such as the lack of support for the correct treatment of uncertainties and rounding errors, shall be overcome.

2 Why Another Modeling Language?

During the work on the global optimization environment COCONUT [12], it became apparent that the currently available modeling languages which were very successful in dramatically simplifying the use of large-scale optimization software for the average user, still have significant limitations in several directions. Most prominent is the inability of the languages to express high level mathematical structures: For example, partial differential equations must be specified not in their natural form but in a fixed discretization, stochastic optimization problems must be reduced explicitly to sequences of ordinary optimization problems, and (probabilistic, worst case, or rounding-based) uncertainty specification in the available data is either impossible or very awkward. Model specification

⋆ The FMathL-Project is funded by the FWF

would be simpler and less error prone if the problems could be specified on their natural level, and the modifications needed to solve them would be done by tools operating on this specification. Another limitation for applications in pure mathematics is the lack of precision in the specification of mathematical problems to be solved. This is due to the fact that data conversion is usually accompanied by uncontrolled rounding errors, since traditionally numerical solvers introduced additional rounding errors anyway. However, there is now software with mathematical verification capabilities even for problems whose solution involves floating point computation, and interfacing to them needs full control of all approximations made. This is particularly important for applications in pure mathematics such as checking the validity of the solution of the Kepler problem, or the still unsolved problem of the minimum norm of the Thompson lattice, where the condition to be checked involves integer coefficients with several hundred digits.

3 Goals of the Project

Instead of augmenting the modeling capabilities of a language each time a new type of application is encountered – as it is currently done –, we propose in this project to create a modeling system with a universal modeling language that allows a semantic representation of arbitrary mathematical constructs. This language shall be interpreted by a system which can "understand" enough of this representation to perform automatic translations to specialized systems with well-defined solution capabilities.

By designing the new modeling language as a slightly formalized version of the traditional mathematical language, users are relieved from unnecessarily doubling of work by first representing the ideas in a general mathematical framework (such as a LaTeX report of the model) and later reducing it to a formal description by coding it in the appropriate formal language (e.g., AMPL).

While we shall concentrate in the implementation on the aspects directly relevant for applications in optimization and numerical analysis (where our main expertise resides), we want to make the design of the language and its basic implementation broad enough to enable others to build upon it a flexible, easy-to-use system for applications in any field of mathematics.

Of course, the best modeling language would be ordinary mathematical language as written in textbooks. But making ordinary mathematical text completely comprehensible for the computer is an exceedingly difficult task [13], at present virtually impossible without much interactive assistance. Therefore, the informal mathematical language must be somewhat restricted to be automatically analyzable.

We expect to design a modeling language which

- is based on traditional mathematical syntax (colloquial english or german mathematical text slightly formalized to ensure a unique interpretation),
- provides tools to translate mathematical specifications of problems into a unified internal representation, thus 100% preserving the content,

– integrates an implemented decision tree for mathematical software which automatically selects the right tools (recognizing a linear system of equations, an optimization problem, a partial differential equation, etc.),
– allows the access of a large variety of systems from a common mathematical platform

We can paraphrase the goal of the project in the following way: Combine the advantages of LaTeX for writing and viewing mathematics with the user-friendliness of mathematical modeling systems such as AMPL for the flexible definition of large-scale numerical applications, with the high-level discipline of CVX [6] for solving convex programming problems, and with the semantical clarity of the Z notation [7] for the precise specification of concepts and statements.

4 Things that Already Exist

Of course, partial progress towards such a modeling language is already visible in many current software systems. There are various existing systems that have realized some of the features we regard as useful, but all of them just accomplish just one of the many goals of the FMathL project. However, none of the systems available has the capabilities required for a universal mathematical modeling language.

Markup languages (OpenMath [1], MathML [5], OMDoc) facilitate the exchange of mathematical information between computers, and the generated display looks like nice ordinary mathematics.

Natural language interfaces enable to translate a non-fully-formalized language close to the common mathematical language into, e.g., Mizar (MathLang [9]), MathML (Hermes [2]) or formal statements (NaProChe).

Mizar represents syntax and semantics of mathematical information.

The Z notation [7] provides a LaTeX interface by the tool set CADiZ [8], and an interface to a theorem prover.

Modeling systems (AMPL, GAMS, NOP) have interfaces to all major state of the art optimization packages.

CVX[6] does automatic verification of the convexity properties for nonlinear optimization problems.

Decision trees for software [4] give advice about which software to use for which kind of optimization problem.

Theorema [3] presents proofs in a form acceptable for mathematicians: proofs can be expanded or shortened in a user-controlled manner.

The Grammatical framework [10] recognizes and generates important fragments of many languages.

5 How We Intend to Reach Our Goals

Initially we shall define a restricted subset of LaTeX as input format, and create a front end that translates restricted LaTeX documents into a simple graph representation based on the Vienna Graph Template Library (VGTL, [11]). This

will enable us to start working on the backend to solvers by using and extending the functionality of the translators in the COCONUT environment.

We shall design a grammatical extension of a simplified subset of LaTeX as preliminary input language. This subset will need a more rigid structuring than standard LaTeX to simplify the semantical analysis. We will start with a small subset of LaTeX which suffices to express the typical modeling problems we are interested in, but soon expand this subset. We hope that using the Grammatical Framework [10] will add flexibility to the rigid structure. At least in the beginning, interaction of the user will be necessary whenever the input is not ambiguous or not comprehensible for the system.

We shall use the experience gained in working with the preliminary input format to gradually evolve its structure and the associated language parser. During this process we shall improve the semantical analysis tool and adapt the language and the internal format. To ensure that we end up with a flexible language close to mathematical practice, we shall also collect and study informal phrases from diverse textbooks, as they are used to introduce a formal context in mathematical writing.

Later, we will try to find ways of resolving ambiguities automatically from context, or interactively if an automatic resolution is infeasible. We shall add multiple output language support for the documentation of models specified in the new input language.

On the other hand, we shall collect and analyze a large number of optimization models and constraint satisfaction models from the literature, described in informal mathematical terms, and derive from these a flexible language which enables us to encode these models with minimal deviations from the informal description as it would be encoded in LaTeX. This will replace the initial LaTeX dialect as input format.

To make the information collected in the graph representation available for actual use, we plan to create a number of interface tools to other systems.

We shall also design and provide a documentation facility for LaTeX output in natural language in several languages, at least English and German. The documentation language should be such that, in full documentation mode, the model can be automatically and unambiguously reconstructed from the documentation, at least from the English version.

6 Benefits from the Project

A huge amount of modeling is performed by mathematically educated people in many areas of application. Thus, we expect that our new modeling system will relieve them of most of the cumbersome translation or implementation work needed to transfer their mathematical knowledge to a formal system; in the ideal case, these modelers will not even need to learn a new formal language, since they usually know LaTeX and the mathematical language. Furthermore, using the system it will be finally possible to rigorously and conveniently formulate mathematical programming problems which have to be solved in a verified way.

As a long-term consequence we hope that our system may gradually develop into a standard for communicating formal mathematics. Ultimately, the system could develop into part of a system managing not only large user-specific models in important applications, but also managing general mathematical knowledge. While this is beyond the scope of this project, this project contributes at a low level towards such a system.

References

1. J. Abbott, A. Diaz, and R.S. Sutor. A report on OpenMath: a protocol for the exchange of mathematical information. *ACM SIGSAM Bulletin*, 30(1):21–24, 1996.
2. R. Anghelache. Hermes – A Reliable Conversion from TeX to MathML. In *New Developments in Electronic Publishing of Mathematics (4-th European Congress of Mathematics), Stockholm*, 2004.
3. B. Buchberger, T. Jebelean, F. Kriftner, M. Marin, E. Tomuţa, and D. Văsaru. A survey of the Theorema project. In *Proceedings of the 1997 international symposium on Symbolic and algebraic computation*, pages 384–391. ACM New York, NY, USA, 1997.
4. P. Bunus. A simulation and decision framework for selection of numerical solvers in scientific computing. In *Proc. 39th ann. Symp. Simulation, IEEE Computer Society, Washington*, pages 178–187, 2006.
5. D. Carlisle, P. Ion, R. Miner, and N. Poppelier. Mathematical markup language (mathml) version 2.0. *W3C Recommendation*, 21, 2001.
6. M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming (web page and software). `http://stanford.edu/~boyd/cvx`, 2000.
7. ISO. International standard ISO/IEC 13568:2002 information technology – Z formal specification notation – syntax, type system and semantics, 2002.
8. D. Jordan. CADIZ-computer aided design in Z. In *Proceedings of the 4th International Symposium of VDM Europe on Formal Software Development-Volume I: Conference Contributions-Volume I*, pages 685–686. Springer-Verlag London, UK, 1991.
9. F. Kamareddine, M. Maarek, and J.B. Wells. MathLang: An experience driven language of mathematics. *Electronic Notes in Theoretical Computer Science*, 93C:123–145, 2004.
10. A. Ranta. Grammatical framework. *Journal of Functional Programming*, 14(02):145–189, 2004.
11. H. Schichl. VGTL (Vienna Graph Template Library) version 1.0. *Manuscript,* `http://www.mat.univie.ac.at/~neum/glopt/mss/Schi02.pdf`.
12. H. Schichl. Global optimization in the COCONUT project. *Lecture notes in computer science*, pages 243–249, 2004.
13. C. Zinn. *Understanding Informal Mathematical Discourse.* PhD thesis, Univ. Erlangen, 2004.