

A FRAMEWORK FOR REPRESENTING AND PROCESSING ARBITRARY MATHEMATICS

Arnold Neumaier, Peter Schodl

Fakultät für Mathematik, University of Vienna, Nordbergstr. 15, A-1090 Wien, Austria

Arnold.Neumaier@univie.ac.at, Peter.Schodl@univie.ac.at

Keywords: Knowledge representation, human-machine cooperation.

Abstract: While mathematicians already benefit from the computer as regards numerical problems, visualization, symbolic manipulation, typesetting, etc., there is no common facility to store and process information, and mathematicians usually have to communicate the same mathematical content multiple times to the computer. We are in the process of creating and implementing a framework that is capable of representing and interfacing optimization problems, and we argue that this framework can be used to represent arbitrary mathematics and contribute towards a universal mathematical database.

1 INTRODUCTION

Mathematicians nowadays rely heavily on computers. They use them to communicate with colleagues, search the web for information, create documents they want to publish, perform numerical and symbolic computations, check their proofs, store the work they have done, etc. However, since mathematicians address very diverse parties with their writing, a mathematician usually has to formulate the same idea (e.g., a proof, a numerical problem, a conjunction) multiple times, depending on the recipient: for a student in great detail, for a foreign colleague working in the same field in less detail but a common language, for a publication in a document markup language, for a numerical solver in an algebraic modeling language, for a proof checker in a special language and at a tremendous level of detail.

Our vision is that, when represented in an adequate way, the same mathematical content only needs to be communicated to the computer once, and the machine can then extract the information in different formats, depending on the addressee. We have achieved some promising results representing and reformulating optimization problems into different formal and (controlled) natural languages, and we envision that our framework used for optimization problems is general enough for representing and communicating arbitrary

mathematics.

If a general representation with rich possibilities to interface the information proves feasible, it may also contribute to a huge electronic database containing essential amounts of the known mathematics. This vision is not new, it dates back at least to the QED project (Boyer, 1994). Its goal was to represent all important mathematical knowledge, conforming to the highest standards of mathematical rigor. Another vision in this direction was the universal automated information system for all sciences (Andrews, 2003). This is even more ambitious than a universal mathematical database, but the prominent role of mathematics in such a system, also discussed by Andrews, would make a mathematical database probably a corner stone of such a system and could be a starting point.

2 THE SEMANTIC MEMORY

The semantic memory is the data structure that is, in our opinion, most adequate to represent arbitrary mathematics. It is not a fixed file format like an Excel-sheet or an XML-file, but rather an abstract concept of a data structure, like a binary search tree or a heap. The semantic entities we want to refer to are called

objects. The set of objects is not fixed, but frequently used mathematical notions like set, function, derivative, manifold, \mathbb{R} , π etc. may be objects. All the information is then stored in a way that is akin to the semantic web, namely we store relations of the form

$$\text{object1.object2} = \text{object3} \quad (1)$$

with the only restriction that

$$\begin{aligned} \text{object1.object2} = \text{object3} \quad \text{and} \\ \text{object1.object2} = \text{object4} \quad \text{implies} \\ \text{object3} = \text{object4}. \end{aligned}$$

One possible way to store such relations is in the form of a matrix, where the relation (1) is represented as an entry `object3` in the row with name `object1` and column with name `object2`. We call this matrix the *semantic matrix*.

Such relations can also be stored and visualized in a labeled, directed graph, where the relation (1) is represented as an arc from vertex `object1` to vertex `object3`, labeled with `object2`. Such a directed graph will be called a *semantic graph*.

For example, the information $7 + 5 = 12$ may be represented by the following set of relations:

```
equation1.RHS = 12
equation1.LHS = term_lhs
equation1.OP = EQUAL
term_lhs.1 = 7
term_lhs.2 = 5
term_lhs.OP = PLUS
```

This corresponds to the semantic matrix given in Figure 1 and to the semantic graph in Figure 2.

	LHS	RHS	OP	1	2
equation1	term_lhs	12	EQUAL		
term_lhs			PLUS	7	5

Figure 1: $7 + 5 = 12$ represented in a semantic matrix

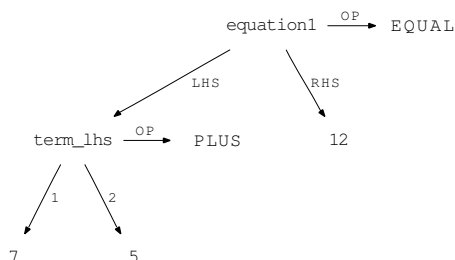


Figure 2: $7 + 5 = 12$ represented in a semantic graph

A semantic graph is a special case of a *concept map*, a graphical tool to organize and represent knowledge,

but that is not defined rigorously (Novak and Cañas, 2006).

3 THINGS THAT ALREADY EXIST

We list some of the facilities that already provide valuable aid for mathematicians. Interfaces to these systems would be desirable.

Semantic markup languages for mathematics:

This is a group of languages that represent mathematics as annotated text. They are used to communicate formulas between machines, display formulas on the web, etc. Examples are OMDoc (Kohlhase, 2001) and OpenMath (Abbott et al., 1996).

Computer algebra systems:

These are programs that are able to perform symbolic manipulations on terms, solve equations, plot graphs, etc. Examples for widely used CAS's are Mathematica, Maple and Maxima.

Algebraic modeling languages:

Numerical optimization problems can be expressed quite comfortable in algebraic modeling languages. The machine reads the description of the problem and the numerical data and is able to interface a variety of solvers. Examples are AMPL, GAMS, GLPK and NOP-2 (Kallrath, 2004).

Proof assistants:

A *proof assistant* is software that checks the validity of a proof, expressed in a special, highly detailed and annotated language. Since translating a proof into such a language is a lot of work, proof checkers are not widely used among mathematicians.

4 NATURAL LANGUAGE IN- AND OUTPUT

For being attractive for a working mathematician, the ability to interface existing systems is one key feature, another one is communication in an almost natural language.

There is a consensus among mathematicians and linguists that the communication of mathematics to a computer is much easier than the communication of arbitrary content. This has several reasons:

- Mathematical discourse has a well-defined domain, is highly structured, and has relatively small set of discourse relations. The reasoning patterns applied in mathematics are widely studied and understood (Zinn, 2004). Building an ontology for, say, number theory, is much easier than for a natural domain, because mathematicians *define* concepts before they use them. It was even claimed that “[...] if we fail to construct an understander for mathematical discourse, then we will also fail to write one for other (non-trivial) domains”, see p. 8 in (Zinn, 2004).
- Due to the fact that mathematicians want to communicate unambiguously, they tend to use a relatively small set of phrases to express their ideas, and there is a standard interpretation for these phrases. About 700 phrases suffice for the essential part of mathematics (definitions, theorems, proofs, etc.) but this does not include the more informal motivational part (Trzeciak, 1995).
- Mathematicians use words and phrases in a very rigid way. The language of mathematics is simple: very few variety in time, person, etc. (Ganesalingam, 2009).
- Another reason why mathematics is apt to be represented by a machine is that in mathematics we are in the (probably unique) position that every meaningful rigorous statement can, at least in principle, be translated into a formal language. Therefore, it is possible for a machine to faithfully represent the complete content of an arbitrary (but meaningful) mathematical statement.

However, we do not intend to allow general natural language as input, even though we expect only relatively simple sentences, but we intend to exploit the fact that mathematical language is simple by defining a controlled natural language (CNL) that is expressive enough to fulfill the needs of mathematicians, while still sounding like natural language.

For formulas, since \LaTeX has been de facto-standard in the mathematical community for decades, we envision a reasonable subset of \LaTeX as the main input format.

5 THINGS ALREADY IMPLEMENTED

- Optimization problems can be represented in the semantic memory, and a description of the problem can then be automatically generated in the algebraic modeling language AMPL and in almost

natural language. Below is an example of a simple optimization problem, in Figure 3 formulated as a mathematician would do, and then in in Figure 4 in the AMPL-format. Both texts have been generated automatically from a common representation in the semantic memory comprising about 550 relations of the form of (1).

- For the semantic memory we have two implementations, one written in Matlab where the semantic memory is a sparse matrix and the objects are natural numbers, and another one in Soprano, a framework for RDF data.
- We created an interface to the controlled natural language of the Naproche project (Kühlwein et al., 2009), a project carried out at the university of Bonn that enables proof checking of proofs written in a controlled natural language.
- Creation of \LaTeX -output of simple general mathematical text represented in the semantic memory: basic forms of definitions, assumptions, interferences, etc.
- Grammatically correct text-output is generated via an interface to the Grammatical Framework (Ranta, 2004), a programming language and software package for multilingual grammar applications.
- We implemented a parser for problem files of the TPTP (Thousands of Problems for Theorem Provers, available at <http://www.tptp.org/>), and parsed and represented large parts in the semantic memory, adding up to several thousand formulas. These have been represented in the semantic memory and written into \LaTeX -files.

6 CONCLUSION

We gave our arguments why we think that our framework to represent optimization problems can be a starting point for a more general system to represent and process arbitrary mathematics. We think that one of the key features of a universal mathematical database, that will actually be used, is the retrieval of information in many different forms, together with the ability to communicate with the user in a way natural to the human. There is already a huge amount of mathematical knowledge on the web, e.g., in online encyclopedias, in archives for scientific papers, libraries for proof assistants etc., but as long as they only serve single purposes, they will stay separated.

Multi-dimensional knapsack.

Let integer N be number of contract, let integer M be number of budget, let c_j be contract volume of project j for $j = 1, \dots, N$, let $A_{i,j}$ be estimated cost of budget i for project j for $i = 1, \dots, M$ and $j = 1, \dots, N$, let B_i be available amount of budget i for $i = 1, \dots, M$ and let $x_j = 1$ if project j is selected, and let $x_j = 0$ otherwise for $j = 1, \dots, N$.

Problem : Given integer N , integer M , vector c , matrix A and vector B find binary vector x such that

$$\sum_{j=1}^N c_j x_j$$

is maximal under the constraint $\sum_{j=1}^N A_{i,j} x_j \leq B_i$ for $i = 1, \dots, M$.

Figure 3: The knapsack-problem in (almost) natural mathematical language

```
param N ;
param M ;
param c{j in 1..N} ;
param A{i in 1..M , j in 1..N} ;
param B{i in 1..M} ;
var x{j in 1..N} binary ;

maximize target : sum{j in 1..N}(c[j]
* x[j]);
subject to constraint_3014{i in 1..M}
: sum{j in 1..N}(A[i , j] * x[j]) <=
B[i];
```

Figure 4: The knapsack-problem in AMPL

ACKNOWLEDGEMENTS

Support by the Austrian Science Fund (FWF) under contract number P20631 is gratefully acknowledged.

REFERENCES

- Abbott, J., Díaz, A., and Sutor, R. (1996). A report on openmath: a protocol for the exchange of mathematical information. In *SIGSAM Bulletin* **30** Nr. 1. ACM.
- Andrews, P. (2003). A universal automated information system for science and technology. In *First Workshop on Challenges and Novel Applications for Automated Reasoning*.
- Boyer, R. e. a. (1994). The qed manifesto. In *Automated Deduction—CADE 12*. Springer.
- Ganesalingam, M. (2009). *The Language of Mathematics*. PhD thesis, University of Cambridge.

Kallrath, J. e. (2004). *Modeling languages in mathematical optimization (Applied Optimization Vol. 88)*. Kluwer Academic Publishers, Boston, Dordrecht, London.

Kohlhase, M. (2001). Omdoc: Towards an internet standard for the administration, distribution, and teaching of mathematical knowledge. In *Artificial Intelligence and Symbolic Computation*. Springer.

Kühlwein, D., Cramer, M., Koepke, P., and Schröder, B. (2009). The naproche system. In *Intelligent Computer Mathematics*. Springer.

Novak, J. and Cañas, A. (2006). The theory underlying concept maps and how to construct and use them. In *Technical report IHMC CmapTools 2006-01*. Florida Institute for Human and Machine Cognition, Pensacola FL.

Ranta, A. (2004). Grammatical framework. In *Journal of Functional Programming*, **14** Nr. 2. Cambridge University Press.

Trzeciak, J. (1995). *Writing mathematical papers in English: a practical guide*. Gdańsk Teacher's Press, Gdańsk.

Zinn, C. (2004). *Understanding informal mathematical discourse*. PhD thesis, University of Erlangen-Nürnberg.