

FMathL – Formal Mathematical Language

**Arnold Neumaier, Peter Schodl,
Kevin Kofler
(University of Vienna, Austria)**

I. The status quo

For our consulting work in Vienna, we'd like to have a system for efficient mathematical modeling at the highest possible level, namely ...

... in formal mathematical language in an easily readable and easily editable form.

For our work on computer-assisted proofs (of existence of solutions of a PDE, say) we'd like to have a system capable of verifying both the programs that do the numerical part of the proof and the theory that shows that the computations done indeed constitute the desired proof.

No existing system satisfies our needs.

The current overhead for automatic verification is horrendous:

Freek Wiedijk estimated that it takes 40 human hours to produce a verified version of one page of normal LaTeX, and this work is quite tedious for a mathematician.

But it takes me on average only 4 hours per page to write LaTeX in publication quality, including polishing the presentation style, working in improvements requested by the referees, and correcting galley proofs.

Current systems for doing formal mathematics on the computer (theorem provers, e.g., Theorema or COQ) are unaware of the preferences, values, and working habits of mathematicians, with the result that these systems are used by extremely few mathematicians.

But many mathematicians make use of

- mathematical typesetting languages (LaTeX),
- computer algebra packages (e.g., Mathematica),
- high-level numerics languages (e.g., Matlab),
- modeling languages (e.g., AMPL).

Each of these systems has numerous large-scale applications.

This leads to the challenge of designing and building a system (called a mathematical research system, MathResS) that

- is as easy to use as these heavily used systems,
- can encode and check arbitrary mathematics
- feels close to what mathematicians are accustomed to.

II. Goals and design criteria

In its ideal form, the desired mathematical research system

- represents arbitrary formulas in their natural context;
- represents arbitrary semantical relations between concepts, formulas, names of variables, etc.;
- produces and understands a large part of natural mathematical text;
- the output feels like high quality mathematical prose (whywihyp: "what you write is what you publish");
- feeds arbitrary solvers (packages for computer algebra, partial differential equations, optimization, theorem proving, etc.);
- is context-aware and allows the use of locally consistent notation in its appropriate context even when the notation is globally inconsistent.

The system should also be user-configurable (many theories, many personal styles), be incremental, and be capable of learning by doing.

We started work towards these goals, and are still far from having a satisfactory solution for practical applications.

But the partial work we have already done is encouraging enough to believe that the goal is reachable in the near future.

We also have far more utopian dreams about an automatic mathematical research system that could aid mathematicians (or other scientists) in doing tedious repetitive work (including checking mathematical text for completeness and correctness) that currently must be done with little computer support.

III. Analysis and linear algebra

Analysis and linear algebra are basic for mathematical reasoning. Hence they must be understood by any mathematical assistant worthy of assisting a mathematician.

Therefore we started to investigate what it takes to automatically analyze the LaTeX source of a text on analysis and linear algebra. Since we have the LaTeX source, the text chosen were my 450 page German lecture notes

”Analysis und Lineare Algebra”,

here referred to as ALA.

ALA contains 23 chapters with standard undergraduate mathematics,

- starting with naive set theory, groups, rings, and fields, and an axiomatic definition of the complex numbers,
- covering (among many other things) standard linear algebra, multivariate calculus, basic complex analysis, exterior forms, the Lebesgue integral, differential geometry in \mathbb{R}^n , and
- ending with the integral theorem of Stokes and the theory of the topological degree of a mapping.

Using LaTeXML and postprocessing its output, we created (with considerable difficulty) a list of about 4000 sentence templates in which all formulas were replaced by the word **FORMULA** or **DISPLAY**.

From this, we created an experimental (still far from satisfying) version of

- a lexicon of about 1500 German basic words,
- a simple morphological grammar for generating inflected and composed words,
- a sentence grammar with about 1000 production rules for generating semantically different phrases and complete sentences, and
- a program automatically generating a documentation of the resulting complete grammar from an internal representation.

The internal representation is expected to change with time as the grammar converges to one faithfully reflecting the subset of the German mathematical language used in ALA.

The current version is available at the FMathL web site

www.mat.univie.ac.at/~neum/FMathL.html

(Type FMathL into Google to find it!)

IV. The semantic matrix

For the representation of the semantics of mathematical texts we chose a record-based conception, called the semantic matrix.

Concepts are thought to be the names of rows and columns of a (potentially infinite) semantic matrix.

The entries of the (only partially filled) semantic matrix in row $\langle \text{concept1} \rangle$ and column $\langle \text{concept2} \rangle$, denoted by $\langle \text{concept1} \rangle . \langle \text{concept2} \rangle$ is again a concept.

The interpretation of the resulting partial binary dot operation is determined by some context-dependent protocol whose details are still under development.

For example, if NN is the name of the set of natural numbers then, in suitable contexts,

```
variable.name=x
```

```
variable.in=NN
```

denotes the assumed knowledge that x is a natural number.

In another context,

```
(x.in).NN=true
```

might encode in the semantic matrix the same information.

In practice, only a small, finite submatrix of the semantic matrix, capable of comprising all of mathematics (and indeed all formalizable knowledge), is explicitly generated and represented.

To test the universality of the semantic representation, we wrote a translator for the TPTP library (“Thousands of Problems for Theorem Provers”).

It translates TPTP problems into records in the semantic matrix.

A second translator takes the result and produces readable formulas in LaTeX.

We also just started to take problem classes from the OR-library (of data sets for a variety of optimization/operations research problems) and to translate their definitions into a semantic representation.

From this representation, solvers can be automatically fed with their required input format, and one can also recover a human-readable LaTeX format.

This is intended to lead to a system that is able

- to represent complete problem descriptions for arbitrary optimization problems,

and to translate them both

- into a human-readable description close to one a mathematician would produce, and
- into an input format for an appropriate solver.

Ultimately, there should be a bijection between

- internally represented data structures for the system to work on, and
- externally represented human-readable text in a well-defined formal mathematical language (FMathL) that can be translated both ways.

In addition, there should be an interactive reader of ordinary mathematical text, with editing facilities to bring it easily into the FMathL format without change in the intended meaning.

This would serve as a projection from the set of all mathemaitcally meaningful formulations to the subset of those formulated in FMathL.

In a second stage, we will also formulate a formal language for describing such problems in a way close to the way mathematicians communicate such problems on the blackboard to peers.

In a third stage, this language shall be extended to cover other problem areas (PDEs, etc.) and ultimately all mathematics.

V. The semantic Turing machine

To enable a future theorem prover to work directly on the semantic matrix, we defined the semantic Turing machine (STM), a machine whose memory is a semantic matrix, but which also has access to external devices (for input, output, and external solvers).

It has a universal, semantics-friendly, assembler-like programming language, for which we wrote a GLR parser and a simple (correct but inefficient) interpreter.

The interpreter (currently written for simplicity in Matlab) accesses, of course, only a finite part of the semantic matrix, and leads to an error message if the finite memory is exceeded.

The language is presently tested on a number of examples including

- a universal Turing machine (to show universality) and
- a universal semantic Turing machine (USTM), a single program that simulates in the STM the action of the STM with arbitrary programs on arbitrary inputs.

From the point of view of trustability, it is remarkable that the description of the USTM needs less than 300 lines of STM program code, including blank lines and comment lines.

Thus, it is expected that, in any implementation, only a very small amount of code must be checked by hand for correctness before the system can be trusted.

VI. MathResS – An automatic mathematical research system

The long term vision is to create an automatic mathematical research system that is able to perform at the level of a good undergraduate student of mathematics,

- both with respect to mathematical background knowledge
- and with respect to capability of understanding ordinary mathematical language and the ability to solve standard exercises.

Moreover, the automatic research system should be efficient in supporting the mathematical modeling of large-scale real life applications.

This would enable mathematicians using this system

- to check their own work for correctness,
- to improve the quality of their presentations,
- to decrease the time needed for routine work in the preparation of publications,
- to quickly and reliably remind them of work done
- to produce multiple language versions of their manuscripts
- and ultimately to quickly disseminate fully checked results to other users of the system.

Such a system will have a high value for every mathematician.

It would also enable visually impaired people to use the system and its mathematical contents.

It is proposed to create the system in such a way that these benefits will begin to be available long before the full capability of the system is reached.

This would make it attractive for mathematicians to use it already at an early stage, and to contribute to its completion.

A successful automatic mathematical research system requires the creation of a database system for storing mathematical theory that

- contains a formal representation of the mathematical theory at the undergraduate level (at least naive set theory, linear algebra, real analysis, basic algebra, and basic numerical analysis);
- ... and much more theory as the system grows;
- allows the systematic storage and retrieval of mathematical concepts propositions, and proofs;
- is fairly independent of notational styles, input language, and underlying mathematical or logical foundation;

[database requirements, continued]

- allows conceptual and structural search;
- has a standardized transfer language for communicating database contents between different database instances and to standard formats like MathML;
- has a versioning and backup/restore system for safe upgrading, etc.
- contains external links for background information, authorities for reference, further explanation;
- gets automatic upgrades from an official web site.

A successful automatic mathematical research system also requires algorithms for

- reading and interpreting mathematical text specified in a LaTeX-like fashion, as close to natural mathematical language as possible,
- displaying contents in natural mathematical language, if possible in different styles and/or languages,
- restructuring mathematical contents according to specific goals,
- comparing different sections of mathematical contents with respect to structural similarity and common information,
- verifying the semantical meaningfulness of mathematical contents,
- automatic calculation of routine computations,

[algorithmic requirements, continued]

- verifying the logical correctness of proofs given, and the validity of auxiliary results used,
- pointing out gaps and errors in proofs,
- finding counterexamples of, from a human point of view, obviously wrong statements,
- hierarchical arrangement of mathematical material in a way comprehensible by humans,
- meaningful handling of mathematically similar fragments of theories,
- automatic recognition and formation of special cases and generalizations,
- heuristic evaluation of the quality of mathematical contents w.r.t. given goals
- heuristic decision making for how to reach assigned goals.

A successful automatic mathematical research system also requires an analysis and efficient formalization of the social process by which a student acquires the capabilities

- to read and understand mathematical contents,
- to verify proofs, arguments given in a book or by a lecturer,
- to guess from context the meaning of ambiguous formulations, and to check whether the guessed interpretation is consistent with the context,
- to ask sensible questions clarifying items that are not well understood or seemingly ambiguous,

[social requirements, continued]

- to relate mathematical contents to one's own understanding, and to classify the material accordingly,
- to recognize the need to learn more about a concept, and to find out where the required information can be found,
- to respond sensibly to statements commenting on current performance,
- to answer questions regarding its understanding and knowledge of mathematical theory,
- to formulate meaningful plans for proceeding in a more complex task,
- to assess the quality of an argument or proof, and to suggest sensible improvements,

[social requirements, continued]

- to adapt to its partners in communication by selecting an appropriate level of detail or overview,
- to recognize particular styles of presenting mathematics,
- to recognize the existence of hidden assumptions, assumed terminology and notation, etc.,
- to recognize and use preferred but not binding notational conventions,
- to recognize minor glitches in spelling or sloppiness in notation or handling degenerate cases,

[social requirements, continued]

- to integrate notation, concepts, algorithms, or proof methods, seen repeatedly or formally described, into its knowledge base,
- to recognize that variations of the same statement say essentially the same thing, or to be able to say how they differ.

A successful automatic mathematical research system also requires a flexible and easy to use human-machine interface, in particular

- a TeXmacs like editor for creating mathematical contents and for editing mathematical manuscripts for publications,
- a batch mode for the automatic interpretation of mathematical textbooks, and their incorporation into the database,
- a facility to easily inspect and edit the database,
- graph based techniques to understand the interdependence of the contents of selected parts of the database,

[human interface requirements, continued]

- a dialogue system for communication with the user; both the system and the user must be able to ask and respond to questions by the other side,
- web based interaction tools to allow remote users to contribute to the growth and quality of the database contents,
- automatic synchronization tools between different instances of the database.

Trust management

All our communication, and all our knowledge, is based on trust and doubt.

Trust in the sources, trust in the tools used to create and/or check statements, trust in one's memory and in one's reasoning power.

Doubt about information that doesn't fit the current expectations.

And ways to check in which cases one should trust, doubt, revoke trust, or resolve doubts.

Trust is important in all proofs, not only the computer-assisted, or even fully formalized ones.

We trust the referees, the experts in the fields, the quoted references, or our own expertise.

We trust that we don't miss a faulty step hidden in a phrase "it is straightforward to show that".

We trust that our computer programs that did some calculations or checked some proofs are correct, and that the processor on which they run have no bug.

Sometimes, trust turns out to be unjustified later.

Nevertheless, without trust there is no knowledge, for everything can be doubted.

Fortunately, in robust, well-structured mathematical proofs, there are many connections between the concepts and arguments used.

This has the consequence that most results remain correct when a particular link is found wanting.

For example, Russell's paradoxon invalidated the logical basis of set theory, but hardly affected the bulk of mathematics.

This explains why mathematics as a whole thrives, in spite of many published “proofs” that contain serious gaps or errors.

But it also shows the importance of plausibility checks (of the kind of informal judgments upon seeing proof sketches by a colleague: “this cannot be true” – “very unlikely” – “seems ok”) and/or formal verification tools.

And it shows that a system should be built in a way that only a small amount needs to be rechecked should some piece no longer be trustworthy.

The proposed mathematical research assistant also needs a trust management system, in order to let the user keep track of information of different (and perhaps differently perceived by different users) quality.

To guarantee the highest possible level of reliability, all mathematical units of content are assigned signatures containing information about what is assumed to be able to trust the statement.

A signature may say, e.g.,

- “from Bourbaki’s Elements”,
- “imported from PlanetMath”,
- “verified by the proof assistant HOL light on the basis of ZFC”,
- “by an approximate Matlab calculation”,
- “based on the Riemann hypothesis”, or
- “unverified claim by user xyz”.

There must be a trust propagation and verification system that recognizes on which assumptions a given assertion is based and, given definition of what is trusted, points out all the untrusted links in existing verifications together with their signatures.

This enables users to specify or even to experiment with the amount of trust they are prepared to put in various sources of information and deduction systems.

Users interested in increasing the quality of the system can also look for theories in the system which still have important gaps on certain trust levels, and work towards closing these gaps.

With time, and organized in a wiki-like fashion, this will make the whole system more and more trustworthy.

Conclusions:

Lots of requirements – a really big challenge.

We shall see how far one can go towards this vision.

A funding application for part of the MathResS project is under way.

More information can be found at

<http://www.mat.univie.ac.at/~neum/FMathL.html>