

Joint Proceedings of the FM4M, MathUI, and
ThEdu Workshops, Doctoral Program, and
Work in Progress at the Conference on
Intelligent Computer Mathematics 2016

Andrea Kohlhase Michael Kohlhase Paul Libbrecht
Hochschule Neu-Ulm Jacobs University PH Weingarten

Bruce Miller Frank Tompa
NIST University of Waterloo

Adam Naumowicz Walther Neuper
University of Bialystok Graz University of Technology

p Pedro Quaresma Martin Suda
University of Coimbra TU Wien

December 22, 2016

Contents

<i>Preface</i> (Michael Kohlhase)	i
<i>FM4M 2016: Formal Mathematics for Mathematicians</i> (Adam Naumowicz)	1
<i>Invited talk: Developments, Libraries and Automated Theorem Provers</i> (Chad Brown)	2
<i>Invited talk: On Differences in Proofs Between Intuitionistic and Classical Logic</i> (Aleksy Schubert)	3
<i>Tarski's geometry and the Euclidean plane in Mizar</i> (Adam Grabowski and Roland Coghetto)	4
<i>Formalization of the prime number theorem and Dirichlet's theorem</i> (Mario Carneiro)	10
<i>Topological Foundations for a Formal Theory of Manifolds</i> (Karol Pąk)	14
<i>Registrations vs Redefinitions in Mizar</i> (Artur Kornilowicz)	17
<i>Linking to Compound Conditions in Mizar</i> (Adam Naumowicz)	21
<i>MathUI 2016: Mathematical User Interfaces</i> (Andrea Kohlhase, Paul Libbrecht)	25
<i>Towards Visual Type Theory as a Mathematical Tool and Mathematical User Interface</i> (Lucius Schoenbaum)	26
<i>Understanding Mathematical Expressions: An Eye-Tracking Study</i> (Andrea Kohlhase, Michael Fürsich)	42
<i>The plain text trap when copying mathematical formulae</i> (Paul Libbrecht, Matija Lokar)	51
<i>Proposal for Coexistence of Mathematical Handwritten and Keyboard Input in a WYSIWYG Expression Editor</i> (Juan Lao-Tebar, Francisco Álvaro, Daniel Marqués)	56
<i>KAT: an Annotation Tool for STEM Documents</i> (Tom Wiesing, Felix Schmoll)	66
<i>Notation-based Semantification</i> (Ion Toloaca, Michael Kohlhase)	73
<i>ThEdu 2016: Theorem Provers Components for Educational Software</i> (Walther Neuper & Pedro Quaresma)	82
<i>Lucas-Interpretation from Users' Perspective</i> (Walther Neuper)	83
<i>Rigor of TP in Educational Engineering Software</i> (Walther Neuper)	90
<i>CICM 2016 Doctoral Program</i> (Martin Suda)	96
<i>Automated Theorem Proving for Elementary Geometry</i> (Marek Janasz UP Cracow)	97
<i>Knowledge Management across Formal Libraries</i> (Dennis Müller)	98
<i>Augmenting Mathematical Formulae for More Effective Querying & Presentation</i> (Moritz Schubotz)	102
<i>Design and development of a tool based on Coq to write and format mathematical proofs</i> (Théo Zimmermann Hugo Herbelin)	104
<i>DML Work In Progress</i> (Frank Tompa)	106
<i>swMATH - Challenges, Next Steps, and Outlook</i> (Hagen Chrapary, Wolfgang Dalitz, and Wolfram Sperber)	107

<i>Formalization of Polynomially Bounded and Negligible Functions Using the Computer-Aided Proof-Checking System Mizar</i> (Hiroyuki Okazaki and Yuich Futa)	117
<i>A Smooth Transition to Modern mathoid-based Math Rendering in Wikipedia with Automatic Visual Regression Testing</i> (Moritz Schubotz and Alan P. Sexton)	132
<i>Getting the units right</i> (Moritz Schubotz, David Veenhuis, and Howard S. Cohl)	146
<i>MKM Work In Progress</i> (Bruce Miller)	157
<i>Lemma Extraction Criteria Based on Properties of Theorem Statements</i> (Karol Pałk)	158
<i>The impact of proof steps sequence on proof readability — experimental setting</i> (Karol Pałk and Aleksy Schubert)	172
<i>Models for Metamath</i> (Mario Carneiro)	187
<i>A first step towards automated conjecture-making in higher arithmetic geometry</i> (Andreas Holmstrom)	204
<i>Initial Experiments with Statistical Conjecturing over Large Formal Corpora</i> (Thibault Gauthier, Cezary Kaliszyk, and Josef Urban)	219
<i>A Standard for Aligning Mathematical Concepts</i> (Cezary Kaliszyk, Michael Kohlhase, Dennis Müller, Florian Rabe)	229
<i>FrameIT Reloaded: Serious Math Games from Modular Math Ontologies</i> (Denis Rochau, Michael Kohlhase, and Dennis Müller)	245

Preface

This joint volume of proceedings gathers together papers from the workshops and Work in Progress section of the 9th Conference on Intelligent Computer Mathematics (CICM), held July 25-29 2016 in Bialystok, Poland. CICM has been held annually since 2008. Papers from the four main tracks at CICM 2016 (CALCULEMUS, Digital Mathematical Libraries, Mathematical Knowledge Management, Surveys & Projects, and Systems & Data) are published in volume 9791 of Springer Lecture Notes on Artificial Intelligence (LNAI).

CICM also gives the opportunity for researchers to present Work in Progress (WiP) papers, of interest to the community but not yet ready for formal presentation. This year, WiP papers have been presented in the MKM and DML tracks.

Finally, CICM traditionally organizes a “Doctoral Program”, where doctoral students can present their achievements and future plans to the community and receive mentoring by experienced members of the CICM community.

This joint volume collects these together with those contributed to three of the workshops held at CICM 2014:

- The 11th Workshop on Mathematical User Interfaces (MathUI),
- The 2014 Workshop on Theorem Provers Components for Educational Software (ThEdu),
- The 2016 Workshop on Formal Mathematics for Mathematicians (FM4M),

The workshop papers were edited by their respective organisers and the Work In Progress papers by the General CICM PC Chair and the CICM track Chairs. Please see the individual prefaces for more details.

Editors:

Michael Kohlhase Joint volume editor and CICM program chair

Andrea Kohlhase MathUI editor

Paul Libbrecht MathUI editor

Bruce Miller MKM WiP editor

Frank Tompa DML WiP editor

Adam Naumowicz FM4M editor

Martin Suda Doctoral Program editor

Acknowledgements: Thanks to all those who served on the program committee or reviewed the work in this volume, and to the organisers of CICM 2016. We acknowledge the publisher CEUR-WS, and the authors of the EasyChair software which greatly aided the publication of these proceedings.

20th July 2016
Michael Kohlhase
Jacobs University Bremen

FM4M 2016: Formal Mathematics for Mathematicians

FMM is a workshop affiliated with CICM 2016 intended to gather together mathematicians interested in computer assistance and researchers in formal and computer-understandable mathematics.

The mathematical community today seeks various ways to support their work by accessing digital libraries and repositories, applying Internet search techniques to better explore and classify the vast mathematical knowledge, and to combine computer calculations with informal mathematics. Related methods have been developed a lot recently by the formal community, allowing the building of very large formal mathematical libraries and full formal verification of large computationally involved proofs such as those of the Kepler conjecture and the Four Color Theorem.

It is very important to establish a platform for both communities to interact. The successful development of computerized formal mathematics and its general availability very much depends on the feedback that the formalized mathematics developers can obtain from the community of working mathematicians. This workshop's main objective is to explore ways of building synergy between the two communities.

Over the last decades, we witnessed a number of successful instances of computer-assisted formalization of mathematical problems. Research in this field has been boosted by the development of systems for practical formalization of mathematics (proof assistants), a creation of large repositories of computer-verified formal mathematics, and integration of interactive and automated methods of theorem proving. Proof assistants provide a very useful teaching tool suitable for undergraduate instruction, in particular for training beginning students in writing rigorous proofs. An expected wider outcome of this workshop is therefore setting up of an annual series of meetings between working mathematicians and researchers in formal mathematics as well as graduate students with strong background in these fields.

July 2016, Adam Naumowicz

Invited talk: Developments, Libraries and Automated Theorem Provers

Chad Brown

Czech Technical University
Prague, Czech Republic

Abstract

When formalizing a mathematical development with an interactive prover, it is helpful if the user can interface with a library (to avoid starting from scratch) and with automated provers (to avoid needing to give full details explicitly). We will consider an example of a development in Mizar, leading to some discussion of how one can interact with Mizar's library and how automated theorem provers can help construct Mizar proofs. With this example in mind, we discuss criteria for three aspects of formalization to work in harmony: formal mathematical developments, working with a global library of theorems and definitions and making use of automation.

Invited talk: On Differences in Proofs Between Intuitionistic and Classical Logic

Aleksy Schubert

Institute of Informatics, University of Warsaw
ul. S. Banacha 2, 02-097 Warsaw, Poland
alx@mimuw.edu.pl

Abstract

The presentation will contrast the complexity results for proving assertions in classical and intuitionistic logic. The comparison will be built upon the known results for propositional logic and predicate one. The predicate case will be based upon the Mints hierarchy in intuitionistic logic which will be contrasted with its counterpart i.e. the prenex hierarchy in classical logic.

The differences in complexity will be illustrated with examples of particular proving mechanisms that are responsible for the divergence, which should facilitate the understanding of where the mathematics is done constructively.

Tarski’s Geometry and the Euclidean Plane in Mizar

Adam Grabowski
Institute of Informatics
University of Białystok
ul. Ciołkowskiego 1 M
15-245 Białystok, Poland
adam@math.uwb.edu.pl

Roland Coghetto
Rue de la Brasserie, 5
B-7100 La Louvière
Belgium
roland.coghetto@hotmail.com

Abstract

We discuss the formal approach to Tarski geometry axioms modelled with the help of the Mizar computerized proof assistant system. We try however to go much further from the use of simple predicates in the direction of the use of structures with their inheritance, attributes as a tool of more human-friendly namespaces for axioms, and registrations of clusters to obtain more automation (with the possible use of external equational theorem provers like Otter/Prover9). The formal proof that Euclidean plane satisfies all eleven axioms proposed by Tarski is an essential development, allowing to show the independence of the parallel postulate, one of the items from “Top 100 mathematical theorems”.

1 Introduction

For years, foundations of geometry attracted a lot of interest of researchers from various areas of mathematics. From the very beginnings, human thought was stimulated by geometrical objects, however from the modern viewpoint of automated theorem-provers, diagrams can deliver some really tough problems. Here an important example is the possibility of ruler-and-compass construction: impossibility of trisecting the angle and doubling the cube (as two out of four problems of antiquity), where the treatment of constructible numbers is way more efficient from the formal point of view.

The choice of the topic is not accidental – recent code available in Coq [BN12] and the use of automated equational provers caught an eye of researchers and, as a by-product, some results which shed some new light on the axiomatization of geometry were published. One of the bright milestones was also the publication of the new issue of the classical textbook *Metamathematische Methoden in der Geometrie* by Schwabhäuser, Szmielew, and Tarski (SST) with the foreword of Beeson.

2 Mizaring Affine Geometry

In Tarski’s system of axioms [TG99] the only primitive geometrical notions are points, the ternary relation B of “soft betweenness” and quaternary relation \equiv of “equidistance” or “congruence of segments”. The axioms are reflexivity, transitivity, and identity axioms for equidistance; the axiom of segment construction; reflexivity, symmetry, inner and outer transitivity axioms for betweenness; the axiom of continuity, and some others. The original set consisted of 20 axioms for two-dimensional Euclidean geometry and was constructed in 1926–27, submitted for publication in 1940, and finally appeared in 1967 in a limited number of copies. There are many modifications of this system, and Gupta’s work in this area [Gup65] offers an important simplification.

Another notable axiomatization, proposed by Hilbert [Hil80] in 1899, has three sorts: planes, points, and lines, and three relations: betweenness, containment, and congruence. In this sense it is a little bit more complex than Tarski's (but not necessarily in terms of numbers of axioms as it has also 20 of these). The two approaches establish a geometrical framework, in which theorems can be proven logically (remember Euclid's *Elements* proofs are mainly pictures or graphical constructions and rely heavily on intuition). This allows to use computerized theorem prover in order to find the proof or proof checker to check the theory for its correctness. We deal with the proof checker Mizar based on classical first order logic and Tarski-Grothendieck set theory (a variant of Zermelo-Fraenkel) and describe, how Tarski's theory was built formally.

Geometry in the Mizar Mathematical Library is based on eight various structures, so it raises communication issues between various approaches. The last article was PROJPL_1 dated back to 1994, and the series was not very actively developed (with the exception of the paper of the combinatorial Grassmanians COMBGRAS). But in 1990 Mizar articles on geometry were about 30% of the whole Mizar repository (out of 140 files). Of course, after that time revisions of this specific area were quite active: one of the main streams (done also by the current author) was to get separate axioms of selected properties instead of a large block for *the mode* (i.e. the constructor of the type in Mizar). Affine approach to geometry was less important as there was another big challenge which for fifteen years stimulated geometry (in analytical setting, however): the proof of the Jordan Curve Theorem.

As notable affine geometrical facts already formalized in Mizar we can enumerate:

- Hessenberg's theorem – HESSENBE;
- Desargues theorem (present in "Top 100 mathematical theorems") – ANPROJ_2;
- Pasch configuration axioms – PASCH;
- Fano projective spaces – PROJRED1;
- Desarguesian projective planes – PROJRED2;
- Pappus, Minor, Major and Trapezium Desargues axioms – AFF_2;
- Minkowskian geometry – ANALORT.

In our opinion, the unifying approach in Tarski's spirit could be quite useful to bind all of these geometrical results together. Among another significant facts in geometry we can point out Morley trisector theorem, Ceva, and Menelaus theorem. These facts however deal with Euclidean plane, so the proofs are in the area of analytic geometry; they were developed more than ten years later than the foundations of geometry in Mizar, when Euclidean spaces were more thoroughly explored in MML.

The distinction between classical and abstract mathematics (i.e. the one based on ordinary axioms of set theory, and all those using the notion of a structure, respectively) is important from the viewpoint of the organization of the Mizar repository. We had to choose between two paths:

- it is possible to formulate Tarski's axioms without the use of a structure, and also set theory could be meaningless for that framework, only the classical logic with Mizar predicates is enough;
- the use of Mizar structures forces us paradoxically to use fundamentals of set theory – defining a signature of Tarski's plane needed to give a type of congruence of segments and betweenness relation, which was set-theoretic (Mizar language is typed, and in the earlier case one should also give a type at least to points, but it can be defined as Mizar `object`).

The latter was also chosen by us as the whole geometry in MML is written in abstract style (as the majority of MML) as structures in Mizar are present for a long time. Even if in ordinary mathematical tradition they are considered as ordered tuples, in the implementation in Mizar they are treated rather as partial functions, with selectors as arguments, and ordinary inheritance mechanism (with polymorphic enabled, which will be extensively used in our formalizations).

`definition`

```

struct (1-sorted) TarskiPlane
  (# carrier -> set,
   Betweenness -> Relation of [:the carrier, the carrier:], the carrier,
   Equidistance -> Relation of [:the carrier, the carrier:], [:the carrier, the carrier:] #);
end;
```

The betweenness relation can be treated as a ternary relation, but the choice of this concrete model was quite arbitrary as the difference between dealing with ternary relations and relations between ordered pairs and elements will not cause any major problems later (we use mainly predicates). Then we can use a type `POINT` of `S` just for elements of structures S , and predicates `between a,b,c` and `a,b equiv c,d` as `[[a,b],c]` in the `Betweenness` of `S` and `[[a,b],[c,d]]` in the `Equidistance` of `S`, respectively.

3 The First Part of Tarski's Axiomatics

Original version of Mizar formalization of Tarski's axioms done by William Richter with the help of `miz3` did not contained any existence proofs. This essentially caused the lack of the appropriate Mizar type. By using attributes instead of predicates we can have modular building of a complex structure, all other can be reused; hence in our Mizar article we focus on pure betweenness-equidistance part, not really mentioning the question of dimensions. We proved 44 theorems (properties of the predicates), with the Gupta's proof of Hilbert's I1 axiom (that two distinct points determine a line).

Our Mizar versions of Tarski's axioms have descriptive names, and follow the ones from SST (using \equiv for congruence of segments and B for betweenness relation):

- `CongruenceSymmetry` (A1):
 $\forall_{a,b} ab \equiv ba,$
- `CongruenceEquivalenceRelation` (A2):
 $\forall_{a,b,p,q,r,s} ab \equiv pq \wedge ab \equiv rs \Rightarrow pq \equiv rs,$
- `CongruenceIdentity` (A3):
 $\forall_{a,b,c} ab \equiv cc \Rightarrow a = b,$
- `SegmentConstruction` (A4):
 $\forall_{a,q,b,c} \exists_x B(q, a, x) \wedge ax \equiv bc,$
- `BetweennessIdentity` (A6):
 $\forall_{a,b} B(a, b, a) \Rightarrow a = b,$
- and `Pasch` (A7):
 $\forall_{a,b,p,q,z} B(a, p, z) \wedge B(p, q, z) \Rightarrow \exists_x B(p, x, b) \wedge B(q, x, a).$

One attribute has the form which substantially differs from SST version: in order to shorten the notation, we introduced technical predicate

```

definition
  let S be TarskiPlane;
  let a, b, c, x, y, z be POINT of S;
  pred a,b,c cong x,y,z means      :: GTARSKI1:def 3
    a,b equiv x,y & a,c equiv x,z & b,c equiv y,z;
end;
```

denoting essentially SSS predicate for triangles. Using this notion, SST axiom (A5) could be encoded as follows:

```

definition let S be TarskiPlane;
  attr S is satisfying_SAS means      :: GTARSKI1:def 9
    for a, b, c, x, a1, b1, c1, x1 being POINT of S holds
      a <> b & a,b,c cong a1,b1,c1 &
      between a,b,x & between a1,b1,x1 &
      b,x equiv b1,x1 implies c,x equiv c1,x1;
end;
```

that is,

$$\forall_{a,b,c,x,a',b',c',x'} (a \neq b \wedge ab \equiv a'b' \wedge bc \equiv b'c' \wedge ac \equiv a'c' \wedge \\ \wedge B(a, b, x) \wedge B(a', b', x') \wedge bx \equiv b'x') \Rightarrow cx \equiv c'x'.$$

Having separate attributes for distinct axioms could had already shown its usefulness in various geometrical settings. It could also allow later for defining equivalent axiom systems for Tarski geometry (and due to mechanism of clusters this equivalence will be obvious for the checker, once proven). But also additional attribute, `satisfying_Tarski-model`, was defined as a shorthand for the above seven axioms.

4 Adding Metric Ingredient

It is well known fact that every metric space can be equipped with the natural topology. This informally obvious mathematical property brings some unexpected difficulties when dealing with structures if automated proof-assistants play a role. Namely then, if one considers topological spaces in a quite natural way, that is $\langle U, \tau \rangle$, and metric space as $\langle U, d \rangle$, respectively, one can rather naturally merge both structures into common $\langle U, \tau, d \rangle$.

During the formalization of the Jordan Curve Theorem in Mizar, however, another approach was chosen (and pushed consequently until the successful finale): `Euclid 2` denoted metric space concerned with the Euclidean plane, and then special functor converting any metric space into the topological space was applied to obtain `TOP-REAL 2` (of course the conversion can be made for arbitrary natural number n , not necessarily 2, but Jordan curves deal with two-dimensional case). The basic signature for metric spaces are `MetrStruct`, where distance is a function defined on the Cartesian square of the carrier with the real values. Then, metrics (or pseudo-, quasi-, semimetrics, etc.) can be defined in terms of attributes [KLS90], that is properties of the distance function.

definition

```
struct (MetrStruct,TarskiPlane) MetrTarskiStr
  (# carrier -> set,
   distance -> Function of [:the carrier, the carrier:], REAL,
   Betweenness -> Relation of [:the carrier, the carrier:], the carrier,
   Equidistance -> Relation of [:the carrier, the carrier:], [:the carrier, the carrier:] #);
end;
```

Then we have two worlds merged: affine, where we have two Tarski's relations, and Euclidean, where in terms of distance function, we can have betweenness relation and the the measure for segments. We argue that, regardless of all the complications caused by merging structures (which increases the number of selectors, hence the chain of notions gets more complicated), such approach – not converting between two contexts, but rather to make reasoning in the world which is successor of both – allows for more flexible reuse of knowledge from two original areas.

definition let M be MetrTarskiStr;

```
attr M is naturally_generated means      :: GTARSKI1:def 15
  (for a, b, c being POINT of M holds between a,b,c iff b is_Between a,c) &
  (for a, b, c, d being POINT of M holds a,b equiv c,d iff dist (a,b) = dist (c,d));
end;
```

In merged structure, we want to have two segments congruent, if they are equal in terms of distance, and betweenness relation uses the predicate `is_Between`, also given in terms of the sum of distances.

Recalling the previous discussion on the structure merging, the construction of such a space from scratch (essentially more or less modified copy-and-paste work on the proof from [Try90]) can be avoided with the help of some useful tricks, investigating knowledge already present in MML with its possible reuse. For example, based on the geometry on the real line, appropriate geometrical structure from metric structure can be just an extension with properly defined distance.

definition

```
func TarskiSpace -> MetrTarskiStr equals  :: GTARSKI1:def 22
  the naturally_generated TarskiExtension of RealSpace;
coherence;
end;
```

Then, we can show that in such extension the metric is well defined, i.e. this space is reflexive, symmetric, and discerning. Furthermore, if we take into account “geometrical part”, all newly introduced Tarski's axioms are true.

5 Euclidean Plane Satisfies Tarski's Axioms

At the beginning, we had to construct an example of the structure which satisfies all these axioms.

definition let n be Nat;

```
func TarskiEuclidSpace n -> MetrTarskiStr equals      :: GTARSKI2:def 1
  the naturally_generated TarskiExtension of Euclid n;
end;
```

Table 1: The statistics of our formalizations

Items	Numbers in [RGA14]	Numbers in [CG16]
attributes	10	4
lines	1522	1926
kBytes	50	73
theorems	47	50

and `TarskiEuclid2Space` is just the shorthand for the above functor with $n = 2$.

First part of the work, which was rather easy, was to show that such Euclidean plane satisfies seven axioms from [RGA14]. It was expressed in terms of cluster registration:

```

registration
  cluster TarskiEuclid2Space -> satisfying_Tarski-model;
  coherence;
end;

```

Then, we defined the remaining four axioms, with both descriptive names and those corresponding to SST namespace, as follows:

```

notation let S be TarskiPlane;
  synonym S is satisfying_Lower_Dimension_Axiom for S is satisfying_A8;
  synonym S is satisfying_Upper_Dimension_Axiom for S is satisfying_A9;
  synonym S is satisfying_Euclid_Axiom for S is satisfying_A10;
  synonym S is satisfying_Continuity_Axiom for S is satisfying_A11;
end;

```

Of course, their meaning is just as in SST. Then, finally we can deal with the remaining axioms:

```

registration
  cluster TarskiEuclid2Space -> satisfying_Lower_Dimension_Axiom satisfying_Upper_Dimension_Axiom
    satisfying_Euclid_Axiom satisfying_Continuity_Axiom;
end;

```

Some of the proofs were proven based on the space with the dimension 2 (mainly those done by the second author), in few places we used Isabelle development and gave our Mizar proofs accordingly. The summary of formalizations in two Mizar files can be found in Table 2.

6 Conclusions

Although the history of the development of the axiom system for geometry by Tarski is not very clear from the beginnings (as the early works by Tarski seem to be postponed by the World War II), and even if the ultimate source of information is

The book by Schwabhäuser, Szmielew and Tarski, reissued with the foreword of Michael Beeson, attracted recently a lot of focus from automated deduction systems. The remarkable item here is of course Narboux’s formal development of Tarski’s system with the use of Coq [Nar07]. We are planning to include some interesting results from GeoCoq in the Mizar system; Nakasho’s et al. MML symbol reference system¹ offers quite user-friendly interface for browsing appropriate content. Of course, GeoCoq provides ready-to-use list of notions, which are connected only with the Tarski’s system – that makes the browsing more convenient. Of course, after rescaling all definitions and theorems can be browsed within MML providing a look in the style of GeoCoq project.

We have created in [RGA14] (and GTARSKI2 [CG16]) complete formal axiomatization of Tarski’s geometry which, at least in our opinion, has the advantage of higher readability for ordinary mathematicians than e.g. Coq or Prover9 proof objects. In the same it is tightly connected with another axiomatization of Euclidean plane, due to Hilbert, already available in MML. Important part of our development was the proof that the real Euclidean plane satisfies all Tarski’s axioms.

¹The system can be browsed at <http://webmizar.cs.shinshu-u.ac.jp/mmlfe/current/> with the official version of the Mizar system.

References

- [BBG15] Bancerek G., Byliński Cz., Grabowski A., Kornilowicz A., Matuszewski R., Naumowicz A., Pał K., Urban J.: Mizar: state-of-the-art and beyond, Intelligent Computer Mathematics, *Lecture Notes in Computer Science* 9150, pp. 261–279 (2015)
- [Bee16] Beeson M.: Mixing computations and proofs, *Journal of Formalized Reasoning*, 9(1), pp. 71–99 (2016)
- [BW14] Beeson M., Wos L.: OTTER proofs in Tarskian geometry, in Proceedings of IJCAR 2014, *Lecture Notes in Computer Science*, 8562, pp. 495–510 (2014)
- [Bor60] Borsuk K., Szmielew W.: *Foundations of Geometry*, North-Holland (1960)
- [BN12] Braun G., Narboux J.: From Tarski to Hilbert, in *Automated Deduction in Geometry*, T. Ida and J. Fleuriot (eds.) Proceedings of ADG 2012 (2012)
- [CG16] Coghetto R., Grabowski A.: Tarski geometry axioms – part II, *Formalized Mathematics*, 24(2), available at <http://mizar.org/library/tarski/> (2016)
- [Gra14] Grabowski A.: Efficient rough set theory merging, *Fundamenta Informaticae*, 135(4), pp. 371–385 (2014)
- [Gra15] Grabowski A.: Mechanizing complemented lattices within Mizar type system, *Journal of Automated Reasoning*, 55(3), pp. 211–221 (2015)
- [GKN10] Grabowski A., Kornilowicz A., Naumowicz A.: Mizar in a nutshell, *Journal of Formalized Reasoning*, 3(2), 153–245 (2010)
- [Gup65] Gupta H.N.: Contributions to the axiomatic foundations of geometry, PhD thesis, University of California, Berkeley (1965)
- [Hil80] Hilbert D.: *The Foundations of Geometry*, Chicago: Open Court, 2nd ed. (1980)
- [KLS90] Kanas, S., Lecko, A., Startek M.: Metric spaces, *Formalized Mathematics*, 1(3), pp. 607–610 (1990)
- [Kor15] Kornilowicz A.: Definitional expansions in Mizar, *Journal of Automated Reasoning*, 55(3), pp. 257–268 (2015)
- [Mak12] Makarios T.: A mechanical verification of the independence of Tarski’s Euclidean Axiom, MSc thesis (2012)
- [MF03] Meikle L., Fleuriot J.: Formalizing Hilbert’s Grundlagen in Isabelle/Isar, in Proceedings of TPHOLs’03, *Lecture Notes in Computer Science*, 2758, pp. 319–334 (2003)
- [Nar07] Narboux J.: Mechanical theorem proving in Tarski’s geometry, in *Automated Deduction in Geometry*, F. Botana and T. Recio (eds.), *Lecture Notes in Computer Science*, 4869, pp. 139–156 (2007)
- [NK09] Naumowicz A., Kornilowicz A.: A brief overview of Mizar, in *Theorem Proving in Higher Order Logics 2009*, S. Berghofer, T. Nipkow, Ch. Urban, M. Wenzel (Eds.), *Lecture Notes in Computer Science*, 5674, 67–72 (2009)
- [RGA14] Richter W., Grabowski A., Alama J.: Tarski geometry axioms, *Formalized Mathematics*, 22(2), pp. 167–176 (2014)
- [SST83] Schwabhäuser W., Szmielew W., Tarski A.: *Metamathematische Methoden in der Geometrie*, Springer-Verlag (1983)
- [Tar59] Tarski A.: What is elementary geometry?, in *Studies in Logic and the Foundations of Mathematics*, North-Holland, pp. 16–29 (1959)
- [TG99] Tarski A., Givant S.: Tarski’s system of geometry, *The Bulletin of Symbolic Logic*, 5(2), pp. 175–214 (1999)
- [Try90] Trybulec, W.A.: Axioms of incidence, *Formalized Mathematics*, 1(1), pp. 205–213 (1990)
- [Wie12] Wiedijk F.: A synthesis of the procedural and declarative styles of interactive theorem proving, *Logical Methods in Computer Science*, 8(1):30 (2012)

Formalization of the prime number theorem and Dirichlet’s theorem

Mario Carneiro
Pure and Applied Logic program
Carnegie Mellon University, Pittsburgh PA, USA
di.gama@gmail.com

Abstract

We present the formalization of Dirichlet’s theorem on the infinitude of primes in arithmetic progressions, and Selberg’s elementary proof of the prime number theorem, which asserts that the number $\pi(x)$ of primes less than x is asymptotic to $x/\log x$, within the proof system Metamath.

1 Introduction

Dirichlet’s theorem, or the Dirichlet prime number theorem, states that for any $N \in \mathbb{N}$ and $A \in \mathbb{Z}$ such that $\gcd(A, N) = 1$, there are infinitely many primes in the progression $A + kN$, or equivalently there are infinitely many $p_k \equiv A \pmod{N}$. Euler was the first to make progress on this theorem, proving it in the case $A = 1$, and it was shown in full generality by Dirichlet in 1837 [Dir37].

The prime number theorem gives an overall order of growth of the number of primes less than x . Letting $\pi(x)$ denote the number of primes in the interval $[1, x]$ (where x is not necessarily an integer), the prime number theorem asserts that $\pi(x) \sim \frac{x}{\log x}$. This was first conjectured by Legendre in 1797, and was first proven using complex analysis and the zeta function by Jacques Hadamard and Charles Jean de la Vallée-Poussin in 1896. Two “elementary” proofs were later discovered by Selberg and Erdős in 1949 [Sel49, Erd49].

The first formal proof of the prime number theorem was written by Jeremy Avigad et. al. in 2004 [Avi07], in the Isabelle proof language, following Selberg’s proof, a landmark result for mathematics formalization. Hadamard and Vallée-Poussin’s proof was formalized in HOL Light by John Harrison in 2009 [Har09]. John Harrison also later formalized Dirichlet’s theorem in HOL Light, in 2010 [Har10].

Metamath is a formal proof language and computer verification software developed for the purpose of formalizing mathematics in a minimalistic foundational theory [Meg07]. On May 12, 2016 and June 1, 2016 respectively, the author formally verified the following theorems in the Metamath formal system:

Theorem 1 (dirith, Dirichlet’s theorem).

$$N \in \mathbb{N} \wedge A \in \mathbb{Z} \wedge \gcd(A, N) = 1 \rightarrow \{p \in \mathbb{P} \mid N \mid (p - A)\} \approx \mathbb{N}$$

Theorem 2 (pnt, The prime number theorem).

$$\left(x \in (1, \infty) \mapsto \frac{\pi(x)}{x/\log x} \right) \rightsquigarrow 1$$

The latter expression is Metamath’s notation for $\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\log x} = 1$, restricted to $x > 1$, which is the domain of definition of the function.

These two theorems are interesting formalization targets as they both have simple statements and “deep” proofs, and they are also both members of the “Formalizing 100 theorems” list maintained by Freek Wiedijk [Wie16], which tracks formalizations of 100 of the most famous theorems in mathematics.

Both proofs were written concurrently, over the course of about seven weeks between April 7 and June 1, 2016. This was done mostly because both theorems are in the same general subject (elementary number theory) and required similar techniques (mostly asymptotic approximation of finite sums of reals). The primary informal text used for the proof was Shapiro [Sha83], which devotes a section to Dirichlet’s theorem and the whole final chapter to Selberg and Erdős’s proof of the prime number theorem.

2 Background

The present work is only a broad overview of the problem and proof method. Interested readers are invited to consult the main theorems `pnt` and `dirith` at [Met16], where the exact proof is discussed in detail.

The main arithmetic functions used in the formalization are:

$$\begin{aligned} \pi(x) &= |\{p \in \mathbb{P} \mid p \leq x\}| = \sum_{p \leq x} 1 & \theta(x) &= \sum_{p \leq x} \log p \\ \Lambda(n) &= \begin{cases} \log p & \exists p \in \mathbb{P}, k > 0 : n = p^k \\ 0 & o.w. \end{cases} & \psi(x) &= \sum_{n \leq x} \Lambda(n) \end{aligned}$$

Additionally, the Möbius function $\mu(n)$ is a very useful tool in sum manipulations. It is the unique multiplicative function such that $\mu(1) = 1$ and $\sum_{d|n} \mu(d) = 0$ for $n > 1$. This yields the Möbius inversion formula: if $f(n) = \sum_{d|n} g(d)$, then $g(n) = \sum_{d|n} \mu(d)f(d)$. Since $|\mu(n)| \leq 1$, this is a very powerful technique for estimating sums “by inversion”.

The proof of Hadamard and Vallée-Poussin relies on some deep theorems in complex analysis, such as Cauchy’s theorem, which were not available at the time of this formalization, so instead we targeted the “elementary” proof discovered half a century later semi-independently by Erdős and Selberg. The key step in both proofs is the Selberg symmetry formula:

Theorem 3 (`selberg`, Selberg symmetry formula).

$$\sum_{n \leq x} \Lambda(n) \log n + \sum_{uv \leq x} \Lambda(u)\Lambda(v) = 2x \log x + O(x).$$

In Selberg’s proof, we leverage this theorem to produce a bound on the residual $R(x) = \psi(x) - x$:

Theorem 4 (`pntlog2bnd`).

$$|R(x)| \log^2 x \leq 2 \sum_{n \leq x} |R(x/n)| \log n + O(x \log x).$$

The goal is to show $\pi(x) \sim \frac{x}{\log x}$, but it is easily shown that $\psi(x) \sim \theta(x) \sim \pi(x) \log x$, so it is equivalent to show that $\psi(x) \sim x$, or $R(x)/x \rightarrow 0$, to establish the PNT. Given an eventual bound $|R(x)| \leq ax$ and the estimation $\sum_{n \leq x} \frac{\log n}{n} = \frac{1}{2} \log^2 x + O(\frac{\log x}{x})$, an application of Theorem 4 reproduces the original estimate $|R(x)| \leq ax + o(x)$, but using improved bounds on $R(x)$ on small intervals we can improve the estimate to $|R(x)| \leq (a - ca^3)x + o(x)$ for a fixed constant c , which produces a sequence of eventual bounds approaching zero, which proves $R(x)/x \rightarrow 0$ as desired.

In Dirichlet’s theorem, the focal point is instead the Dirichlet characters mod N , which are group homomorphisms from $(\mathbb{Z}/N\mathbb{Z})^*$ to \mathbb{C}^* , extended to $\mathbb{Z}/N\mathbb{Z}$ with value 0 at non-units, but the general theme of estimation of sums involving μ, Λ, \log and the characters $\chi(n)$ is the same.

3 Formalization

In keeping with Metamath’s tradition of minimal complexity, we used a minimum of definition. Asymptotic estimations are reduced to the class $O(1)$ of eventually bounded functions, partial functions $\mathbb{R} \rightarrow \mathbb{C}$ such that for some c, A , $x \geq c$ implies $|f(x)| \leq A$. An equation such as $f \in O(g)$ is rewritten as $f/g \in O(1)$ (which is correct

as long as g is eventually nonzero, which is always true in cases of interest), and similarly $f \in o(g)$ is rewritten as $f/g \rightsquigarrow 0$.

A few finite summation theorems take us a long way; two number-theory specific summation theorems are the following divisor sum commutations:

$$\sum_{k|n} \sum_{d|k} A(k, d) = \sum_{d|n} \sum_{m|n/d} A(dm, d)$$

$$\sum_{n \leq x} \sum_{d|n} A(n, d) = \sum_{d \leq x} \sum_{m \leq n/d} A(dm, d)$$

A small amount of calculus was used in the proof, mostly through the following sum estimation theorem, which for example evaluates $\sum_{n \leq x} \frac{\log n}{n} = \frac{1}{2} \log^2 x + O\left(\frac{\log x}{x}\right)$:

Theorem 5 (dvfsumrlim). *If F is a differentiable function with $F' = f$, and f is a positive decreasing function that converges to zero, then $g(x) = \sum_{n \leq x} f(n) - F(x)$ converges to some L and $|g(x) - L| \leq f(x)$.*

4 Comparison and Conclusion

Table 1: Comparison of the present proof with [Har10, Avi07]. “?” marks an estimated or unknown value.

	Dirichlet (author)	PNT (author)	Dirichlet [Har10]	PNT [Avi07]
Total time spent	2 weeks	5 weeks	5 days	12 weeks?
Lines of code	3595	5100	1183	19713
Compressed bytes (gzip)	109683	156226	11762	97470
Informal text	10 pp.	37 pp.	192 lines	37 pp.
Informal text (gzip)	5500?	20350?	2524	20350?
de Bruijn factor	19.9?	7.67?	4.66	4.78?
Verification time	0.18 s	0.23 s	450 s	1800 s?

The comparison of parallel proof attempts in different systems is usually confounded by the many other factors, so these statistics should not be given undue credence. According to [Avi07], Avigad’s PNT project was a year-long project by four people, with the majority of the work happening during one summer, while this was a solo project over about seven work weeks. Dirichlet’s theorem is 10 pages of informal text of [Sha83], and the PNT is 37 pages. Although the number of lines in the current proofs seem competitive, this is lost in the gzipped version, because the stored Metamath proof is already largely compressed, while the Isabelle and HOL scripts are plain text.

The de Bruijn factors for this work had to be estimated because the TeX source for the informal text was not available, but indications suggest that it fares poorly with comparatively large factors 19.9 and 7.67, respectively. However, when reading these statistics it is important to realize that Metamath stores *proofs*, not *proof scripts* like Isabelle and HOL. Every inference in the proof is an axiom or theorem of the system, and no proof searches are conducted by the verifier. This is reflected in the incredibly small verification time, which is normal for Metamath proofs. We do not have exact data on verification time for HOL Light, but it is believed to be on the order of minutes to hours.

These proofs are important milestones for the Metamath project. They demonstrate that even the largest of formalization projects in high level languages can also be conducted in a “full transparency”-style system like Metamath, with entirely worked-out proofs and with all automation offloaded from the verifier to the proof generation.

References

- [Dir37] Dirichlet, P. G. L.: Beweis des Satzes, dass jede unbegrenzte arithmetische Progression, deren erstes Glied und Differenz ganze Zahlen ohne gemeinschaftlichen Factor sind, unendlich viele Primzahlen enthält. Abhand. Ak. Wiss. Berlin **48**, 313–342 (1837)

- [Sel49] Selberg, A.: An elementary proof of the prime-number theorem. *Ann. of Math. (2)*, Vol. 50, pp. 305–313; reprinted in *Atle Selberg Collected Papers*, Springer–Verlag, Berlin Heidelberg New York, 1989 **1**, 379–387 (1949)
- [Erd49] Erdős, P.: On a new method in elementary number theory which leads to an elementary proof of the prime number theorem. *Proc. Nat. Acad. Scis. U.S.A.* **35**, 374–384 (1949)
- [Avi07] Avigad, J., Donnelly, K., Gray, D., Raff, P.: A formally verified proof of the prime number theorem. *ACM Trans. Comput. Logic* **9** (1:2), 1–23 (2007)
- [Har09] Harrison, J.: Formalizing an analytic proof of the Prime Number Theorem (dedicated to Mike Gordon on the occasion of his 60th birthday). *Journal of Automated Reasoning*, 43:243–261 (2009)
- [Har10] Harrison, J.: A formalized proof of Dirichlet’s theorem on primes in arithmetic progression. *Journal of Formalized Reasoning*, [S.l.], **2** (1), 63–83 (2010)
- [Wie16] Wiedijk, F.: Formalizing 100 Theorems, <http://www.cs.ru.nl/~freek/100/> (accessed 20 May 2016)
- [Meg07] Megill, N.: *Metamath: A Computer Language for Pure Mathematics*. Lulu Publishing, Morrisville, North Carolina (2007)
- [Sha83] Shapiro, H.: *Introduction to the theory of numbers*. John Wiley & Sons Inc., New York (1983)
- [Met16] Metamath Proof Explorer, <http://us.metamath.org/mpegif/mmset.html> (accessed 20 May 2016)

Topological Foundations for a Formal Theory of Manifolds

Karol Pałk
Institute of Computer Science,
University of Białystok, Poland
karol@mizar.org

Abstract

Topological manifolds form an important class of topological spaces with applications throughout mathematics. However, the development of this scientific area, even at the initial stage, requires non-trivial Brouwer's theorems: the fixed point theorem, the topological invariance of degree, and the topological invariance of dimension, where each of them is provided for n -dimensional case.

We present a formalization, that is checked in the Mizar system, of several results in algebraic topology that is sufficient to show the basic properties of manifolds with boundary of dimension n .

Keywords: Topological manifolds, Formalization, Mizar.

1 Introduction

The Mizar Mathematical Library (MML) [1] can be considered as a compendium of proven theorems from textbooks and papers, together with the necessary definitions. However, in contrast to the many informal considerations, different approaches to the same theory are generally merged in the MML. It is a consequence of the fact that the overall goal is not only expanding MML by a new formalization – though this still is an important part – but also facilitating the further development of the database based upon the already existing formalizations. Therefore, some theorems or even whole theories contained in the MML are adapted to the needs of new formalizations [4, 6]. To illustrate such situations we can consider the development of topological manifolds in the MML.

2 Development of Topological Manifolds

The first Mizar style [2] formalization of a topological manifold in Mizar was created by M. Riccardi [17, 18]. There, a topological manifold was defined as a topological space that is second-countable Hausdorff and locally Euclidean, i.e. the space resembles locally an open ball of the n -dimensional Euclidean space \mathcal{E}^n near each point for some n . M. Riccardi obtained several basic facts, e.g. if two topological spaces are homeomorphic and one of them is an n -dimensional manifold, then the other one inherits its dimension; the work also showed some examples of topological manifolds: whole planes, spheres in Euclidean space. However, in his approach the parameter n was given a priori. As a justification for this approach we recall that every connected component of a compact manifold is a manifold with a determined dimension. However this fact cannot be clearly formulated in this approach.

To resolve this problem, we can consider a slightly more general definition, where for each point there exist some n and a neighborhood of the point being homeomorphic to an open ball of \mathcal{E}^n . Moreover, if we replace an open ball by a closed ball in the definition (see [3]) we can consider a topological manifold with a boundary, but also without it. Such a generalization of the first approach was made by K. Pałk [8]. He proved that every

connected component of a compact manifold is a manifold with a determined dimension and also he proved the dependence between interior and boundary points. He showed for an n -dimensional manifold that its interior is a manifold without boundary of dimension n and its boundary is a manifold without boundary of dimension $n - 1$. Additionally, he showed that the Cartesian product of manifolds also forms a manifold whose dimension is the sum of the dimensions of its factors and also determined the interior and boundary of this product.

3 Contributions

To establish these results, K. Pąk had to develop mainly the algebraic topology in the MML. It includes theory of simplicial complexes in real linear spaces and its barycenter subdivision. The theory was necessary to provide Brouwer's fixed point theorem based on Sperner's lemma [13, 16]. The theorem has been used in the formalization of the Jordan curve theorem in Mizar [5]. But for this purpose, the 2-dimensional case was sufficient. Therefore, this statement has been provided by A. Kornilowicz, only for this case using basic arguments concerning the fundamental groups of the respective spaces [7]. Note that this approach for higher-dimensional cases requires incomparably more difficult facts about these groups. Another approach based on Sperner's lemma requires only intuitively clear facts about the standard n -dimensional simplex and its arbitrarily small subdivision (see [11, 12]). Additionally, the simplex structure is explored in one of the approaches to prove Brouwer's invariance of the domain theorem that is helpful to distinguish points from the internal and the boundary of a manifold (see [3]).

Obviously the realization of the selected approach required to develop several other areas of topology and algebra. The most important of these are the small inductive dimension of topological spaces [9, 10]; an affine independence of points and barycentric coordinates in an affine space including the theory that barycentric coordinates with respect to an affine independence set is continuous [14]; the rotation group of Euclidean topological spaces [15].

Acknowledgements

The paper has been financed by the resources of the Polish National Science Center granted by decision n^oDEC-2012/07/N/ST6/02147.

References

- [1] G. Bancerek and P. Rudnicki. Information Retrieval in MML. In A. Asperti, B. Buchberger, and J.H.Davenport, editors, *Proc. of Mathematical Knowledge Management 2003*, volume 2594, page 119–131. Springer, Heidelberg, 2003.
- [2] G. Bancerek, C. Bylinski, A. Grabowski, A. Kornilowicz, R. Matuszewski, A. Naumowicz, K. Pąk, and J. Urban. Mizar: State-of-the-art and Beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer, 2015.
- [3] R. Engelking. *General Topology*. PWN - Polish Scientific Publishers, Warsaw, 1977.
- [4] A. Grabowski and Ch. Schwarzweller. Improving Representation of Knowledge within the Mizar Library. *Studies in Logic, Grammar and Rhetoric*, 18(31):35–50, 2009.
- [5] A. Grabowski, A. Kornilowicz, and Adam Naumowicz. Mizar in a Nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
- [6] A. Grabowski and Ch. Schwarzweller. Revisions as an essential tool to maintain mathematical repositories. In *Proceedings of the 14th Symposium on Towards Mechanized Mathematical Assistants: 6th International Conference, Calculemus '07 / MKM '07*, pages 235–249, Berlin, Heidelberg, 2007. Springer-Verlag.
- [7] A. Kornilowicz and Y. Shidama. Brouwer Fixed Point Theorem for Disks on the Plane. *Formalized Mathematics*, 13(2):333–336, 2005.
- [8] K. Pąk. Topological Manifolds. *Formalized Mathematics*, 22(2):179–186, 2014.

- [9] K. Pałk. Small inductive dimension of topological spaces. *Formalized Mathematics*, 17(3):207–212, 2009.
- [10] K. Pałk. Small inductive dimension of topological spaces. Part II. *Formalized Mathematics*, 17(3):219–222, 2009.
- [11] K. Pałk. Abstract simplicial complexes. *Formalized Mathematics*, 18(1):95–106, 2010.
- [12] K. Pałk. Sperner’s lemma. *Formalized Mathematics*, 18(4):189–196, 2010.
- [13] K. Pałk. Brouwer fixed point theorem for simplexes. *Formalized Mathematics*, 19(3):145–150, 2011.
- [14] K. Pałk. Continuity of barycentric coordinates in Euclidean topological spaces. *Formalized Mathematics*, 19(3):139–144, 2011.
- [15] K. Pałk. The rotation group. *Formalized Mathematics*, 20(1):23–29, 2012.
- [16] K. Pałk. Brouwer invariance of domain theorem. *Formalized Mathematics*, 22(1):21–28, 2014.
- [17] M. Riccardi. The definition of topological manifolds. *Formalized Mathematics*, 19(1):41–44, 2011.
- [18] M. Riccardi. Planes and spheres as topological manifolds. Stereographic projection. *Formalized Mathematics*, 20(1):41–45, 2012.

Registrations vs Redefinitions in MIZAR

Artur Kornilowicz

Institute of Informatics, University of Białystok
K. Ciołkowskiego 1M, 15-245 Białystok, Poland
arturk@mizar.org

Abstract

In this paper we briefly discuss two constructions of the MIZAR language – redefinitions and registrations. We focus on practical aspects of using them in MIZAR texts to be effectively processed by the MIZAR VERIFIER. We describe situations when redefinitions can be and should be replaced by corresponding registrations.

1 Introduction

The MIZAR system [Ban15, Gra15] is a computer system invented for computer-assisted verification of mathematical papers. One of the main components of the system is the MIZAR language – a formal language designed for writing mathematical papers readable for humans and effectively processed by computers. The language consists of rules for writing first-order mathematical formulas, proofs, and also syntactic constructions to launch specialized algorithms increasing computational power of the VERIFIER (e.g. term identifications, term reductions [Kor13], flexary connectives [Kor15:b], property registrations [Nau04], etc.).

In this paper, in Sec. 2, we focus on two particular constructions – redefinitions and registrations, especially how they can be used in MIZAR texts to be effectively processed by the MIZAR VERIFIER. In Sec. 3, we present two examples taken from the Mizar Mathematical Library which illustrate our discussion.

2 Redefinitions and Registrations

In MIZAR, redefinitions can be used for three different purposes: a) to specialize (if provable) result types of defined functors and modes, b) to change definitia of notions (predicates, attributes, functors, and modes), and c) to declare properties of particular constructors [Nau04]. The first application influences the identification of operations and types, the second application can be used for the processing definitional expansions [Kor15:a], and the third one can be used for the justification of chosen statements automatically (without explicit references to appropriate theorems). An important feature of the processing redefinitions performed by the VERIFIER is that the order of redefinitions accessible in a given text (redefinitions imported from the database MML or declared in the text) is important. The last redefinition of a notion applicable for given arguments is applied.

On the other hand, registrations can be used for three other purposes¹: a) to register the existence of objects satisfying required properties written as adjectives (existential registrations), b) to declare that some objects possess chosen properties (functorial registrations), and c) to declare that all elements of some type which satisfy a set of properties satisfy also another set of properties (conditional registrations). A significant feature of the processing registrations is that the order of registrations accessible in a given text plays no role (in opposite to the processing redefinitions), all registrations which can be applied for given arguments are applied.

From the point of view of this paper, the fact that only the last redefinition of a given notion is considered in the process of computing types of objects, but all registrations are, is crucial. It is the fundamental reason for

¹The MIZAR word **registration** is also used in other contexts, to introduce items like: **identify**, **sethood**, **reduce**.

which registrations should be used instead of redefinitions, whenever possible. The question now is what are the situations when redefinitions can be replaced by corresponding registrations. Before we answer this question, let us mention that the MIZAR system provides two ways to introduce new types: not expandable (really new constructors) and expandable (shortcuts for collections of adjectives assigned to radix types)². Now, we can formulate a rule which defines situations when redefinitions can be replaced by registrations: *If the result type of a functor and the mother type of a mode to which the original type of the functor and the mode is redefined is an expandable type, then such redefinitions are replaceable by a functorial registration in the case of functors and by a conditional registration in the case of modes.* Practical examples which depict the rule are presented in the next section.

3 Examples from the Mizar Mathematical Library

3.1 Functors

Let us consider the functor `Balls(x)` [Sko98] which defines a family of balls in a topological space generated by a metric space:

```
definition
  let M be non empty MetrStruct, x be Point of TopSpaceMetr(M);
  func Balls(x) -> Subset-Family of TopSpaceMetr(M)
  ...
end;
```

To declare that `Balls(x)` constitutes a base in the space, the authors used the redefinition:

```
definition
  let M be non empty MetrSpace, x be Point of TopSpaceMetr(M);
  redefine func Balls(x) -> Basis of x;
end;
```

But, because the mode `Basis of x` [Try97] is defined as an expandable mode:

```
definition
  let T be non empty TopStruct, x be Point of T;
  mode Basis of x is open x-quasi_basis Subset-Family of T;
end;
```

the redefinition can be replaced by the functorial registration:

```
registration
  let M be non empty MetrSpace, x be Point of TopSpaceMetr(M);
  cluster Balls(x) -> open x-quasi_basis;
end;
```

3.2 Modes

To exemplify how redefinitions of modes can be replaced by conditional registrations, let us consider the mode `Element of D` [Ban92], where `D` is a set containing only trees:

```
definition
  let D be constituted-Trees non empty set;
  redefine mode Element of D -> Tree;
end;
```

Because the mode `Tree` [Ban90] is defined as an expandable mode:

```
definition
  mode Tree is non empty Tree-like set;
end;
```

²In fact, there is another way to define new types – structures, but it is not relevant to the topic.

the redefinition can be replaced by the conditional registration:

```
registration
  let D be constituted-Trees non empty set;
  cluster -> non empty Tree-like for Element of D;
end;
```

3.3 Experiments

To detect all cases in the Mizar Mathematical Library when redefinitions of functors and modes could be replaced by corresponding registration, a special tool has been implemented by the author of the paper. In the MIZAR Version 8.1.05 working with the MML Version 5.37.1267, 89 possible replacements of redefinitions were found.³ A revision [Ala11, Gra07, Pak14] of the MML was proposed to the Library Committee.

References

- [Ala11] Alama, J., Kohlhase, M., Mamane, L., Naumowicz, A., Rudnicki, P., Urban, J.: Licensing the Mizar Mathematical Library. In: Davenport, J.H. et al. (eds.) *Proceedings of the 18th Calculemus and 10th International Conference on Intelligent Computer Mathematics*. Lecture Notes in Computer Science, vol. 6824, 149–163. Springer-Verlag, Berlin, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-22673-1_11
- [Ban90] Bancerek, G.: Introduction to trees. *Formalized Mathematics* 1(2), 421–427 (1990), http://fm.mizar.org/1990-1/pdf1-2/trees_1.pdf
- [Ban92] Bancerek, G.: Sets and functions of trees and joining operations of trees. *Formalized Mathematics* 3(2), 195–204 (1992), http://fm.mizar.org/1992-3/pdf3-2/trees_3.pdf
- [Ban15] Bancerek, G., Byliński, C., Grabowski, A., Kornilowicz, A., Matuszewski, R., Naumowicz, A., Pał, K., Urban, J.: Mizar: State-of-the-art and beyond. In: Kerber, M. et al. (eds.): *Intelligent Computer Mathematics – International Conference, CICM 2015, Washington, DC, USA, Proceedings*, Lecture Notes in Computer Science, vol. 9150, 261–279, Springer (2015), http://dx.doi.org/10.1007/978-3-319-20615-8_17
- [Gra15] Grabowski, A., Kornilowicz, A., Naumowicz, A.: Four decades of Mizar. *Journal of Automated Reasoning* 55(3), 191–198 (2015), <http://dx.doi.org/10.1007/s10817-015-9345-1>
- [Gra07] Grabowski, A., Schwarzweller, C.: Revisions as an essential tool to maintain mathematical repositories. In: *Proceedings of the 14th Symposium on Towards Mechanized Mathematical Assistants: 6th International Conference*. 235–249. *Calculemus '07 / MKM '07*, Springer-Verlag, Berlin, Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-73086-6_20
- [Kor13] Kornilowicz, A.: On rewriting rules in Mizar. *Journal of Automated Reasoning* 50(2), 203–210 (2013), <http://dx.doi.org/10.1007/s10817-012-9261-6>
- [Kor15:a] Kornilowicz, A.: Definitional expansions in Mizar. *Journal of Automated Reasoning* 55(3), 257–268 (2015), <http://dx.doi.org/10.1007/s10817-015-9331-7>
- [Kor15:b] Kornilowicz, A.: Flexary connectives in Mizar. *Computer Languages, Systems & Structures* 44, 238–250 (2015), <http://dx.doi.org/10.1016/j.cl.2015.07.002>
- [Nau04] Naumowicz, A., Byliński, C.: Improving Mizar texts with properties and requirements. In: Asperti, A. et al. (eds.) *Mathematical Knowledge Management, Third International Conference, Proceedings*. Lecture Notes in Computer Science, vol. 3119, 290–301 (2004), http://dx.doi.org/10.1007/978-3-540-27818-4_21
- [Pak14] Pał, K.: Improving legibility of natural deduction proofs is not trivial. *Logical Methods in Computer Science* 10(3), 1–30 (2014), [http://dx.doi.org/10.2168/LMCS-10\(3:23\)2014](http://dx.doi.org/10.2168/LMCS-10(3:23)2014)
- [Sko98] Skorulski, B.: First-countable, sequential, and Frechet spaces. *Formalized Mathematics* 7(1), 81–86 (1998), <http://fm.mizar.org/1998-7/pdf7-1/frechet.pdf>

³Computations were carried out at the Computer Center of University of Białystok <http://uco.uwb.edu.pl>

[Try97] Trybulec, A.: Baire spaces, Sober spaces. Formalized Mathematics 6(2), 289–294 (1997), http://fm.mizar.org/1997-6/pdf6-2/yellow_8.pdf

Linking to Compound Conditions in Mizar

Adam Naumowicz

Institute of Informatics, University of Białystok
Konstantego Ciołkowskiego 1M, 15-245 Białystok, Poland
adamn@math.uwb.edu.pl

Abstract

The processing and analysis of the Mizar library has been performed using the infrastructure of the University of Białystok High Performance Computing Center.

In the current Mizar language, direct linking to statements formulated as compound conditions is prohibited. This particular language feature has often been considered by the users as an unnecessary disruption of the natural course of proof steps. In this paper, we present an analysis of the linking structure in the current contents of the Mizar Mathematical Library which provides statistical data needed to implement a usability-based solution to this problem and propose corresponding changes in the library.

1 Introduction

One of the main design principles of the Mizar [Ban15] language has been to enable writing formal mathematical documents checked by a computer for syntactical, semantical and logical correctness, while at the same time the language should as much as possible resemble standard mathematical papers. The language's richness of features adopted from the natural language contributed to its popularity, which in turn allowed to collect a vast body of formalized mathematical data available as the Mizar Mathematical Library (MML) over the four decades of Mizar's active use [Gra15]. However, there is still a lot of space for improvement, adding new features and eliminating some of the language's idiosyncrasies which hinder its comfortable use. A detailed analysis highlighting a number of difficulties posed by the complexity of the language was presented in a paper by Cairns and Gow [Cai04]. Among such problematic features which users often complain about is that direct linking to choice statements is prohibited in current Mizar. It is perceived as an unnecessary disruption of the natural course of proof steps. In fact, this problem have re-appeared several times in the Mizar Forum mailing list discussions¹. Apparently, this is a more general issue which concerns linking to many sorts of statements formulated as compound conditions.

2 Compound Conditions in Mizar

A brief look at the Mizar syntax² shows that there are quite a few constructs of the language that make use of compound conditions. A complete list of such constructs includes: `Loci-Declaration`, `Case`, `Suppose`, `Generalization`, `Collective-Assumption`, `Existential-Assumption`, and finally the `ChoiceStatement`. Their grammar specifications either explicitly mention the `that` and `and` keywords, or use the `Conditions` nonterminal symbol:

```
Conditions = "that" Proposition { "and" Proposition } .
```

¹See e.g. the following discussion threads in the Mizar Forum mailing list's archives: <http://mizar.uwb.edu.pl/forum/archive/0104/msg00002.html>, and <http://mizar.uwb.edu.pl/forum/archive/0203/msg00001.html>.

²The Mizar language is described by its grammar available in the system's distribution (`syntax.txt` and `syntax.xml` files) as well as on-line on the project's website: <http://mizar.org/language/mizar-grammar.xml>

Using a Weakly-Strict Mizar (WSM) [Nau16, Byl12] parser³, we have collected some statistical data which shows the usage of these constructs in the MML version (5.33.1254) distributed together with the current official Mizar version (8.1.04)⁴. The table below shows the number of occurrences of these language constructs.

Table 1: Occurrences of compound conditions in current MML.

Language: construct	Assumption	Loci-declaration/ Generalization	Case	Suppose	Existential assumption	Choice statement
With conditions:	17311	13811	59	587	2761	58909
W/o conditions:	91914	136733	4782	32136	n/a	n/a

As we can see, some of the constructs are significantly less frequent than others. Notably, **case** and **suppose** are used only in special kinds of proofs (reasoning *per cases*) and the use of compound statements in this context is usually connected with applying definitional expansion [Kor15]. Let us also note that loci declarations and generalizations both use the same **let** keyword and so syntactically they are similar (what makes them different is the context, since loci declarations are only available within definitions). In most contexts the **case** and **suppose** constructs can be used interchangeably and most users tend to prefer the more common **suppose**. The syntax of choice statements (**consider**) and existential assumptions (**given**) requires a **that** phrase, so they never occur without the conditions part, even if they are followed by a single proposition. In all other cases, a user might decide to use only the simple forms of the constructs, e.g. a sequence of single assumptions, and never use compound conditions. And so there are articles in the Mizar library whose authors avoided using compound conditions completely. However, the compound conditions are in general used quite often in the MML. The number of conditions used in an article ranges from 0 to 412 (total 94392). If we assume (very roughly) that each condition is represented by one line of formal text and we compare it to the total number of 2473054 lines of the WSM representation of all the articles, this gives about 4% of all the text content.

It is also worth mentioning that the existential assumption is actually a syntactic sugar construct which replaces an assumption followed by a choice statement - this language feature is rarely used by less experienced Mizar users who are interested primarily in the correctness of their proofs and not in their brevity or “good style”.

Now let us look at the simplest example which demonstrates the underlying linking problem. Here we have a common choice statement with one chosen constant and two labelled compound conditions potentially related to this constant, and we want to immediately refer to this statement in the next proof step:

```
consider x such that A1:  $\alpha$  and A2:  $\beta$  by references1;
then  $\gamma$  by references2;
```

In the original Mizar parser the linking (here using the **then** keyword) is not allowed, because it is ambiguous what it should link to. In consequence, the parser marks the second line with the “164: Nothing to link” error message. In fact, the comment might as well say “Too many options to link”, because in this case there are four possible interpretations of the linked statement, namely:

1. $\text{ex } x \text{ st } \alpha \ \& \ \beta$
2. $\alpha \ \& \ \beta$
3. α
4. β

Although there are cases when option 1. would be useful, usually reasoning in Mizar requires an existentially quantified statements to be eliminated in order to advance the proof. As for option 2., it would be quite useful for a small number of compounds. The average number of compounds in the current MML is 1.92, which makes this option potentially useful for accumulating available facts rather than being very selective in inference statements. However, there are cases of compound statements in the library composed of as many as 33 propositions⁵ and in this case linking to a conjunction of all these propositions would produce too complicated inferences to be practically justified. Option 3. seems least useful and would be most misleading for the users, especially if there were more than two compound propositions involved. This leaves us with option 4., and apparently this

³The parser can be downloaded from a dedicated Git repository: <https://github.com/MizarProject/wsm-tools>.

⁴<http://mizar.org/>

⁵E.g. the GATE_3 article developing the theory of digital circuits.

is the semantics adopted by two Mizar-inspired systems: Mizar Light for HOL Light developed by F. Wiedijk [Wie01] as well as Makarius Wenzel's implementation of the Isabelle/Isar language [Wen02]. Therefore, we have implemented the fourth option in an experimental Mizar version. Mizar verifiers pre-compiled for main supported software platforms are available for download and further experimentation at the author's website⁶. There you can also find an example of a simple set-theoretical Mizar article adapted to this new linking method (file `xbool_0.miz`) to demonstrate its usefulness.

3 Conclusions

First of all, the proposed enhancement of the Mizar parsing module might be beneficial for some Mizar users. But apparently, the inability of linking to compound statements is also one of the main obstacles that complicate automatic improving the legibility of natural deduction proofs in Mizar [Pak14] and make lemma selection based on the close reference principle equivalent to an NP-hard graph problem [Pak15]. So the enhancement might help advance the research on proof legibility. And finally, Mizar has been widely known for its influence on other proof assistant systems and their proof languages - this work is an example that the influence might as well work in the other direction.

3.0.1 Acknowledgement

The processing and analysis of the Mizar library has been performed using the infrastructure of the University of Bialystok High Performance Computing Center.

References

- [Ban15] G. Bancerek, C. Byliński, A. Grabowski, A. Kornilowicz, R. Matuszewski, A. Naumowicz, K. Pāk, J. Urban, Mizar: State-of-the-art and beyond, in: M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, V. Sorge (Eds.), *Intelligent Computer Mathematics – International Conference, CICM 2015, Washington, DC, USA, July 13–17, 2015, Proceedings*, Vol. 9150 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 261–279. doi:10.1007/978-3-319-20615-8_17.
- [Gra15] A. Grabowski, A. Kornilowicz, A. Naumowicz, Four decades of Mizar, *Journal of Automated Reasoning* 55 (3) (2015) 191–198. doi:10.1007/s10817-015-9345-1.
- [Cai04] P. A. Cairns, J. Gow, Using and parsing the Mizar language, *Electr. Notes Theor. Comput. Sci.* 93 (2004) 60–69. doi:10.1016/j.entcs.2003.12.028.
- [Nau16] A. Naumowicz, R. Piliszek, Accessing the Mizar library with a Weakly Strict Mizar parser, in: *Intelligent Computer Mathematics - 9th International Conference, CICM 2016, Bialystok, Poland, July 25-29, 2016, Proceedings*, 2016, pp. 77–82. doi:10.1007/978-3-319-42547-4_6.
- [Byl12] C. Bylinski, J. Alama, New developments in parsing Mizar, in: J. Jeuring, J. A. Campbell, J. Carette, G. D. Reis, P. Sojka, M. Wenzel, V. Sorge (Eds.), *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings*, Vol. 7362 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 427–431. doi:10.1007/978-3-642-31374-5_30.
- [Kor15] A. Kornilowicz, Definitional expansions in Mizar, *Journal of Automated Reasoning* 55 (3) (2015) 257–268. doi:10.1007/s10817-015-9331-7.
- [Wie01] F. Wiedijk, Mizar light for HOL light, in: *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001, Edinburgh, Scotland, UK, September 3-6, 2001, Proceedings*, 2001, pp. 378–394. doi:10.1007/3-540-44755-5_26.
- [Wen02] M. Wenzel, F. Wiedijk, A comparison of Mizar and Isar, *J. Autom. Reasoning* 29 (3-4) (2002) 389–411. doi:10.1023/A:1021935419355.

⁶<http://mizar.uwb.edu.pl/~softadm/linking>

- [Pak14] K. Pał, Improving legibility of natural deduction proofs is not trivial, *Logical Methods in Computer Science* 10 (3) (2014) 1–30. doi:10.2168/LMCS-10(3:23)2014.
- [Pak15] K. Pał, Improving legibility of formal proofs based on the close reference principle is NP-hard, *Journal of Automated Reasoning* 55 (3) (2015) 295–306. doi:10.1007/s10817-015-9337-1.

MathUI 2016: Mathematical User Interfaces

MathUI is an international workshop to discuss how users can be best supported when doing/learning/searching for/interacting with mathematics using a computer.

- Is mathematical user interface design a design for representing mathematics, embedding mathematical context, or a specific design for mathematicians?
- How is mathematics for which purpose best represented?
- What specifically math-oriented support is needed?
- Does learning of math require a platform different than other learning platforms?
- Which mathematical services can be offered?
- Which services can be meaningfully combined?
- What best practices wrt. mathematics can be found and how can they be best communicated?

We invite all questions, that care for the use of mathematics on computers and how the user experience can be improved, to be discussed in the workshop.

July 2016, Andrea Kohlhase & Paul Libbrecht

Towards Visual Type Theory as Mathematical Tool and Mathematical User Interface

Lucius T. Schoenbaum
Louisiana State University
Department of Mathematics
Baton Rouge, Louisiana, USA 70803-4918

Abstract

A *visual type theory* is a cognitive tool that has much in common with language, and may be regarded as an exceptional form of spatial text adjunct. A mathematical visual type theory called *NPM* has been under development that can be viewed as an early-stage project in mathematical knowledge management and mathematical user interface development. We discuss in greater detail the notion of a visual type theory, report on progress towards a usable mathematical visual type theory, and discuss the outlook for future work on this project.

1 Introduction

Suppose that you are a working mathematician with some promising new results that you would like to share with other mathematicians in your field. It is exactly for this purpose that the library of mathematical symbols and the language and style of mathematics were originally developed. It is no surprise, then, that this very style is an excellent vehicle for your purposes.

On the other hand, suppose that you are a worker at a company in a data-intensive field, and you have a problem for which it is necessary to utilize the tools of mathematics. For example, you may wish to automate a finitary procedure, solve a problem in software verification, or apply a physical or economic model. The aforementioned traditional language and style of mathematics was not designed for your purposes. The needs of such users for new styles and new forms of expression are well known and have been felt now for many decades. Many mediums of communication and tools for interacting with other members of such fields, as well as with software and computers, have been developed in recent decades. This effort has been enormously fruitful and transformative, for example, from them an entirely new branch of science (namely, computer science) has arisen.

However, for those in any third walk of life, different from either of these, a communication gap exists whose natural tendency is not to shrink, but to further widen with the growth of scientific knowledge. At the present time there is rising interest in mathematics and other STEM areas [Mau], a growing recognition of the growing importance of STEM knowledge, and a parallel rise in awareness of this communication gap, as well as broader challenges for STEM education. These developments have already attracted the interest of scientists in several fields, and has been (in part) a driver of the development of new areas such as mathematical knowledge management [Far] and mathematical user interface development and design [JP2014], fields that intersect and interact with multiple large knowledge domains, particularly mathematics itself and computer science, but also others such as, e.g., educational and cognitive psychology.

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ20Workshop, Location, Country, DD-MMM-YYYY, published at <http://ceur-ws.org>

In this article, two questions are addressed. First, we address the general question of what theoretical space a tool for mitigating the communication gap named above might occupy. In seeking an answer to this question we introduce the notion of a *visual type theory* in the long section 2, after establishing that such a notion is distinct from existing notions in computer science (in particular programming language theory) and educational psychology and therefore necessitates some kind of new terminology and a somewhat new, developing theoretical framework. Then secondly, in the context of the general body of mathematical knowledge, e.g., the broad fields of analysis, topology, geometry, and algebra, each of which contain numerous specializations, we ask: is it possible to construct a *mathematical* visual type theory? We report that the answer is yes, and that a prototype model is presently in development. In section 3, we describe in outline the prototype system, entitled *NPM*. We conclude our preliminary report with an overview of the outlook of the project in section 4.

2 Visual Type Theory: Form and Function

2.1 Cognitive Model: Comprehension of External Representations, Ordinary Memories

According to the cognitive model of Schnotz and Bannert [SB2, Sch, SB1], the brain’s function of comprehension of external imagery or text involves the creation of a mental model and propositional representation, which are (respectively) depictive and descriptive representations constructed via schema-mediated structure mappings from internal representations that more directly correspond to the given external representations (the imagery, or the text, respectively). The distinction between depictive and descriptive representations is understood in terms of the distinction between *symbols* and *icons* due to Pierce [Pie]. A depictive representation consists of iconic signs associated to the content they represent through common structural features (or “similarity”). A descriptive representation consists instead of symbols having arbitrary structure that are associated to the content they represent by means of convention [Sch]. This model can be viewed as a refinement of the earlier dual coding theory of Paivio [Pai] and the multimedia learning model of Mayer [May]. Such models are used by psychologists to form hypotheses which can subsequently be tested by experiment.

We will adopt this model of visual and text comprehension as our basic framework, and we will refer to the propositional representation and associated mental model generated by an external representation, collectively, as an *ordinary memory* (an idealized but convenient notion). An ordinary memory is distinguished from other memory types, for example spatially organized (as opposed to visual) memories [SB1, p. 616]. In [SB1] the ordinary memory of a graphic (imagery consisting of a graph or chart) is divided into a deep structure and a surface structure. The surface structure is “the outward form of the graphic,” and the deep structure is “a semantic construct which expresses the meaning of the graphic” [SB1]. For our purposes, we seek an analogous definition that would apply to cases other than graphics, in particular to descriptive as well as depictive representations. Therefore we define the *sketch component* of an ordinary memory to be whatever serves (pragmatically) as the individual’s immediate internal depictive representation of the ordinary memory when prompted. The reader may perhaps wish to view this as the associated communication level representation [GMZ, Sch] of the ordinary memory, but we will stick here to the more colloquial terminology of “sketch component”, since our entire argument is quite heuristic and only meant to suggest possible refinements, and since the terminology of [GMZ] does not particularly isolate the depictive portion of the communication level representation as our definition seeks to do. Notice that this definition can indeed be applied equally to external depictive as well as external descriptive representations (as prompts). It is reasonable, and we will assume, that this sketch component or depictive communication level representation is immediately placed in the working memory of the individual upon exposure to an external representation of the associated concept or ordinary memory. Due to the capacity limitations of working memory [SC], this can only be an incomplete part of the total content of the ordinary memory, and we might regard whatever lags in a measure of time-of-recall to belong to the associated *deep component* of the ordinary memory.

We can observe that the sketch component is not well-defined or uniform across space or time (in different individuals, or even in the same individual as time progresses). However, often significant uniformity can be observed in populations. For example, we might expect that a well-known person’s face is the sketch component of the descriptive representation given by the person’s name (for example, an internal visualization of Bill Gates is the sketch component of **Bill Gates**). Likewise, a steaming cup of coffee could be expected to be the sketch component of the ordinary memory of the depictive representation **coffee**. It is also perhaps noteworthy that the sketch component of a memory can be expected to be more well-defined (i.e., uniform/canonical), in general, than the corresponding deep component.

Images presented alongside associated text, or *text adjuncts*, have been shown to improve mnemonic performance, see, for example, [LAC, DSL], and references cited there. In [LAC] a taxonomy of decorative, representational, organizational, interpretational, and transformational text adjuncts is defined, and experimental evidence is presented according to which the effectiveness of text adjuncts for mnemonic and comprehension purposes varies according to the taxonomic type, a point that will be significant for the discussion in section 3.3. In [DSL] evidence is given that other variables can affect the magnitude of such effectiveness, such as setting and period of exposure. In addition they found a significant effect on preservation of learning against decay during the one-week time interval between testing that took place immediately after exposure to a text passage and follow-up testing, due to the presence or absence of a graphic (paired with the text) depicting an illustrative visual metaphor (which they classify as a decorative graphic). Altogether, their study lends support to the view that “extra processing prompted by the presence of the graphics creates a cognitive trade-off for learners” [DSL, p. 202] that can be simplistically described as a short-term cost (the extra energy demanded, e.g., by assimilation of the graphic with the text) that has long-term benefits (e.g., reduced memory decay).

2.2 Mathematical Memories

We next turn specifically to mathematical concepts in the setting of the cognitive model of section 2.1. While many studies have been done on integrating pictures into learning of science and mathematics (see, for example, [BSC], [CL, p.13]), the author is not aware of any that focus on mathematical *structures*. As is well known, during the late nineteenth and early twentieth century, mathematics underwent several new developments due to the contributions of many people including Dedekind, Hilbert, Ore, Noether, and many others, which transformed mathematics, in the minds of many, from a numerical science to an abstract, structural science [Cor]. In this section we approach this latter, structural view with a philosophical discussion based largely on personal experience and introspection, in order to draw several conclusions that are crucial to our argument. At the time of publication it is too late to explore further relationships with psychology and mathematical knowledge management here (but see for example [KK] which suggests an approach via the theory of framings). We will also ignore for simplicity the *numeracy* or *numerosity* function of the brain [Deh], as our focus is particularly on the production of ordinary memories of mathematical structures.

Let us try a simple thought experiment. Suppose that an individual tries to think of a group (that is, tries to think of *groups*, the well-known objects defined in any first course in abstract algebra). He or she would normally recall, after a little thought, particular experiences or encounters with groups. He or she might perhaps recall properties groups have (e.g., “all elements are invertible”), or a theorem about groups (e.g., the Jordan-Hölder Theorem). These memories would, depending on the individual, vary widely. This is particularly true for the *depictive* memory of groups—and it is just this which we would like to draw a focus on. While there are *examples* of groups that evoke depictive representations (symmetric groups, dihedral groups, and orthogonal groups, for example), and there are *heuristic images* that help us to see what groups in general are like, there is no telltale visual cue, no “first-order association” of a group with a depictive representation—something that, when visualized, prompts the entire mental model of a group. Depending on the individual, he or she might have an intuition at hand—for example, spatial intuition for Lie groups, or model-theoretic intuition for polish groups, or set-theoretic intuition for finite groups—but the group concept itself is beyond all such intuitions, or rather, it is apparently a puzzling marriage of all these intuitions, and more besides that is apparently beyond an individual’s power to clarify. In brief, the group concept can be regarded as a “dark” concept.

Indeed, (whether it is fair or unfair to use the evocative term “dark”) many other examples can be given to illustrate that this “dark” quality is in fact commonly encountered in mathematics. While in the example of a group given above it arises due to the formal application of mathematics and logic (the axiomatic method), the same dark quality (the lack of a sketch component in memory) also arises due to the general application of idealized infinite (i.e., nonfinitary or nonterminating) processes in derivations of mathematical formulas and relations, as well as the ubiquitous use of quotient operations in mathematics (see below). Consider, for example, the difference between a standard visual representation of the rational numbers (commonly denoted Q) and the real numbers (denoted R or $(-\infty, +\infty)$) as ordered structures. In both cases, the ordering is linear and *dense*: between any two elements in the ordering, there can be found another distinct element. Only in the latter case, however, does the ordering have the additional property of *completeness*, which is perhaps most easily understood in terms of a topological structure assumed to be present in both cases, or alternatively, by way of a construction of the latter ordered set from the former one. This property, however, defies visualization: density and completeness are indistinguishable in an external or internal depiction, except via a heuristic (i.e., a diagram that

heuristically indicates some phenomenon that cannot be directly visualized). Other examples demonstrating this phenomenon include space-filling curves, real-valued dimension (fractals), and so-called “pathological” functions (for example, the continuous, nowhere-differentiable function discovered by Weierstrass [GO]), many of which arose in the nineteenth century and drove the earliest efforts to place calculus on firm foundations. Or, to reach for examples from other fields, we can think of trying to visualize manifolds (abstract spaces that are curved, like the surface of the earth) that are not embedded in three-dimensional space. The (perhaps) most well-known example of such a manifold, the so-called Klein bottle, has a very famous and lovely depictive representation, but in general, for higher-dimensional manifolds with arbitrarily chosen topological invariants, no such depictive sketch component is available. The same can be said for structures defined only up to some equivalence class (“only up to isomorphism”), i.e., concepts which are obtained via a quotienting operation applied to existing families of concepts. It is not difficult to find examples in which elements of an isomorphism class—for example, the isomorphism class of the plane R^2 up to topological isomorphism (i.e., homeomorphism)—are themselves accessible via a natural depictive representation, and yet there is nevertheless no meaningful or pragmatically useful depictive representation of the entire isomorphism class.

If we set these examples in the context of our cognitive model, we can observe that the ordinary process of creating a mental model and a propositional representation, and an associated sketch component, is confused by the richness and complexity of the concept—in particular the lack of a fixed, canonical, uniform visualization. To couch this point in empirical terms, an experiment could be performed in which mathematicians (or some other chosen population) could be asked to draw pictures of mathematical objects. Our hypothesis, then, would be threefold: first, that there would not be found any uniformity in these sketches, second, in fact in many cases such an experiment might well yield descriptive representations serving as proxies to depictive representations (in other words, there is in that individual’s mind no properly depictive representation at all, and the individual has simply “doubled up” the descriptive representation to serve dual purposes, depictive and descriptive), and third, that there would oftentimes be little or no pragmatic utility (for the purpose of problem solving or mnemonic value) in the sketches produced. Assuming this is accurate, what is therefore observed is a certain inefficient or wasteful use of the brain’s cognitive system: immediate or sketch components of the brain’s recall of mathematical concepts is of little use to the user or learner of mathematics.

2.3 Haptical Memories, Barriers to Entry

It is unclear just how it is, exactly, that mathematicians construct and maintain memories of mathematical concepts. However, if the author ventures to argue on the basis of his own experience, he would cautiously lean towards the view that mathematicians rely often on memory mechanisms of the brain other than the ordinary memory system. In particular, they often rely on what we will refer to here as “muscle memory” or “haptical memory”. Units of such haptical memory are produced in the course of problem solving: passing from a question whose answer is not clear *prima facie* to an answer via logical inference and computation, which may be abridged by prior knowledge or by assumptions and conjectures. (This, we believe, is what mathematicians often refer to simply as “working”.) Let it be granted, then, for the sake of argument, that mathematicians, at least in large part, rely for their memories of mathematical concepts on a posited haptical memory system that is broadly available in the human population. This haptical memory system is assumed to interact, via structure mappings, with the ordinary memory comprehension system, hence it influences the construction of the propositional representation and—perhaps directly, or perhaps indirectly—the mental model of a concept as well.

Haptical memory tactics can be very powerful. However, while an ordinary memory can be formed in an instant, a haptical memory takes several minutes at the very least, and as noted in [SK] places significant cognitive demands on the brain. This matters little in many situations, but with the sheer number of abstractions present in modern mathematics, and the increasing demands for mathematical knowledge in fields outside of mathematics, the acquisition of mathematics through haptical memory tactics is increasingly impractical in many settings. Moreover, mathematical information, once acquired, still must be managed and maintained, activities for which the brain demands energy (due to being so-called biologically secondary knowledge [SK, p.474]). This establishes high barriers to entry at the boundary between those who are willing and able to invest significant resources to acquire, manage, and maintain haptical memories of mathematical concepts, and those who are not.

Cognitive load theory [SC, SK] poses many potential strategies for confronting this conundrum. The findings in [DSL] noted in section 2.1 lend partial support to an approach focusing on the use of visual adjuncts to the normal descriptive representations of mathematical structures (i.e., the rigorous development and study of

mathematical structures, written down in the language of logic). This, by itself, would not seem to be a new idea, indeed modern math and science textbooks are rife with illustrations of mathematical concepts intended to facilitate learning and boost concept retention, and there has been no lack of reflection on and criticism of the famously influential “Bourbaki style”. However, in light of the observation in section 2.2 that mathematical concepts are dark, we can focus attention in particular on the sketch component of memory for mathematical concepts, which is, as observed there, (depending on the individual) missing and/or dysfunctional (serving little or no practical purpose). This begs the question: what if this particular dysfunctional part of the full ordinary memory was replaced by some sort of (in the language of [LAC]) transformational representation? The findings in [DSL] indicate that through such a strategy the problems with the significant time demands and rapid decay of haptical memories might be mitigated.

2.4 Visual Type Theory: Definition

A *visual type theory* is a system of depictive representations created in the course of implementation of the strategy suggested at the end of the previous section. Such a system must have several properties in order to properly be considered a visual type theory:

1. It should depictively reflect logical or natural relationships between the objects being depicted.
2. It should be *almost regular* in the sense that it should be regular in its prescribed use as a general principle, however it should tolerate irregularity when such an irregularity compellingly suggests itself.
3. It should be *generative* in the sense that there is a natural way to generate new representations based on the patterns of existing representations.
4. It should possess the property of *density*: the informational content [Sch, p.103] of each representation of the system should be high in proportion to its visual complexity. For this it might rely on *absence loading* (section 3.3).

A visual type theory can, in theory, function as a communication medium as language does. It differs from language, however, by consisting of depictive, not descriptive representations. Thus we arrive at the “visual” part of the term visual type theory. We use the terminology “type theory” because in the visual type theory of section 3 the representations are very similar to types (for variables) written in a “visual” style of notation, and also because calling it a form of type theory reminds us to distinguish it from language while preserving the idea that such a system, like language, is nevertheless (almost-) regular and generative. Visual type theory provides a system of representations that could be described as “anchors” or “visual anchors” that are particularly useful in situations where concepts are otherwise highly abstract (or, in the language of section 2.2, dark). Through the use of a visual type theory an individual acquires memory tactics in addition to the haptical tactics of problems solving and computation, rote associative memory building, etc.

Now that we have defined visual type theory, a word of caution may be in order. No memory is as simple as a sketch component (section 2.1), taken by itself, would indicate. Beyond the sketch memory there may exist an effectively limitless complex of information. In practice, visual type theory can do no more than provide a user with integrated systems of well-chosen sketch memories of concepts. It leaves to the user the job of filling in deeper memories, and acquiring other abilities a user might need that the visual type theory is possibly unable to provide alone. This is true in particular for a mathematical visual type theory and thus, in this case we caution the reader that visual type theory is not (and is not advertised to be) a “Royal Road to geometry,” and does not disprove the famous ancient dictum of Euclid. Sketch memories are simplistic and can contain inaccuracies that only deep memories can correct. Sketch memories are accompanied by the hazard of false assurances, overhasty opinions, and misguided actions. However, in light of the observation of section 2.2 that abstract mathematical concepts often lack a sketch component (creating a vacuum that, as argued in section 1 and section 2.2, can also impede mathematical cognition and comprehension), a trade-off is expected between which, perhaps, a balance can be struck.

2.5 Visual Type Theory as Language

To conclude this section, we now ask the question: Is a visual type theory a form of language? If so, just what kind of language is it? As a device with a formal definition and prescribed rules of use, a visual type theory (if perfectly regular) may be viewed as being something like a formal language. As a device with a pattern

of growth and an internally consistent set of heuristic (though not unbending) principles, it may be viewed as being something like a natural language. Visual type theories also have many properties in common with both types of language: their purpose, like that of all languages, is to express thoughts, display relationships quickly and memorably, to facilitate the possession, understanding, and management of complex and abstract ideas, to port well through different environments, and, potentially, to serve as a method of communication between users who share the necessary prior knowledge. But, as we have argued in section 2.4, visual type theory technically cannot be considered a language at all: it is not a mapping between syntax and semantics, or in other words, a cognitive tool whose elements travel via the descriptive channels of the ordinary memory system—instead, unlike in a language, they travel via the depictive channels. However, with so many commonalities with existing language notions (and because the use of generally familiar vocabulary to discuss visual type theory is, by all indication, inevitable), it might be wise to bend the notion of language slightly by referring to visual type theory as a *language paradigm*. As such a paradigm, it is distinct from any of the major paradigms that currently exist in computer science: it is not a formal language, nor a compilable programming language, nor a specification language, nor a natural language. From this observation, we can draw several closely related and important corollaries. First, we have seen that visual type theories are not developed as a *language* would be, in order to replace existing mathematical language—neither in whole nor in part. It is not the case that the use of visual type theory would impinge in any way upon mathematical practice if it is used as prescribed here. Second, a visual type theory is not a universal formal system for expressing mathematics, for it operates within an entirely different language paradigm. Rather, it is a supplementary system that runs in parallel with formal languages, whether locally or globally defined. In other words, the depictive character of visual type theory provides it with the property of *independence*: the mathematical formalization a user chooses can be switched out easily while the visual type theory, say, T , stays in place, just as T can also itself be removed or added, turned off or turned on at any moment, without affecting the rigor expressed by formal syntax. These formal languages can be of virtually any kind—past, present, or future. This suggests particular uses for visual type theory in mathematical knowledge management, in its role as custodian of past and present mathematical documents and for confronting the problems of informality and logical heterogeneity [RK].

It may be noteworthy that an attempt to change this prescribed role—to use a visual type theory T as a basis for an imagined “visual” formalization of mathematics, for example—regardless of success or failure, would take away this property of independence, and force T to cease being a visual type theory, and to become instead a “visual” formal language.¹ The independence property removes the most stringent pressures that a language can be subject to: the demands of rigor, compilability, security, searchability, etc. In contrast, these pressures fall directly on formal languages and programming languages. Such languages must cater to the needs of machines first, and human beings, second. In contrast, visual type theories, like natural languages, occupy a space in which this priority is exactly reversed.

3 Progress Report: A Mathematical Visual Type Theory

3.1 Overview, Principles of Design

Now that we have explained at length the simple idea of setting up a visual type theory, we can discuss such a visual type theory for mathematics that is currently under development, called NPM. (The abbreviation NPM originally stood for “not *Principia Mathematica*,” but we will not dwell on this.) A system at minimum requires a supply of depictive glyphs (which are intended to serve as canonical sketch components for ordinary memories of mathematical abstractions, cf. section 2.2), rules that govern their use, and tools that make their use in the real world possible. More concretely, we have:

1. a set of glyphs called the *NPM glyphs*. (more concretely, a scalable font, the *NPM font*).
2. the grammatical rules governing the use of glyphs (Briefly, the *NPM grammar*),
3. the “Principles of Design” of the NPM system. (Guidelines for the ongoing development of NPM.)
4. the correspondence between glyphs and mathematical concepts (the *NPM meaning map* or *interpretation*), and
5. supporting software/technology (see section 4).

¹Many such languages (and similar tools) exist, cf. [Mye].

By the Principles of Design we mean to include both the informal or heuristic relationships between mathematical concepts and visual elements of NPM glyphs, and the underlying motives and principles of visual type theory. We also include the following basic principles (hereafter: Principles of Design #'s 1, 2, 3, and 4), which lay at the heart of NPM. They have no relationship with the general notion of visual type theory explained above.

- #1. NPM should aspire to be as beautiful as the mathematics it expresses. It should “fit in” with the notation and methods of ordinary mathematical practice.
- #2. NPM should be both printable and hand-writable (with a pen or similar stylus).
- #3. NPM should have a many-one meaning map between the logical space and conceptual space of mathematics, and be as close to a one-to-one map as possible.
- #4. NPM should aspire to be *universal*: to provide one internally consistent set of glyphs for all of mathematics (or more precisely, all reasonably well-established knowledge in all mathematical areas).

Principle of Design #1 and #2 impact various design decisions in the development of NPM glyphs. Principle of Design #2 primarily aims, without specifying a reason, to preserve the traditional bond between haptical experience and mathematical memory. Principle of Design #1, together with the density requirement of visual type theory (section 2.4), suggests that the design be simple, with a minimum of flourishes. This conclusion is supported by studies on text adjuncts which find that learning and memory outcomes may decrease due to interference effects [SB2] and in accordance with a so-called coherence effect or coherence principle [MM, p. 95]. The glyphs should also strive to conform to typographical design standards and principles. For example, it should aim to “induce a state of energetic repose, which is the ideal condition for reading” [Bri, p. 24].

Principle of Design #3 has two parts. First, it says that there should be no overloading of NPM glyphs. Just like the syntax of a strictly typed formal computing language, every NPM glyph should have a definite, fixed meaning, with nothing left to context to interpret. Second, it says that as much as possible, the sketch memory of a mathematical concept provided by NPM should be like a fingerprint: it should be a unique identifier. There are many cases in which this specification cannot be fulfilled. Consider notions that exhibit so-called “cryptomorphism,” or nontrivial equivalences between several different definitions. The concept of a matroid, the concept of a lattice, or the concept of a topos are just a few examples of cryptomorphic notions. It is usually sensible to express such cryptomorphism in NPM, rather than choosing one notion and expressing it visually, ignoring the others. Principle of Design #3 tells us, however, to select one of these glyphs and give it precedence over the others. Principle of Design #3 provides a counterweight to the common experience of encountering overloaded notation in mathematics, and disagreement between the notations and terminology used by different authors. It is notable that without changing a single letter or a single word in the canon of mathematical terminology and notation, we obtain in this way a tool for disambiguation that is always near at hand, yet unobtrusive because it is depictive and thus does not share the same cognitive channel as the descriptive symbolism. We note, echoing section 2.5, that this benefit to the use of NPM can be enjoyed without explicitly writing or reading NPM glyphs. We also note that, although NPM commits itself to disambiguation just as formal languages and programming languages do, it does so in response to the needs of a reasoning human being maintaining a large, abstract, technical language, not those of a mechanical parser.

Principle of Design #4 states that NPM should be developed to serve as a tool on *the broadest possible base*, where “base” refers to the potential users of the glyphs and grammar as a cognitive tool—even though, as stated in the introduction, a major goal and motivation of NPM is to increase understanding and lower barriers to entry to mathematics, particularly for those in career paths or life situations which do not permit them to pursue focused, rigorous mathematical studies. Although this may seem like a point of ambition for the NPM project, in the author’s view a mathematician would in fact consider it to be the only path forward, given the deep, inescapable connections between mathematical knowledge in different fields. Any attempt to draw a boundary around the parts of mathematics NPM does and does not attempt to assimilate into its linguistic pattern would inevitably be disrupted, due to the natural course that mathematics itself tends to run.

Before taking steps to finalize a mathematical visual type theory, it must be checked whether such a construction could really exist. In particular, it must be determined if it can in fact be universally applicable across all (or at least, some significant majority) of mathematical specializations. Due to the minute, technical details involved, this task cannot be considered completed until a real mathematical visual type theory has been found meeting the specifications to an extent that convincingly indicates that the project can be continued all the way to completion. Until then, it is still not determined whether a visual type theory could exist which holds,

in one grammar and one glyph set, all the concepts of mathematics at once, or whether any attempt to create such a system would inevitably burst at the seams, overwhelmed by unexpected, accidental conflicts between patterns inspired by different background knowledge areas. An extensive effort to find an answer to this question has recently been concluded, and it has been confirmed that a visual type theory, conforming to all the above-named specifications, is indeed theoretically possible. (However, difficult challenges of a mathematical nature still remain, as will be discussed further in section 4.) We now describe this system in outline.

3.2 Grammar

NPM (viewed as a language) has a large vocabulary of glyphs, but a simple grammar. NPM glyphs are primarily used as identifiers and as modifiers on function and relation symbols. They may appear under arrows, under relations, or standing alone. These are the only prescribed uses of NPM glyphs, though they may also theoretically appear anywhere else they are determined to be useful.

The decoration of mathematical calculations with NPM glyphs performs several functions. It distinguishes calculations in one setting from calculations in a different setting, when the two are formally similar. It also allows users to assemble objects and notions into coherent conceptual wholes, or in other words, to build new abstractions out of preexisting abstractions in a convenient and regular way. Often, these new wholes take the form of categories, or sometimes (depending on the setting) 2-categories (more rarely, even higher categories can appear). It is convenient to express such categories using NPM for several reasons. Such categories are abundant throughout mathematics and can be difficult to manage due to their number. In a given mathematical specialization it is not uncommon for many dozens of categories to arise, even considering only the basic prerequisite knowledge. Categories are also particularly susceptible to the property of having a complicated definition in spite of playing a very basic role in some derived setting or field of investigation. NPM assists in managing such levels of suppressed complexity, creating room to focus on the most relevant details in a given setting. NPM is good at encapsulating complexities, allowing them to be stored neatly, extracted when needed, and stored again when the need is met. Glyphs may be combined in natural ways, providing “ramps” leading from basic ideas to more complex concepts. An NPM expression of a subtle and complex mathematical notion can be a helpful guidepost, and a landmark in the memory of the user that can easily be recalled later, even sometimes used (to one’s delight) to reconstruct a forgotten definition. It can also be helpful in drawing attention to relationships between concepts. This is particularly helpful when (for whatever reason) terminology and notation cannot perform the same role, as is sometimes the case. Nonetheless, as the reader can imagine, NPM does not do everything well. Statements which are “decorated” with NPM glyphs can become heavy, and this can become counterproductive and wasteful. This phenomenon reinforces the principle (section 2.5) that NPM might remain in its role as a visual type theory as opposed to being viewed as an extension of mathematical formalism. In this role its explicit appearance in mathematical calculations and statements would, a priori, be optional and left to the user’s discretion.

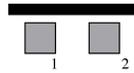
3.3 Glyphs

Finally, we address the question of the appearance and design of the glyphs themselves. Although glyphs of NPM bears some resemblance to other sign systems—most notably perhaps, Chinese Han characters and chemical nomenclature—they rely for much of their power on a tool called *absence loading*, which we now define, and which is not a part of the prior knowledge of either of these sign systems.

The space in which a glyph is written or imprinted is called the *body*, or *bounding box*. Absence loading is the association of a spatial region in the bounding box (a region anywhere in the bounding box that may have any size or shape) to a logical constraint Φ . Once this region is assigned, information concerning Φ may then be associated, via convention, to hand-drawn or printed lines which are placed in that region. In particular, an empty or vacant region (i.e., the absence of any line or mark of any kind) indicates that either nothing is assumed about Φ , nothing is known about Φ , or that information about the status of Φ is not relevant to the context where the glyph is drawn or printed. (It is left to context to determine which of these cases applies.)

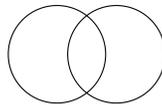
Here is an example. Consider two sets A and B (in some fixed universe U). We can construct a simple glyph using absence loading that would tell us whether something, say x , is in the set A , and whether it is in the set B . Suppose that we take the symbol  as a guide for the eye, and “load” or “charge” the two regions below

the horizontal line $\overline{\quad}$ with information about membership in A and B , respectively:



region 1: $x \in A$
 region 2: $x \in B$

Now let us assume that a dot in the loaded regions shall denote: positive, certain membership in A , and positive, certain membership in B , respectively. Then we can see that our entire language consists of the symbols $\overline{\quad}$, denoting the general universe of sets and elements U , $\overline{\bullet}$, denoting the set of elements of A , $\overline{\circ}$, denoting the set of elements of B , and $\overline{\bullet\bullet}$, denoting the set of elements in both A and B . If we close off the grammar and glyph system here, we can see that we have the power to denote only the sets A , B , and $A \cap B$, along with the universe U . However, we can fill the loaded regions with any symbol we wish. Next, suppose we place, say, an open circle \circ in the regions to denote positive, certain *non*-membership in A and B , respectively. Then we generate the usual four disjoint classes: $B - A$, $A - B$, $A \cap B$ and $U - A - B$ via the symbols $\overline{\circ\bullet}$, $\overline{\bullet\circ}$, $\overline{\bullet\bullet}$, and $\overline{\circ\circ}$. Notice the additional flexibility: we can, using the same symbol system, leave the region assigned to A or B absent, leaving us with a smaller set of symbols germane to that setting, for example, $\overline{\quad}$, $\overline{\bullet}$, $\overline{\circ}$ can serve in any setting in which we are not interested in B , only in the set A . Hence if we were in a setting in which we wish to ignore some information, we could continue use the glyphs we have defined rather than switching to another set of glyphs. This is achieved by making use of natural visual processing mechanisms: upon exposure to visual information, relative locations in memory are maintained, but visual information considered irrelevant is filtered out in favor of visual information that is the focus of attention. For this reason, the notation is interpreted as a depictive representation, not descriptive one. Certainly, in this notation the amount of depictive power obtained is rather slight, but it's not so bad for a line, two dots, and two circles. What we really mean by this, of course, is that the notation has the property of density that was defined in section 2.4. This density is more noticeable if one compares an absence loading notation to other depictive representations of the situation. Consider the most common depictive representation, a Venn diagram:



While we will not draw them, the Venn diagram above suggests other possible “visual” notations for the two sets and their subsets. Note that while it is also possible to *extend* such a Venn-diagram-inspired notation to denote three sets, already in the case of four sets, Venn notation is stretched to its breaking point and has practical value only in exceptional cases where some of the intersections are empty. This is not so for the notation using absence loading: given three sets A, B, C , we extend the line to produce an immediately evident *extension* of the sign system:



Given more sets A_1, \dots, A_7 , we can continue further: $\overline{\circ\bullet\bullet\circ\circ\bullet}$, etc. This, for example, would generate a language of 3^7 or 2187 unique² glyphs, or something close to the number of Chinese Han characters in frequent use [L]. This extension of the notation naturally suggests itself because the use of spatial regions of the bounding box naturally induces visual patterns. The corresponding descriptive representation (using formal logical symbols) can easily be derived from the depictive representation, and vice versa. Although this set of examples has little practical value, absence loading can provide a potentially useful notation alternative to descriptive notation in any setting in which frequently recurring conditions Φ, Ψ , etc. arise in highly variable combinations. Such settings are found in many technical and scientific knowledge areas.

Now we come to apply absence loading to the generation of mathematical glyphs, and the design choices specifically made in the development of NPM. For the typeset glyphs we use in this article we employ a design based on scripts of several natural languages.³ We begin with the glyph for a set. If pressed, we will say this is

²Unique up to equivalence defined by the Gestalt laws.

³Elements of the design were influenced by, e.g., the Hiragana syllabary and Sanscrit alphabet, Arabic and Khmer scripts, and Chinese Han characters.

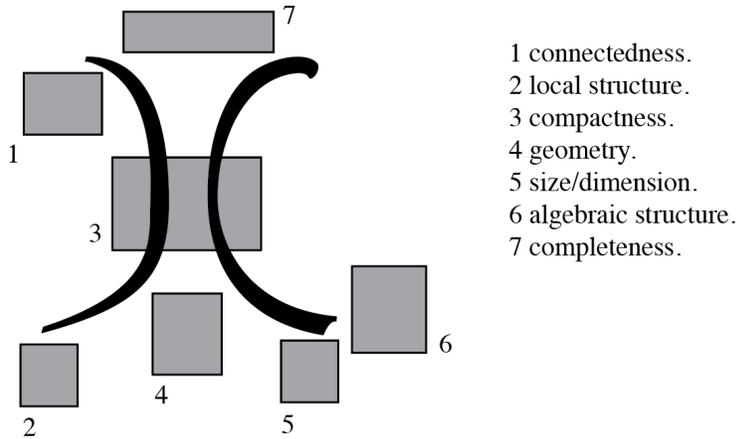
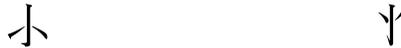


Figure 1: Absence-loaded regions of the Hausdorff space radical.

the set concept determined formally by the Zermelo-Frankel axioms, including the axiom of choice:



We observe a vertical center line, to serve as a visual anchor, partially enclosed by a pair of dots (represented as they might be drawn using a brush). We can now begin developing the glyph as structural features are added to a set. This will give rise to the first of the two major families of glyphs, the *structure* glyphs. Following terminology for Chinese and Japanese, we call the most basic glyphs *radicals*, and thus we say that the set glyph is one of the basic *radicals* of the NPM system. From the set glyph immediately follow two of the most important radicals, that of groups and semigroups (left), and that of categories (right).



On the left is the NPM glyph of the category concept, and on the right is the glyph of the mathematical group concept. The depictive elements in these glyphs are briefly the following: a group is a set with a little bit of extra structure added (a binary operation satisfying various properties). This additional structure is represented by extending the dots of the set glyph to the ascending diagonal lines of the group glyph. Thus, via absence loading, when we write the glyph of a *set*, or *invoke* it, we simultaneously invoke all *groups*. (In other words, it is unspoken whether or not the sets we invoke are in fact groups.) On the other hand, the relationship between the category radical and the set radical is slightly irregular: a set is what is known as a *zero category*: in the language of category theory, this is a set of objects with no morphisms between them. There is an ascending sequence of 0-categories, 1-categories (which are the usual categories), 2-categories, 3-categories, etc. Since this sequence and the “higher category theory” it leads to is foundational to many areas of mathematics, we build into NPM the relationship between sets and categories. In other words, NPM views a set as a zero category—as well as a structure with a coherent definition and theoretical area of study of its own.

These characters can now be recombined to give the character for a *groupoid*, a category in which all morphisms are invertible:



Coincidentally, this character is reminiscent of the Han character for water, 水. Guided by the Han character, we make the strokes slightly asymmetric, for this increases its aesthetic appeal and evokes the experience of writing the glyph by hand. There are numerous variants of these basic glyphs that will not be discussed in this overview.

Now we come to the second major family: the topological glyphs, which differ from the structure glyphs semantically, grammatically, as well as typographically. The topological family has two radicals instead of one,

the *Kolmogorov space* radical (left) and the *Hausdorff space* radical (right):



The treatment of the Hausdorff glyph as a radical reflects the importance of Hausdorff spaces, particularly in analytic areas of study (for example, harmonic analysis) where a topological space is needed that can support a notion of measure (in the sense of Lebesgue theory). Kolmogorov spaces, on the other hand, are often studied very differently. For example, they frequently arise in situations in which the so-called specialization ordering is an important part of the structure (the specialization ordering is trivial in the case of a Hausdorff space). These differences are reflected directly (depictively) in the associated NPM glyphs, in the sense that the Hausdorff property is not added to the Kolmogorov space radical, even though a priori this is how one might have expected to proceed.

For the sake of brevity, we will not develop the topological radical's many variants here. Figure 1 indicates absence-loaded regions of the radical, along with the class of properties reflected (or “loaded” or “held”) in each region. Note that positions are disjoint and reflect design choices regarding proximity to the center of the glyph. In particular, compactness is placed directly in the center of the glyph: this reflects the general fact that compactness across virtually all of mathematics is among the most important and most impactful properties of a topological space. We note, in passing, that the glyph of a compact Hausdorff space is denoted in NPM by



Here the intuition of compactness is represented depictively by a closed circle. It may also be noteworthy that the algebraic structure region's location is deliberately chosen to allow maximum flexibility concerning size adjustment (or “expansion” of a region), as a great deal of the variation in topological spaces arising in practice is due to variations in algebraic structure.

We will now give a demonstration of how NPM grammar patterns can be combined to create complex glyphs out of simple ones. For this we will study the development of structures arising in analysis. Let us go back to the group glyph



We can add the assumption that it is abelian by transforming the enclosing strokes into a cross:



This depictively evokes the symbol of addition, the Greek cross used by Widman (in 1489) and Descartes [Caj]. Next we can transform to a *vector space* by assuming the action of a field:



Now we begin to see that, in but a single glance, we can perceive via NPM the close relationship between many diverse structures, each with a distinct theory and semantic space of their own: vectors spaces, groups, sets, and categories.⁴ Now we combine the topological and structure glyphs to generate the notion of a *topological vector space*:



⁴The concept of a *module* (in the mathematical sense) is also very closely related to the rest of these concepts, and is therefore represented in NPM by an evocatively similar glyph:



The line depicting the scalar action of a ring or algebra is not so far extended (ascends not as far as the cap height).

When such structures are studied, several properties come to the fore, generating a family of glyphs that in NPM take the form:



The additional strokes in the glyphs reflect structural and topological properties: *completeness*, the presence of an inner product structure, the presence of a norm structure, and several properties related to the presence of an involution [Dix]. These glyphs (at left, at center, at right) depict structures that are known in analysis as *Banach spaces*, *Hilbert spaces*, and *C*-algebras*, respectively, each of which is the basis of a substantial mathematical theory.

Next, we illustrate how NPM provides representations that can concisely express otherwise obscure relationships between mathematical concepts. For this example, we return to the category radical,



and we develop it to the notion of a topos, whose glyph is



which is abbreviated,⁵ for convenience, to



There is more to tell, however, in unpacking this glyph. The terminology “topos” is in fact used to denote two mathematical concepts, the first known as *elementary topos* and the chronologically earlier concept of *Grothendieck topos*. The Grothendieck topos can be defined as an elementary topos that is cocomplete and has a small generating set. However, this definition does not reflect the (so to speak) “correct” intuition in all situations where the notion of a Grothendieck topos is used. Therefore the position of NPM is that a Grothendieck topos is denoted using a glyph based on a different radical than that of the elementary topos:



This glyph, which irregularly displays the category glyph in the geometric region (see Figure 1), evokes the Kolmogorov space radical above—in other words, it evokes a spatial object, not a structural one. This reflects the language of generalized spaces developed in the original work by Grothendieck and his school [SGA4], based on geometric morphisms. The fact that the two approaches to topos theory in fact converge on the same notion and a common underlying theory was the product of years of development by mathematicians; for better or for worse, NPM can capture the entire thrilling story, more or less, simply by invoking their names.

For the last demonstration, we show how NPM may be used to express logical relations between structures, and in particular its grammatical role with respect to morphisms. We choose a relation in order theory, known

⁵This abbreviation is a regular, recurring NPM pattern. The vertical line which is suppressed is called the *limit file*, and is decorated with absence-loaded regions for expressing the limit properties of a category. Thus a category glyph with limit file

extended is written , and extra strokes are added which, in the case above, for example, reflect the fact that the structure is assumed to be cartesian closed and to have all finite limits. This creates a depictive relationship between this and the order glyph

 (see Table 1) that once again reflects underlying mathematical structure. Cf., for example,



which is the Heyting algebra glyph.

as *Priestley duality*:

$$\begin{array}{c} \text{⌈} \\ \hline \text{⌋} \end{array} \approx \begin{array}{c} \text{⌈} \\ \hline \text{⌋} \end{array}$$

Here, a glyph set above an arrow denotes the objects of a category whose morphisms are described using the glyph centered below the arrow, and the symbol \approx denotes a duality (i.e., an equivalence between one and the opposite of the other category). Within order theory there are at least two important cases of this duality: *Birkhoff duality*

$$\begin{array}{c} \text{⌈} \\ \hline \text{⌋} \end{array} \approx \begin{array}{c} \text{⌈} \\ \hline \text{⌋} \end{array}$$

and *Stone duality*

$$\begin{array}{c} \text{⌈} \\ \hline \text{⌋} \end{array} \approx \begin{array}{c} \text{⌈} \\ \hline \text{⌋} \end{array}$$

Notice that the NPM representations suggest, via highly automated cognitive processing mechanisms, that there are many relationships and patterns to be recognized between the three formulas. The task of bringing to the fore the same parallels in the structure of these statements cannot be as easily achieved via descriptive methods.⁶ Variations of this broad duality relationship, which is a form of spectral duality, can be found not only in order theory, but in many other mathematical disciplines as well. NPM captures each one of these and often succeeds in capturing something of the metaphorical relationship between these different dualities in an evocative and information dense, calligraphic, yet useful form.

Finally, the entire collection of NPM radicals is presented in Table 1.

4 Conclusion and Outlook

The problem of developing an implementation of a visual type theory for mathematics such as NPM demands ventures in both the sciences and the arts, across the boundaries between mathematics, computer science, and graphic design, and poses many challenges. For its grounding as a tool for cognition, it appeals to fields such as educational and cognitive psychology for a model on which to base its mechanical function, and for empirical results that may clarify its role. The empirical results reviewed in section 2.1 are encouraging in this respect, but as these studies arise from experiments involving text adjuncts like (among others) graphs, charts, and representational images, we could regard this evidence as not directly hitting the bull’s eye. For instance, the evidence on transformational and decorative text adjuncts in [LAC, DSL] suggests that an information dense, grammatically regular, depictive device like a visual type theory may in some settings boost learning and memory performance. However, absence loading (section 3.3), construed as a mnemonic device and tool for schema-construction (cf. sections 2.5, 3.1, where it was noted that a visual type theory could be used as a public or as a private language for this purpose), still has not itself been scientifically tested or evaluated. It is

⁶We cannot fully explain the glyphs appearing in the dualities here; however, ⌈ is the glyph of finiteness (the three strokes of the glyph correspond to ascending and descending chain condition, ⌋ cochain-finiteness, respectively), and ⌋ denotes distributivity. The rest can be inferred from formal statements.

\updownarrow	set	\mathcal{X}	sheaf (geometric view)*
\mathcal{K}	Kolmogorov space	\mathcal{I}	(dynamical) system, triple
\mathcal{H}	Hausdorff space	\mathcal{P}	process
\mathcal{R}	ring, field, algebra	\mathcal{V}	topological vector space
\mathcal{M}	module, vector space	\mathcal{C}	CW complex
\mathcal{E}	pairs, extensions	\mathcal{S}	simplicial set, Kan complex*
\mathcal{G}	group	\mathcal{C}	category
\mathcal{G}_{\downarrow}	group (topological)	\mathcal{G}	globular set, generalized category**
\mathcal{L}	Lie algebra	\mathcal{E}	enriched category
\mathcal{M}_{∞}	manifold, bundle	\mathcal{O}	order, lattice
\mathcal{V}	classical variety	\mathcal{D}	deduction system, graph*
		\mathcal{L}	lambda calculus

* subject to revision

** recent/experimental

Table 1: NPM: The Basic Radicals, July 2016.

also unclear how the prior knowledge demanded for the use of a visual type theory would affect its performance as a mnemonic and organizing tool in practice. However, the author would regard it as too much to argue that the existing evidence falls off the target altogether, and would suggest that results so far are promising from the point of view of visual type theory, and that further research is warranted.

As regards potential NPM enthusiasts who are mathematically trained, there appears to be, fortunately, only two major axes along which NPM must be brought to smoother interoperability if NPM is to become usable: TeX platforms, since TeX is the near-universal language of mathematical and technical writing, and the space within and around the large family of proof assistants currently available (Coq, Agda, Isabelle, CompCert, Lean, and others), perhaps via adaptation of existing middleware such as the Proof General [Asp]. A first-generation NPM system could provide such inter-operability, however minimal. The creation and maintenance of a full library of NPM glyphs even on this minimal basis would require not only facility in typographical design, but also broad knowledge of mathematics, in order to maximize the quality and realism of the language. The author's hope is that the project is continued without losing the sense of purpose it has carried since its earliest beginnings: for the NPM developer, the task faced is not only to be of service to its users, but also to be a participant in the relationship, as old as civilization itself, between people and mathematics.

A working NPM platform would involve a usable glyph set easily numbering in the tens of millions. If such a platform is one day available, the problem of how to improve user interface can then be approached in

conformity with a plan for growth. This suggests connections to the fields of mathematical user interface design and human-computer interaction, as well as programming language theory, which supplies an understanding of the anatomy and life cycles of formal languages existing in diverse and rapidly changing environments. Further study in these areas can perhaps provide models and guidelines which can be adapted to the visual type theory paradigm. All research related to NPM's development would also have to follow developments in a number of areas including proof assistants and proof engineering, mathematical typesetting, mathematical software, and mathematical knowledge management.

5 Acknowledgements

It is a privilege to do interdisciplinary research, and I deeply thank everyone who helped me put this still-incomplete puzzle together. I would especially like to thank the CICM 2016 and MathUI referees for their criticism and suggestions, which led to many significant improvements of this work.

References

- [SGA4] M. Artin, A. Grothendieck, and J. L. Verdier. *Theorie des topos et cohomologie étale des schémas*. Séminaire de Géométrie Algébrique du Bois-Marie, année 1963-64. second edition published as Lecture Notes in Mathematics, Vols. 269, 270, and 305, Springer-Verlag, 1972.
- [Asp] D. Aspinall. "Proof General: A generic tool for proof development." *Tools and Algorithms for the Construction and Analysis of Systems*. LNCS Vol 1785, pp. 38-43, 2000.
- [BSC] J. Bobis, J. Sweller, and M. Cooper. "Cognitive Load Effects in a Primary-School Geometry Task." *Learning and Instruction*, Vol. 3 pp. 1-21, 1993.
- [Bri] R. Bringhurst. *The Elements of Typographic Style*, 4th ed. Hartley and Marks Publishers, 2013.
- [CL] R.N. Carney and J.R. Levin. "Pictorial Illustrations Still Improve Students' Learning From Text." *Educational Psychology Review*, Vol. 14, No. 1, 2002.
- [Caj] F. Cajoli. *A History of Mathematical Notations*. Open Court, Chicago, 1932 (Dover reprint, 1993).
- [Cor] L. Corry. *Modern Algebra and the Rise of Mathematical Structures*, 2nd ed. Birkhauser Verlag, 2004.
- [DSL] R.W. Danielson, N.H. Schwartz, and M. Lippmann. "Metaphorical graphics aid learning and memory." *Learning and Instruction*, Vol. 39 pp. 194-205, 2015.
- [Deh] S. Dehaene. *The Number Sense*. Oxford University Press, 1997.
- [Dix] J. Dixmier. *Les C^* -algèbres et leurs représentations*. Gauthiers-Villars, 1969.
- [JP2014] M. England, J. H. Davenport, A. Kohlhase, Michael Kohlhase, P. Libbrecht, W. Neuper, P. Quaresma, A.P. Sexton, P. Sojka, J. Urban, and S.M. Watt. *Joint Proceedings of the MathUI, OpenMath and ThEdu Workshops and Work in Progress track at CICM, 7-11 June 2014, Coimbra, Portugal*. url: <http://ceur-ws.org/Vol-1186> CEUR Workshop Proceedings Vol. 1186, 2014.
- [Far] W.M. Farmer. "MKM: A New Interdisciplinary Field of Research." *ACM SIGSAM Bulletin*, Vol. 38, No. 2, pp. 47-52, 2004.
- [GO] B.R. Gelbaum and J.M.H. Olmsted. *Counterexamples in Analysis*. Holden-Day, San Francisco, 1964 (Dover reprint, 2003).
- [GMZ] A.C. Graesser, K.K. Millis, and R.A. Zwaan. "Discourse comprehension." *Annual Review of Psychology*, Vol. 48 pp. 163-189, 1997.
- [LAC] J.R. Levin, G.J. Anglin, and R.N. Carney. "On empirically validating functions of pictures in prose." In Willows, D.M., and Houghton, H.A. (eds.), *The Psychology of Illustration: I. Basic Research*. Springer, New York, pp. 51-85, 1987.

- [L] *List of Frequently Used Characters in Modern Chinese*. Ministry of Education of the People's Republic of China, 1988.
- [KK] A. Kohlhase and M. Kohlhase. "Compensating the Computational Bias of Spreadsheets with MKM Techniques." J. Carette et al. (eds.) *Calculemus/MKM 2009, Lecture Notes in Artificial Intelligence* Vol. 5625, pp. 357–372, 2009.
- [Mau] W.C.J. Mau. "Characteristics of US Students that Pursued a STEM Major and Factors that Predicted Their Persistence in Degree Completion." *Universal Journal of Educational Research*, Vol. 4 No. 6, pp. 1495–1500, 2016.
- [MM] R.E. Mayer and R. Moreno. "Animation as an Aid to Multimedia Learning." *Educational Psychology Review* Vol. 14 No. 1, pp. 87–99, 2002.
- [May] R.E. Mayer. "Multimedia learning: Are we asking the right questions?" *Educational Psychology Review* Vol. 32, 1–19, 1997.
- [Mye] B.A. Myers. "Taxonomies of Visual Programming and Program Visualization." *Journal of Visual Languages and Computing* Vol. 1 pp. 97–123, 1990.
- [Pai] A. Paivio. *Mental Representations: A Dual Coding Approach*. Oxford University Press, Oxford, 1986.
- [Pie] C.S. Pierce. "Prolegomena to an apology for pragmatism." *Monist* pp. 492–546, 1906.
- [RK] F. Rabe and M. Kohlhase. "A scalable module system." *Information and Computation*, Vol. 230 pp. 1–54, 2013.
- [SB1] W. Schnotz and C. Baadte. "Surface and deep structures in graphics comprehension." *Memory and Cognition*, Vol. 43 pp. 605–618, 2015.
- [SB2] W. Schnotz and M. Bannert. "Construction and interference in learning from multiple representation." *Learning and Instruction*, Vol. 13 pp. 141–156, 2003.
- [SK] W. Schnotz and C. Kürschner. "A Reconsideration of Cognitive Load Theory." *Educational Psychology Review*, Vol. 19 pp. 469–508, 2007.
- [Sch] W. Schnotz. "Towards an Integrated View of Learning From Text and Visual Displays." *Educational Psychology Review*, Vol. 14, No. 1 pp. 101–119, 2002.
- [SC] J. Sweller and P. Chandler. "Why some material is difficult to learn." *Cognition and Instruction*, Vol. 12, No. 3 pp. 185–233, 1994.

Understanding Mathematical Expressions: An Eye-Tracking Study

Andrea Kohlhase

Michael Fürsich

Neu-Ulm University of Applied Sciences

Abstract

Intuitive user interfaces need a good understanding of the respective users and practices. One important mathematical practice consists in using mathematical expressions, which evolved as concise and precise representations of mathematical knowledge and as such guide and inform mathematical thinking. We present an exploratory eye-tracking study in which we investigate how humans perceive and understand mathematical expressions. The study confirms that math-oriented and not math-oriented users approach them differently and reveals implicit mathematical practices in the decoding and understanding processes. Further experiments are needed to confirm and study them. We discuss how these practices could be used to improve mathematical user interfaces.

1 Introduction

The art of expressing mathematical knowledge in mathematical expressions evolved over the last three centuries. From the standpoint of Human-Computer Interaction they have a perfect usability score (at least for mathematicians) as they are rated highly for the three components of usability: effectiveness, efficiency and satisfaction.

With the rise of the World Wide Web mathematical knowledge (in a very broad sense) can be shared easily. The field of Mathematical Knowledge Management (MKM) has achieved crucial breakthroughs to digitalize mathematics, for instance the development of MathML or OpenMath with their respective content and presentation parts as Web formats for math. In a nutshell, the technology for the traditional tools-of-the-mathematical-trade on the Web exist, but are they still as usable on the new medium?

Let us have a closer look at the usability of mathematical expressions in digital documents. The effectiveness and efficiency of formalizing mathematical information via mathematical expressions stays the same, but the satisfaction component is influenced by traditional intuitions how they are to be used and modern intuitions about Web services for non-technical documents. In our research, we are interested in the integration of traditional mathematical intuitions into modern user interfaces for math (“**mathUIs**”) on the Web. With this study we aim at a deeper understanding of mathematical expressions as a tool so that we ultimately can provide new best-practice guidelines for the design of mathUIs.

For this we investigate implicit mathematical practices by conducting an eye-tracking experiment on the reading of formulae. In several studies (see for example [11] for an overview) it was shown that there is a correlation between what a participant attends to and where she is looking at. The “eye-mind hypothesis” [7] even claims a correlation between the cognitive processing of information and the person’s gaze at the specific location of the information. In a former study [10] it was shown that professional mathematicians have different requirements for search user interfaces not only compared to non-mathematicians, but even to educated mathematicians who don’t do mathematics any longer. In particular, implicit mathematical practices can be made explicit by direct comparison of practices by math-oriented and not math-oriented people.

Copyright © by the paper’s authors. Copying permitted for private and academic purposes.

In: A. Kohlhase, M. Fürsich: Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at <http://ceur-ws.org>

GRAY AND TALL note in [5, p. 1] about mathematical expressions that their “*ambiguity of notation allows the successful thinker the flexibility in thought to move between the process to carry out a mathematical task and the concept to be mentally manipulated as part of a wider mental schema*”. For example, the very simple mathematical expression “ $1/2$ ” can be looked at as the process of dividing 1 by 2 or as the product of this division, i.e., 0.5. Such expressions can be very elaborate and intricate, e.g. look at the complex formula Expression 3 given in Fig. 1. As mental models [8] shape how people interpret and experience the world, they can be very powerful.

Mathematical expressions provide a trained person with a specific mental model. They serve – as mathematical tool-of-the-trade – the distinct purpose to enable mathematicians to **proceptualize**, that is to provide focus on either the process (or procedure) of obtaining an information chunk, or on the conceptualization (or the idea) of the information chunk itself; see [5]. ANDRA ET AL. found in [2] that a formula condenses its information in a shorter but more implicit inscription. They note that a “*formula constitutes both a procedure and a symbolic narrative*” [2, p. 242, original emphasis].

As mathematical expressions can neither be considered text nor image, it was previously suggested that they form a separate category (see e.g., [6]) which humans perceive differently. This perception was coined by ARCAVI “symbol sense”, i.e., a “*complex and multifaceted ‘feel’ for symbols [...] a quick or accurate appreciation, understanding, or instinct regarding symbols*” [3, p. 31]. In 2005 W. SCHNOTZ presented a study proving that “*comprehension is highly dependent on what kind of information is presented and how it is presented*” [12, p. 73]. Even though he used only text and image information chunks, it is suspected that mathematical expressions build a category that enables math-oriented persons to understand math in their own way. The “formula shock” [13] effect is well-known, but how exactly do math-oriented people read mathematical expressions differently? If we can get a deeper understanding, then we might get new insights for the design of math search engines.

In [4] an eye-tracking study showed that persons with a high mathematical expertise related proof items with an according supportive image. They found that the participants tend to jump between text and image and it was suggested that relating the different representations enables the relation in different memory stores. Therefore, mathematical expressions with their visual and textual aspects present a cognitive tool for mathematical creativity by enabling proceptualization.

In [9] KAMALI AND TOMPA studied the retrieval for content in mathematical documents. Their empirical research also indicates that mathematical expressions are special. They conclude that mathematical expressions should neither be used as conventional document fragments nor with a too exact retrieval algorithm as both result in very poor search results. Instead they showed that algorithms that are based on making use of the Content MathML representation of the document corpus fare much better. So we know now that math search research should take the specifics of mathematical expressions into account. But we still don’t know why and how exactly. In our experiments we approached the answer in a very broad and exploratory way, so that we get clues for the answer. Our results should be viewed as hypotheses about the process of understanding mathematical expressions. These need to be confirmed or rejected in future research.

In Section 2 we first describe the set-up of our eye-tracking experiment. Next we present our findings and discuss them to develop first hypotheses for mathematical practices with respect to mathematical expressions. Section 3 concludes the paper.

2 Our Study

Ultimately, we are interested in the potentially distinct user groups of technical documents in a broad sense and want to create fitting mathUIs. Here, our goal consists in the comprehension of the intuitive use of mathematical expressions. One strength of mathematical expressions is the exact and concise formalization of mathematical information. Therefore we wanted to understand the process of reading and understanding mathematical expressions: How are they read and what are the mathematical intuitions of use? To get a better understanding of the relevance of mathematical expressions and user group behaviour w.r.t. them we set up an eye-tracking study.

With an eye-tracker a subject’s eye movements can be followed and measured while she is performing a task. According to e.g. [7], semantically relevant information increases the number of fixations. Hence, we were interested in the specific spots in a mathematical expression fixated the most or the longest and the order in which this was done by our users. The location where people look at first is the one from which they decode the meaning of the expression. In turn, this might indicate what they are trying to comprehend first and thus, what is the most relevant ‘object’ when searching for the mathematical expression.

2.1 Set-Up

We invited 23 participants to look at concrete mathematical expressions, particularly the three expressions seen in Fig. 1 in the order of numbering. The first expression represents a very simple equation system, the second one is slightly more

Expression 1

$$\begin{aligned} b &= 11 \\ a + b &= 16 \\ a &= ? \end{aligned}$$

Expression 2

$$\begin{aligned} x + 4 &= y \\ y &= 5 \\ ? &= x \end{aligned}$$

Expression 3

$$c_1(\delta(x))^{-\lambda_1} \exp\left(\int_{\delta(x)}^{\eta} \frac{z_1(s)}{s} ds\right) \leq a(x) \leq c_2(\delta(x))^{-\lambda_2} \exp\left(\int_{\delta(x)}^{\eta} \frac{z_2(s)}{s} ds\right)$$

Figure 1: Mathematical Expressions Presented to Test Subjects (Expression 3 cited from [1])

complex as the variables are not on the same side of the equation symbol, and the third one is a rather complex mathematical expression. After Expression 1 and 2 the participants were asked to select the correct answer.

To differentiate between mathematical practice and typical reading behaviour, we invited math-oriented and not math-oriented persons to participate in the test. They were invited via e-mail and were asked to enrol online, not knowing what kind of test to expect. This information was withheld to avoid losing participants that dislike mathematics in general. Before starting the test, the test subjects were asked to indicate their mathematical abilities and interest in mathematics on a range from low, medium and high. Based on this self-assessment and our judgement based on the correctness of their answers given as solutions to math problems in the test the subjects were grouped by “math-affinity” into MATH and NO-MATH participants respectively (see Fig. 2.1).

Group	Female	Male
MATH	5	5
NO-MATH	7	6

Figure 2: Participants

The mathematical expressions were shown to the participants as images on a Tobii t60 Eye Tracking Screen (17” and 4:3 ratio). Note that the seating position of some participants differs from normal positions while solving mathematical problems. Therefore the screen and integrated eye tracker were adjusted accordingly and had to be calibrated. In this set-up the user is able to speak and move

his head and upper body moderately in front of the screen which has its limitations in the tracking box of the eye tracker. To avoid losing tracking data, the eye tracker had to be recalibrated after finishing a task with major tracking losses. This could be the case if participants changed seating because of concentration.

2.2 Results and Discussion

As we were interested in the specific spots in a mathematical expression fixated the most or the longest over time, the visualization of our results in this respect in form of a heatmap or a gaze opacity map seemed the most suitable. **Heatmaps** as the ones in Fig. 5 were originally used to visualize the heat distribution on a surface for a set period of time, where typically the color “red” indicates the hottest transitioning over yellow, green and light blue to the color “deep blue” representing the coldest area. In an eye-tracking study the longest and most fixated areas are the hottest, the completely ignored spots the coldest. A **gaze opacity map**, used for example in Fig. 3, works like a mask based on a heatmap: The most looked at areas are the transparent, the ignored ones are the opaque areas of the mask. Therefore, with a gaze opacity map one can only see those areas that the participant group has looked at and thus, seen. Everything else is blacked out. To get a better understanding of the order in which objects in mathematical expression are looked at, we also used the visualization in form of a **gazeplot**, in which fixations are represented by dots that are connected and numbered according to their accumulated occurrence in a given time frame and whose size indicates the length of the gaze.

Let us first have a look at the result for the very simple Expression 1 in Fig. 1. We gathered the gaze opacity maps after the indicated time for the MATH and NO-MATH participant group in Fig. 3. Human eyes are adapted to fast and efficient information processing. In particular, there are certain visual patterns like vertical lines that are registered by optical cells specialized for the perception of this kind of visual pattern. If nothing strikes the eye, then a typical grasp of the input given starts in the centre. Thus participants in the NO-MATH group began to cope with the input in the center of the mathematical expression as expected. Interestingly, the MATH members did not. Instead they started by looking at the first equation symbol. We hypothesize that

Mathematical Practice 1: “*Math-oriented people use visual patterns for math detection.*”

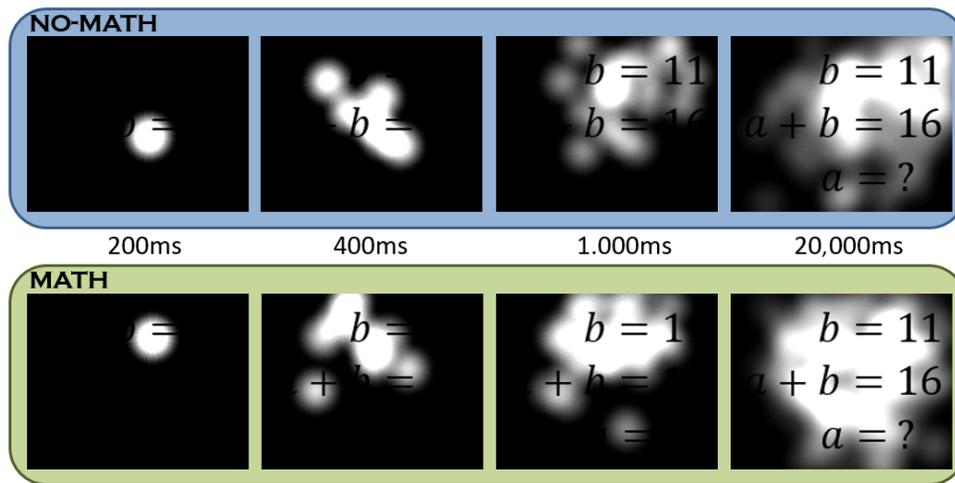


Figure 3: Development for MATH vs. NO-MATH Participants for Expression 1

If this could be confirmed, then this might have broad consequences. For instance, the formula shock which many people suffer could be overcome by new *visual* training methods. Also, math search engines could offer new kinds of query interfaces according to such visual patterns and they could be another facet for sorting search results.

The gaze opacity map taken after 400ms is also revealing. Here, NO-MATH members start to explore the symbols in the mathematical expression near the center, whereas MATH members seem to gather information about the variable name “*b*” and the operator “+”. This behavior is continued over the next milliseconds. After 1000ms the MATH participants have discovered the last equation symbol, only afterwards they are interested in the actual numbers. In contrast, the NO-MATH participants searched for another occurrence of the element “*b*” and its concrete value “11”. We noted earlier that mathematical expressions have a procedural and a symbolic/calculational aspect at the same time. Using the procedural aspect of the equation system, one has to gather information about the procedural structure first. That is, one observes the kind of mathematical expression first (here after 100ms “equation”), then one looks for the **major** variable, i.e., the one that is essential to understand the task, and the operator (here after 400ms “*b*” and “+”), and finally, one is interested in the concrete values (here after 20,000ms “11” and “16”). Note that

Mathematical Practice 2: “*The decomposition of a mathematical expression is organized along its procedural character.*”

The second variable “*a*” in the second line of the equation system is not looked at at all: soon after 20,000ms all participants of the MATH group progressed to the solution task. We suspect that the math-oriented subjects know that the plus-operator is binary, so the *name* of the minor variable is not relevant at this point.

Expression 2 was the second mathematical expression shown to our suspects (see Fig. 4) and already more complex than the first one because of its unusual structure. Note that the first clear fixation could already be noted after 100ms in contrast to the first for Expression 1 which took 200ms. Our participants didn’t know before the test what kind of test to expect. We therefore suspect that this time difference of fixations for Expression 1 and 2 is due to an adjustment to the test. Now everyone was aware that some mathematical expression was to be expected.

It is quite stunning, how the strategy of members of the two groups differed from each other for Expression 2 in Fig. 4. If we look at the first two screenshots taken at 100 and 400ms respectively, we notice that the numbers are singled out by the NO-MATH participants whereas the variables were in focus by the MATH members. This means the grasp of these two distinct types of objects (literals and variables) can be easily done as it is the first means to comprehend Expression 2 and maybe generally mathematical expression. So we hypothesize:

Mathematical Practice 3: “*Literals and variables can be easily differentiated by most people (with an educated background).*”

If we look back at our subjects’ eye movements in Expression 1 in Fig. 3, we notice that the MATH members started with focusing on the first equation symbol. According to Math Practice 1 they might do so because they perceive a specific visual pattern. This pattern could be the one of an “equation system”, e.g. three columns with the middle one containing “=”-symbols. Once this is established, as for example can be seen in the screenshot in Fig. 3 at 400ms, they look for the variables (and arguably operators). The interpretation of “*b*” being perceived as variable in Fig. 3 and not just as

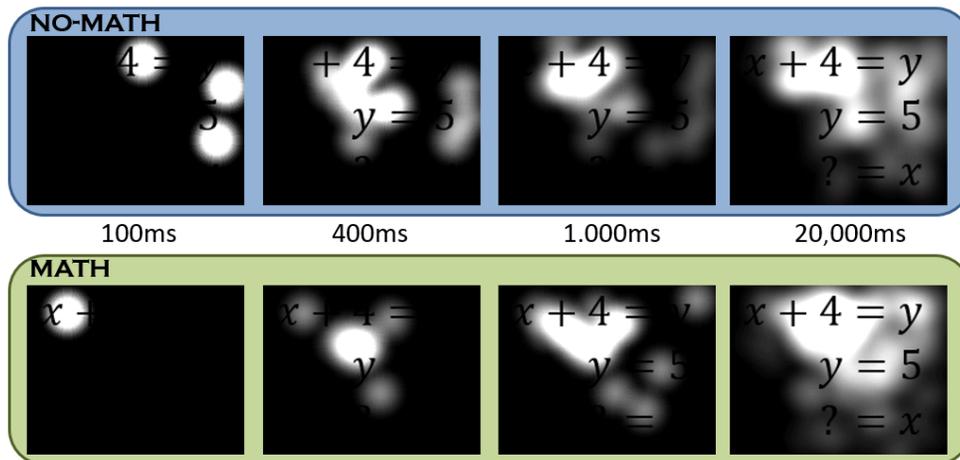


Figure 4: Development for MATH vs. NO-MATH Participants for Expression 2

surrounding ‘text’ as in the screenshot at 400ms for the NO-MATH users is based on the contemporaneous gaze at the operator symbol. The Expression 2 was shown to participants directly after having been presented Expression 1. Therefore this first phase may have been dropped when reading Expression 2 and the subjects jump right in by looking for variables in Fig. 4. We observe that the variable “ a ” in Expression 1 is not fixated at the beginning, whereas both variables “ x ” and “ y ” are in Expression 2.

Note that the unusual form of Expression 2 makes the variable “ y ” into a suspected second major variable, whereas the variable “ a ” in Expression 1 was a minor one in the sense that it was just the other argument for the binary “+”-operator as can be seen in Fig. 1. Not only are variables placeholders for values, their names are also “holding a place” for operator arguments. This specific placeholder role of variable names can be extended to simple mathematical expressions. As this seems to be integrated in the decomposition of mathematical expressions, we hypothesize:

Mathematical Practice 4: “Simple mathematical expressions can be treated as placeholders for argument positions and therefore as neglectable in the mathematical expression decomposition process.”

Subexpressions are treated like variables in the perception of mathematical expressions just as if they were placeholders for syntactic constituents that make a formula functionally well-formed.

Now we might ask, why do the MATH members look at the variables first, that is, why is this the second phase in the decomposition of a mathematical expression? We suggest that math-oriented persons want to grasp the task first. So here they make out the “equation system” first, then they try to understand what the distinct elements of the task are. We noticed the prioritization with Math Practice 4, but we can also observe that variable names as signifiers for variables are important for the task set-up. We argue that the MATH subjects’ focus on the variables in Fig. 3 can be attributed to their specific placeholder role in a mathematical expression. Thus, as they were singled out so markedly, we like to note:

Mathematical Practice 5: “Variable names are interchangeable as they signify arbitrary but fixed values.”

This placeholder role of variables is already made use of in many math search engines. Concretely, content MathML is used to allow the user to input arbitrary variable names, but search for the generic expression in the underlying corpus. Our hypothesis rather confirms that math-aware search engines need the semantic capture of math knowledge to be intuitive. It is a question whether the exchangeability of variable names as *argument* placeholders (and not just value placeholders) has yet been considered. We observed that the name is not looked at if the operator is so well known that the existence of this variable can be assumed, which we framed as their characteristic of being “minor” or “major”. This hypothesis may be used to differentiate importance among substitutions in search results.

The difference of accessing the complex mathematical expression between math-oriented and not math-oriented subjects in the respective heatmaps after one minute of watching – presented in Fig. 5 – is striking. The focus on the centre of the expression by NO-MATH members supports Math Practice 1. We suspect that they also didn’t even try to comprehend the formula, so they retreated to answer a question on the meta-level “What kind of mathematical expression is this?” with “an inequality”. This is corroborated by the absence of the concepts “lower/upper bound (of an integral)” in the think-aloud protocol by NO-MATH participants.

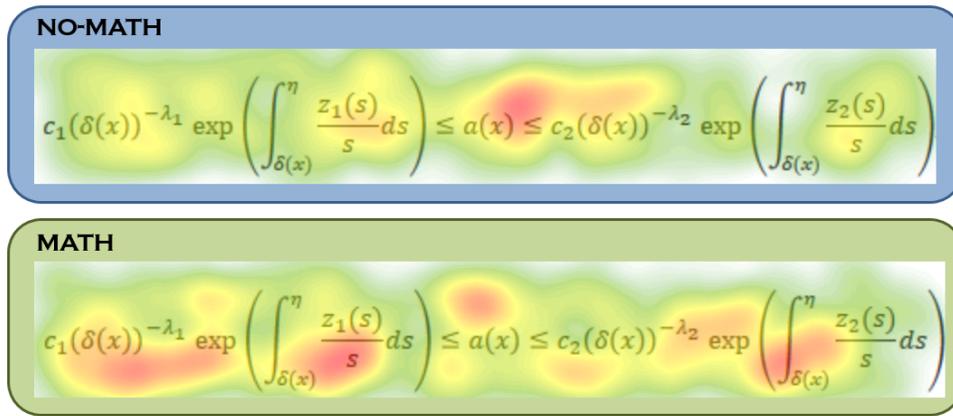


Figure 5: HeatMap (1min) of Complex Formula for MATH vs. NO-MATH Participants

Interestingly, neither the integral symbols nor their limits were glanced at by NO-MATH subjects, whereas even the integral symbol itself was fixated by some MATH ones. Note that the precision of the hotspots in the heatmaps for Expression 3 suffered especially as all participants changed their seating position due to the complexity of the mathematical expression. As a consequence the calibration of the eye-tracker to the individual pair of eyes deteriorated accordingly. Assuming that MATH members really looked at the integration symbol itself, we may hypothesize:

Mathematical Practice 6: “In the decomposition of a mathematical expression, some symbols (e.g. the integral sign) carry structural information, which is read independently from its functional information. The structural information does not include parameter values.”

Math Practice 5 strengthens this statement, as major variables help to understand the structure, whereas minor ones the function. For math search interfaces this might mean that \LaTeX is not a superior input language for search queries after all, as it requires syntactic correctness, which according to Math Practice 6 might be too strict.

In Fig. 6 it is obvious that the MATH participants started out from the left to decode the mathematical expression. One

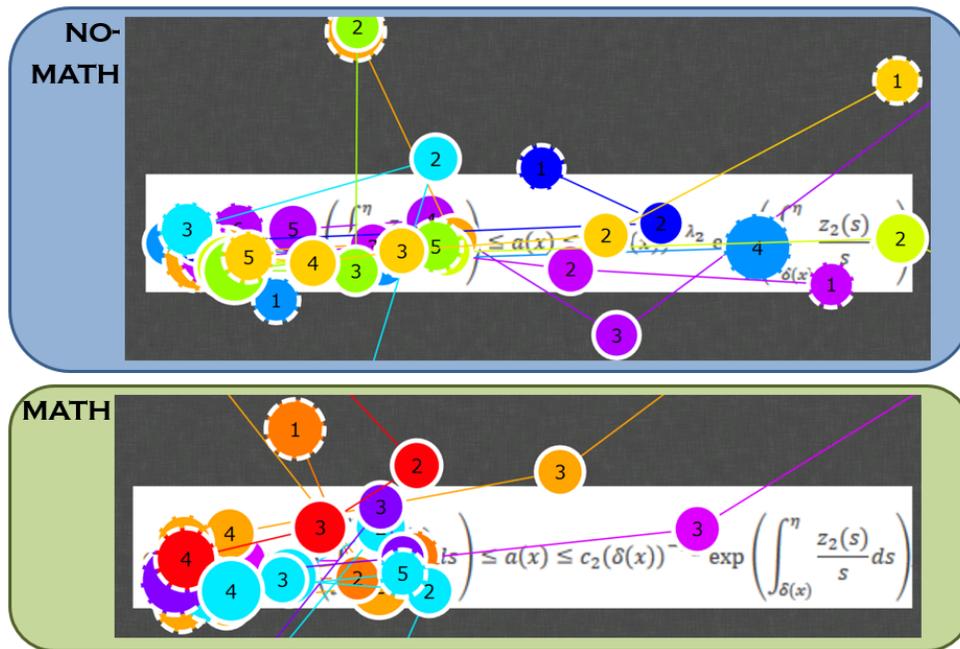


Figure 6: GazePlot (1sec) of Complex Formula for MATH vs. NO-MATH Participants

could think that this is due to the natural reading direction in Western societies and indeed, the very first deciphering of the mathematical expression has to be expected to be on the very left. But it is not evident that they continued to do so until the

first meaningful sub-expression, i.e., the lower limit, was completed (compare with the overview scanning behaviour from left to right by the NO-MATH members). For the MATH participants the entire decoding process starting after the grasp of the first sub-expression contained back and forth jumps to comprehend the intra-relations of the expression.

Moreover, from Fig. 5 we can gather that they looked at each part of the formula separately, whereas the NO-MATH subjects only understood it on the meta-level, that is the mere structure of the formula being an inequality that tells us something about a function $a(x)$. From this finding we can hypothesize that

Mathematical Practice 7: “The decoding of a mathematical expression starts from the left until a first meaningful sub-expression is grasped. Further comprehension is chunked into understanding sub-expressions and their relations.”

This makes it very probable that they remember the parts in a formula that are located to the left better than ones to the right as these sub-expressions will tend to be the recognition anchors. A potential consequence of this hypothesis consists therefore in the suggestion that the ranking of search results has to include a factor that is concerned with the position of the found fragment where more to the left should rank higher than one to the right.

2.3 Results and Discussion with respect to Proceptualization

To assess whether the MATH members scan the mathematical expression using proceptualization, we have to find out whether the participants switch between its procedural and conceptual aspect. Instead of conducting a visual cluster analysis for the data elicited in our study, here we had to turn to a visual inspection of each participant’s data in Expression 3, particularly the order in which he or she deconstructed the complex formula. We exemplify this with one math-oriented subject, but we verified this with all.

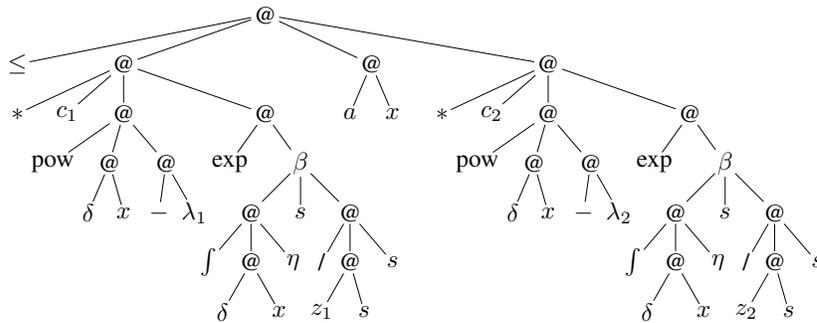


Figure 7: Functional Structure of Expression 3 in Fig. 1

Let us first look at the functional structure of the complex formula, that is the content tree. Fig. 7 shows a tree representation of Expression 3 in Fig. 1, where the tokens (variables and symbols) are at the leaves and the applicative/binding structure is given by internal nodes labeled with “@” (application) and “β” (binding). The top of the tree is an application of the “≤”-relation to two products (apply “*” to multiple arguments) and so on down the tree. The subtree rooted and the node “β” is the representation of the integral $\int_{\delta(x)}^{\eta} \frac{z_1(x)}{s} ds$. The binding construction has three arguments: a binding operator $\int_{\delta(x)}^{\eta}$, the bound variable s and the body $\frac{z_1(x)}{s}$.

Now if we superimpose the sequence of fixations identified in our eye-tracking experiment (see Fig. 8), we obtain the situation in Fig. 9 – we restrict our attention on the first/left argument of the inequality.

We interpret the first two steps (1,2) as formula segmentation process: the proband looked at large left bracket for orientation, identified the first factor $c_1(\delta(x))^{-\lambda_1}$ to its left, and then found the matching right bracket. Steps 3, moves to the integral symbol and step 4 fixes the lower bound – the upper bound does not seem to receive much by the MATH group. Finally, step 5 passes to the body of the integral before step 6 discovers that the integral has the exponential function applied to it. Then the attention shifts to the first factor again and explores its base and exponent (steps 7 and 8).

This exploration of the left-hand-side of the inequality is followed by an orientation towards the right-hand-side via the two “≤” symbols, and then an exploration of the right hand side that is very similar to the one detailed in Fig. 9. We observe even though the constant c_1 had not been fixated on the left –presumably, since it is very simple,

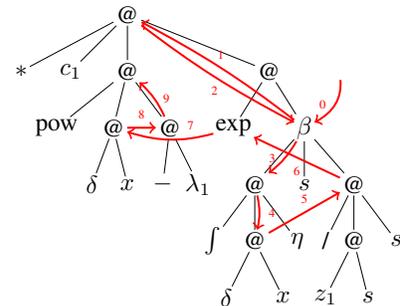


Figure 9: Exploration of Expression 3

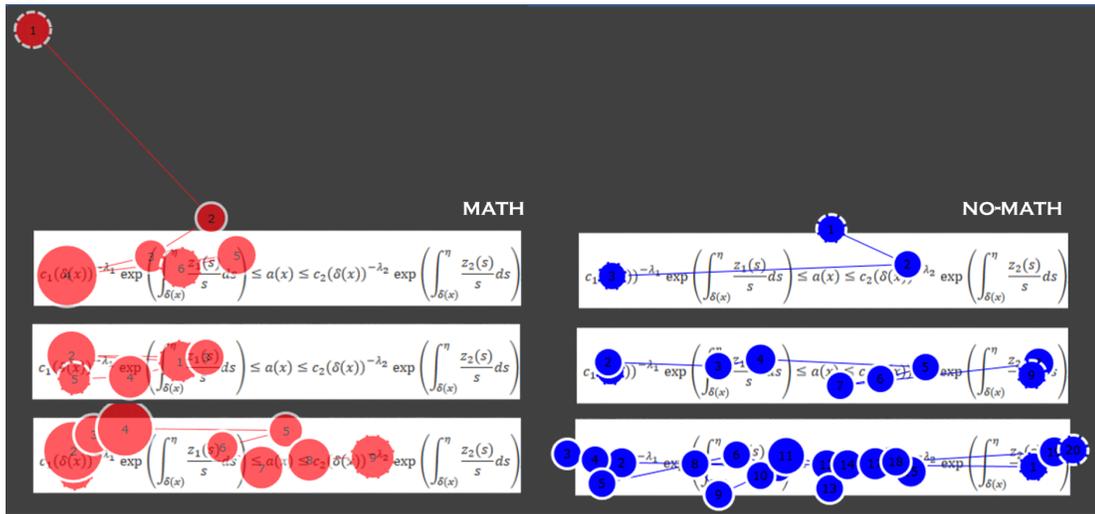


Figure 8: Gazeplot of the Order of Single Saccades by a MATH and a NO-MATH Participant

the corresponding c_2 was – presumably the reader noticed the contrast. The final part of the formula explanation by the MATH group consisted in a systematic comparison of the left and right sides of the inequality: the fixations jump between the corresponding subtrees (c_1 vs. c_2 , λ_1 vs. λ_2 , and z_1 vs. z_2 , but $\delta(x)$ used uniformly).

Those patterns of visual exploration can be detected in recordings of multiple MATH participants which supports Math Practice 7 and moreover, leads to

Mathematical Practice 8: “Proceptualization is not only used when using mathematical expressions but also when decoding.”

In contrast to this the NO-MATH subjects explored the inequality in two essentially left-to-right sweeps. The only significant reversals in this are the decoding of the two infix “ \leq ” symbols in the middle (see the second frame on the right in Fig. 8) and some consideration for the large brackets on the left (see the third frame on the right in Fig. 8). After these two sweeps, structured exploration of the inequality seems to cease: the sequencing of fixations seems essentially random. We also observe that the not math-oriented group seems to have less fixations and those that are identified seem less intensive, even though the overall time spent looking at the inequality is the same.

3 Conclusion and Future Work

Mathematical expressions are widely used in technical documents, therefore fitting mathUIs have to be created that empower their users to better utilise those tools of the trade. With our eye-tracking experiment we have identified a set of mathematical practices when decoding a math expression.

We observed that math-oriented participants have visual patterns for math detection and are able to decompose mathematical expressions with clearly organized procedures. Literals and variables are easily distinguished by most readers, whereas variable names are not the pivotal point essential for problem solving. This might be different with mathematical symbols, that do not necessarily include all parameters of their information. The comprehension of mathematical expression by math-oriented participants is chunked into understanding sub-expressions and their relations. They start decoding the mathematical expression from left to right, until a first meaningful sub-expression is grasped. Those six hypotheses stated, enable a deeper understanding of the way mathematical expression are perceived and show that further research is needed.

This set of math practices is still hypothetical but nevertheless give rise to interesting new perspectives for the design of mathUIs, particularly math search engines. We admit that the observed practices are not yet orthogonal and that we have to refine the primitives which we feel obliged to do in the near future. Moreover, it would be interesting to understand what mathematicians memorize when having looked at mathematical expressions and what parts they indeed recall.

References

- [1] Ramzi Alsaedi, Habib Mâagli, and Noureddine Zeddini. “Existence and global behavior of positive solution for semilinear problems with boundary blow-up.” English. In: *J. Math. Anal. Appl.* 425.2 (2015), pp. 818–826. ISSN: 0022-247X. DOI: 10.1016/j.jmaa.2014.12.066.
- [2] Chiara André et al. “READING MATHEMATICS REPRESENTATIONS: AN EYE-TRACKING STUDY”. In: *International Journal of Science and Mathematics Education* 13.2 (2015), pp. 237–259. ISSN: 1573-1774. DOI: 10.1007/s10763-013-9484-y. URL: <http://dx.doi.org/10.1007/s10763-013-9484-y>.
- [3] Abraham Arcavi. “The role of visual representations in the learning of mathematics”. In: *Educational Studies in Mathematics* 52.3 (2003), pp. 215–241.
- [4] Jana Beitlich and Kristina Reiss. “Das Lesen mathematischer Beweise - Eine Eye Tracking Studie”. In: *Beiträge zum Mathematikunterricht 2014*. Ed. by J. Roth and J. Ames. WTM-Verlag, 2014, pp. 157–160.
- [5] David O. Tall Eddie M. Gray. “Duality, Ambiguity, and Flexibility: A ‘Proceptual’ View of Simple Arithmetic”. In: *Journal for Research in Mathematics Education* 25.2 (1994), pp. 116–140. ISSN: 00218251, 19452306. URL: <http://www.jstor.org/stable/749505>.
- [6] Hans Freudenthal. *Didactical Phenomenology of Mathematical Structures (Mathematics Education Library)*. Dordrecht: Reidel. ISBN: 9027715351.
- [7] John M. Henderson, Phillip A. Weeks Jr., and Andrew Hollingworth. “The effects of semantic consistency on eye movements during complex scene viewing”. In: *Journal of Experimental Psychology: Human Perception and Performance* 25.1 (1999), pp. 210–228. URL: <http://dx.doi.org/10.1037/0096-1523.25.1.210>.
- [8] Philip N. Johnson-Laird. “Mental Models in Cognitive Science”. In: *Cognitive Science* 4.1 (1980), pp. 71–115. DOI: 10.1207/s15516709cog0401_4. URL: http://dx.doi.org/10.1207/s15516709cog0401_4.
- [9] Shahab Kamali and Frank Wm. Tompa. “Retrieving Documents with Mathematical Content”. In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Ed. by Gareth J. F. Jones et al. Dublin, Ireland: ACM, 2013, pp. 353–362. ISBN: 978-1-4503-2034-4. DOI: 10.1145/2484028.2484083. URL: <http://doi.acm.org/10.1145/2484028.2484083>.
- [10] Andrea Kohlhase. “Search Interfaces for Mathematicians”. In: *Intelligent Computer Mathematics 2014. Conferences on Intelligent Computer Mathematics*. (Coimbra, Portugal, July 7–11, 2014). Ed. by Stephan Watt et al. LNCS 8543. Springer, 2014, pp. 153–168. ISBN: 978-3-319-08433-6. URL: <http://arxiv.org/abs/1405.3758>.
- [11] Keith Rayner. “Eye Movements in Reading and Information Processing: 20 Years of Research”. English. In: 124.3 (1998), pp. 372–422.
- [12] Wolfgang Schnotz. “An Integrated Model of Text and Picture Comprehension”. In: *The Cambridge Handbook of Multimedia Learning*. Ed. by R. E. Mayer. Cambridge University Press, 2005 (2014), 49–69 (72–103).
- [13] Alexander Strahl, Julian Grobe, and Rainer M’uller. “Was schreckt bei Formeln ab? - Untersuchung zur Darstellung von Formeln”. In: *PhyDid B - Didaktik der Physik - Beiträge zur DPG-Frühjahrstagung 0.0* (2010). URL: <http://phydid.physik.fu-berlin.de/index.php/phydid-b/article/view/169>.

The plain text trap when copying mathematical formulæ

Paul Libbrecht
PH Weingarten
Weingarten, Germany
paul@hoplahup.net

Matija Lokar
University of Ljubljana
Ljubljana, Slovenia
Matija.Lokar@fmf.uni-lj.si

Abstract

When an object, of any nature, is displayed and selectable on a computer screen, users expect it to be copy-and-paste-able: one can invoke the copy function and insert (paste) it at other places, within the same programme or beyond. This holds for many different kinds of objects: texts and images, at least. Unfortunately, for mathematical objects, this is rarely so.

Most operating systems offer multiple channels to carry exchanged content but most mathematical systems do not take advantage of it: they transfer the content in plain text, expecting it to have the right syntax or, if necessary, expecting the user to use a different copy function so that the right syntax is exchanged.

While ways to circumvent these issues are available, they are mostly not used by mathematical software. We explore potential justifications and describe for which type of users, these justifications do not apply.

To support this, we report briefly on the experiment students about their expectations and observations on the above mentioned process.

1 Intro: Transferring naturally between Competent Softwares

Copy and pasting mathematical formulæ between different systems is a desirable and common action. It is widely known that most mathematical systems have areas where they are very effective and while they only provide an approximate service in other areas. For example, dynamic geometry systems are good at constructive geometric figures and letting them be manipulated but also offer other functions in which they are rather less good: for example, most of them support some part of the $\text{T}_{\text{E}}\text{X}$ language to display formulæ and some offer computer algebra features; each of these extra features are very limited.

Most computer algebra systems (Maple, Mathematica, Macsyma, MuPad, Reduce) themselves have their own user-input parser and use TeX for display purposes mostly [Zha03]

Instead of relying on such limited extra features, users have the possibility to transfer the mathematical object between a system and another so that the receiving system offers its high quality features to solve the translated problem.

To perform the transfer of mathematical objects, the natural procedure of copy and pasting is often expected by users but, in the mathematical world, this procedure is decorated with special methods of many sorts, to

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at <http://ceur-ws.org>

work somewhat properly. Switching between different systems, transferring relevant data, worrying about things getting “out of sync”, differences in command sets and capabilities between different applications, soon becomes overwhelming[Lie00].

A simple sequence select, copy, switch, insert, paste is mostly expected by users. However, it is not rare a more complex procedure is needed such as the invocation of a special copy or adjusting the pasted content before it is further processed. For example[Kin02], the Stanford Interactive Workspaces’smart clipboard can copy and paste data between incompatible applications on different platforms[Kic00]. The smart clipboard must transparently invoke the machinery whenever the user performs a copy and paste operation. A more sophisticated but less general approach, semantic snarfing, as implemented in Carnegie Mellon’s Pebbles project, captures content from a large display onto a small display and attempts to emulate the content’s underlying behaviors [Mey01].

In the middle of this process lies the exchanged content. Most operating systems offer multiple channels to carry this exchanged content but most mathematical systems do not take advantage of it: they transfer the content in plain text, expecting it to have the right syntax or, if necessary, expecting the user to use a different copy function so that the right syntax is exchanged.

Why is this a problem? There are ranges of issues which are encountered by users and are all due to this choice of plain text. They range from syntax mismatch to unmasterable expressions, from the failure of apparent syntax compatibility to the somewhat arbitrary text-linearization of the text appearing within the formula.

While ways to circumvent these issues are available since long in a standardized form (clipboard flavours or alternative representations in MathML), they are not used by mathematical software. We propose potential justifications and describe for which type of users, these justifications do not apply.

2 Observation Methods for Copy and Paste of Formulæ

To be able to evaluate what are the expectations of users when transferring between systems, the authors employ their usage and teaching experience. These includes courses dedicated to the introduction of various systems (e.g. introduction to LaTeX, to computer algebra systems, or to dynamic geometry) in undergraduate teacher education classes. While this class of users is clearly not representative of the complete population of users of mathematical systems, it represents the important share of moderately technical users and also precludes those that will educate broad masses of citizens.

With this class of users, a practical experiment has been done at the University of Ljubljana for about 30 students in the first cycle professional study program Practical Mathematics: a mathematical task was given, explicitly requiring the exchange of mathematical expressions between various systems, of which comments and reports were expected.

In the first weeks of the subject called *Computer tools in mathematics* they get used to typical examples of mathematical software they are likely to apply in their career. They obtain mostly the basic knowledge, so they know only the most common functions and functionality of the applications. After a few weeks they received the experiment’s task as follows (one of the weekly assignments): They should solve a certain mathematical task (and report in detail the process of obtaining the solution). In all the tasks given we foreseen the usage of different tools. Most often the combination of computer algebra system, dynamic geometry system and numerically oriented matrix software was needed. So even when certain tasks could be solved within one software, their their limited familiarity with the software lead them to use multiple softwares. For reporting they mostly used the most common word processing software. We were interested how they will cope with the process of exchanging the mathematical object between programs and how they will report on that where transfer of various mathematical objects has been expet. The subjects were instructed beforehand to report on all difficulties and obstacles.

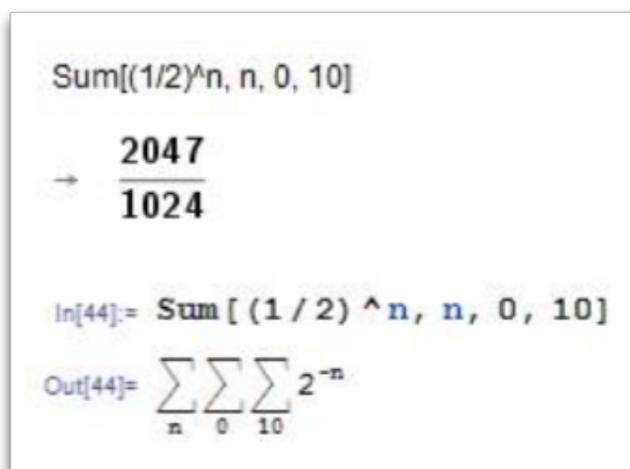
To be able to evaluate what the mathematical systems are able to import, clipboard inspectors are used: On Windows the ClipSpy utility which shows the bytes allocated for each flavour within the clipboard. The application seems to show most of the information accessible by API as documented on [MSCI]. On MacOSX, ClipboardViewer (an example application of the Apple Developer Tools) has been used. It shows, similarly, an association between the “Uniform Type Identifiers” (the name of the clipboard flavour types defined by Apple, see [ApUTI]) and the byte-values. Both the Uniform Type Identifiers (on MacOSX) and the Windows flavour names are a flexible mechanism to encode the type of content-flavours: the strings define, in a kind of cooperative agreement way, which data-type is put in the clipboard. Platform makers define basic types (e.g. basic images and texts) while applications are free to define new formats. In the case of UTI, a mechsanim of inheritance is provided.

3 Issues when employing plain text in copy and pasting formulæ

The first family of issues that we have met is that default copy mechanisms are rarely universally applicable and are, very commonly, restricted to only copy and paste internally, for which the system clipboard is not used (as copying from the system to itself is much safer). This same default copy function tends to copy a representation that is close to a source format within the text flavours. While this representation may be readable by users, it is not effective to input in most other mathematical systems as the input syntax is specific to each system. Mismatches of syntaxes then need to be manually fixed by the user who has to understand both syntaxes; something that is cognitively demanding when the mathematics thinking also demands cognitive resources.

Examples of such incompatibilities include an incomplete LaTeX compatibility of a dynamic geometry system: the user is left to explore bits by bits what works and what does not and, contrary to LaTeX, the documentation for the supported features is far smaller.

The second family of issues lies in the apparent compatibility of syntaxes between mathematical systems: While some expressions seem naturally exchangeable, e.g. simple polynomials written with the “^” as exponent, this compatibility breaks very quickly (e.g. these polynomials might be usable in a JavaScript source but not in a Python source because the exponent there is “**”). Much more delicate is the exchange of formulæ between Mathematica and Geogebra as is displayed in the picture below, produced by one of the experiment subjects: it shows that the Sum operator between the two computer algebra systems does not follow a completely similar syntax even if it shares quite some similarity as is the case for elementary formulæ between these systems.



Sum[(1/2)^n, n, 0, 10]

→ $\frac{2047}{1024}$

In[44]:= Sum [(1 / 2) ^ n , n , 0 , 10]

Out[44]:= $\sum_n \sum_0 \sum_{10} 2^{-n}$

Figure 1: The same expression in Mathematica and GeoGebra...

A third issue lies in the need to use dedicated copy functions. It forces the user to navigate through sequential menu choices to decide which format to copy (at worst she needs to use a dedicated command before). Among the particularly invasive ones lies the frequent function “Copy as MathML” which puts the (generally big) MathML source code as text source and which is intercepted by paste actions (“this looks like MathML”): the difficulty to reach the menu and the rather wild appearance of the XML expression makes this operation very far from natural.

Finally, another issue in using plain-text is on web-browsers: As MathML there is generally represented as part of an HTML text page, the copied content is text, sometimes styled text. The copy function generally copies only an unordered sequence of characters which may be valid only if fractions or roots are not present; moreover, the copy function is supported by a selection mechanism which completely ignores the underlying mathematical structure (even that of MathML).

4 User types and their affinity to fiddling with text

While we acknowledge that using plain text to copy and paste often yields readable syntax and, what is probably more important, repairable syntax, it can be accepted differently depending on the users.

Experts we have met generally accept well the syntax differences; they can even be a good hint to the capacity spectrum of the computing system. However, there is a clear need to master well the different syntaxes, which is generally not available in many user types we met.

Such comments appeared in our experiment to illustrate our case:

- *“.. that Copy/Paste method does not work with drawing plots in different programs. For drawing plots each one of used programs has its own ‘language’ which we have to use. “*
- *“... we were very rarely able to use the Copy/Paste method to transfer expressions between programs and even then some corrections were usually needed for programs to work”*
- *“After several tries I gave up. The only way to transfer an expression between programs is to manually type it...”*
- *“To input matrix in X is surprisingly identical to input matrix in Y. But unfortunately here the joy ends.”*

In the discussion of the task afterwards students we performed experimentation with all express their deep disappointment about the softwares. When started the task they mostly all expected there will be just some minor problems. As one students said “it is all mathematical software and mathematics is just one, so I expected no problems in using Copy/Paste”

We conclude that the learning of syntaxes is the biggest hurdle for these students and that alternative ways need to be found to make a more productive use of the interchanges between pieces of software.

5 Conclusion: Standards that Support Copy and Paste

While copying plain text formulæ has been considered a universal way, we have described a population of users for which this approach fails and for which different exchange mechanisms are needed.

Standards exist to this end. They include the clipboard flavour names of Windows and UTIs on MacOSX as specified in the media-types of MathML: these describe the names of three clipboard flavours that can encode MathML for several applications. Using them, it is possible for a computing system to use its internal mechanisms to export MathML content in the form of a parseable tree and MathML presentation in the form of a rendered LaTeX fragment.

Similarly, embedding MathML within HTML fragments might potentially include the wealth of MathML exchanges as indeed a considerably bigger consideration exists for Web-pages’ content. However, many of the efforts towards empowering web-oriented copy-and-paste such as the Clipboard API draft tend to consider [Che15].

Exchanging MathML offers another possibility to combine multiple types: through the `semantics`, `annotation`, and `annotation-xml`, the markup may contain alternative representations of the formulæ in different media-types (which we consider equivalent to clipboard flavour names), see [Car13, §6.4]. It is not clear if this *competing* alternate-representations mechanism may produce different results.

If doing this, the receiving party can choose the format it best understands from the list of clipboard flavour names and from the alternate representations in MathML. Thus a layout software might use a MathML-presentation or Rich Text Format fragment, or even a picture; however another computing system would use the MathML-content fragment or another flavour which is potentially better suited (e.g. to accommodate for bilateral import-export functions). Moreover, receiving multiple representations is possible with clipboard flavours and with MathML alternate representations. This allows the recipient to request all of the data-streams for each of the formats and potentially evaluate further its processability or offer a choice to the user using the internal display mechanisms (as investigated in [Lib09]).

As for systems which do not understand any of the mathematical oriented flavours: a picture, a rich text format or... a fragment of plain text (as a last resort) might be satisfactory.

References

- [ApUTI] Apple Inc. Introduction to Uniform Type Identifiers *Mac Developer Library* Available at https://developer.apple.com/library/mac/documentation/FileManagement/Conceptual/understanding_utis/understand_utis_intro/understand_utis_intro.html.
- [Car10] D. Carlisle. MathML on the Clipboard, Blog entry. <http://dpcarlisle.blogspot.de/2010/01/mathml-on-clipboard.html>.

- [Car13] D. Carlisle and P. Ion and R. Miner. Mathematical Markup Language, Version 3 (2nd edition) W3C. <http://www.w3.org/TR/MathML3/>.
- [Che15] D. Cheng Clipboard API: remove dangerous formats from mandatory data types public-webapps mailing list post on 2015-06-09. Archive of the thread visible at <https://lists.w3.org/Archives/Public/public-webapps/2015AprJun/0819.html>.
- [Kic00] E. Kiciman and A. Fox. Using Dynamic Mediation to Integrate COTS Entities in a Ubiquitous Computing Environment. in *Handheld and Ubiquitous Computing: Second International Symposium, HUC 2000 Bristol, UK, September 25–27, 2000 Proceedings*, 221–226, 2000.
- [Kin02] T. Kindberg and A. Fox. System software for ubiquitous computing. *IEEE pervasive computing*, 1(1):70–81, 2002.
- [Lie00] H. Lieberman and T. Selker. Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 3.4(39):617–632, 2000.
- [Lib09] P. Libbrecht, É. Andrès, and Y. Gu Smart pasting for ActiveMath Authoring *Proceedings of MathUI 09 Workshop*, <http://www.activemath.org/workshops/MathUI/09/>, 2009.
- [MSC1] Microsoft Inc. Clipboard Class, API Documentation available at <https://msdn.microsoft.com/en-us/library/system.windows.clipboard.aspx> (checked 2016-07-20).
- [Mey01] B. A. Myers and C. H. Peck and J. Nichols and D. Kong and R. Miller. Interacting at a Distance Using Semantic Snarfing. in *UbiComp 2001: Ubiquitous Computing: International Conference Atlanta Georgia, USA, September 30–October 2, 2001 Proceedings*, 305–314, 2001.
- [Pad04] L. Padovani and R. Solmi. An Investigation on the Dynamics of Direct-Manipulation Editors for Mathematics. in *Mathematical Knowledge Management: Third International Conference, MKM 2004, Białowieża, Poland, September 19–21, 2004. Proceedings*, 302–316, 2004.
- [Zha03] L. Zhang and R. Fateman. Survey of user input models for mathematical recognition: Keyboards, mice, tablets, voice. *Computer Science Division, University of California*, 2003.

Proposal for Coexistence of Mathematical Handwritten and Keyboard Input in a WYSIWYG Expression Editor

Juan Lao-Tebar
juan@wiris.com

Francisco Álvaro
falvaro@wiris.com

Daniel Marquès
dani@wiris.com

WIRIS math, Barcelona, Spain

Abstract

Despite math notation is well-known and extensively used in many fields, introducing math expressions into a computer device is not straightforward and requires specific notations. For this reason, users commonly use editors that make this task easier, and even handwriting can be used directly as input method. In this paper, we propose a design of interface and behaviour for the coexistence of a keyboard-based mathematical editor and a handwritten mathematical expression recognizer. Advantages of both modes are analyzed and it is shown that they are complementary, so a design based on coexistence is more appropriate for the final user than single modes. Also, regarding accessibility support, we provide some hints and guidelines on an implementation that offers a good experience to users with disabilities.

1 Introduction

In recent years, the World Wide Web Consortium (W3C) and major web browsers have invested heavily in developing and expanding web standards to provide a greater separation between content and presentation to improve user experience. Specifically, HTML5 introduced new types of input fields for web forms. In the new standard, an input field can be used to enter dates, time, colors, emails, phone numbers, ranges or even something as simple as a floating number, taking in account users local designation of decimal mark.¹

Years ago, if a developer needed to use one of those input types in a web form, the developer had to include javascript code in the page. (S)he had to write his own code or use third-party libraries. This caused several problems to the user. Since every page had to implement their own way of entering special input types, cohesion in the user interface and behaviour did not exist. In addition, language support, accessibility features and responsiveness depended on each implementation.

Web browsers do not have a native mathematical input field for web forms, since HTML5 specification does not define it.² In addition, due to the complexity of the edition of mathematical expressions (and even if HTML5 defined a mathematical input field), the existence of third-party solutions would improve the user experience and notation coverage. Something similar to the case of rich-text editors, where third-party solutions such as TinyMCE³ or CKEditor⁴ are used to provide a full featured web rich-text editor since W3C and web browsers

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at <http://ceur-ws.org>

¹<https://www.w3.org/TR/html5/>

²<https://www.w3.org/TR/html-markup/input.html>

³<https://www.tinymce.com/>

⁴<http://ckeditor.com/>

do not provide a standardized solution for this requirement.

There are different libraries available to deal with math expressions in previously discussed scenarios. Some of them let the user insert expressions using L^AT_EX, like WikiEditor⁵ (content editor for Wikipedia), while other libraries offer a more user-friendly interface, where what you see is what you get (WYSIWYG) and no previous knowledge of any language or format is needed. Examples of this kind of libraries are DragMath,⁶ MathQuill,⁷ CodeCogs⁸ or WIRIS editor.⁹

With the rise of touch devices and artificial intelligence systems, new systems are appearing for recognizing handwritten mathematical expressions, like MyScript¹⁰, MathBrush [ML15], m_{in} [SHP⁺12] or WIRIS hand.¹¹ These systems have some advantages compared to keyboard input: they offer a more comfortable interface on touch devices and they are more user-friendly and intuitive than classic overloaded toolbars that keyboard-based editors usually have. However, sometimes a user may prefer to insert expressions using the classic keyboard for several reasons:

- Lack of a touch device or not achieving the required accuracy with the mouse when drawing an expression.
- Suffering from visual impairment.
- In general, keyboard input offers a wider variety of mathematical expressions than a handwriting recognizer.
- Creation of expressions with a specific style and formatting (colors, position of elements, etc.).

For these reasons, the user should decide what input method (s)he wants to use, and should be able even to change it during the editing session.

In summary, as a developer, there should be a mathematical input field, independent to the input method (keyboard or handwritten); and as a user, the interface should offer a way to manage the input methods. Some vendors already implemented this duality in their systems, like Learnosity¹² and WIRIS.¹³

2 Keyboard Input Interface

A keyboard-based mathematical editor is traditionally composed by a toolbar and an editing area. Of course, this structure can vary between implementations by adding or removing interface elements but these variations are out of the scope of this paper. An example of this structure is shown in Figure 1.

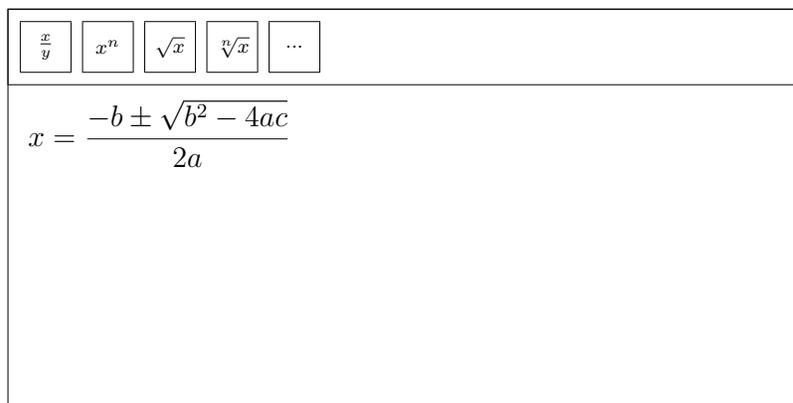


Figure 1: Basic structure of a keyboard-based mathematical editor

⁵<https://www.mediawiki.org/wiki/Extension:WikiEditor>

⁶<http://dragmath.bham.ac.uk/demo.html>

⁷<http://mathquill.com/>

⁸<http://www.codecogs.com/latex/about.php>

⁹<http://www.wiris.com/editor/demo/en/>

¹⁰<http://webdemo.myscript.com/#/demo/equation>

¹¹<http://www.wiris.com/hand>

¹²<https://www.learnosity.com/>

¹³<http://www.wiris.com/editor/demo/en/>

The toolbar of a keyboard-based mathematical editor is used for changing content formatting, switching element properties, managing clipboard (copy, cut, paste), performing undo/redo actions and inserting mathematical structures that cannot be typed using a keyboard (like fractions or square roots), among other features.

A keyboard-based mathematical editor also offers advanced features that are difficult to implement in a handwriting recognition system, such as support for different character sets (chinese, japanese, korean, arabic), Right To Left (RTL) support, syntax checking, copy/cut/paste, formatting or pixel-perfect element position. In summary, a keyboard-based mathematical editor is a powerful tool that can be used not just to let the user insert expressions, but also to decide exactly how they should look.

3 Handwritten Input Interface

It is difficult to determine the interface structure of a handwritten mathematical expression recognizer, since the technology is still new and there are different implementations in the market without any established design pattern.

The common characteristic in most of them is that there are two main components: the drawing area and the action buttons, that can be displayed inside a tiny toolbar (Figure 2) or floating over the drawing area (Figure 3).

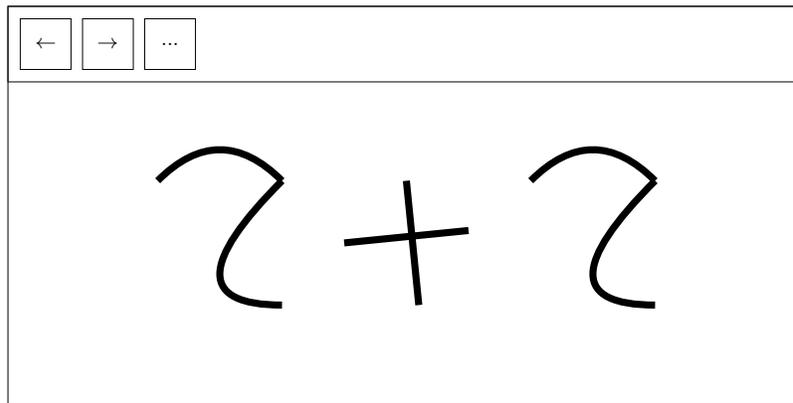


Figure 2: Basic structure of a handwritten mathematical expression recognizer with toolbar

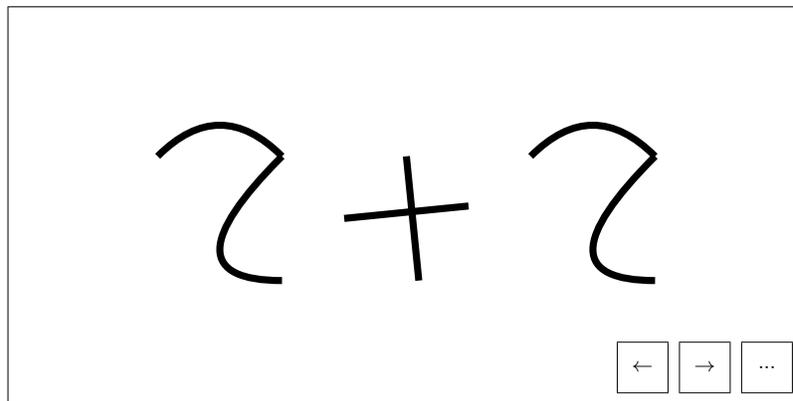


Figure 3: Basic structure of a handwritten mathematical expression recognizer with floating buttons

Action buttons can be used to perform simple tasks, as undo/redo actions or clearing the drawing area.

The drawing area allows the user to write mathematical expressions using a touch screen or a mouse, but it can be used also to enter gestures. A gesture is a special input that the system interprets as a command. For example, drawing several strokes over an existing drawn character can be interpreted as a desire to delete the

character (Figure 4); or touching with two fingers at the same time and changing the distance between them can be interpreted as a desire to increase/decrease the zoom of the drawing area.

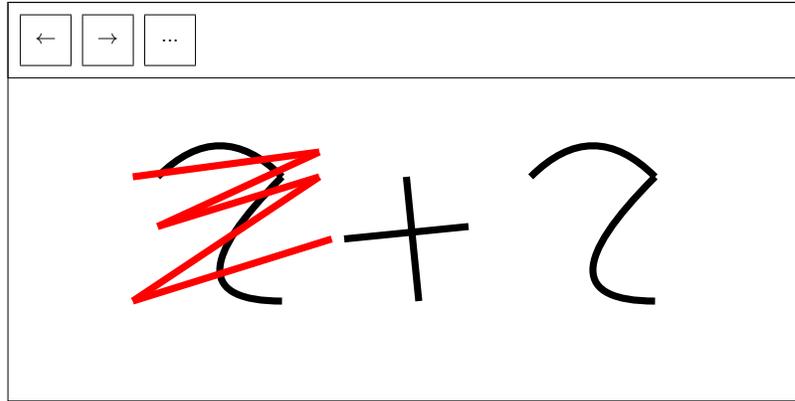


Figure 4: Deletion gesture

This kind of interface is simpler than the interface of a keyboard-based mathematical editor. It can be useful in environments where the semantic of the entered value is more important than the formatting.

4 Coexistence of Handwritten and Keyboard Input

As said previously, both interfaces have different and complementary characteristics. In some scenarios, just one of them is enough to fulfill the given specific requirements, but in other cases the final decision is up to the user.

In this paper, we propose the coexistence between both interfaces. On the startup, the system detects and uses the appropriate interface depending on the device, user preferences or developers decision, but the user can interchange the interface at any moment, even during the editing session.

4.1 Interchange of Interface

There are several ways to interchange between a keyboard-based mathematical editor interface and a handwritten mathematical expression recognizer interface. In this paper, we suggest the usage of an element that indicates clearly the concept of duality, avoiding approaches based on gestures like swiping without any previous visual clue. The described proposal can be implemented via toolbar (Figure 5) or float buttons.

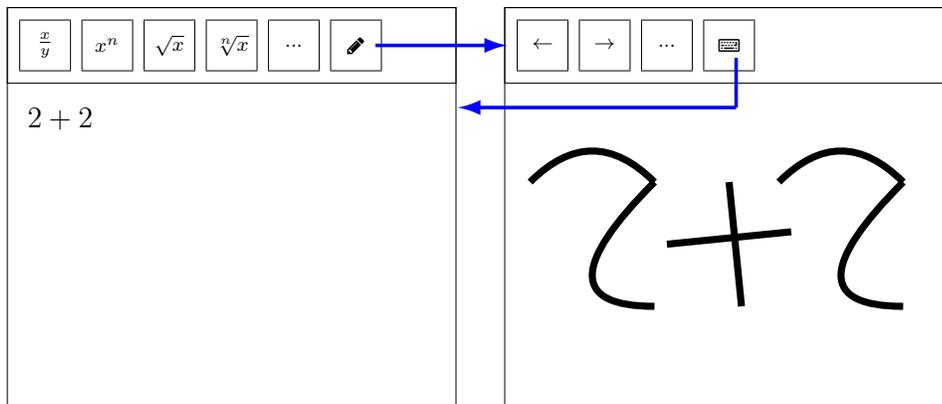


Figure 5: Interchange between a keyboard input interface and a handwritten input interface via toolbar buttons

In this proposal, we also recommend the conservation of the existing content (if there is any) as long as possible, converting it from one input method to another, as explained below.

4.2 Conversion of Formal Content to Strokes and Vice Versa

Given a handwritten math input, once recognized, its conversion to a formal content for the keyboard-based mathematical editor is trivial: the formal result of the recognizer is just passed to the editor, assuming that all the pieces use the same format.

The conversion of content typed with a keyboard-based editor to a set of strokes understandable by the handwritten expression recognizer is not immediate, because it is necessary to create a correct 2D structure containing handwritten symbols. The technical details are out of the subject of this paper, but one possible solution can consist on reusing the existing painting technology of the WYSIWYG editor, sending the paint instructions directly to the drawing area of the recognizer.

For the conversion of text characters to strokes, an existent font made with drawing instructions can be used, and the system just have to execute the corresponding paint instructions when a character must be drawn. A real implementation of this proposal has been done in the WIRIS editor. Figure 6 shows the automatically generated handwritten expression.

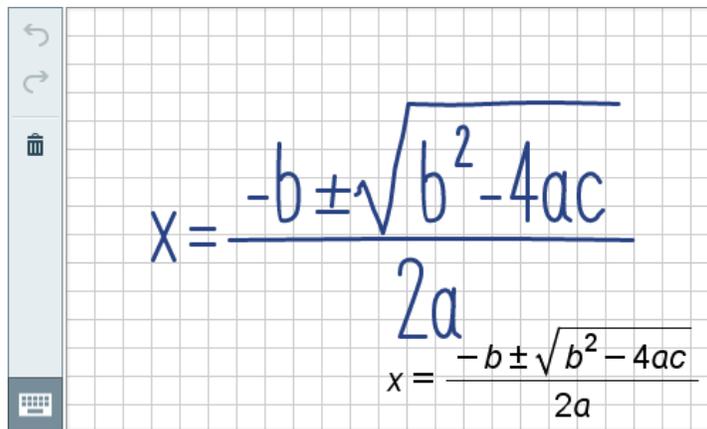
The image shows a screenshot of the WIRIS editor interface. On the left, there is a vertical toolbar with icons for undo, redo, delete, and a keyboard icon. The main area is a grid where the quadratic formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ is written in blue ink. The formula is written in a handwritten style, with the denominator $2a$ positioned below the fraction line. Below the main formula, there is a smaller version of the same formula, also in blue ink, which appears to be a zoomed-in or alternative view of the same expression.

Figure 6: Automatically generated handwritten expression in WIRIS editor

4.3 Developer Tools

In this section, we present some guidelines for API development where keyboard and handwriting inputs coexist.

4.3.1 Providing Clues and Hints About the Expected Value

In some platforms, like in assessment environments, the mathematical input field is used to let students provide an answer to a problem or question.

Since recognition of handwritten math expressions is not 100% accurate and due to handwriting ambiguities, it is recommended to pass hints and clues to the recognizer in order to increase the success rate.

For example, if the platform knows that the student has to give an answer that is composed just by numbers, notifying it to the recognizer can increase the success rate since the recognizer will not classify confusing drawings of “2” like “z”.

Technical details about the format of those clues and hints go beyond the purpose of this paper.

4.3.2 Obtaining Information About the Input Mode

In some cases it is interesting for the platform to store the drawing of the user. For example, in assessment environments, when the student answers using the handwriting recognizer and the system is unable to recognize the expression with success, a teacher can inspect manually the strokes drawn by the student and determine if the answer is correct or not.

In some formats like MathML it is possible to insert semantic content next to the formal representation of the expression (in this case, using a semantics tag¹⁴). In other formats like L^AT_EX, this semantic content can be

¹⁴<https://www.w3.org/TR/REC-MathML/>

inserted just as a \LaTeX comment, and the real code that represents the expression is not affected by it.

This paper proposes the insertion of the user drawing as a semantic content inside the returned value when the developer explicitly retrieves it in this way from the input field.

In the future, if new input methods are implemented (like voice recognition), other semantic content can be returned next to the real representation of the expression using this method.

4.3.3 Interface Events

An API for managing interface events increases extensibility and adaptation to customer requisites, and allows the developer to have full control on the input field behaviour. This paper proposes the following events, not entering in details about the technical implementation. In general:

- An event that is fired when the value of the field changes.
- An event that is fired when the interface input mode changes.

For the keyboard-based mathematical editor:

- An event that is fired when the selection or caret position changes.
- An event that is fired when the formatting of the expression or the selection changes.

For the handwritten mathematical expression recognizer:

- An event that is fired when the drawing area changes.
- An event that is fired when the system recognizes an expression.
- An event that is fired when the system cannot recognize an expression.

With these events a developer can, for example, prevent the submission of a form if the entered strokes have not been still recognized (or there has been an error in the recognition), in order to prevent wrong submissions.

4.4 Default Interface Mode on Startup

On the startup the system has to decide the appropriate interface to be displayed for entering mathematical expressions. This decision is based on several factors.

The developer has all the power to decide what interface must be used, since the developer knows the context of the page and, for example, if the entered value is going to be evaluated later or is going to be inserted in a page with a specific formatting. The developer knows also the main user target (primary education, secondary education, university students, etc.), and can decide that a complex expression should be entered just with the keyboard-based editor (like, for example, a Taylor series development).

If the developer does not take a decision over the default interface, there are several scenarios. If there are stored user preferences from previous sessions, the system can load them and display the default interface based on the current configuration.

If a user configuration is not defined, this proposal suggests that the system should determine, as far as possible, if the platform expects a given value and the user can use the handwriting recognizer to enter it.

In other words: if the handwriting recognizer is not able to recognize the expected value, the system must display the keyboard-based editor interface by default (and in some cases, it is recommended to disable completely the handwritten input). Of course, in this scenario the developer needs to provide a clue about the expected value, as seen previously.

Otherwise, the system can detect characteristics from the device. For example, if the user is using a device with a touch screen or a digital pen, the handwriting recognizer can be used as default interface. The system can detect other properties, like if the user is using a screen reader due to a visual impairment and, in that case, enable the keyboard-based editor by default.

5 Accessibility

Web accessibility refers to the inclusive practice of removing barriers that prevent interaction with, or access to websites, by people with disabilities. When sites are correctly designed, developed and edited, all users have equal access to information and functionality.

The needs that Web accessibility aims to address include:

Visual Visual impairments including blindness, various common types of low vision and poor eyesight, various types of color blindness.

Motor/mobility e.g. difficulty or inability to use the hands, including tremors, muscle slowness, loss of fine muscle control, etc., due to conditions such as Parkinson’s Disease, muscular dystrophy, cerebral palsy, stroke.

Auditory Deafness or hearing impairments, including individuals who are hard of hearing.

Seizures Photo epileptic seizures caused by visual strobe or flashing effects.

Cognitive/Intellectual Developmental disabilities, learning disabilities (dyslexia, dyscalculia, etc.), and cognitive disabilities of various origins, affecting memory, attention, developmental “maturity”, problem-solving and logic skills, etc.

By the nature of a mathematical input field and the subject of this paper, only visual and motor/mobility disabilities are taken in account in this proposal.

In the case of motor/mobility disabilities, the major part of the work is already performed by the web browser, the operating system and third-party native tools, performing the required zoom on the page and enabling special features on mouse and keyboard (like enabling special keys or modifying the behaviour of classical mouse drag&drop). In this paper we assume that these features are enough to offer a good experience to motor/mobility disabled users and It does not propose any extra work. But, of course, implementations are free to include new features that help on the removal of this barrier.

In the case of visual disabilities, due to the nature of a handwriting recognizer, this proposal recommends the exclusive usage of a keyboard-based mathematical editor, which can include more advanced features for visual impaired users than a handwriting recognizer, like keyboard navigation and support for screen readers.

5.1 Keyboard Navigation

On the keyboard-based editor, the system must implement a way of accessing all elements (toolbar, buttons, editing area and others) using just the keyboard. This feature can be already implemented using the focus property defined by W3C.¹⁵

When a user activates the button that changes the interface to the handwriting recognizer, we propose to make focusable just the button to go back to the keyboard-based editor. By this, a user that suffers from visual impairment or a motor/mobility disability can go easily back to the classical editor that offers full accessible support.

5.2 Screen Reader Support

Modern accessible web technologies offer advanced support for screen readers, allowing the developer decide what the screen reader should synthesize. On the keyboard-based mathematical editor, this feature can be used for two purposes:

- Providing description of interface elements when they are focused, like tabs, action buttons and editing area.
- Providing a description of the context when the caret is moved, like in a classic input field. When the caret is moved in a classic input field, the screen reader usually spells the name of the character at the caret position (or other details, like if the caret is at the end of a line). This feature can be also implemented in a keyboard-based mathematical editor using the standard aria-live region,¹⁶ providing a brief description of the context of the caret (beginning of a square root, denominator of a fraction, etc).

¹⁵<https://www.w3.org/TR/html5/editing.html#focus>

¹⁶https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/ARIA_Live_Regions

6 Examples and First Integrations

The model of interface proposed in this paper has been already implemented in WIRIS quizzes. WIRIS quizzes (Figure 7) is a software component that allows teachers to create questions for students, containing random variables, providing automatic evaluation of answers and using WIRIS editor as mathematical input field for students.

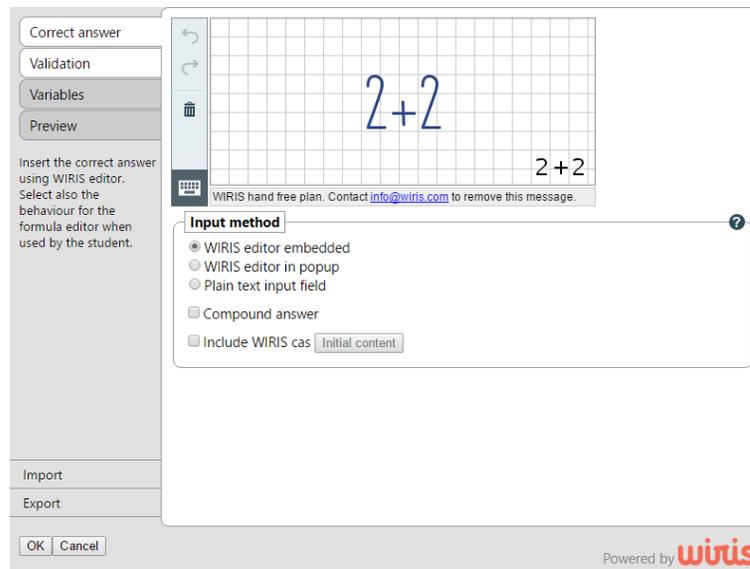


Figure 7: Question creation in WIRIS quizzes

When a teacher creates a question and provides the structure of a correct answer (e.g. specifying what variables, numbers or symbols are involved in it), WIRIS quizzes analyzes it and passes a set of constraints to WIRIS editor to increase the success rate of the handwriting recognizer. For example, if the correct answer contains only numbers and does not contain any “z”, the ambiguous strokes that might either be a number “2” or a letter “z” will be recognized as numbers. Figure 8 shows an example of failed recognition due to unknown information about the context, while Figure 9 shows how, with information, the recognizer has a higher success rate.

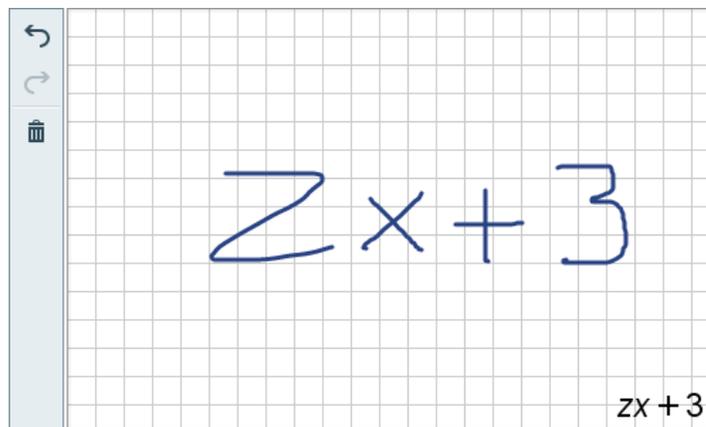


Figure 8: Failed recognition of number “2” without context information about the expected input

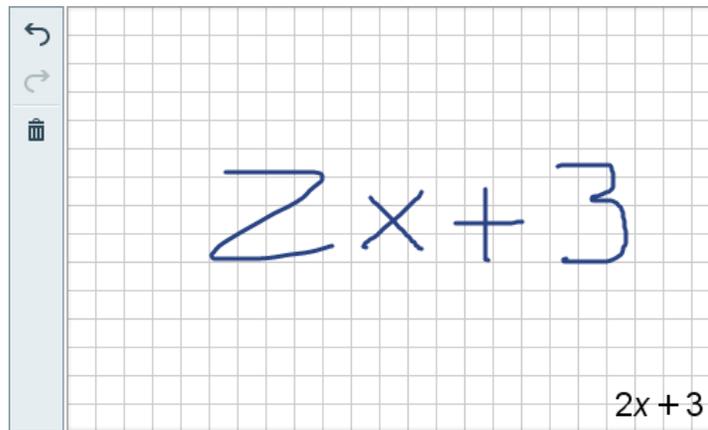


Figure 9: Once the system has information about the context, it recognizes correctly the character “2”

7 Discussion and Future Work

7.1 Technical Details and Standardization

This paper is a proposal for coexistence of mathematical handwritten and keyboard input in a WYSIWYG expression editor, and is not an API proposal or a definition of a standard.

From the point of view of the authors of this paper, the following subjects are still pending of discussion and standardization:

- Inclusion of a math type on standard HTML input field types.
- Definition of an API for the control of “math” input fields, including configuration parameters and event listeners.
- Election of a standard mathematical format for “math” input field values, like MathML.

7.2 Other Input Methods

Advancements in artificial intelligence and machine learning are promoting the development of new technologies that can analyze and classify information impossible to handle until now. Specially, advancements in ASR (Automatic Speech Recognition) introduce speech as a new input method of mathematical expressions [WHP⁺09]. There are even some studies that combine both handwriting and speech input to enter a mathematical expression [MMPVG13].

The interface proposal of this paper can be extended to these new input methods, adding a new button for interchanging the interface to the new input modes, extending the functionality of the existing interchanger button or adding a new button that just opens a modal dialog for speech input, like traditional text input already do (e.g. Google, Apple, Microsoft).

7.3 Drawing over the Current Formal Expression

Another possible approach for the problem handled in this paper can consist in allowing the user to draw new subexpressions over an existing formal expression in the mathematical editor. But this approach presents some problems:

- Using the mouse or the touch screen to handle the caret position and selection can conflict with the feature of drawing or performing gestures.
- Sometimes there is not enough space to draw a subexpression. For example, in this case it would be difficult to draw a new fraction inside the square root:

$$\frac{\sqrt{x}}{2}$$

- Recognizing incomplete subexpressions has lower success rate than recognizing complete expressions.

8 Conclusions

In this paper we proposed a design of interface and behaviour for the coexistence of a keyboard-based mathematical editor and a handwritten mathematical expression recognizer. Advantages of both modes have been analyzed and it has been shown that they are complementary, so a design based on coexistence is more appropriate for the final user than single modes.

In this proposal we also present some guidelines about the communication channel between developers and mathematical input fields that fulfills the needs a platform can have on a real scenario.

Regarding accessibility support, we proposed some hints and guidelines on an implementation that offers a good experience to users with disabilities.

Finally, future work should be focused on the definition of a standard API and format for real case scenarios and exposed.

References

- [ML15] Scott MacLean and George Labahn. A bayesian model for recognizing handwritten mathematical expressions. *Pattern Recognition*, 48(8):2433–2445, 2015.
- [MMPVG13] Sofiane Medjkoune, Harold Mouchere, Simon Petitrenaud, and Christian Viard-Gaudin. Multimodal mathematical expressions recognition: Case of speech and handwriting. In *Human-computer interaction. interaction modalities and techniques*, pages 77–86. Springer Berlin Heidelberg, 2013.
- [SHP⁺12] Christopher Sasarak, Kevin Hart, Richard Pospesel, David Stalnaker, Lei Hu, Robert Livolsi, Siyu Zhu, and Richard Zanibbi. min: A multimodal web interface for math search. *Symp. Human-Computer Interaction and Information Retrieval, Cambridge, MA*, 2012.
- [WHP⁺09] Angela Wigmore, Gordon Hunter, Eckhard Pflugel, James Denholm-Price, and Vincent Binelli. Using automatic speech recognition to dictate mathematical expressions. *Journal of Computers in Mathematics and Science Teaching*, 28(2):177–189, 2009.

KAT: Enabling the Semantification of STEM Documents

Felix Schmoll
f.schmoll@jacobs-university.de

Tom Wiesing
t.wiesing@jacobs-university.de

Jacobs University Bremen

Abstract

Considering a constantly growing body of mathematical knowledge it becomes more and more difficult for individuals to take full advantage of all information available. To semantify documents, we need to be able to create annotations efficiently and conveniently – marking definitions or declarations as well as usages of concepts – on a large corpus of documents. Eventually this can be achieved automatically, but as a first step a gold standard has to be created by humans. KAT – the KWARC Annotation Tool – has the goal of allowing users to create, view and update annotations on arbitrary (X)HTML documents. We have presented our approach before and in this paper we want to give an update on our progress.

1 Introduction

STEM – Science, Technology, Engineering and Mathematics – documents often not only introduce the user to new areas but build heavily on previous knowledge. We want to make STEM documents, in particular mathematical documents, more accessible and lower the burden of gaining access to a new topic. Users should be able to intuitively navigate through existing knowledge as to make the process of understanding more efficient.

In the context of mathematical documents this requires annotating documents and marking up definitions, declarations and other linguistic phenomena. Furthermore we want to mark up all usages of these concepts within the document to allow readers to, for example, click on a concept and navigate to its definition. In order to bring us closer to this goal it is necessary to do this on a large scale.

As it is non-trivial to annotate a huge corpus of documents one wants eventually to do this automatically, with only little human interaction. During the development of tools to achieve this it is however common to annotate a small subset of documents manually, creating a “gold standard”, that can then be used as a basis for further the development – either using automated machine learning approaches or smart rule-based software. Annotation tools to create, edit and view annotations are here necessary and can further be helpful during later phases of development – in order to evaluate performance¹. Additionally they can they can be used by mathematical readers to interactively navigate through annotated content.

STEM documents usually consist of a multitude of content, ranging from pure text over mathematical/chemical formulae to tables and diagrams. Most common annotation tools, such as `brat` [BR], `Yawas` [YW], and `Annotatie` [AN], only work properly with textual content – they store the position of annotations as offsets in a character string. While some of them can work with more advanced content, they do so by treating it as a blackbox and replacing it with a placeholder. Such a treatment however prevents for example the annotation of sub-formulae as any embedded formulae are considered as a single object.

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

¹Indeed our tool has been used during the development of a declaration spotter in order to evaluate performance, we refer interested readers to [Sch16].

We have developed KAT – the KWARC Annotation Tool – to address and solve these problems. KAT is a browser-based tool that will allow users to annotate arbitrary (X)HTML documents. In this paper we want to present the newest iteration of it, building on the previous versions presented in [DGKMMW14] and [GLKW15]. We proceed in Section 2 by giving a short review of the existing KAT system architecture. In Section 3 we then proceed by giving a more detailed look into how annotations can be created in the browser and in particular focus on the progress we have achieved since the last iteration. We then conclude in Section 4 with an outlook on how we plan to develop the system further.

2 Recap of KAT system architecture

KAT is implemented as a JavaScript library working with (X)HTML5 documents and can be integrated into arbitrary websites with little effort. The basic system architecture can be seen in Figure 1. This architecture consists of four main components: *i*) the KAT Annotator, the KAT tool running in the browser; *ii*) the KAnnSpec, which serves as a description of *iii*) the Annotation ontology and *iv*) a document management system (here called CorTeX).

Instead of describing this architecture in detail, we refer interested readers to [DGKMMW14] and [GLKW15]. KAT is capable of working as a standalone tool, however it is best used in the context of a corpus management system such as CorTeX

system [CT]. This scenario is intended for creating and improving a “gold standard”. Users receive a document from the document store in CorTeX and create annotations for it using KAT. These annotations are then sent back to the system in RDF form. Alternatively the users receive an already annotated document and check existing annotations. These two scenarios drive the KAT development.

KAT annotations are represented as RDF subject / predicate / object triples. The subjects are the text fragments that we are annotating whereas the objects are either concepts from the annotation ontology (in the case of classificational annotations) or other text fragments (in the case of relational annotations). KAT is not tied to a particular annotation ontology. On the contrary, at startup time it loads a set of annotation descriptions referred to as KAT Annotation Specification, or KAnnSpec for short. These are a set of custom XML documents that describe the annotation ontology, its concepts and additional constraints for the KAT user interface.

As KAT is XHTML based we reference to text fragments using URIs. For this we make use of the XPointer framework [GMMW03] and developed a custom XPointer scheme. Each text fragment is a contiguous range of elements in the DOM tree and can be identified by giving the first and last elements contained in it. This obviously imposes the limitation that text fragments must consist of entire elements, making it very difficult to annotate linguistically meaningful concepts. In previous iterations of the system we worked around this by TEI-tokenizing document, that is adding elements around each word in the document, however we are in progress of implementing an updated XPointer scheme that allows us to reference text ranges within elements and thereby eliminating the limitation entirely.

3 Editing Annotations In The Browser

KAT offers three different operating modes that can be navigated via a sidebar. Each of them facilitates a different kind of working with annotations:

1. *Annotation Mode*. This mode provides a user interaction to create and edit annotations on documents. This is semantically separate from the viewing and evaluating of annotations by a different mode, allowing specialized operations e.g. on right-clicking.
2. *Reading Mode*. Once annotations have been created they can be used to obtain a better understanding of the structure of a mathematical document as a whole. This mode attempts to provide as much information as possible about all given annotations in an intuitive way.

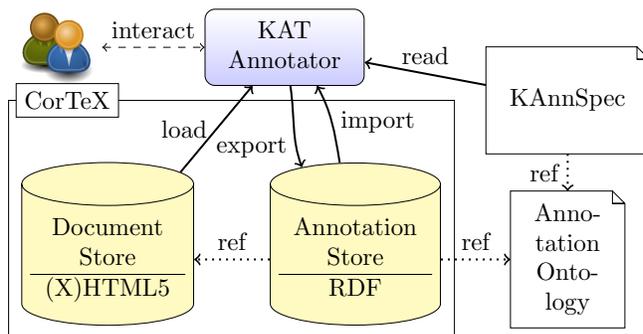


Figure 1: The KAT System Architecture

3. *Review Mode.* When automating the creation of annotations, one wants to go over the created annotations and evaluate them. This mode allows an evaluation of annotations by providing an interface for judging them as good (thumbs up) or bad (thumbs down).

3.1 Creating Annotations

As KAT is a browser-based tool, the process of creating annotations is a heavily form-oriented process. A range of text is selected and then an annotation category is chosen from a right-click context menu. Subsequently a form appears in the sidebar where more specific information can be entered. An example of this can be seen in Figure 2.

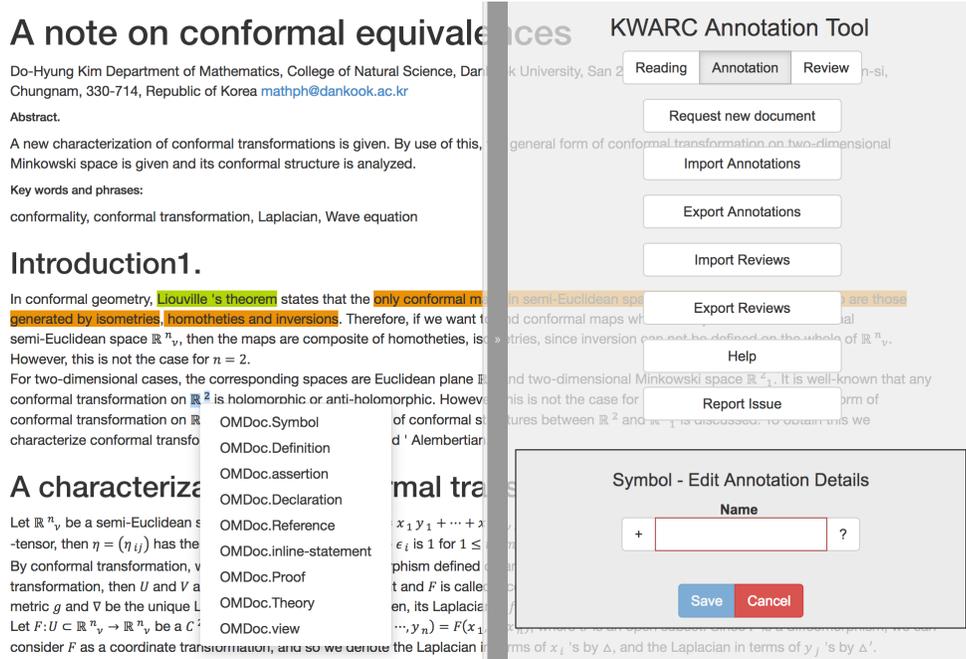


Figure 2: Creating annotations in KAT.

It is possible to make restrictions on input in the KAnnspec, such as the format of a name by providing a Regular Expression to be fulfilled. To ensure that entered content is valid it is only possible to save an annotation once all conditions are fulfilled. To visually indicate this the color of a text field changes from red to green, and only once the validation constraints are fulfilled the save button becomes active. Additionally the question mark right next to the input-field can be used to obtain more information about said constraints (in this particular case the Regular Expression restricting the input format would be shown).

3.2 Visualizing Annotations And References

In Reading mode all annotations are displayed to the user. The respective information are conveyed via multiple means:

- *Highlighting.* Each concept from the KAnnspec is assigned its own color and each annotation is highlighted in the appropriate color.
- *Tooltip.* When hovering an annotation, information about its fields is displayed. The tooltip is generated using the `<template/>` tag in the KAnnspec.
- *Relations.* Some annotations may have referential fields. These are visualised on demand.

Upon creation of an annotation it is assigned a UUID (universal unique identifier). This is later on the only way to refer to a specific annotation. As this is not a very intuitive way to reference an annotation there is a need for visual cues in displaying relations.

Key words and phrases:

conformality, conformal transformation, Laplacian, Wave equation

Introduction¹.

In conformal geometry, **Liouville's theorem** states that the **only conformal maps in semi-Euclidean space of dimension $n > 2$ are homotheties and inversions**. Therefore, if we want to find conformal maps which are bijective on n -dimensional homotheties, isometries, since inversion can not be defined on the whole of \mathbb{R}^n . However, this is not the case For two-dimensional cases, the corresponding spaces are Euclidean plane \mathbb{R}^2 and two-dimensional Minkowski

Figure 3: Paths can be used to visualize references.

To achieve this, KAT provides the ability to show the relations using paths between annotations that appear by clicking on them. The paths are labeled with the type of reference between two given annotations. The direction of a path becomes clear due to the user interaction that causes them to be displayed.

Internally this is implemented using an SVG-overlay over the document. Paths are then drawn from the selected annotation to all other annotations that are referenced in respective fields of the KAnnSpec and decorated with captions describing the kind of relation. As start and end points of the paths the upper left corners of their selections are used.

3.3 Import And Export Of Annotations

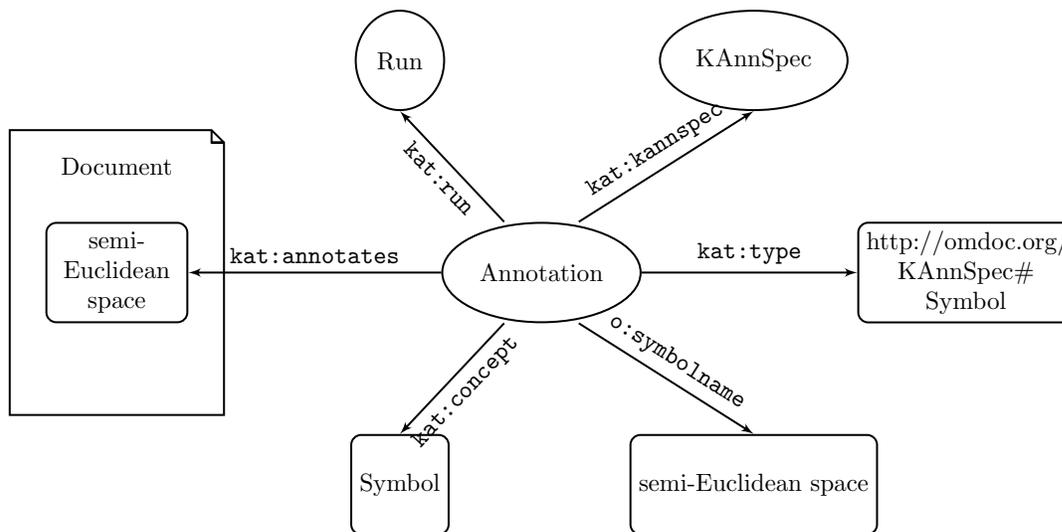


Figure 4: An Annotation Graph

By itself KAT only saves the state of annotations for the length of a given session. To make annotations persistent, it is possible to export and later re-import annotations via buttons in the side pane, which is implemented based on the Resource Description Framework [SR14]. This approach has been successfully tested in the creation of a gold standard of declarations in math [Sch16].

The current implementation also features a prototype for retrieving new documents from a document storage and submitting annotations to it. If one would fully integrate the system with CorTeX it would then no longer be necessary to store annotations manually but could store it centralised together with the document.

In Listing 1 we show a sample of how an annotation is exported to RDF. Each annotation consists of a single node (lines 6-12). Additionally, each annotation has meta-data, such as the used KAnnSpec, which is omitted here. To generate this node, we use the properties of the annotation graph shown above in Figure 4.

Listing 1: Exported RDF generated for a single annotation of an OMDOC Symbol

```
1 <rdf:RDF xmlns:o="http://omdoc.org/KAnnSpec#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:kat="https://github.com/KWARC/KAT/">
```

```

2 <!-- omitted a lot of meta-information here -->
3
4 <rdf:Description rdf:nodeID="KAT_1433087821332_4477">
5   <kat:annotates rdf:resource="https://kwarc.github.io/KAT/content/sample1.html#cse(%2F%2F
      *%5B%40id%3D'sentence.11'%5D%2C%2F%2F*%5B%40id%3D'word.202'%5D%2C%2F%2F*%5B%40id%3D'
      word.203'%5D)" />
6   <kat:run rdf:nodeID="kat_run"/>
7   <kat:kannspec rdf:nodeID="KAT_1433087757661_OMDoc"/>
8   <kat:concept>Symbol</kat:concept>
9   <kat:type rdf:resource="http://omdoc.org/KAnnSpec#Symbol" />
10  <o:symbolname>semi-Euclidean space</o:symbolname>
11 </rdf:Description>
12
13 </rdf:RDF>

```

The export starts by declaring the text fragment it annotates using the *kat:annotates* relation (line 5). For this we use the KAT XPointer scheme – in this case given by the URI `https://kwarc.github.io/KAT/content/sample1.html#cse(//*[@id='sentence.11'],//*[@id='word.202'],//*[@id='word.203'])`. In this URI *cse* stands for *C*ontainer, *S*tart and *E*nd. The URI itself consists of

- the document URI `https://kwarc.github.io/KAT/content/sample1.html`, the document in which the annotated text fragment is located;
- an XPath to the deepest element that fully contains the annotated text fragment, here `//*[@id='sentence.11']`;
- an XPath that points to the start of the annotated text fragment – the first element that is contained in it – here `//*[@id='word.202']` and
- an XPath that points to the end of the annotated fragment – the last element that is contained in it – here `//*[@id='word.203']`.

Next, a *kat:run* is passed (line 6). This is intended to provide meta-information such as when and how this annotation was generated, which has been omitted from the listing. Continuing, it references a KAnnSpec (line 7) and then the actual concept it annotates (line 8). We further specify the type of the annotation with the *kat:type* relation (line 9) as given in the KAnnSpec. Finally, we provide all the fields and their values. In this case, we just give the concept the name *semi-Euclidean space* (line 10).

3.4 Evaluating Annotations In The Review Mode

After annotations have been created it is possible to evaluate them using a special review mode. Here automatically created annotations can be assessed by a human operator. An example of the review mode can be found in Figure 5.

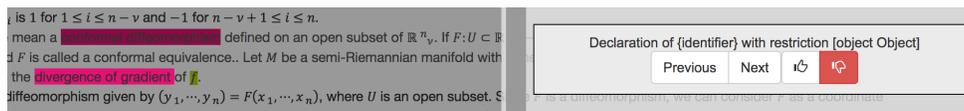


Figure 5: Evaluating annotations using the review mode.

At any given time exactly one annotation is in focus – the one that is supposed to be assessed. This annotation is visually contrasted by a dark overlay over the rest of the document. An information field in the side pane displays all available information for the current annotation (which would appear in a tooltip when hovering over an annotation in reading mode) and facilitates endorsing or flagging of a given annotation, such that the automatic annotation generation can be improved based on the given feedback. The user can then iterate through the annotations in the order of their appearance in the document.

Using this field it is possible to evaluate the accuracy of automated annotation-systems or to use ensure the quality of existing annotations. One possible use case is as a stage in the development cycle in order to obtain relevant feedback, as was done in the development of the declaration spotter in [Sch16].

The reviews of annotations made in review mode can likewise be exported, whereas the format is essentially a list of (UUID, Boolean)-pairs indicating which annotation was reviewed and what the binary review choice was.

4 Conclusion And Future Work

We have presented the KAT system, an open and browser-based annotation system for STEM documents encoded in XHTML5. In particular we have presented the new and improved visualisation of referential annotations and the new review mode. The code base is released freely under the terms of the GNU Public License and available at [KG]. A running demo is further available at [KD].

While the basic utility of annotating documents is functional, there are some aspects that we want to improve upon in order to allow a smoother user experience. The most important ones are

1. *Allowing annotated text fragments within elements.* At the moment it is only possible to annotate text fragments on a DOM Element basis. Thus KAT works best with documents are TEI-tokenized into words in order to process them. We are working on an XPointer-implementation that would allow any XHTML5 document to be immediately marked up. This would furthermore allow a better integration with CorTeX [CG].
2. *Distinction between overlapping annotations.* Currently different annotation categories are visually set apart by color. This is however not sufficient once there are overlapping annotations. In future versions the ranges of overlapping annotations should be discernable by giving each of them a different height.
3. *Adding feedback and other improvements to the review mode.* It should be possible to provide more specific feedback in review mode, as it is currently only possible to make a binary choice. One way would be to provide additionally an input field in order to give an explanation, another one to pick a specific reason why an annotation is inappropriate from a drop-down menu. Furthermore it is an open question how to properly export the feedback given in review mode.
4. *Using arrows for visualising paths.* One might want to extend the visualisation of relations as to use actual arrows indicating a direction. The current paths are suitable while using the system, however insufficient when looking at a static image of the representation.
5. *Allowing changing document content.* Published papers are rarely changed, but one might nevertheless want to accommodate the ability to handle a changing document structure for example when annotating just sections of a document and merging them later on. The current implementation does not allow this.

We will also try to get more immediate user feedback by providing a version of the system to a larger audience.

4.1 Acknowledgements

We thank Frederik Schaefer for his feedback on further improvements of KAT especially in terms of usability and Deyan Ginev for his input on how to avoid the need to tokenize documents. Additionally we would like to thank Michael Kohlhase for his supervision.

References

- [AN] Annotation tool. URL: [Http://www.annotatiesysteem.nl](http://www.annotatiesysteem.nl) (visited on 02/15/2014).
- [BR] Brat rapid annotation tool. URL: <http://brat.nlplab.org> (visited on 02/15/2014).
- [CG] GitHub repository. URL: <https://github.com/dginev/CorTeX/>.
- [CT] CorTeX framework. URL: <http://cortex.mathweb.org> (visited on 02/14/2014).
- [DGKMMW14] Mircea Alex Dumitru, Deyan Ginev, Michael Kohlhase, Vlad Merticariu, Stefan Mirea, and Tom Wiesing. System description: KAT an annotation tool for STEM documents. 2014. URL: <http://kwarc.info/kohlhase/submit/cicm14-kat.pdf>.
- [GLKW15] Deyan Ginev, Sourabh Lal, Michael Kohlhase, and Tom Wiesing. KAT: an annotation tool for STEM documents. In *Mathematical user interfaces workshop at CICM*. Andrea Kohlhase and Paul Libbrecht, editors, July 2015. URL: http://www.ceremat.org/events/MathUI/15/proceedings/Lal-Kohlhase-Ginev_KAT_annotations_MathUI_15.pdf.

- [GMMW03] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. W3c xpointer framework. W3C Recommendation. World Wide Web Consortium (W3C), March 25, 2003. URL: <http://www.w3.org/TR/2003/REC-xptr-framework-20030325/>.
- [KD] URL: <http://kwarc.github.io/KAT/> (visited on 07/18/2016).
- [KG] GitHub repository. URL: <https://github.com/KWARC/KAT/>.
- [Sch16] Jan Frederik Schaefer. Declaration spotting in mathematical documents. B.Sc. Thesis. Jacobs University Bremen, 2016.
- [SR14] Guus Schreiber and Yves Raimond. RDF 1.1 primer. W3C Working Group Note. World Wide Web Consortium (W3C), 2014. URL: <http://www.w3.org/TR/rdf-primer>.
- [YW] Yawas - the original web highlighter. URL: <http://www.keeness.net/yawas/> (visited on 02/15/2014).

Notation-based Semantification

Ion Toloaca

Michael Kohlhase

Jacobs University Bremen

1 Introduction

The scientific community produces a large number of mathematical papers (approximately 108.000 new papers per year [arX]), which raises the importance of machine based processing of such documents. Unfortunately, the most popular formats in which these papers are found (for instance, \LaTeX) do not contain much information that would allow the computers to infer the human-understandable knowledge contained within a paper. Since, at this point, changing these formats is not practically possible, the other solution is to add a semantic flavor to the existing documents by translating them into a more suitable format, for instance, Content MathML.

The current scientific community publication went through a series of evolutions in search of the best method of writing scientific documents. This process was highly influenced by the invention and spreading of the internet. Scientists understood the necessity of a standard that could help them write and exchange their findings in an efficient way. A lot of effort has gone into translating books into digital documents.

The next step in this evolution is translating digital documents into knowledge-rich digital documents. This next step can only happen if a new feasible way of transition appears, which **MathSemantifier** attempts to become.

Ambiguity is one of the main reasons that makes semantification complex. Mathematical documents are not a simple collection of symbols. The main use of these documents emerges only when the intended semantics of a document is accessible. However, humans tend to be lazy in writing down the whole graph, but instead rely on implicit human knowledge to decipher these documents. This is where ambiguity comes into play, when the author relies on the ability of the human to use the context of document in order to pinpoint the actual meaning an expression. Ambiguities can be largely divided into two: structural and idiomatic ambiguities.

A simple example that demonstrates the concept of structural ambiguities can be “ $\sin x / 2$ ”. It can mean one of the following:

1. \sin applied to $\frac{x}{2}$
2. $\frac{1}{2}$ times \sin applied to x

Copyright © by the paper’s authors. Copying permitted for private and academic purposes.

In: Alexandre Rademaker and Vinay K. Chaudhri (eds.) Proceedings of the 6th Workshop on Formal Ontologies meet Industry, Rio de Janeiro, Brazil, 22-SEP-2014, published at <http://ceur-ws.org>

Contrary to structural ambiguities, idiomatic ambiguities are not due to different parse trees. Given one single parse tree, some formulae allow for multiple readings. A standard example would be B_n . This could be:

1. The sequence of Bernoulli numbers
2. A user defined sequence
3. The vertex of one of a series of geometric objects

Consider $c(a(b))$. The most natural interpretation is c of a of b . However, there arise multiple meanings if we consider that the application could be interpreted as multiplication. The human reader discards such meanings by convention and experience of handling mathematical documents, however, teaching this to a computer is a complex task. The approach that **MathSemantifier** takes is simply extracting all the possible meanings, while trying to apply heuristics to weed out impossible ones.

Possible Applications of Semantification include:

- **MathWebSearch** [HPK14] is a search engine for mathematical formulas. Such search engines could greatly benefit from semantification. The idea is that a search engine is only as good as the database of information is. Semantification allows a new kind of queries could be possible: semantic queries. Rather than searching for strings, or formulas with free form subterms, the user could specify the meaning of the sought expression. This would improve a lot the relevance of the results, since there will be no result that matched just because it was presented in a similar manner.
- Another possibility for using semantification is theorem proving and correctness checking. One possible application would be assisting authors in writing semantic content by providing real-time feedback.
- The possibilities extend even beyond this. Using semantified content, rather than having a database of CMML expressions, it is possible to create a smart knowledge management system that could be used to create expert systems. The user could then ask questions, or create complex queries, to exploit the full power of semantic content.

Semantification will not make all of the above directly possible, however it is a necessary step towards achieving goals similar to the ones described above, that require more knowledge about the used content than just how it is rendered.

1.1 An Introduction to MathSemantifier

In a nutshell the **MathSemantifier** algorithm converts Presentation MathML (PMML) to Content MathML (CMML [ABC⁺10]) it is the converse of the presentation algorithm that converts CMML to PMML [KMR08] using a database of notation renderings. In Figure 1.1 a typical example of what the presentation algorithm produces is displayed. The first child of the **semantics** node contains the PMML that corresponds to the CMML contained in the **annotation-xml** node. In order to produce this output, the algorithm used the notation **natarith addition** (shown in Figure 1.1 as well). OMDoc Notations like **natarith addition** are initially written in sTeX [Koh08], and then converted to OMDoc using L^AT_EX₄XML [Mil].

```

1 <math id="-1759683794" xmlns="http://www.w3.org/1998/Math/MathML">
2   <semantics>
3     <math>
4       <math-selected data-mmt-position="">
5         <mi data-mmt-position="1"></mi>
6         <mo data-mmt-symref="http://mathhub.info/smglob/numberfields
7           /natarith.omdoc?natarith?addition" data-mmt-position="0">+</mo>
8         <mi data-mmt-position="2"></mi>
9       </math>
10    </semantics>
11  </math>

```

```

1 <omdoc:notation cd="natarith" name="addition" stex:macro_name="natplus"
2   stex:nargs="1" xml:id="natarith.notation8" about="#natarith.notation8"
3   stex:srcref="smglob/numberfields/source/natarith.tex#textrange(from=7;0,to=7;76)">
4   <omdoc:prototype>
5     <om:OMA>
6       <om:OMS cd="natarith" cr="fun" name="addition"/>
7       <omdoc:explist name="args">
8         <omdoc:expr name="arg"/>
9       </omdoc:explist>
10    </om:OMA>
11  </omdoc:prototype>
12  <omdoc:rendering precedence="500">
13    <omdoc:iterate name="args">
14      <omdoc:separator>
15        <m:mo cr="fun">+</m:mo>
16      </omdoc:separator>
17      <omdoc:render name="arg"/>
18    </omdoc:iterate>
19  </omdoc:rendering>
20 </omdoc:notation>

```

Figure 1.1: Presentation Algorithm Output and the corresponding Notation Definition

MathSemantifier converts PMML into valid CMML as described above. In order to perform this task, it needs to match PMML against a list of notations. This is achieved by compiling the notations into a context free grammar, and using a CFG parsing engine to parse the PMML. A parse returns a list of possible parse trees, out of which **MathSemantifier** extracts information regarding what notations matched at top level and continues with the arguments of the found notation recursively. We use the **Marpa Grammar Engine** (see [Keg]).

We generate a database of notations, we use the SMGloM corpus [GIJ⁺, Koh], a mathematical thesaurus, which contains over 1000 notation definitions for elementary mathematical concepts. This sTeX-encoded corpus is converted to OMDoc using L^AT_EXML. **MathSemantifier** is an extension of the MMT system [Rab] a scalable mathematical knowledge processing engine that reads the OMDoc notation definition and generates a context free grammar from them.

MathSemantifier takes this generated CFG and input from the **Web UI** in order to produce possible parse trees of the input. The Marpa grammar engine is the proposed tool for this purpose. The main advantages are that Marpa handles ambiguous grammars and provides control over the parsing process. The author of Marpa provides a more detailed analysis of the advantages of Marpa.

2 The MathSemantifier System

The major idea of **MathSemantifier** is, as already described in the introduction, finding possible Content MathML readings for Presentation MathML input expressions.

The general flow of a single semantification can be described as follows:

1. Generation of Context Free Grammar from MMT Notations
2. Parsing using the Marpa Grammar Engine and the generated CFG to detect the top level notation
3. Parsing the arguments of the top level notation recursively
4. Using the parse trees from step 2 and 3 to generate an internal representation of the meaning trees
5. Converting the meaning trees to Content MathML
6. Displaying the Content MathML trees in the frontend

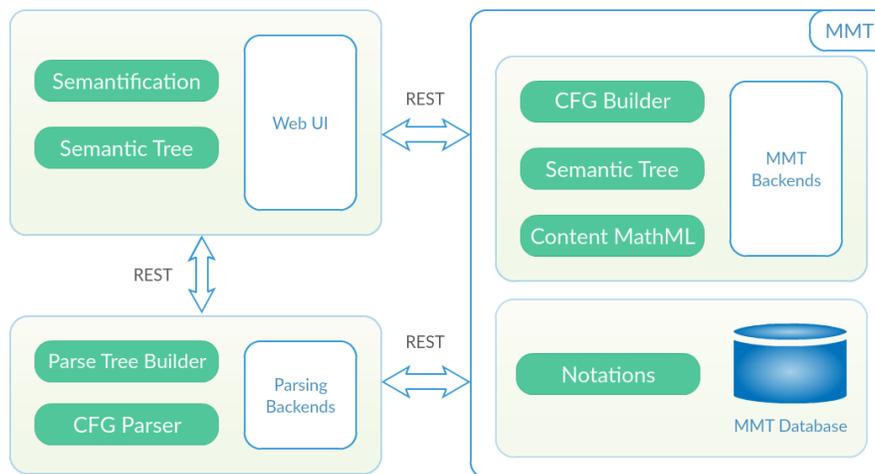


Figure 2.1: **MathSemantifier** Architecture

A CFG-based solution was chosen because of the preexisting parsing frameworks like the Marpa Grammar Engine. Such frameworks solve all the parsing-related technical problems, like parsing ambiguous expressions, different kinds of recursion, while also providing a high degree of freedom.

The main goals of the system can be expressed in a concise manner as follows:

1. Generating the correct set of parses efficiently and effectively
2. Providing opportunities for improvement for further research in the area

The architecture can be roughly divided into four parts as shown in [Figure 2.1](#).

2.1 The Web User Interface

The **Web UI** is a core component of **MathSemantifier**. It is intended to be a lightweight solution that queries a server for the results of more computationally-intensive tasks.

The interface consists of an input area, where **MathML** needs to be inputted, and three options:

1. Semantify (The user can guide the semantification of the top symbol directly, by choosing the correct matching range, notation and argument positions)
2. Show Semantic Tree (The other option is to ask for all the possibilities and get all the semantic trees)
3. Evaluation (The user is walked through a series of examples to demo the functionality)

The repository containing the **Web UI** can be found on GitHub [[Tolb](#)].

User Guided Semantification is performed as follows: The user provides Presentation MathML as input to the system, then uses the **Semantify** button to reveal a list of top level notation names. The names of the notations are derived from the notation paths as follows: **archive name + symbol name**, for example, **natarith addition** refers to the addition of natural numbers. After determining which notation is the correct one, the user needs to make sure that the arguments were detected properly and, finally, the resulting Content MathML tree is displayed (see [Figure 2.2](#) for a similar representation of the result).

The easier but computationally more expensive alternative is to simply generate all the possible parse trees at once and display them. The user simply needs to click the **Show Semantic Tree** option.

For the example shown in [Figure 2.2](#), there is a total of 342 different readings. This can be easily explained, since **Invisible Times**, **Arithmetic Plus** and **Mod** have each multiple notations definitions (that originate from different MMT archives, for instance), and there are no limitations on what kind of notation definitions can go together in the same CMML tree.

In order to minimize the Content MathML, the standard allows subtree sharing. To enable this option, the **Use term sharing** checkbox should be checked. In that case, the terms is shared among different readings.

2.2 The MMT Backend

The MMT Backend is a **Server Extension** that is part of MMT. As shown in [Figure 2.1](#), it is linked via **REST** with the **Web UI** and the **Parsing Backends**.

Its role can be summarized to the following core functions:

1. Compile the **MMT Notations** into a CFG Grammar
2. Receive the input from the **Web UI** and build its Semantic Tree
3. Delegate the parsing to the **Parsing Backends**

We decided to put the core logic of the application in MMT in order to make it easier to interoperate with the MMT Notation Database, as well as with any other MMT components that may need **MathSemantifier**.

The code can be found as part of the MMT codebase [\[KT\]](#).

Let us look at the components of the MMT Backends in more detail below.

2.3 The Grammar Generator

The Grammar Generator aggregates all the knowledge contained in the MMT Notations into one Context Free Grammar. The grammar is shaped into the normal form accepted by the

The screenshot displays the **Math Semantifier** interface. At the top, a list of MathML tokens is shown: `<mn>2</mn>`, `<mo>+</mo>`, `<mo>i</mo>`, `<mo>+</mo>`, `<mi>x</mi>`, `<mo>mod</mo>`, and `<mn>5</mn>`. Below this is a **MathML Preview** section with a text input containing `2 + i + x mod 5` and a checkbox for **Term Sharing**. Three green buttons are visible: **Semantify MathML**, **Show Semantic Tree**, and **Evaluation**. The **Show Semantic Tree** button is active, leading to the **Semantic Tree Result** section. This section shows a tree structure for **Reading 1**, where the root node is **apply**. It branches into three **csymbol** nodes, each with a URL. The first **csymbol** node branches into an **apply** node (which further branches into **cn** 2 and **csymbol** for imaginary unit) and another **csymbol** node. The second **csymbol** node branches into **ci** x and **cn** 5.

Figure 2.2: **MathSemantifier** in Action

Marpa Grammar Engine. To achieve this, the format used to store the notations in MMT is decomposed into CFG rules. Otherwise said, the tree-like structure of each formula, that is stored as nested applications of **MMT Markers** needs to be serialized into CFG rules.

This is done in several steps:

1. Break apart the **MMT Marker** trees into level by level representations
2. Transform the intermediate representation into valid CFG rules

The fundamental structure of the CFG is established using a preamble as shown in [Figure 2.3](#). Note the default action is the **Grammar Entry Point**. It is precisely what tells the Grammar Engine to build parse trees, and also determines their structure.

The entry point into the grammar is the **Expression** rule. It can be an MMT Notation, or Presentation MathML. The **prec0** rule should be read as precedence zero, that is - the lowest precedence there is.

Precedence handling is done using a commonly-used method for including it in CFGs. The **Notation Precedences** section in [Figure 2.3](#) gives a quick glance at how exactly it is done. The number of precedences needs to be known in advance, then, for each precedence value a corresponding **precN** rule is created.

The rule contains all the notations with that precedence, and, one of the alternatives is going to a higher precedence value. In the example below there are 15 used precedence values (**prec0** corresponds to precedence $-\infty$). However, the **Notation Precedences** section shows only half of the concept.

To make this approach work, what is also needed is that the arguments in a rule of a certain precedence N can only contain notations with precedence K if $K > N$. This is done by the **Argument Precedences** part in the preamble (as shown in [Figure 2.3](#)).

Finally, [Figure 2.4](#) presents an example of what the **natharith addition** MMT notation from the **smglom/numberfields** archive translated to CFG rules looks like.

```

1 #GRAMMAR ENTRY POINT
2 :default ::= action => [name, start, length, values]
3 lexeme default = latm => 1
4 :start ::= Expression
5 ExpressionList ::= Expression+
6 Expression ::= Presentation
7 | | | | | Notation
8
9 #Notation Precedences
10 Notation ::= prec0
11 prec0 ::= prec1 | #Rules with precedence zero
12 prec1 ::= prec2 | #Rules with precedence one
13 ...
14
15 #Argument Precedences (depends on the number of precedences)
16 argRuleP0 ::= prec0 | Presentation
17 argRuleP1 ::= prec1 | Presentation
18 argRuleP2 ::= prec2 | Presentation
19 ...
20
21 #Presentation MathML
22 Presentation ::= mrowB Notation mrowE
23 | mrowB ExpressionList mrowE
24 | moB '(' moE ExpressionList moB ')' moE
25 | moB text moE
26 | miB text miE
27 | mnB text mnE
28 ...
29
30 #Presentation MathML Parts
31 mrowB ::= '<mrow' attribs '>'
32 mrowE ::= '</mrow>'
33 mathB ::= '<math' attribs '>'
34 mathE ::= '</math>'
35 miB ::= '<mi' attribs '>'
36 miE ::= '</mi>'
37 ...
38
39 #Lexemes
40 ws ::= spaces
41 spaces ~ space+
42 space ~ [ \s ]
43 text ::= textRule
44 textRule ::= char | char textRule
45 char ~ [ ^< ]
46 ...

```

Figure 2.3: CFG Preamble

```

1 _natharith_additionP7N213 ::= rule443
2 rule443 ::= argRuleN213A1ArgSeq rule32 rule443_
3 rule443_ ::= argRuleN213A1ArgSeq | argRuleN213A1ArgSeq rule32 rule443_
4 rule32 ::= moB '+' moE

```

Figure 2.4: MMT Notation in CFG

2.4 The Parsing Backend

The **Semantic Tree Generator** works by recursively querying the **Parsing Backend** and using the result to construct the tree of possible meanings.

The parse trees stored in MMT have the following structure:

- **Variants** - represents a list of possible readings. It is always the top node in any parse tree.
- **Notation** - a notation detected in the input. This node contains the name and arguments of the notation it represents.
- **Argument** - an argument of a notation. The plugin is recursively called on it to construct its meaning subtree as well.
- **RawString** - the ground term representation.

The final part of the semantification processes is converting the internal representation to a standard one, which is CMML in this case. MMT provides a simple API which requires:

1. MMT notation path
2. Argument maps (maps from the argument number to the corresponding substring)

Both of which are available in the representation structure. The argument path is obtained by extracting the argument number from the argument name, and looking up in a map of paths created at the time of grammar creation. This implies the grammar rule names are overloaded with meaning, however, the possibilities are very limited in this aspect since the parsing framework used does not give complete control over the parsing process.

The Parsing Backend [Figure 2.1](#) receives requests from the **Web UI** directly for Guided Semantification and from the **MMT Backend** for Semantic Tree Generation. All the parsing related work is delegated to this backend.

The code of the Parsing Backend can be found on GitHub [[Tola](#)].

The parsing backend consists of two parts.

First of all, the CFG needs to be queried and parsed. This is implemented using lazy evaluation, which means that it is only done when a request actually comes.

The serialized CFG is unpacked and fed to the Marpa Parser Generator.

Second, useful information needs to be extracted from the generated parse trees. Note that going through all the parse trees is not practical, so only the first N (currently 1000) parse trees are processed. This still gives the correct results in most cases since the grammar rules are optimized for giving preference to parse trees that are more likely to be correct.

3 Evaluation

This section presents an evaluation of **MathSemantifier** from the point of view of efficiency and effectiveness of semantic tree generation. Next, the interoperability of the system with other possible applications is examined, which mostly depends on the backend APIs.

Testing **MathSemantifier** through normal operation is not a trivial task, because the results need a human expert to check whether the results are indeed correct. Fortunately, the **MathHub Glossary** [[KWA](#)] contains a sufficient number (about 3000) of examples of Presentation

MathML with Content MathML annotations that result from applying the **Presentation Algorithm** discussed in the introduction. Since **MathSemantifier** is the partial inverse of the **Presentation Algorithm**, checking whether the results are indeed correct boils down to comparing two Content MathML trees.

For the purpose of testing **MathSemantifier** on the **MathHub Glossary**, the **Evaluation** option mentioned in the **Web UI** section is used. It processes examples from the **MathHub Glossary**.

A visual control of approximately 500 examples from the **MathHub Glossary** revealed the following results: About 40% of the examples are semantified correctly. However, this is largely because of reasons that do not depend on **MathSemantifier** itself.

Contrary to what the percentage number suggests, for expressions with less than 1000 parse trees that are correctly rendered, **MathSemantifier** produces correct results with a very high probability.

Let us look into the reasons that make **MathSemantifier** fail. First of all, the reasons that do not depend on the system itself.

1. The Presentation Algorithm fails to render CMML into PMML properly in about 30-40% of the cases.
2. About 10% of the examples include symbols from archives that were not included in the CFG

Up to this point, if we exclude the above mentioned examples, the effective success rate of **MathSemantifier** is about 80-90%.

1. Some notations are omitted because they create cycles in the CFG
2. Input data that is too long or complex having more than 1000 parse trees
3. Assuming the input is more knowledge-rich than it actually is

While the effective success rate of 80-90% is inspiring for a proof of concept system, the main issue of **MathSemantifier** is simply poor performance on long or complex input. Therefore, **MathSemantifier** is a successful proof of concept, but not yet a practical tool.

4 Conclusions

The conclusion of this study is the proof of concept architecture and implementation of a system capable of converting **Presentation MathML** to all the possible meanings, which is a list of **Content MathML**. The testing revealed that the system is able to recognize correctly single top level symbols, as well as the whole set of readings of expressions with less than 1000 parse trees (this is not the limit, but no testing is done beyond that). Naturally, **MathSemantifier** can only recognize the symbols that were used when building the Context Free Grammar it uses. This shows that it is certainly possible to aggregate the knowledge from the MMT Notations and to use it for parsing purposes. However, both the Notations and the Parsing Framework need significant improvements in order for the system to be scalable beyond what is presented in the previous section. The most important part of this study is, therefore, the optimizations and heuristics used, and other techniques presented below that further research could benefit from.

References

- [ABC⁺10] Ron Ausbrooks, Stephen Buswell, David Carlisle, Giorgi Chavchanidze, Stéphane Dalmas, Stan Devitt, Angel Diaz, Sam Dooley, Roger Hunter, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Paul Libbrecht, Bruce Miller, Robert Miner, Murray Sargent, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 3.0. W3C Recommendation, World Wide Web Consortium (W3C), 2010.
- [arX] arXiv. Monthly Submissions Statistics. https://arxiv.org/stats/monthly_submissions. [Online; accessed 8-May-2016].
- [GIJ⁺] Deyan Ginev, Mihnea Iancu, Constantin Jucovski, Andrea Kohlhase, Michael Kohlhase, Heinz Kroger, Jurgen Schefter, and Wolfram Sperber. The SMGLoM Project and System.
- [HPK14] Radu Hambasan, Corneliu C. Prodescu, and Michael Kohlhase. MathWebSearch at NTCIR-11. 2014.
- [Keg] Jeffrey Kegler. The Marpa Parser. <https://jeffreykegler.github.io/Marpa-web-site/>. [Online; accessed 19-April-2016].
- [KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. Notations for living mathematical documents. In Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki, and Freek Wiedijk, editors, *Intelligent Computer Mathematics*, number 5144 in LNAI, pages 504–519. Springer Verlag, 2008.
- [Koh] Michael Kohlhase. The SMGLoM Archive. <https://mathhub.info/smgglom>. [Online; accessed 4-May-2016].
- [Koh08] Michael Kohlhase. Using L^AT_EX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.
- [KT] KWARC and Ion Toloaca. MathSemantifier Parsing Backend. <https://svn.kwarc.info/repos/MMT/>. [Online; accessed 4-May-2016].
- [KWA] KWARC. MathHub Glossary. <https://mathhub.info/mh/glossary>. [Online; accessed 1-May-2016].
- [Mil] Bruce Miller. LaTeXML: A L^AT_EX to XML converter. Web Manual at <http://dlmf.nist.gov/LaTeXML/>. seen September 2011.
- [Rab] Florian Rabe. The MMT API: A Generic MKM System.
- [Tola] Ion Toloaca. MathSemantifier Parsing Backend. <https://github.com/itoloaca/Notation-Based-Parsing>. [Online; accessed 4-May-2016].
- [Tolb] Ion Toloaca. MathSemantifier Web UI. <https://github.com/itoloaca/MathSemantifier>. [Online; accessed 4-May-2016].

ThEdu 2016: Theorem Provers Components for Educational Software

ThEdu is a forum to gather the research communities for computer Theorem Proving (TP), Automated Theorem Proving (ATP), Interactive Theorem Proving (ITP) as well as for Computer Algebra Systems (CAS) and Dynamic Geometry Systems (DGS). The goal of this gathering is to combine and focus systems of these areas and to enhance existing educational software as well as studying the design of a next generation of educational mathematical tools. ThEdu Web-page: <http://www.uc.pt/en/congressos/thedu/thedu16>.

Educational software tools have integrated technologies from CAS, from DGS, from Spreadsheets and others, but not from TP with few exceptions: the latter have been developed to model mathematical reasoning in software – rigorous reasoning as a companion of calculating, which guarantees the unsurpassed reliability of mathematics. Providing students with insight in and experience with this kind of reliability is considered an essential aim of mathematics education.

Tps intrude into science as well as into industry: They are used to tackle difficult proofs in the science of mathematics, like the Four Color Problem or the Kepler Conjecture. In industry Tps are successfully used to verify safety critical software. This workshop addresses a window of opportunity during the currently open development of TP.

The workshop provides a meeting place for educators and developers of educational mathematics software and experts in TP. The discussions shall clarify the requirements of education, identify advantages and promises of TP for learning and motivate development of a novel kind of educational mathematical tools probably establishing a new generation of such tools.

July 2016, Walther Neuper & Pedro Quaresma

Lucas-Interpretation from Users' Perspective

Walther Neuper

IICM, Institute for Computer Media, University of Technology. Graz, Austria

wneuper@ist.tugraz.at

Requirements-engineering for educational software [14] raised the question, how much efforts would be required for implementing substantial material from mechanics [16, 17] in a system based on technology from Computer Theorem Proving (TP). The question appears relevant for several kinds of “users”: for decision makers, for course designers and last not least for staff from faculties of engineering, who is interested to implement their own examples and exercises in the future.

Since TP is new in the field, there are some general informations about how to implement material in a TP-based system: First there needs to be a so-called “theory” which collects or imports all definitions used in the material and which formalises respective theorems and proofs. The system under consideration is Isac [3], a prototype based on the TP Isabelle [2]. Isabelle’s standard distribution contains most of the mathematics required, multivariate analysis¹ etc. Further material is in the Archive of Formal Proofs [1]. And what is not yet covered by these sources, can be defined as axioms preliminarily, see for instance². Specification of problems is not much effort, see some prototype implementations³. The focus of the paper are the methods solving the problems.

The paper is organised as follows: §1 presents LI as a slight extension of usual interpreters for programming languages and introduces an example used throughout the paper⁴, §2 discusses the present state and future development of Isac’s programming language. Because the latter is purely functional, without any statement for input or output, there is the question “Where are the Interactions from?” raised in §3 before a conclusion reshapes LI’s advantages for educational software.

1 A Slightly Extended Interpreter

An interpreter of a programming language works as sketched in Fig.1 on p.2: The interpreter reads a statement at a certain *location* in a *program*; the statement is *interpreted* such that the *location* moves on to another statement to be read next; the interpreter also maintains an *environment*, which pairs identifiers encountered in statements with respective values; a step of interpretation updates the *environment* according to the interpreted statement.

A *Lucas-Interpreter (LI)* extends the above with additional elements and actions: First a step of *calculation* is constructed by each step of interpretation. Guarantee of correctness for steps in *calculations* is the purpose of the additional elements; for logical details see [12], here follows a general explanation according to Fig.1:

A *theory* provides the language elements for certain logical expressions collected in a *context* [18]. In each step of interpretation the *context* provides the logical facts required to correctly deduce the next

¹https://isabelle.in.tum.de/dist/library/HOL/HOL-Multivariate_Analysis/index.html

²<https://intra.ist.tugraz.at/hg/isa/file/e7afa662670b/src/Tools/isac/Knowledge/Biegelinie.thy>

³http://www.ist.tugraz.at/projects/isac/www/kbase/pbl/index_pbl.html

⁴The running example is part of another example used with another perspective in [14].

step of *calculation*; this action is called *prove* in Fig.1. A blue square in this figure indicates, that input of a *formula or tactic* to the *calculation* is *proved* correct by automated provers using the current *context*, which is updated at each step of interpretation.

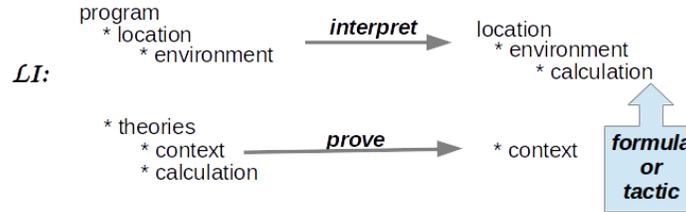


Figure 1: Survey on Lucas-Interpretation

The scope of environments with respect to programs with nested sub-programs has been clarified in the discipline of compiler-construction. However, the scope of contexts in the presence of theories and “locales” is not yet settled [18]. Isac uses the following scoping rules: a context is initialised by the pre-conditions at the start of interpretation, it is visible in sub-programs; the context of a sub-program remains local except predicates containing variables declared in the output of the respective sub-program.

The execution of the program’s statements is done by rewriting, as usual with functional programs (where one speaks about “evaluation” of “functions” instead of “execution” of “programs” as we do here). In the present state of Isac’s prototype there are lots of evaluators: for list-expressions in programs, for normalisation of user input (for checking correctness). for pre-conditions, etc. Compiling these evaluators is an elaborate, expensive and error-prone task. So migration of Isac’s programming language to Isabelle’s function package [5] shall free the programmer from these tasks. As soon as this integration is done, all the functions available for Isabelle/HOL are ready for LI, including those which implement computer algebra (see for instance [7, 13]).

A program is accompanied by a “guard”; guard and program together are called a “method” in Isac. The guard is not contained in Fig.1; below an example is copied from [14] §.1:

```

1   Guard:
11  Given: Masses  $m = 2 \text{ kg}$ , Length  $l_0 = 0.3 \text{ m}$ , Consts  $\{c_1 = 1.1 \frac{\text{kg}}{\text{s}^2}, c_2 = 2.2 \frac{\text{kg}}{\text{s}^2}\}$ , Damper  $d = 0.4 \frac{\text{Ns}}{\text{m}}$ 
12  Where:  $0 < m \wedge l_0 < 0$ 
13  Find: Matrixes  $\{M(m), D(d), C(c_1, c_2)\}$ , DiffEq  $M \cdot \ddot{x} + D \cdot \dot{x} + C \cdot x = F$ 
14  Relate:  $\exists x. \forall t. t > 0 \Rightarrow M \cdot \ddot{x} + D \cdot \dot{x} + C \cdot x = F$ 

```

Given in line 11 lists the concrete input items to the program, Find declares the output item(s). Where is the pre-condition, which shows, that the Guard is a conjunction of predicates restricting input (and restrictions imposed by physical constants could be added here as well). In Relate the post-condition relates input and output (in the sense of [4]); this particular post-condition can be proved by the theory of differential equations, but it is not immediately useful for an engineer, who wants the solution of an equation and not only some promise by \exists .

2 The Interpreted Language

Isac’s programming language has been implemented [9] before the “function package” [5] has been introduced to Isabelle. But Isac’s language has been designed such, that it anticipated the function package and now the former can be explained in terms of the latter.

Functions in Isabelle/HOL must be total in order to keep the logic consistent. Isac’s programs are not total in general, their input is restricted by pre-conditions as shown in §1. Thus Isac’s programs are declared as `partial_function` in Isabelle.

A major difference between Isabelle and Isac concerns the purpose of functions: while the former is built for proving properties of functions, for evaluating them and probably generating efficient code from them, the latter is built for stepwise construction of calculations solving problems in (applied) mathematics. Construction of calculations comprises interactively specifying and solving the respective problem. Below the guard from §1 above is re-used by the `Specification` (thus not shown again and folded in) for a `Problem` with a `Solution` as follows:

```

21  Problem [determine, 2-mass-oscillator, DiffEq]:
211  Specification:
212  Solution:
2121                                     forces of springs
2122      [Fc1 = c1x1, Fc2 = c2(x2 - x1), Fc3 = c1x2]
2123                                     forces of dampers
2124      [Fd1 = d $\dot{x}_1$ , Fd2 = d $\dot{x}_2$ ]
2125                                     mass times acceleration equals sum of all forces
2126      [m $\ddot{x}_1$  = -Fc1 + Fc2 - Fd1 - F1, m $\ddot{x}_2$  = -Fc2 - Fc3 - Fd2 + F2]
2127                                     Substitute [Fc1, Fc2, Fc3, Fd1, Fd2]
2128      [m $\ddot{x}_1$  = -c1x1 + c2(c2 - x1) - d $\dot{x}_1$  - F1, m $\ddot{x}_2$  = -c2(c2 - x1) - c1x2 - d $\dot{x}_2$  + F2]
2129                                     Rewrite.Set normalise
212a      [m $\ddot{x}_1$  + d $\dot{x}_1$  + c1x1 - c2(x2 - x1) = F1, m $\ddot{x}_2$  + d $\dot{x}_2$  + c2(x2 - x1) + c1x1 = F2]
212b                                     switch to vector representation
212c       $\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ F_1 \end{pmatrix} + \begin{pmatrix} 0 \\ F_2 \end{pmatrix}$ 
22       $\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \ddot{x} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \dot{x} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} x = \begin{pmatrix} 0 \\ F_1 \end{pmatrix} + \begin{pmatrix} 0 \\ F_2 \end{pmatrix}$ 

```

The above `Solution` is considered as close to a calculation written by hand on a blackboard as possible; on the left margin there are the formulas of the calculation (indented according to the calculation’s structure), on the right margin there are the tactics and hints; the rest should be self-explanatory. The `Solution` is the result of Lucas-Interpretation of this program:

```

.  partial_function diffeq_2_mass_oscil (m, l_0, [c_1, c_2], d, springs, dampers, sums) =
1  let
11  begin_parallel
1101      springs = Take springs “forces of springs”
111  parallel
1111      dampers = Take dampers “forces of dampers”
112  parallel
1121      sums = Take sums “mass times acceleration equals sum of all forces”
12  end_parallel
13  diffeq = Take sums “”
14  diffeq = Substitute [ springs, dampers ]
15  diffeq = Rewrite.Set normalise

```

```

16    diffeq = Rewrite_Set vectorify “switch to vector representation”
2    in
21    diffeq

```

The partial function *diffeq_2_mass_oscil* gets the arguments (*m*, *l_0*, [*c_1*, *c_2*], *d*, *springs*, *dampers*, *sums*) from the preceding specification phase. The first lines 2122..2126 of the calculation could have been constructed in arbitrary order; this is reflected by the lines 11..12 in the above function: the statement `parallel` models parallel execution, while the remaining statements 13..17 represent a sequence.

Above there are specific statements, called “tactics” in weak analogy to TP: Take, Substitute and Rewrite.Set as examples for some dozen others. Tactics are handled by LI like “break points” by debuggers: interpretation halts at the tactics and passes control to the dialog component, which decides how to pass control back to LI, see the next section.

3 Where are the Interactions from?

As shown in §2 by example, a program in Isac is purely functional as is an Isabelle function, without side-effects and without input or output. However, LI creates side-effects in a specific way and cooperates with a dialogue component. The architectural design is shown in Fig.2. The *WorksheetDialog* implements the observer pattern [6] and listens to two active components: the *Worksheet* as interface to the user on the one side and to the *MathEngine* as interface to LI on the other side; the latter is active in the sense, that response may be delayed due to heavy prover activity:

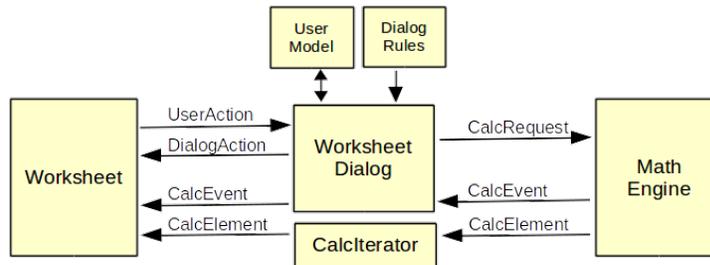


Figure 2: Isac’s dialog component

Here come some details, how automated dialog generation works, by use of the running example: LI starts after successful completion of interactive specification, reads line 11 in the function on p.3 and halts at the tactic Take. By delivering a *CalcEvent* the *MathEngine* passes control to the *WorksheetDialog*. Now the dialogue is free to choose various kinds of interaction from the *DialogRules* according to the *UserModel*: the user can be a novice and might prefer to watch passively, or the dialog-mode is set to “exam” and forces the student to be active, etc. So, the dialog might ...

- allow the student to request the next formula (passive watching)
- suggest the kind of next step by “forces of springs”
- provide a partial next formula like $[F_{c1} = \dots, F_{c2} = \dots, F_{c3} = \dots]$ to be completed as shown by [14].Fig.2.
- suggest further parts of the formula (some more activity)
- enforce input of the formula (maximal activity on the student’s side): each of the lines 2122, 2124 or 2126 could be input due to the `parallel` statement. And the input formula can have

arbitrary format as long as it is algebraically equivalent, due to simplification to a normal form in the *MathEngine*.

- allow to review the method helping in construction of the calculation (if not in “exam” mode)
- allow to investigate Isac’s mathematics knowledge base, look at other examples and continue with the calculation eventually.
- etc.

Depending on the decision of the *WorksheetDialog* after some *UserActions* and *DialogActions* the respective *CalcRequest* is sent to the *MathEngine*. The latter responds with a *CalcEvent*, which notifies the *Worksheet* (by the way controlled by the *WorksheetDialog*) that a new *CalcElement* waits to be fetched from the *MathEngine* via a *CalcIterator*, which has read-access to the whole calculation under construction by LI.

The kind of interaction described above can be considered a dialog between partners on an equal base: both, the student and the system, are able to do steps of calculation more or less actively. The architecture reflects this balance, where LI is the source of power on the system side during interactive calculation.

In the present state of development the *DailogRules* are implemented only to an extent which allows demonstration of LI by Isac’s prototype. A general machinery [8] is ready to cope with the complexity of interaction expected in the future. Respective “dialog authoring” will be an efficient investment: interactions in doing mathematics are considered independent from various areas of mathematics; also differences in behaviour of novices versus experts are considered the same in all areas.

The present state of development, the *UserModel* is still a stub. The stub is designed such, that each student is assigned an individual data set, which can be preset as well as updated during a session: error rates on specific knowledge items (rule, problem, method, example), preferences in interaction and current dialog mode. Three modes are envisaged at least: investigation, exercise, examination (the modes are subsets of *DailogRules*).

Relevant in the context of questions about efforts for implementation of interactive TP-based course material is: the implementation of functions is not concerned with interaction at all, a “mathematics author” can focus mathematics and nothing else.

4 Conclusion

In spite of higher complexity of TP-based systems with Lucas-Interpretation as compared with a Computer Algebra System, the implementation of course material does not require more efforts in principle, as soon as Isac’s programming languages has migrated to Isabelle’s function package. Purely functional programs are specifically interpreted by so-called Lucas-Interpretation (LI), which generates steps of calculation and respective dialogue guidance as side-effects of steps in interpretation.

Lucas-Interpretation maintains a context, which provide the most powerful technology available with logical facts for checking correctness of user-input. Thus there is maximal freedom for input – for algorithmic sequences as well as for formula representation: equivalence modulo a theory is checked with maximal reliability. The ability to propose a next step in calculations is the novel feature contributed by Lucas-Interpretation.

No additional efforts are required when programming methods to solve engineering problems: checking steps and proposing steps is done by Lucas-Interpretation automatically and in cooperation with a dialogue module user-guidance is generated automatically.

Finally, after fifteen years of conceptual work and of prototyping, a successful proof of concept [10, 11, 15] some time ago and after the recent requirements engineering, Isac appears ready to start development for a professional release usable at universities of applied sciences.

References

- [1] *Archive of Formal Proofs*. <http://afp.sourceforge.net>.
- [2] *Generic proof assistant "Isabelle"*. <http://isabelle.in.tum.de/>.
- [3] *Isac-project*. <http://www.ist.tugraz.at/isac/History>.
- [4] Dines Bjørner (2006): *Software Engineering. Texts in Theoretical Computer Science 1,2,3*, Springer, Berlin, Heidelberg.
- [5] Lukas Bulwahn, Alexander Krauss, Florian Haftmann, Levent Erkk & John Matthews (2008): *Imperative Functional Programming with Isabelle/HOL*. In Otmane Mohamed, Csar Muoz & Sofine Tahar, editors: *Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science 5170*, Springer Berlin / Heidelberg, pp. 134–149, doi:10.1007/978-3-540-71067-7_14. Available at http://dx.doi.org/10.1007/978-3-540-71067-7_14.
- [6] Ralph Johnson Erich Gamma, Richard Helm & John M.Vlissides (1994): *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- [7] Florian Haftmann, Andreas Lochbihler & Wolfgang Schreiner (2014): *Towards abstract and executable multivariate polynomials in Isabelle*. Isabelle Workshop 2014, <http://www.infsec.ethz.ch/people/andreloc/publications/haftmann14iw.pdf>.
- [8] Markus Kienleitner (2012): *Towards "NextStep Userguidance" in a Mechanized Math Assistant*. Master's thesis, IICM, Graz University of Technology. Bakkalaureate Thesis.
- [9] Walther Neuper (2001): *Reactive User-Guidance by an Autonomous Engine Doing High-School Math*. Ph.D. thesis, IICM - Inst. f. Softwaretechnology, Technical University, A-8010 Graz. <http://www.ist.tugraz.at/projects/isac/publ/wn-diss.ps.gz>.
- [10] Walther Neuper (2006): *Angewandte Mathematik und Fachtheorie*. Technical Report 357, IMST – Innovationen Machen Schulen Top!, University of Klagenfurt, Institute of Instructional and School Development (IUS), 9010 Klagenfurt, Sterneckstrasse 15. http://imst.uni-klu.ac.at/imst-wiki/index.php/Angewandte_Mathematik_und_Fachtheorie.
- [11] Walther Neuper (2007): *Angewandte Mathematik und Fachtheorie*. Technical Report 683, IMST – Innovationen Machen Schulen Top!, University of Klagenfurt, Institute of Instructional and School Development (IUS), 9010 Klagenfurt, Sterneckstrasse 15. http://imst.uni-klu.ac.at/imst-wiki/index.php/Angewandte_Mathematik_und_Fachtheorie_2006/2007.
- [12] Walther Neuper (2012): *Automated Generation of User Guidance by Combining Computation and Deduction*. pp. 82–101, doi:10.4204/EPTCS.79.5. <http://eptcs.web.cse.unsw.edu.au/paper.cgi?THedu11.5>.
- [13] Walther Neuper (2014): *GCD — A Case Study on Lucas-Interpretation*. In: *Joint Proceedings of the MathUI, OpenMath and ThEdu Workshops and Work in Progress track at CICM*, Coimbra, Portugal. <http://ceur-ws.org/Vol-1186/paper-17.pdf>.
- [14] Walther Neuper (2016): *Rigor of TP in Educational Engineering Software*. In: *submitted to CICM*, Bialystok, Poland. <http://www.ist.tugraz.at/projects/isac/publ/tp-engin-sw.pdf>.
- [15] Walther Neuper & Johannes Reitingner (2008): *Begreifen und Mechanisieren beim Algebra Einstieg*. Technical Report 1063, IMST – Innovationen Machen Schulen Top!, University of Klagenfurt, Institute of Instructional and School Development (IUS), 9010 Klagenfurt, Sterneckstrasse 15.

- http://imst.uni-klu.ac.at/imst-wiki/index.php/Begreifen_und_Mechanisieren_beim_Algebra-Einstieg.
- [16] Wolfgang Steiner (2012): *Vorlesungsskriptum Technische Mechanik II. Sommersemester 2012*. FH OÖ Campus Wels.
- [17] Wolfgang Steiner (2015): *Vorlesungsskriptum Technische Mechanik III*. FH OÖ, Fakultät für Technik und Umweltwissenschaften.
- [18] Makarius Wenzel (2015): *The Isabelle/Isar Implementation*. <http://isabelle.in.tum.de/website-Isabelle2015/dist/Isabelle2015/doc/implementation.pdf>.

Rigor of TP in Educational Engineering Software

Walther Neuper

IICM, Institute for Computer Media, University of Technology, Graz, Austria

wneuper@ist.tugraz.at

The discipline of Computer Theorem Proving (TP) distinguishes itself by formal rigor in doing mathematics in various application domains [1]. This short paper is, however, *not* on TP but on educational *software based on TP* components. Such software promises advantageous features [7] some of which are demonstrated by a prototype [3] called Isac. Isac is based on the TP Isabelle [2] and generates dialogues similar to interaction with chess software: moves in chess are considered as rigorous formal as steps in calculations are when solving problems in engineering disciplines. Isac checks input of students by use of Isabelle’s automated provers, which in turn are provided with necessary logical context by Lucas-Interpretation [6]. This interpreter also allows to propose next steps towards a solution, so roles can be arbitrarily switched between student and system.

This paper reports work in progress in cooperation with universities of applied sciences in Austria. The work concerns a feasibility study on how Isac could serve in engineering education at these universities. Since Isac has been designed for “pure” mathematics, the study encounters several challenges. Below one running example presents three major challenges for discussion; the example is from [9] and slightly changed for reasons discussed in §2:

Given is a system with two oscillating masses, $m = 2 \text{ kg}$, connected by linear springs with length $l_0 = 0.3 \text{ m}$ and damped with $d = 0.4 \frac{\text{Ns}}{\text{m}}$ as shown in Fig.1. The respective spring constants are $c_1 = 0.11 \frac{\text{N}}{\text{m}}$ and $c_2 = 0.22 \frac{\text{N}}{\text{m}}$. The masses are located such that $x_1 = x_2 = 0$ with relaxed springs; initially the masses are dislocated with $x_1 = x_2 = 0.05 \text{ m}$ and have velocities $v_1 = 0.1 \frac{\text{m}}{\text{s}}$ and $v_2 = 0.2 \frac{\text{m}}{\text{s}}$ respectively. The right mass is excited by force $F = 0.6 \sin(3t) \text{ N}$. Change the given spring constant c_2 such that the left mass becomes a vibration absorber for the right one (i.e. make the masses oscillate in opposite directions such that the system shows no vibration to the outside).

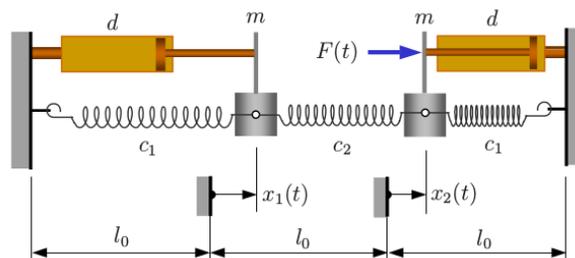


Figure 1: System with two oscillating masses ©W.Steiner 2015 [9]

A solution for c_2 involves modelling the system and comprises several sub-problems: determine the differential equation, solve the homogeneous part, determine the particular solution for F and finally calculate c_2 . Below we start with the first subproblem and demonstrate the first challenge raised by geometric descriptions typical for modelling physical systems.

1 Formal Specification and Geometric “Intuition”

A formal specification in the sense of [4] makes the input `Given` and the output `Find` to a system’s model explicit as shown below; it restricts input by a pre-condition `Where` and relates input with output by a post-condition `Relate`. The first subproblem of the running example is formally specified as follows:

```

21 Problem [determine, 2-mass-oscillator, DiffEq]:
211 Specification:
2111 Model:
21111 Given: Masses  $m = 2 \text{ kg}$ , Length  $l_0 = 0.3 \text{ m}$ , Consts  $\{c_1 = 0.11 \frac{\text{N}}{\text{m}}, c_2 = 0.22 \frac{\text{N}}{\text{m}}\}$ , Damper  $d = 0.4 \frac{\text{Ns}}{\text{m}}$ 
21112 Where:
21113 Find: Matrixes  $\{M(m), D(d), C(c_1, c_2)\}$ , DiffEq  $M \cdot \ddot{x} + D \cdot \dot{x} + C \cdot x = F$ 
21114 Relate:  $\exists x. \forall t. t > 0 \Rightarrow M \cdot \ddot{x} + D \cdot \dot{x} + C \cdot x = F$ 
2112 References:
212 Solution:

```

The Problem in line 21 is named such, that a reference into Isac’s knowledge base is given ¹. The other References addressed by line 2112 point to a theory, which imports language elements like \ddot{x} , and to a method which can create a Solution, are collapsed here (as well as the pre-conditions in `Where`). The notation $M(m)$ establishes a literal connection between `Given` and `Find`; the notation is up to discussion. The post-condition in 21114 comprises an \exists not relevant for engineers and might be omitted.

In interactive construction of a Solution the challenge for students is to relate forces, for instance

$$m\ddot{x}_1 = -F_{c1} + F_{c2} - F_{d1}, \quad m\ddot{x}_2 = -F_{c2} - F_{c3} - F_{d2} + F(t)$$

And for the task of relating the forces, figures like Fig.2 are used to capture coordinates and forces. Now the problem with Isac’s design is, that such figures capture relations in a precise representation, but this representation is geometric, not formal — and Isac is designed to work with formulas (which would be clumsy in capturing geometric structure here), which can be handled by Isabelle’s components in the background. So the section’s headline advocates “intuition” as opposed to formal specification.

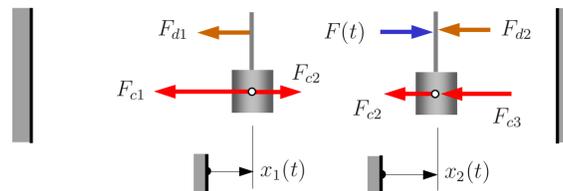


Figure 2: Forces on the oscillating masses ©W.Steiner 2015 [9]

Such figural representations are used frequently and in different engineering disciplines, not only in mechanical engineering. So efforts seem well invested to tackle this design challenge, to sustain Isac’s claim to be a “system that explains itself” and to develop a generally usable component for that purpose.

Such a component shall allow to add coordinates, arrows and associated identifiers at certain positions in a figure. This component also shall provide feedback automatically generated from a formalisation, which has been prepared for each example by Isac’s math-authors. Generation of figural representations like Fig.2 shall become another duty of math-authors (while managing interaction is concern of the component).

¹http://www.ist.tugraz.at/projects/isac/www/kbase/pbl/index_pbl.html

2 Learning by Switching Levels of Abstraction

Learning to comprehend abstraction requires experiencing a multitude of **concrete examples** — a fact experienced in the practice of education in general, not only in engineering education and with abstract models in mechanics. Mathematical abstractions, like differential equations, however, have an advantage: they can be computed with concrete values, they even can be dynamically simulated given such values.

Another characteristics of complex learning processes, like comprehension of abstract models, is that they succeed **not in one go**. Learning happens in phases in the brain, which usually are separated by latency periods – and these cannot be planned from outside an individual brain; so, a lecture on the behaviour of two oscillating masses (e.g. [9].p.122–129) is only a part of respective learning processes.

So, how to cope with these challenges in learning to comprehend abstraction? Good old L^AT_EX provided wide margins for personal notes in papers and textbooks again and again; interactive media can do better nowadays, if they are designed appropriately. And it appears obvious: the more such media cover the process of problem solving, the better. Isac claims to cover the whole process; §1 showed, how problem specification is covered by Isac. Below is shown, what else can be done.

Include creation of models into concrete examples as done with the running example: The problem statement on p.1 contains a concrete request for a particular value of c_2 (below folded into `Specification` in order to save space) — nevertheless the `Solution` should comprise the creation of the underlying abstract model as follows:

```
. Problem [absorber, 2-mass-oscillator]
1 Specification:
2 Solution:
21 Problem [determine, 2-mass-oscillator, DiffEq]
22 
$$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \ddot{x} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \dot{x} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} x = \begin{pmatrix} 0 \\ F \end{pmatrix}$$

23 Problem [solution, 2-mass-oscillator, homogen, DiffEq]
24 
$$x(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} (A_1 \cos \omega_1 t + B_1 \sin \omega_1 t) + \begin{pmatrix} 1 \\ -1 \end{pmatrix} (A_2 \cos \omega_2 t + B_2 \sin \omega_2 t),$$

25 Problem [particular, solution, 2-mass-oscillator, DiffEq]
26 
$$x_1(t) = \begin{pmatrix} 0 \\ a_1 \end{pmatrix} \sin \Omega t, x_2(t) = \begin{pmatrix} 0 \\ a_2 \end{pmatrix} \sin \Omega t, a_1 = \frac{F_0 c_2}{(c_1 + c_2 - m \Omega^2)^2 - c_2^2}, a_2 = \frac{F_0 (c_1 + c_2 - m \Omega^2)}{(c_1 + c_2 - m \Omega^2)^2 - c_2^2}$$

27 Problem [complete, solution, 2-mass-oscillator, DiffEq]
28 
$$x(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} (A_1 \cos \omega_1 t + B_1 \sin \omega_1 t) + \begin{pmatrix} 1 \\ -1 \end{pmatrix} (A_2 \cos \omega_2 t + B_2 \sin \omega_2 t) + \begin{pmatrix} 0 \\ a_1 \end{pmatrix} \sin \Omega t,$$


$$a_1 = \frac{F_0 c_2}{(c_1 + c_2 - m \Omega^2)^2 - c_2^2}, a_2 = \frac{F_0 (c_1 + c_2 - m \Omega^2)}{(c_1 + c_2 - m \Omega^2)^2 - c_2^2}$$

29 Problem [compute, spring]
2a  $c_2 = 1.2345 N$ 
.  $c_2 = 1.2345 N$ 
```

Since above the modelling process is included into the `Solution`, students are enabled, not just to advocate some formula from somewhere (which seduces to use formal models without understanding). A student can regard the (sub-)Problems as black-boxes, of course (if not forced by Isac’s dialog guide to actively do the step). But Isac is designed to elicit experimentation, for instance to experiment with different input values to a problem in order to approximate a solution by trial and error. Since solving the above problem with trials soon turns out hopeless, a student might be motivated to make the sub-Problems white-boxes, look into them, study details and to rework creation of the abstract model.

Switch levels of abstraction: In the above calculation the concrete result $c_2 = 1.2345 N$ was possible, because the `Specification` contains the concrete values given on p.1, which were input at the beginning of the calculation:

```
. Problem [absorber, 2-mass-oscillator]
1 Specification:
2 Solution:
21 Problem [determine, 2-mass-oscillator, DiffEq]
22 [2x1 + 0.4x1 + 3.3x1 - 0.22x2 = 0, 2x2 + 0.4x2 - 0.22x1 + 3.3x2 = 0.6]
23 Problem [solution, 2-mass-oscillator, homogen, DiffEq]
24 [x1(t) = 0.05e-0.1t(cos 0.81t + 3.85 sin 0.81t),
x2(t) = 0.05e-0.1t(cos 0.81t + 3.85 sin 0.81t)]
25 Problem [particular, solution, 2-mass-oscillator, DiffEq]
26 [x1(t) = -0.05e-0.1t0.59 sin 1.69t, x2(t) = 0.05e-0.1t0.59 sin 1.69t]
27 Problem [complete, solution, 2-mass-oscillator, DiffEq]
28 [x1(t) = 0.05e-0.1t(cos 0.81t + 3.85 sin 0.81t - 0.59 sin 1.69t),
x2(t) = 0.05e-0.1t(cos 0.81t + 3.85 sin 0.81t + 0.59 sin 1.69t)]
29 Problem [compute, absorber]
2a c2 = 1.2345 N
. c2 = 1.2345 N
```

Note that above also intermediate results are numeric values (copied from a Mathematica notebook): both representation, symbolic and numeric, have their advantages: the first tells about the structure of the model, the latter tells about concrete results (which can be observed in dynamic simulation of the enclosed differential equations, ultimately, see [5]).

Switching between symbolic representation and numeric representation can be done by computer software: this novel feature seems of utmost importance for learning, so it shall be included to Isac within the next development phase. Realisation requires to extend Isac's Lucas-Interpreter [6] such that the interpreter's environment not only takes identifier-value pairs, but associates identifiers with lists of values.

Both design features together, the feature to include models into concrete examples and the feature to switch levels of abstraction, lead to radically new learning scenarios

- The student is offered to review the construction of the abstract model any time (every example concerning, for instance, the two-mass-oscillator includes the respective model: this is accomplished by copy& past for some sub-problems in the respective program [8]).
- The student is not bothered by unwanted details: he or she can skip in a calculation whatever steps they want to (if the dialog [8] allows to do so, which would not be the case in exams, for example).
- Even students of introductory courses can be offered to look into advanced examples like the two-mass-oscillator under consideration: the dialog jumps to some subproblem in the calculation, which is up to exercise in the course (e.g. differentiation, equation solving, etc). Then the student interactively works on the respective subproblem, and if the problem is solved, the system finished the calculation automatically. This way questions like "What for do we learn this method" are anticipated in an unobtrusive way.

3 Formal Deduction and Physical Arguments

Isac is designed such that a student can rely on the system, that wrong steps in a calculation are rejected reliably. From the assumptions (the input in `Given` and the pre-conditions in `Where`) the steps are

formally deduced such that finally the post-condition (partially represented in `Relate`) can automatically be proven; for details see [6].

Now, in §1 we encountered an example, where essential assumptions are given by geometric structures (arrows as forces in a figure) and not by formulas. So it seems straight forward, to add informal arguments to the steps of formal deduction. We come back to this sub-Problem and proceed from the `Specification` (folded in below) to the `Solution`:

```

21  Problem [determine, 2-mass-oscillator, DiffEq]:
211  Specification:
212  Solution:
2121                                     forces of springs
2122      [Fc1 = c1x1, Fc2 = c2(x2 - x1), Fc3 = c1x2]
2123                                     forces of dampers
2124      [Fd1 = d $\dot{x}_1$ , Fd2 = d $\dot{x}_2$ ]
2125                                     mass times acceleration equals sum of all forces
2126      [m $\ddot{x}_1$  = -Fc1 + Fc2 - Fd1, m $\ddot{x}_2$  = -Fc2 - Fc3 - Fd2 + F]
2127                                     Substitute [Fc1, Fc2, Fc3, Fd1, Fd2]
2128      [m $\ddot{x}_1$  = -c1x1 + c2(x2 - x1) - d $\dot{x}_1$ , m $\ddot{x}_2$  = -c2(x2 - x1) - c1x2 - d $\dot{x}_2$  + F]
2129                                     Rewrite_Set normalise
212a      [m $\ddot{x}_1$  + d $\dot{x}_1$  + c1x1 - c2(x2 - x1) = 0, m $\ddot{x}_2$  + d $\dot{x}_2$  + c2(x2 - x1) + c1x1 = F]
212b                                     switch to vector representation
212c      
$$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ F \end{pmatrix}$$

22      
$$\begin{pmatrix} m & 0 \\ 0 & m \end{pmatrix} \ddot{x} + \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \dot{x} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_1 + c_2 \end{pmatrix} x = \begin{pmatrix} 0 \\ F \end{pmatrix}$$


```

Above the numbers at the left do not belong to an Isac calculation, they are for referencing only. In the middle there are the formulas of the calculation (with tree-like indentation). On the right there are the justifications for the formulas.

In the lines 2127 and 2129 above there are tactics which contribute to formal deduction in the `Solution`. But the lines 2121, 2123 and 2125 do not contribute to the formal semantics, they are arguments addressing intuitive understanding of physical aspects of modelling. In line 212b there is even a non-physical argument, useful just for structuring the calculation.

The two latter kinds of arguments are not contained in Isac’s original design. But now requirements analysis suggests to extend Isac’s programming language with an additional tactic for textual arguments; this extension seems simple. For details, how Isac guides the student during step-wise construction of a calculation and how this guidance is generated automatically, see [8].

Conclusions

The paper demonstrated examples from mechanics in order to show, that Isac’s original design is appropriate for education also in engineering disciplines. Requirements analysis in cooperation with experts in engineering education identified additional requirements, which appear realizable with reasonable effort due to TP’s power. In particular, Isac as a prototype for TP-based educational systems seems ready for the following three extensions, which have been introduced in §1, §2 and §3 respectively:

1. An interactive graphics-component for input of coordinates, arrows and associated identifiers at certain positions in a figure, while correctness is checked by use of hidden “formalisations” already present in Isac — this shall support geometric intuition in creating physical models.

2. Extension of Isac’s Lucas-Interpreter such that the interpreter’s environment not only takes identifier-value pairs, but associates identifiers with lists of values (symbolic and numeric) — this shall support comprehending abstract models by switching levels of abstraction.
3. An additional tactic, which plays no role in formal deduction in a calculation, but displays informal arguments for certain steps of calculation — this shall support additional arguments for intuitive understanding.

As soon as these features are implemented, Isac can be considered a “system that explains itself” also for mathematics applied to various engineering disciplines — a system which promises to establish novel learning scenarios in the field.

Acknowledgements

Several experts from the Universities of Applied Sciences in Wels, Hagenberg and Salzburg are involved in the current requirements engineering: Stephan Dreiseitl, Günther Eibl, Klaus Schiefermayr, Wolfgang Steiner and Stefan Sunzenauer. The author owes these persons a great debt of gratitude for their precious time spent and the novel ideas contributed to the design of Isac and to this paper.

Disclaimer

There was no time asking the mentioned experts to review the paper at hand; so any flaws and mistakes in the paper are in full responsibility of the author.

References

- [1] *Archive of Formal Proofs*. <http://afp.sourceforge.net>.
- [2] *Generic proof assistant “Isabelle”*. <http://isabelle.in.tum.de/>.
- [3] *Isac-project*. <http://www.ist.tugraz.at/isac/History>.
- [4] Dines Bjørner (2006): *Software Engineering. Texts in Theoretical Computer Science 1,2,3*, Springer, Berlin, Heidelberg.
- [5] Sarah Lichtblau: *Motion of Two Masses Connected by Springs*. <http://demonstrations.wolfram.com/MotionOfTwoMassesConnectedBySprings/>. Wolfram Demonstrations Project.
- [6] Walther Neuper (2012): *Automated Generation of User Guidance by Combining Computation and Deduction*. pp. 82–101, doi:10.4204/EPTCS.79.5. <http://eptcs.web.cse.unsw.edu.au/paper.cgi?THedu11.5>.
- [7] Walther Neuper (2013): *On the Emergence of TP-based Educational Math Assistants*. 7, pp. 110–129. Available at <https://php.radford.edu/~ejmt/ContentIndex.php#v7n2>. Special Issue “TP-based Systems and Education”.
- [8] Walther Neuper (2016): *Lucas-Interpretation from Users’ Perspective*. In: submitted to CICM, Bialystok, Poland. <http://www.ist.tugraz.at/projects/isac/publ/lucin-user-view.pdf>.
- [9] Wolfgang Steiner (2015): *Vorlesungsskriptum Technische Mechanik III*. FH OÖ, Fakultät für Technik und Umweltwissenschaften.

CICM 2016 Doctoral Program

The Doctoral Programme provides a dedicated forum for PhD students to present and discuss their ideas, ongoing or planned research, and achieved results in an open atmosphere. It consisted of presentations by the PhD students to get constructive feedback, advice, and suggestions from the research advisory board, researchers, and other PhD students. Each PhD student was assigned to an experienced researcher from the research advisory board to act as a mentor and provide detailed feedback and advice on their intended and ongoing research.

This year, four best submissions were selected for the Doctoral Programme. We thank the students for their participation and the mentors for their valuable feedback.

July 2016, Martin Suda

Marek Janasz UP Cracow

Thesis: Automated theorem proving for elementary geometry.

This study analyses automated proofs of theorems from Euclidean Elements, book VI, using the area method. The theorems we will be discussing concern Euclidean field theory about equality of non-congruent figures and similarity of the figures [1]. The proofs are generated by the program WinGCLC.

My proposed hypotheses:

1. The way of modification of the elimination lemmas while adding the elementary constructions to extending the abilities of the area method in proving theorems from Euclidean Elements, book VI.
2. The method of extension of the axiomatic system from [5] to proving theorem for ordered geometry.

I would like to use my results to implement the prover to the programming languages in logic (Prolog, Haskell) or to the applications like Coq. My dissertation plans contain the analysis of the automated proofs of segment's field from Cartesian arithmetic using the area method by the program WinGCLC. My study is concerned with proving equality of line segments using various methods of constructions corresponding with multiplication of line segments (uniqueness) and commutative property of multiplication of line segments.

Bibliography:

1. B laszczyk P., Mrówka K., *Euclides Elementy. Ksiegi V-VI teoria proporcji i podobieństwa, z tłumaczeniem i komentarzem*, Copernicus Center Press, Kraków 2013 (in polish)
2. B laszczyk P., Mrówka K., *Kartezjusz. Geometria*, Universitas, Kraków 2015 (in polish)
3. Chou S. C. , Gao X. S., Zhang J. Z., *Machine Proofs in Geometry*, World Scientific, Singapore 1994.
4. Euclid, *Elements*, edited, and provided with a modern English translation, by R. Fitzpatrick;
<http://farside.ph.utexas.edu/Books/Euclid/Elements.pdf>.
5. Janičić P., Narboux J., Quaresma P., The area method, *Journal of Automated Reasoning* 48, 4 (2012), 489-532.

Knowledge Management across Formal Libraries

Dennis Müller
Jacobs University Bremen

May 23, 2016

Abstract

In my Ph.D., I want to contribute to developing an open archive for formalizations, a common and open infrastructure for managing and sharing formalized mathematical knowledge such as theories, definitions, and proofs, based on a uniform foundation-independent representation format for libraries, integrate existing formal libraries into this archive and develop methods to efficiently transfer and share information between them.

Research Problem: In the last 10 years, the formalization of mathematical knowledge (and subsequent verification/automation of formal proofs) has – especially in conjunction with problems such as as Kepler’s conjecture, the classification theorem for finite simple groups etc. – become of growing interest to mathematicians. By now, there is a vast plurality of formal/symbolic systems and corresponding libraries; almost all of which are, however, non-interoperable because they are based on differing, mutually incompatible foundations (e.g., set theory, higher-order logic, constructive type theory, etc.), library formats, and library structures, and much work is spent developing basic libraries for mathematics in each system.

Also, formalizations in current systems are usually based on the *homogeneous method*, which fixes one foundation with all primitive notions (e.g., types, axioms, and rules) and uses only conservative extensions (e.g., definitions, theorems) to model domain knowledge. For this purpose, most systems support complex conservative extension principles, such as type definitions in the HOL systems [HOL], provably terminating functions in Coq [Tea] or Isabelle/HOL [NPW02], or provably well-defined indirect definitions in Mizar [Miz].

The combination of fixed foundation and homogeneous method means that a lot of – expensive – formalization work is needed just to build the setting of interest (e.g., the real numbers) as a conservative extension of the fixed foundation. However, the resulting formalizations are actually less valuable: It becomes virtually impossible to move them between foundations. Therefore, almost all current systems are mutually incompatible, with only a few ad hoc translations between them (e.g., [KW10; KS10]).

On the other hand, the *heterogeneous method*, going back to the works by Bourbaki [Bou64], focuses on defining theories that may introduce new primitive notions, and considers truth relative to a theory. The heterogeneous method optimizes reusability by stating every result in the weakest possible theory and using *theory morphisms* to move results between theories in a truth-preserving way. This is often called the little theories approach [FGT92]. Similarly, scientific practice prefers the heterogeneous method. For example, while all mathematics can be reduced to first principles (e.g., using the homogeneous method based on axiomatic set theory), it is usually carried out in highly abstracted settings that hide the foundation. For example, the category of categories is used routinely without focusing on its foundational subtleties. Correspondingly, there is an inconvenient discrepancy between the currently existing formal libraries and the way (informal) mathematics is usually done in practice.

Research Objectives: We want to tackle these interoperability and plurality problems by developing an open archive for formalizations (**OAF**), a ⁹⁸common and open infrastructure for managing and sharing formalized mathematical knowledge such as theories, definitions, and proofs, designed to be scalable with respect to both the size of the knowledge base and the diversity of logical foundations.

Theoretically, the main prerequisite has been established in the LATIN project [KMR09]. The LATIN logical framework [Rab13; Cod+11] integrates institutional representations of model theory and type theoretical representations of proof theory and thus permits combining the benefits of both worlds. A paradigmatic example was published as [HR11]. However, whereas LATIN provides a logical framework, there still remains the problem of integrating the existing formal libraries.

There are two facets of library integration. Firstly, one can *refactor a single library* to increase reuse through modularity, sharing, and inheritance. Secondly, one can *connect or merge two libraries* from different systems. This requires translating the libraries into a common language (namely MMT) and then identifying and eliminating overlap between the two libraries.

My initial focus will be on integrating the specification language PVS [ORS92], with others to follow. PVS is a specification language integrated with support tools and a theorem prover under active development at the Computer Science Laboratory of SRI International.

On this basis, I then want to tackle the problem of refactoring and merging libraries.

Methodology: The aim is to integrate the syntax, underlying foundation and (ultimately) available libraries of different theorem provers into MMT [RK13; HKR12; KRSC11], a uniform foundation-independent representation format for libraries, which allows formalizing the logical foundations alongside the libraries and thus acts as framework for aligning libraries.

Integrating each such library entails four things:

1. Writing an MMT-Plugin, that allows for importing the existing archives into the MMT API,
2. writing an MMT theory, that provides the underlying foundational theory,
3. (potentially) implementing desirable features on the logical framework level (subtyping features, recursive definition principles etc.) to provide syntactical constructors for the specific peculiarities of the theorem prover under consideration and
4. (potentially) adapting and improving the MMT language and API in the process.

In the case of PVS, the particular work will be in translating the inductive/coinductive types, record types and the sophisticated subtyping mechanism that PVS provides into the MMT system and to match the conceptually different module systems of both languages (PVS uses theories and namespaces quite differently than MMT).

Furthermore, I investigate methods for refactoring and integrating/connecting the various theorem prover libraries. Useful notions in that regard are *alignments* [Kal+16] between semantically equivalent symbols across different libraries and foundations, *interface theories* [KRSC11] that abstract from the specifics of a given foundation and *theory intersections* [MK15] for generating interface theories.

Preliminary results: Apart from the preliminary results due to others (mentioned in the previous paragraphs), personal results include:

- A specification of the theorem prover TPS and a translation of its library into MMT (in progress),
- extensions of the logical framework LF to allow for specifying more complex foundational theories with (among others) judgmental and predicate subtypes, an infinite type hierarchy and record types, including a logical framework based on homotopy type theory as a case study [Mmta],
- an almost complete specification of PVS in MMT using the aforementioned LF extensions [Mmtc],
- a corresponding translation of PVS’s Prelude and NASA libraries into the MMT language [Mmtb],
- a survey of different alignment types and a simple language to specify different kinds of alignments [Kal+16],
- a first implementation of an *expression translation* machinery in MMT, that uses alignments and various theory morphisms to translate arbitrary expressions between different formal libraries,

- an implementation of an algorithm for finding alignments between libraries,
- various improvements on the MMT API (parsing, type checking, new language features, interfaces to external databases and applications).

Future work: I am currently working on survey papers on subtyping principles and type hierarchies – these are intended to serve as a starting point for implementing the corresponding features in MMT as generic as possible.

The PVS specification and translation need to be improved with respect to record types, (co-)inductive definition principles and more obscure language features, such as update expressions, to faithfully capture the actual behaviour of the PVS system. Also, I want to additionally import PVS’s proof files, to allow for e.g. exporting and translating proof sketches. The latter will require a generic specification of various proof tactics.

The expression translation and alignment finding algorithms are in an early stage and need to be improved, evaluated as to their usefulness and extended to allow for more complex translations. Furthermore, I want to investigate methods for generating interface theories from alignments and theory morphisms.

Last but not least, more formal systems will be imported into MMT, providing additional challenges for all the presented research objectives.

References

- [Bou64] N. Bourbaki. “Univers”. In: *Séminaire de Géométrie Algébrique du Bois Marie - Théorie des topos et cohomologie étale des schémas*. Springer, 1964, pp. 185–217.
- [Cod+11] M. Codescu et al. “Towards Logical Frameworks in the Heterogeneous Tool Set Hets”. In: *Recent Trends in Algebraic Development Techniques*. Ed. by H. Kreowski and T. Mossakowski. LNCS 7137. Springer, 2011.
- [FGT92] W. Farmer, J. Guttman, and F. Thayer. “Little Theories”. In: *Conference on Automated Deduction*. Ed. by D. Kapur. 1992, pp. 467–581.
- [HKR12] Fulya Horozal, Michael Kohlhase, and Florian Rabe. “Extending MKM Formats at the Statement Level”. In: *Intelligent Computer Mathematics*. Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 65–80. URL: <http://kwarc.info/kohlhase/papers/mkm12-p2s.pdf>.
- [HR11] F. Horozal and F. Rabe. “Representing Model Theory in a Type-Theoretical Logical Framework”. In: *Theoretical Computer Science* 412.37 (2011), pp. 4919–4945.
- [Kal+16] Cezary Kaliszyk et al. “A Standard for Aligning Mathematical Concepts”. Submitted to CICM2016. 2016. URL: <http://kwarc.info/kohlhase/submit/alignments16.pdf>.
- [KMR09] M. Kohlhase, T. Mossakowski, and F. Rabe. *The LATIN Project*. see <https://trac.ondoc.org/LATIN/>. 2009.
- [KRSC11] Michael Kohlhase, Florian Rabe, and Claudio Sacerdoti Coen. “A Foundational View on Integration Problems”. In: *Intelligent Computer Mathematics*. Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 107–122. URL: <http://kwarc.info/kohlhase/papers/cicm11-integration.pdf>.
- [KS10] A. Krauss and A. Schropp. “A Mechanized Translation from Higher-Order Logic to Set Theory”. In: *Interactive Theorem Proving*. Ed. by M. Kaufmann and L. Paulson. Springer, 2010, pp. 323–338.
- [KW10] C. Keller and B. Werner. “Importing HOL Light into Coq”. In: *Interactive Theorem Proving*. Ed. by M. Kaufmann and L. Paulson. Springer, 2010, pp. 307–322.
- [Miz] *Mizar*. <http://www.mizar.org>. 1973–2006. URL: <http://www.mizar.org>.
- [MK15] Dennis Müller and Michael Kohlhase. “Understanding Mathematical Theory Formation via Theory Intersections in MMT”. 2015. URL: http://cicm-conference.org/2015/fm4m/FMM_2015_paper_2.pdf.

- [Mmta] *MathHub MMT/LFX Git Repository*. URL: <http://gl.mathhub.info/MMT/LFX> (visited on 05/15/2015).
- [Mmtb] *MathHub PVS/NASA Git Repository*. URL: <http://gl.mathhub.info/PVS/NASA> (visited on 05/15/2015).
- [Mmtc] *MathHub PVS/Prelude Git Repository*. URL: <http://gl.mathhub.info/PVS/Prelude> (visited on 05/15/2015).
- [NPW02] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Springer, 2002.
- [ORS92] S. Owre, J. Rushby, and N. Shankar. “PVS: A Prototype Verification System”. In: *11th International Conference on Automated Deduction (CADE)*. Ed. by D. Kapur. Springer, 1992, pp. 748–752.
- [Rab13] F. Rabe. “A Logical Framework Combining Model and Proof Theory”. In: *Mathematical Structures in Computer Science* 23.5 (2013), pp. 945–1001.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <http://kwarc.info/frabe/Research/mmt.pdf>.
- [Tea] Coq Development Team. *The Coq Proof Assistant: Reference Manual*. INRIA. URL: <https://coq.inria.fr/refman/>.
- [HOL] HOL4 development team. <http://hol.sourceforge.net/>.

Augmenting Mathematical Formulae for More Effective Querying & Presentation

1 Summary

Scientists and engineers search regularly for well-established mathematical concepts, expressed by mathematical formulae. Conventional search engines focus on keyword based text search today. An analogue approach does not work for mathematical formulae. Knowledge about identifiers alone is not sufficient to derive the semantics of the formula they occur in. Currently, for formula related inquiries the solution is to consult domain experts, which is slow, expensive and non-deterministic.

Consequently, core concepts to enable formula related queries on potentially large datasets are needed. While earlier attempts addressed the problem as a whole, I identify three mutually orthogonal challenges to formula search.

The first challenge, *content augmentation*, is to collect the full semantic information about individual formula from a given input. Most fundamentally, this might start with digitization of analogue mathematical content, captures the conversion from imperative typesetting instructions (i.e. TEX) to declarative layout descriptions (i.e. presentation MathML) but also deals about inferring the syntactical structure of a formula (i.e. the expression tree often represented in content MathML). In addition, this first challenge involves the association of formula metadata such as constraints, identifier definitions, related keywords or substitutions with individual formulae.

The second challenge is *content querying*. This ranges from query formulation, to query processing, actual search, hit ranking to result presentation. There are different forms of formula queries. Standard ad-hoc retrieval queries, where a user defines the information need and the math information retrieval system returns a ranked list given a particular data set. Similar is the interactive formula filter queries, where a user filters a data set interactively until she derives at the result set, which is relevant to her needs. Different are unattended queries that run in the background to assist authors during editing

or readers to identify related work while viewing a certain formula.

The third challenge is *content indexing* for growing data sets. This challenge includes the scalable execution of the solutions to the two aforementioned challenges. While well-established from the area of database systems i.e. XML processing and indexing can be applied, math specific complexity problems require individual solutions.

Augmented content (challenge 1) opens up additional options for similarity search, and potentially improves the search results regardless of the applied similarity measure. In order to separate the effect of content augmentation from intrinsic improvements in the applied similarity features, I develop measures for *formula data quality* that separates those aspects. Afterwards I compare similarity measures given a certain data quality based on that quality measure. The quality measure itself is a valuable contribution. Its use is not limited to search. For example a quality measure can assist authors to check their documents for (1) missing definitions, (2) ambiguities, (3) dependency problems, and (4) redundancy. Note that I focus on quality measure for individual formulae assuming that the relevant meta-information has already been extracted from the surrounding text. Since the developed data quality measures are tailored to my approach of similarity search I'll give more details on the data quality measures after having introduced the similarity factors below.

Math search engines have used some form of similarity measure since the early days of Math search in the 2000'th. However, Youssef and Zhang [1] were the first who branded 5 factors that contribute to similarity measures in July 2014. Those factors are the starting point for my systematic approach to formula similarity measures, which extends their work in the following way:

First, I differentiate between *proper-* and *contextual* formula similarity factors. Proper factors are quantified by applying a distance measure to a

formula-formula pair. I identify three proper factor groups that correspond to *lexical*, *structural* and *semantic* similarity. Typical uses of proper factors are formula clustering, auto correction and completion for formula input while editing, and hit candidate retrieval in the search context. In contrast, contextual formula similarity factors describe the relatedness between a formula *search* pattern and a candidate hit list. Even though, contextual factors are less formal and do not necessarily depend on the concrete search pattern, they play an essential role for Math Information Retrieval Applications, i.e. in search result ranking and cannot therefore be neglected.

In addition, I will describe existing MIR systems using the notation of the discussed similarity factors. This will provide a template, on using formula similarity factors as building blocks for specialized applications or future search MIR systems. Eventually, not all features can be described by the identified factors. In that case I would refine the factor list to ensure that all systems participated in the NTCIR 10 and 11 challenges can be modelled using the factors as building blocks.

Second, I analyze the impact of each individual factor and the inference patterns between different factors. Additionally, I create case studies and templates on how different factors can be combined into use-case specific similarity measures.

The identified factors imply dimensions of the aforementioned formula quality measure. Namely, I define four dimensions of formula quality: (1) typographic and lexical quality; (2) syntactic structure quality; (3) semantic quality; and (4) metadata quality. An example for low syntactic data quality is misinterpretation of $f(a + b)$ as $fa + fb$ rather than f "applied at" $(a + b)$. In a search context this might hinder relevant results from being matched. The associated quality measure for the structural data quality needs to measure to which degree the structure of the formula was captured correctly. Note that the quality measure for contextual information needs to be related to a main unit of possible relevant meta-information.

To assess the similarity measures used by current MIR systems, I will describe existing MIR systems using the uniform similarity factor approach, and perform a comparative evaluation in NTCIR Tasks. This evaluation is twofold and depends on human

relevance judgement for document sections on the one hand, and on known item retrieval, on the other hand. For the impact analysis of individual similarity factors, I was using (1) gold standard driven sensitivity analysis [2] and (2) known item based evaluation [4]. The two metrics discussed above measure not only the performance of MIR systems, but they also evaluate the performance of similarity factors individually. The similarity measures evaluated were taken from existing MIR systems and additional measures proposed in the literature and taken from other disciplines. One result of this evaluation is that some factors, namely, those in the group of proper semantic similarity measures, require a minimum level of data quality to contribute to search result quality in a meaningful way.

By collecting a large and heterogeneous sample of similarity measures, I am confident to have laid a good foundation to evaluate measures that will be developed in the future. I used the existing arXiv corpus for the evaluation. In addition I created, based on the experiences gained with that corpus, an additional corpus from Wikipedia. Since the HTML formats generated in this corpus was obtained automatically from LaTeX or Wikitext, respectively, data quality is not perfect, but I consider it good enough to get qualitative insights about the impact on individual measures. In addition, I use the DLMF/DRMF data-sets which are partially available in different levels of data quality to analyze the impact of data quality on formula search and individual factor effect. I expect to see that the original version of LaTeXML without any content enrichment has the lowest data quality and lowest precision in search result. The DRMF content augmentation process will have raised the data quality and also improved search results. The best results with regard to data quality and search results are expected to be obtained by using the manually generated dataset of DLMF chapter 1-4 by Zhang and Youssef.

The main contribution of this thesis is a systematic analysis of the opportunities and limitations of formula similarity search for context free formula, in dependence of the formula data quality and application scenario. Incorporation of formula unrelated information that is given in the text only, is beyond the scope of this work.

Thesis abstract: “Design and development of a tool based on Coq to write and format mathematical proofs”

Théo Zimmermann Hugo Herbelin (supervisor)

Coq is an interactive proof assistant relying on a foundation language which is both a logical framework and a strongly-typed programming language. It has recently increased in popularity thanks to two ACM prizes and some significant proof developments by George Gonthier and his team. Foundational mathematicians have started to be really interested in Coq, in particular in the links between type theory and homotopy theory. Yet, it is still a tool that requires some “hacker” skills to use.

A general aim of my PhD thesis is to progress along the path of making Coq easier to use for any mathematician. In particular, I wish to make it attractive by transforming it into a tool to write and format mathematical proofs.

Mathematical writing

There has already been a lot of work in this direction, for various proof assistants. We can cite in particular Mizar and its Journal of Formalized Mathematics, and Martijn Oostdijk’s and Yann Coscoy’s PhD theses.

I plan to create a tool to transform proof scripts through a number of steps, each adding constraints to the form the proof script should take. Coq’s tactic language is indeed very flexible and the proof scripts can take many forms and styles. We will target a style that is as close as possible to informal mathematical writing so that the last step, which will be to transform a proof script into English (and LaTeX), is as straight-forward as possible. Preliminary experiments have shown that it is indeed possible to produce such a “mathematical” style in Coq, using its tactic language, unmodified. Additionally, we want to make the last translation (from Coq to English) reversible, so that the generated articles can be re-interpreted and checked by Coq.

Technically, this tool will be based on Arnaud Spiwack’s infrastructure to instrument proof script execution, which is available in Coq 8.5. This will allow transforming the scripts while they are being written, thus providing useful feedback to the mathematician.

On a theoretical level, one of the goals is to understand how transformations between various proof styles maintain the ability to reconstruct the implicit information (information which can be reconstructed by the unification algorithm or some decision or semi-decision procedure).

In this thesis, we restrict ourselves by targeting only arithmetical proofs. However, we also want to make the tool easily extensible, so that other domains can be added. I will evaluate the quality of the tool by using it on some examples of formalization projects and promote its testing by enthusiastic early adopters.

CICM being a community in which this project perfectly fits, I will try to attend each of its meetings where I will strive to publish and present my progress.

Internship work continued: Theorem transfer

I had previously started working on ideas which make proof formalization closer to pen-and-paper mathematics during an internship with Hugo Herbelin, now my PhD supervisor. The main goal of the internship was to help reason modulo isomorphism. The work I did then was to devise and implement an algorithm to automatically transfer theorems between two related types, given the right kind of user-provided declarations.

I generalized upon the work of Matthieu Sozeau, on generalized rewriting, and of Cyril Cohen, et al., on type refinements, and obtained a set of inference rules which allow the transfer to take place. I presented this work at CICM 2015 in the work-in-progress track. The anonymous referees then pointed me to other works on the topic, in the context of Isabelle, of which I had not yet been aware. The ideas behind Isabelle's Transfer package, which are described in great detail in Ondřej Kunčar's PhD thesis, are very close to my own, which, while validating the path I had taken, also make it harder to publish my own work as novel.

Since the conference, I have finished implementing a prototype of my transfer method. The implementation is a small library¹, relying on Coq's type class mechanism. It is able to transfer all sorts of theorems, including theorems about predicates, such as induction principles. Following my work and Jérémie Koenig's binary logical relation library, Matthieu Sozeau is now, with my help, extending once more the rewriting capabilities of Coq to include rewriting with heterogeneous binary relations (relations between elements of two distinct types). I foresee that theorem transfer will then become a special case of generalized rewriting, benefiting at the same time from a more robust implementation.

¹This library is available at <https://github.com/Zimmi48/transfer>.

DML Work In Progress

Mathematicians dream of a digital archive containing all peer-reviewed mathematical literature ever published, properly linked, validated and verified. It is estimated that the entire corpus of mathematical knowledge published over the centuries does not exceed 100,000,000 pages, an amount easily manageable by current information technologies.

The track objective is to provide a forum for the development of math-aware technologies, standards, algorithms, and formats that can lead towards fulfilling the dream of a global Digital Mathematical Library (DML). The DML track also serves as an interdisciplinary venue to share experience and best practices among projects in digital libraries, natural language processing, optical character recognition, pattern recognition, information retrieval, and other areas that could change the paradigm for creating, storing, preserving, searching, and interacting with a mathematical corpus.

July 2016, Frank Tompa

swMATH - Challenges, Next Steps, and Outlook

Hagen Chrapary^{1,2}, Wolfgang Dalitz², and Wolfram Sperber¹

¹ FIZ Karlsruhe/zbMATH, Franklinstr. 11, 10587 Berlin, Germany

² Zuse Institute Berlin (ZIB), Takustr. 7, 14195 Berlin, Germany

Abstract. swMATH [?] is currently one of the most comprehensive specialized information services and search engines for Mathematical SoftWare (MSW). It was the intention of the project to support the user community by providing information about MSW and searching relevant MSW. Currently swMATH lists information of more than 13,000 items and 120,000 publications which refer to MSW. Maintaining and updating of the service is mainly done automatically, the number of requests is permanently increasing. This sounds like a perfect solution. But swMATH is only a first step to a powerful information system on mathematical research data. This talk addresses some open problems for the further development of the swMATH service. It is shown that some problems for swMATH lead to central questions for the information management of mathematical research data, especially for MSW. This contains an extended content analysis of MSW, versioning and citation standard of MSW, a typing of the swMATH resources and the presentation of context information and high-quality control within the swMATH service.

1 Introduction

The advent of computers has led to the development of a new mathematical branch: MSW. MSW has some unique features which are different from the classical form of the presentation of mathematical knowledge, the publications

- MSW is written in a special programming language, not in a natural language as publications. The source code of a MSW is not self-explaining and often inappropriate for human's reading and automatic content analysis.
- MSW depends on hard- and software, programming languages or environments, etc.
- For human users of MSW, special forms of instructions (documentations, manuals, etc.) have been established. This information is often provided on own Web sites of a MSW.
- MSW realizes a mathematical concept or an algorithm for solving special classes of mathematical models ("computational model") which can be presented independently from the MSW.
- Typically, MSW has instances, e.g. versions and develops dynamically. Moreover, MSW has a life cycle. This requires a version management of the MSW (in the past, mathematical publications were static).
- The dynamic character of MSW has led to the development of repositories for maintaining and providing MSW. There are some commercial companies but also a lot of developer communities, e.g. the *R* community, who run own repositories.

- The increasing use of MSW in mathematics and other scientific fields, industry, services, and administration requires an exact citing of the MSW. Results without exact references of the used MSW instance significantly complicate the reproduction of the results.
- Standardized data models for the description of and a citation standard for MSW are missing. Metadata schemes for publications cannot be transferred one-to-one for MSW.
- Methods for the quality control of MSW are under development. Some specialized journals on MSW were founded. Typically, publications on Open Source MSW consist of a text-based publication describing the algorithm or the MSW supplemented by the software code and testdata. The quality control of these publication contains also a peer-reviewing of the MSW, typically an installation of the MSW and a validation of the results.

The demands on information services for MSW are diverse, especially search, access, version management, and archiving. This has led to the development of different kinds of information services for MSW: directories which list MSW and provide information about it, repositories, which maintain and archive MSW for a special subject and programming language, and individual websites for a MSW or specialized mathematical services.

The swMATH service is a directory for MSW. There are a lot of other directories for MSW. The manual maintenance is expensive. Therefore we have developed a concept for a machine-based concept.

2 swMATH in a nutshell

A central concept of swMATH is the so-called publication-based approach. MSW is often accompanied by publications. By analyzing the mathematical literature, MSW is identified and information about it is extracted. Instead of analyzing the full texts of publications, the information of the database zbMATH [?] is used, especially titles, reviews or abstracts and references. Software citations in mathematical publications are often short and restricted to the name of the MSW, e.g. "SAGE", without precise information about instances, e.g. versions. Often an exact information on the type of the resource is missing. Sometimes also information which is closely related to MSW but not MSW itself is detected, e.g. services, languages and environments, testdata and benchmarks, etc. We have decided to insert also these kinds of information into swMATH. In the following, all entries in swMATH are labeled as MSW.

The heuristic analysis starts with searching for characteristic patterns for MSW. Each publication which has a link to a MSW is classified either as "standard publication" if a MSW is the main content of the publication or as "user publication" if the MSW is used as a tool for calculation. As said above, publications and also zbMATH data often don't contain information about different instances of a MSW. Hence there can exist more than one standard publication (a standard publication can exist for each instance of a MSW). A citation standard for MWS would significantly improve the situation. Some questions on a citation standard will be discussed in detail below.

But the analysis of the publications offers great potential to gain relevant information about MSW. This is done in a second step. Currently the main information in swMATH contains the following information (metadata) on MSW:

- the name of the MSW
- a short description of the content of the MSW. This description uses the review or the abstract of a standard publication of the MSW and provides information about the content of the MSW addressing such questions ‘Which problems can be solved using MSW?, Which algorithms are realized by the MSW?, What are the special features of the MSW? .:
- a keyword cloud which aggregates the keywords of standard and user publications
- classification codes of the standard and user publications correspondingly to the Mathematics Subject Classification 2010 (MSC2010) [?]
- a list of publications which cites the MSW
- a list of “related software” is calculated by the data of publications
- a profile of the acceptance of the MSW measured by the number of publications per year

Moreover, a simple and an advanced search feature over the swMATH entries is provided. The number of references of MSW is used for a ranking of information. For example, the importance of a keyword in the keyword cloud results from its frequencies, the MSC codes provide relevant information about the mathematical subjects and the applications areas. The number of references in peer-reviewed publications can be also interpreted as an indicator for the quality of MSW.

Publications contain not all information about a MSW. Typically, the Web sites of a MSW are excellent resources for specific information, e.g. they contain manuals or documentations, provide technical specifications, legal rights, etc. By a websearch the URL of the website of a MSW – if existing – is determined. We are working on more specific methods for the automatic analysis of the websites of MSW. We have started to identify manuals and documentations of MSW on the websites.

Figure 1 shows the swMATH entry for the MSW “Singular” .

Commonly, websites of a MSW contain more or less information about the current instances of a website. But the publications refer not only to a current instance of a MSW but also to outdated instances. Web archives which periodically store the Web could enrich the information about the different stages of a MSW and help to assign the publications to the corresponding instances. Therefore the swMATH directory of swMATH will be used as a seed list for to search information about MSW. It is the plan to combine this information with the existing information in swMATH.

All swMATH entries (families of MSW) get a persistent identifier, the running number of the swMATH input of the MSW. The identifier could be extended by adding data of an instance.

The publication-based approach and the use of the zbMATH database allows a widely automatic maintaining and updating of swMATH. The publication-based approach also ensures the presentation of MSW in the context with the mathematical literature.

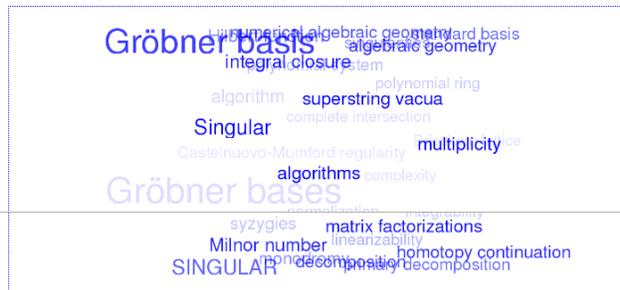
swMATH is an open access service for the mathematical community.

SINGULAR

SINGULAR is a Computer Algebra system (CAS) for polynomial computations in commutative algebra, algebraic geometry, and singularity theory. SINGULAR's main computational objects are ideals and modules over a large variety of baserings. The baserings are polynomial rings over a field (e.g., finite fields, the rationals, floats, algebraic extensions, transcendental extensions), or localizations thereof, or quotient rings with respect to an ideal. SINGULAR features fast and general implementations for computing Groebner and standard bases, including e.g. Buchberger's algorithm and Mora's Tangent Cone algorithm. Furthermore, it provides polynomial factorizations, resultant, characteristic set and gcd computations, syzygy and free-resolution computations, and many more related functionalities. Based on an easy-to-use interactive shell and a C-like programming language, SINGULAR's internal functionality is augmented and user-extendible by libraries written in the SINGULAR programming language. A general and efficient implementation of communication links allows SINGULAR to make its functionality available to other programs.

This software is also referenced in ORMS.

Keywords for this software



URL: www.singular.uni-kl.de
Manual: www.singular.uni-kl.de...
Authors: Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, Hans Schönemann
Platforms: ix86-Linux, SunOS-5, IRIX-6, ix86-Win (runs on Windows 95/98/NT4/2000/XP/Vista), FreeBSD, MacOS X, x86_64-Linux (AMD64/Opteron/EM64T), IA64-Linux
Licence: free and open-source under the GNU General Public Licence.

[Add information on this software.](#)

Related software:

- Macaulay2
- CoCoA
- Magma
- Maple
- primdec
- Sage
- Plural
- FGb
- GAP
- Risa/Asir

[Show more...](#)

References in zbMATH (referenced in 802 articles, 4 standard articles)

Showing results 1 to 20 of 802.

Sorted by year (citations)

1 2 3 ... 39 40 41 next

1. Bivía-Ausina, Carles; Fukui, Toshizumi: Mixed Lojasiewicz exponents and log canonical thresholds of ideals (2016)
2. Bobol, Nicolás; Dickenstein, Alicia: Implicitization of rational hypersurfaces via linear syzygies: a practical overview (2016)
3. Dimca, Alexandru; Sticlaru, Gabriel: Syzygies of Jacobian ideals and weighted homogeneous singularities (2016)
4. Dumnicki, M.; Farnik, L.; Głowska, A.; Lampa-Baczyńska, M.; Malara, G.; Szeemberg, T.; Szpond, J.; Tutaj-Gasińska, H.: Line arrangements with the maximal number of triple points (2016)
5. Dumnicki, M.; Szeemberg, T.; Tutaj-Gasińska, H.: Symbolic powers of planar point configurations. II. (2016)
6. Ellis, Graham: Cohomological periodicities of crystallographic groups. (2016)
7. Eröcal, Burçin; Motsak, Oleksandr; Schreyer, Frank-Olaf; Steenpaß, Andreas: Refined algorithms to compute syzygies (2016)
8. Ferčec, Brigita; Giné, Jaume; Romanovski, Valery G.; Edneral, Victor F.: Integrability of complex planar systems with homogeneous nonlinearities (2016)
9. Giesbrecht, Mark; Heinle, Albert; Levandovskyy, Viktor: Factoring linear partial differential operators in n variables (2016)
10. Giné, Jaume; Valls, Claudia: Center problem in the center manifold for quadratic differential systems in $\mathbb{m}at_{\mathbb{C}}\mathbb{R}^3$ (2016)
11. Hausen, Jürgen; Keicher, Simon; Laface, Antonio: Computing Cox rings (2016)
12. Marats, Magdaleen S.; Steenpaß, Andreas: The classification of real singularities using S ingular. II: The structure of the equivalence classes of the unimodal singularities. (2016)
13. Marcolla, Chiara; Pellegrini, Marco; Sala, Massimiliano: On the small-weight codewords of some Hermitian codes (2016)
14. Margulies, S.; Morton, J.: Polynomial-time solvable #CSP problems via algebraic models and Pfaffian circuits (2016)
15. Ma, Yue; Wang, Chu; Zhi, Lihong: A certificate for semidefinite relaxations in computing positive-dimensional real radical ideals (2016)
16. Rollenske, Sönke: A new irreducible component of the moduli space of stable Godeaux surfaces (2016)
17. Adamus, Janusz; Seyedinjad, Hadi: A fast flatness testing criterion in characteristic zero (2015)
18. Atzal, Deeba; Pfister, Gerhard: A classifier for simple isolated complete intersection singularities (2015)
19. Albert, Mario; Fetzer, Matthias; Sáenz-de-Cabezón, Eduardo; Seller, Werner M.: On the free resolution induced by a Pommaret basis (2015)
20. Balekoročau, W.; Ma'u, S.: Chebyshev constants, transfinite diameter, and computation on complex algebraic curves (2015)

1 2 3 ... 39 40 41 next

Further publications can be found at: <http://www.singular.uni-kl.de/index.php/publications/singular-related-publications.html>

Article statistics & filter:

Search for articles

MSC classification

- Top MSC classes
- 13 Commutative algebra
- 14 Algebraic geometry
- 32 Functions of several...
- 34 Ordinary differential...
- 68 Computer science
- Other MSC classes

Publication year

- 2010 - today
- 2005 - 2009
- 2000 - 2004
- before 2000

Chart: cumulative / absolute

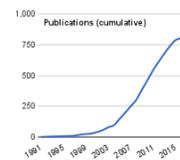


Fig. 1. The detailed information for the software "Singular"

3 Challenges

3.1 MSW instances and citation standard for MSW

The analysis of mathematical publications seems to be an efficient way to search for MSW. But the use of heuristic methods can fail, e.g. if the name is a word from the natural language. Also, the heuristic methods don't allow to determine instances. A standardized referencing of MSW would have important advantages: a precise information about MSW used, better visibility of the MSW, reproducibility of research results, secure identifiability of the MSW, etc.

Software is ubiquitous on computers. A precise citing of the used instance of a software is a precondition for reproducing the results which were achieved by using a software. Different instances of a software can be incompatible, are derived from other algorithms and provide other features. Software has other characteristics as publications, especially its dynamic character. Citing of software is a widely discussed topic in different communities, see the general discussion Mike Jackson's blog [?] "How to cite and describe software". The standardization of citations has different aspects. MSW is distributed in various different ways which has an influence on a citation standard. Some publishers and software providers have developed and implemented own models and recommendations for citing its MSW. A main requirement to a citation standard for MSW is that the citation standard must guarantee the unique identifiability of the software used, typically such a reference must refer the instance of the software used. A MSW can have instances, e.g. versions or revision numbers. Instances can characterize the state of the art of a MSW, dependencies from operating systems, programming languages, or other software, or different legal rights and usability conditions. MSW can be accessible via a repository and explicitly defined versions are missing, here timestamps or hash tags could be helpful information for the verification of the used MSW instance.

The current situation is characterized by proprietary standards and recommendations for the citation of software. A widely used approach for the citation standard is the "Who – When – What – Where" -format, propagated, e.g. by the American Psychological Association. This approach is derived from the citation standard for publications. In principle, this recommendation can be used but the details of this recommendations are not specified.

The *R* community, a community-driven initiative in mathematics statistics, recommends to cite the corresponding manuals - if existing - instead of directly referencing the MSW. Typically, there is an 1:1 relation between a manual and an instance of a MSW. But, the link to the software cannot be generated from the manual. For the MSW based on *R* this is not important because the CRAN repository allows a fast access to the *R* packages.

Jackson has published some recommendations which contain necessary information. His general hint: Publications on a software should be cited as a complement not as a replacement for citing software.

He distinguishes between the following four types for referencing:

- software purchased off-the-shelf which can be described by the following data:
Product name; Version; Release date; Publisher; Location

- software downloaded from the Web which can be described by the following data:
Product name; Version; Release date; Publisher; Location; DOI or URL; Download date
- software checked out from a public repository which can be described by the following data:
Product name; Publisher; Location; URL; Repository specific checkout information
- software purchased by a researcher which can be described by the following data:
Product name; Author; Location; Contact details; Received date

These data describe the used software in a unique way. But the proposal requires to apply different metadata schemes which hampers the acceptance and use. Further information especially operating systems or used packages or libraries could be optional parts of the citation standard.

A further problem of a citation standard of MSW is its implementation. For the mathematical community, $\text{T}_{\text{E}}\text{X}$ is the lingua franca to write mathematical documents. References are done in $\text{BibL}_{\text{A}}\text{T}_{\text{E}}\text{X}[?]$ which has widely substituted the outdated $\text{BibT}_{\text{E}}\text{X}$ software. $\text{BibL}_{\text{A}}\text{T}_{\text{E}}\text{X}$ is more flexible than $\text{BibT}_{\text{E}}\text{X}$ and allows to define new types and corresponding metadata schemes but doesn't support a type "software" until now. This type must be specified, mandatory and optional fields describing the MSW must be defined and the backend software. A first $\text{BibL}_{\text{A}}\text{T}_{\text{E}}\text{X}$ proposal has been created with "Biber" as backend software.

Remark: The problem of a citation standard for MSW should be discussed more generally for all types of mathematical research data, not only for MSW. We discuss this problem in more detail in the following section.

3.2 Context, resource types, and classification of swMATH entries

The publication-based approach described above links a MSW to the publications which cite it. The publications contain data about the investigated problems and models, their mathematical treatment, data of measurements which are used for the input, parameters, etc. But often these data are incomplete, not explicitly linked to MSW (e.g. which model is solved by the MSW?) which makes it difficult to reproduce and reuse scientific results.

The mathematical subject, concepts, models, and methods as the application of mathematics are very heterogeneous for MSW. A classification of mathematical research data by its subject and type would be useful for retrieval and navigation. It allows to present MSW in its context, especially the relation to the mathematical models which are solved by using MSW, the algorithm, the used data and parameters for calculations.

A comprehensive and structured content analysis starting with the original mathematical models, the parameter and input data, the computational algorithms and the computational models used for MSW as well as the computational results would make the research process more transparent and improve the reproducibility of results.

Subject classification in swMATH

The MSC2010 is the most accepted subject classification scheme for mathematical publications. The scheme covers pure mathematics as well as a broad spectrum of application areas. For publications with the focus on MSW a special classification code XX-04 for the most of the 63 top classes of mathematical subject in the MSC is provided. Last but not least all zbMATH entries are classified by the MSC.

The publication-based approach of swMATH suggested to use the MSC also for the characterization of the mathematical subjects of a MSW. Of course, other classification schemes, e.g. the Guide to Available Mathematical Software [?] index could be added.

Type classification in swMATH

Currently, swMATH contains different types of mathematical data with the focus on MSW. Currently, entries of the following types of objects are inside swMATH:

- MSW
A MSW could be defined as source code, computer programs, or software libraries for analyzing or calculating mathematical problems or models
- Mathematical languages and environments for MSW
Languages and environments are programming languages for special mathematical subjects, e.g. the language and environment *R* [?] for statistics.
- Mathematical services
The common property of mathematical services is the presentation of information. The information can result from retrieval in databases or from online computing or a mix of both. The Online Encyclopedia of Integer Sequences or the Digital Library of Mathematical Functions are impressive examples of widely used mathematical services.
- Testdata and benchmarks
Testdata and benchmarks are important for the verification of results of the use of MSW, to test its performance and to decide which MSW is used for solving a mathematical problem.
- Further mathematical research data, especially models and model parameters, visualizations and simulations
This class is very heterogeneous and covers numerous model parameters but also complex mathematical objects and models, e.g. polynomials or ideals for computer algebra.

This first approach for classifying the types is resulting from a pragmatic point of view on the swMATH entries.

But, the scheme reveals also the difficulties of a classification scheme for mathematical research data: The classes are characterized by different properties. Different classes can strongly overlap. The swMATH entries can be elements of different classes. On the other hand, a classification of the resource types for the swMATH entries is useful, e.g. for the presentation of related software. The feature "Related software" is important for the users of swMATH. It provides an overview about similar swMATH entries. Up to now the similar entries are determined by common references in publications and common MSC codes. But if we don't want to compare apples to oranges,

e.g. a MSW to a benchmark, we need a typing of the swMATH entries. The resource type scheme allows a structured presentation of the environment of a MSW, not only of similar MSW but also of related benchmarks, models etc.

As usual we propose a hierarchic classification scheme for the types in swMATH. All these classes can be subdivided. Up to now, a first flat type scheme is implemented in the production database of swMATH.

The typing could be presented as an additional facet in the UI of swMATH.

3.3 Applied specific information in swMATH

Here we restrict ourselves to two aspects. At first we discuss the enrichment of the existing information of application aspects in swMATH. Currently, information which is of special interest for users outside of mathematics is included in the MSC classification, especially the MSC codes for the application areas, and in the keyword cloud by application area-specific keywords. But there is no explicit tagging of mathematical and application-specific keywords. Moreover, keywords and a classification codes of the application areas are not mandatory. Typically, non-mathematicians actively participate in the modeling and the evaluation process. They are especially interested to get detailed information about the models, MSW and the results of the calculations. This information would help the users to decide which models and MSW could be suitable for solving their problems. Publications in applied mathematics often contain a description of the original problem and the mathematical models. Heuristic methods could be developed for searching specific information for users in the publications which are listed in the swMATH entries, especially for models.

A more general aspect is the ambiguous borderline between MSW and non-mathematical software. The publication-based approach in swMATH is restricted to publications which are reviewed by zbMATH. swMATH provides an interface to inform about MSW outside of the scope of zbMATH.

3.4 Quality of MSW

Quality measures of MSW are difficult. The quality control of MSW cannot be reduced to a single factor. It is a complex process. Up to now, swMATH doesn't make any statement about the quality of the MSW. References to a MSW within the zbMATH data are a necessary condition for the inclusion of the MSW to the database swMATH. Reference to a MSW in a peer-reviewed publication can be interpreted as an indirect quality measure for the cited MSW. The number of references is also an indicator for the quality. In the last 20 years, special journals with the focus on scientific software were founded, in mathematics especially the journals "ACM Transactions on Mathematical Software" [?], "Archive of Numerical Software" [?], "Journal of Machine Learning Research" [?], "Journal of Software for Algebra and Geometry" [?], "Journal of Statistical Software" [?], "Mathematical Programming Computation" [?], "Numerical Algorithms" [?], or "The R Journal" [?]. For a complete list of journals with the focus on software see Neil Chue Hong's blog [?] "In which journals should I publish my software". A typical publication in these journals consists of an article which describes the algorithm and/or the MSW and the code of the MSW. All these journals have defined

explicit more or less standardized policies for submission and peer reviewing which involve also the software code. The policies include the installation and feasibility of the MSW for the peer-reviewing process and accepted dependencies from other software and programming languages. The verification of the results which were achieved by using the MSW is a further important criteria for the quality control. Therefore, the submissions must include the original testdata. Of course, the evaluation of MSW articles is more expensive than this of a traditional publication. Some journals have technical editors which are responsible for the reviewing of MSW.

Quality-controlled MSW, especially the one which is published in the journals mentioned above could be labeled in swMATH. This would strengthen trust into the MSW, increase the usability of zbMATH, and be of great value for the user community.

3.5 Further problems

Long-term archiving of MSW is an open problem. It requires an archiving of the different software instances but also of the complete environment. The swMATH service plans to check Internet archives to provide information of the lifecycle of MSW and of outdated MSW. Also the linking of swMATH entries in swMATH with the software code is on the agenda. The granularity of MSW is a further problem. There exists very complex and powerful MSW which can be used for solving different classes of problems but also small MSW which is focused on solving a special problem. Until now there are no standardized parameters which characterize the size of MSW. Often complex MSW, e.g. Mathematica, Matlab, or R have a substructure. Packages are MSW entries with the focus on special classes of problems. swMATH has to define a policy for the handling of complex MSW. To improve the actuality of swMATH we plan to extend the publication-based approach of swMATH can be to other resources, especially identifying MSW in the e-prints of ArXiv [?].

4 Final remarks

Mathematical research data and a qualified access are useful for the mathematical community and outside users. swMATH is a new information service for mathematical research data especially for MSW. It was shown that heuristic methods can be successfully used for content analysis and the maintenance of swMATH. Moreover the development of the swMATH service has caused discussions about standards and classification. We hope that swMATH can contribute to build up a lean and efficient infrastructure for mathematical information.

References

1. The database swMATH (2012 -), <http://www.swmath.org>
2. The database zbMATH (1868 -), <http://www.zbmath.org>
3. The Mathematics Subject Classification 2010, <http://www.msc2010.org>
4. Netlib (1987 -), <http://www.netlib.org>
5. CRAN, <https://cran.r-project.org/web/packages/>

6. Decision Tree for Optimization Software (Plato), <http://plato.asu.edu/guide.html>
7. A portal for computer algebra and symbolic mathematics, <http://mathres.kevius.com/comal.html>
8. Gert-Martin Greuel, Wolfram Sperber, swMATH – an information service for mathematical software, in: Hong, Hoon (ed.) et al., Mathematical software – ICMS 2014. 4th International Congress, Seoul, South Korea, August 5–9, 2014. Proceedings. Springer. Lecture Notes in Computer Science 8592 (2014) , 691-701, http://www.mathematik.uni-kl.de/~greuel/Paper/GeneralType/2014-ICMS_swmath.pdf swMATH, ICMS2014
9. Mike Jackson, How to cite and describe software?, <http://www.software.ac.uk/how-cite-and-describe-software?mpw>
10. BibL^AT_EX, <https://www.ctan.org/pkg/biblatex>
11. Guide to Available Mathematical Software (GAMS), <http://gams.nist.gov/>
12. The R Project for Statistical Computing, <https://www.r-project.org/>
13. ACM Transactions on Mathematical Software (TOMS) (1975 -), <http://toms.acm.org/>
14. Archive of Numerical Software (2013 -), <https://journals.uni-heidelberg.de/index.php/ans/index>
15. Journal of Machine Learning Research (2003 -), <http://www.jmlr.org/>
16. Journal of Software for Algebra and Geometry (2009 -), <http://msp.org/jsag/about/journal/about.html>
17. Journal of Statistical Software (1996 -), <https://www.jstatsoft.org/index>
18. Mathematical Programming Computation (2009 -), <http://mpc.zib.de/>
19. Numerical Algorithms (1991 -), <http://www.springer.com/computer/theoretical+computer+science/journal/11075>
20. The R Journal (2009 -), <https://journal.r-project.org/>
21. Neil Chue Hong, In which journals should I publish my software? <http://www.software.ac.uk/resources/guides/which-journals-should-i-publish-my-software>
22. arXiv.org e-Print archive, <http://arxiv.org/>

Formalization of Polynomially Bounded and Negligible Functions Using the Computer-Aided Proof-Checking System Mizar

Hiroyuki Okazaki and Yuich Futa

Graduate School of Science and Technology, Shinshu University
4-17-1 Wakasato Nagano-city, Nagano 380-8553, Japan
Japan Advanced Institute of Science and Technology
`okazaki@cs.shinshu-u.ac.jp`

Abstract. In recent years, formal verification applications have attracted significant attention. In particular, verification of the security of cryptosystems has been investigated extensively. In this study, we attempt to develop various mathematical libraries for cryptology using the Mizar proof checking system. Polynomially bounded and negligible functions play very important roles in cryptology. Therefore, we introduce formalized definitions of polynomially bounded and negligible functions for formalizing cryptology in Mizar.

Keywords: polynomially bounded functions, negligible functions

1 Introduction

In recent years, formal verification applications have attracted significant attention. Verification of the security of cryptosystems is has been investigated extensively. The objective of this study is to develop mathematical libraries to verify the security of cryptographic systems using the Mizar proof checking system [1]. Mizar is one of the best-known projects in formalized mathematical knowledge management. To achieve our objective, we attempt to formalize various topics related to cryptology, such as computational complexity, probability and probability distributions [2–7], algorithms [8], and number theory [9, 10]. The security of most current public-key cryptosystems is based on computational complexity. A public-key cryptosystem is secure if attacks against the cryptosystem are not easier than solving a problem for which there is no known efficient algorithm. Discrete-logarithm and factorization problems are considered to be hard to solve and are therefore employed in public-key cryptosystems [11, 12]. The difficulty of such problems is classified according to their computational time.

Discrete-logarithm and factorization are subexponential-time problems that are difficult to solve if the input size is sufficiently large. It is well known that polynomial-time problems can be solved with a relatively low computational cost. We consider a problem P_a to be as hard to solve as another problem P_b if we can solve P_b by calling the algorithm that can solve P_a a polynomially

bounded number of times. We are interested in the formalization of computational complexities. Polynomially bounded functions play an important role in practical computational complexity theory. The class P is a fundamental computational complexity class that contains all polynomial-time decision problems [13]. It takes a polynomially bounded amount of computation time to solve polynomial-time decision problems using a deterministic Turing machine. Therefore, we formalize a rigorous definition of polynomially bounded functions.

To analyze the security of cryptographic primitives such as symmetric encryption or hash functions as well as that of cryptographic protocols, we evaluate the success probability of any attack against them. In general, a cryptosystem is secure if the probability of any attack succeeding against the cryptosystem is negligible. Therefore, we formalize a rigorous definition of the negligible functions [14]. Both polynomially bounded and negligible functions are described by the behavior of polynomials. Polynomially bounded functions do not increase faster than polynomials themselves, and negligible functions do not decrease slower than the multiplicative inverse of polynomials. There are interesting relationships between polynomially bounded and negligible functions.

In this paper, we first introduce our formalized definition of polynomially bounded functions. Next, we formalize the definition of negligible functions and prove that the set of negligible functions is a subset of the set of polynomially bounded functions. Moreover, we introduce an equivalence relationship between polynomially bounded functions using negligible functions. All formalized definitions and theorems in this paper are described in the Mizar language, and their proofs are verified by the Mizar proof checker. Our formal descriptions have been stored in the current version of the Mizar Mathematical Library (MML)* and are available online on the Mizar Project official website [1].

1.1 Our contribution

Asymptotic notations allow us to describe the behavior of given real-valued functions. In particular, O notation is often used to analyze the increase in real-valued functions and to classify the computational complexity of solving the given problems. Fortunately, there are formal definitions and theorems regarding asymptotic notation in the MML. Therefore, we can use these definitions to offer a formalized definition of polynomially bounded functions that do not increase faster than polynomials. We then formalize some theorems regarding polynomially bounded functions. Next, we formalize the definition of negligible functions that do not decrease slower than the multiplicative inverses of polynomials and formalize some theorems about them. It is well known that linear combinations of polynomially bounded functions are also polynomially bounded and that those of negligible functions are also negligible. These facts are very useful for evaluating computational complexity and for proving the security of cryptosystems. Thus, we formalize proofs for these theorems. Specifically, we

* Indexed by the MML identifiers ASYMPT_2[15] and ASYMPT_3[16], for Mizar version: 8.1.04 5.32.1246 or later.

formalize the algebraic structure of polynomially bounded functions. We then introduce an equivalence between the elements of the algebra of polynomially bounded functions using negligible functions.

2 Mizar

Mizar [1] is one of the best-known projects in formalized mathematical knowledge management. The Mizar system comprises the Mizar language, an automated proof verifier, and mathematical libraries. The Mizar language normally uses first-order predicate logic; however, it can use second-order predicate logic with conditions.

3 Preparation

3.1 Asymptotic notation

In this section, we briefly review the asymptotic \mathcal{O} notation and introduce related formal definitions available in the MML. For our purposes (i.e., computer science and cryptology) it is sufficient to analyze the behavior of discrete functions. In fact, asymptotic notation in the MML is formalized on sequences, which are synonymous with functions whose domains comprise natural numbers.

Definition: 1 *Let $f(\cdot)$ and $g(\cdot)$ be functions from \mathbb{N} to \mathbb{R} . $g(\cdot) \in \mathcal{O}(f(\cdot))$ iff $\exists N$ is a natural number and c is a real number s.t., $\forall n$ s.t. $N \leq n$ holds $0 \leq g(n) \leq c \cdot f(n)$.*

The \mathcal{O} notation is defined in Mizar as follows:

Definition: 2 (ASYMPT_0:def 9)
 definition
 let f be
 eventually-nonnegative Real_Sequence;
 func Big_Oh(f) -> FUNCTION_DOMAIN of NAT,REAL
 equals
 { t where t is Element of Funcs(NAT, REAL)
 : ex c,N st c > 0 &
 for n st n >= N holds t.n <= c*f.n &
 t.n >= 0};
 end;

In Mizar, Real_Sequence is an alias for a function from \mathbb{N} to \mathbb{R} and “eventually-nonnegative” is an attribute for Real_Sequence and is defined as follows:

Definition: 3 (ASYMPT_0:def 2)
 definition
 let f be Real_Sequence;
 attr f is eventually-nonnegative means

```

ex N being Nat st for n being Nat
st n >= N holds f.n >= 0;
end;

```

The sequence of the monomial $f(n) = n^a$ is defined in Mizar as follows:

Definition: 4 (ASYMPT_1:def 3)
definition
let a be Real;
func seq_n^(a) -> Real_Sequence means
it.0 = 0 &
for n st n > 0 holds
it.n = n to_power a;
end;

The sequence of the exponential $a^{b \cdot n + c}$, where a, b , and c are given constant real numbers, is defined in Mizar as follows:

Definition: 5 (ASYMPT_1:def 1)
definition
let a,b,c be Real;
func seq_a^(a,b,c) -> Real_Sequence means
it.n = a to_power (b*n+c);
end;

Note that for a given real number x , $f(n) = x^n$ is represented as “seq_a^(x,1,0)” in Mizar.

3.2 Polynomially bounded functions

In this section, we briefly review polynomially bounded functions.

Definition: 6 *A real valued function, $f(\cdot)$, is polynomially bounded if there exists a natural number, k , such that*

$$f(x) \in \mathcal{O}(x^k)$$

3.3 Negligible functions

In this section, we briefly review negligible functions[14].

Definition: 7 *Let $\mu(\cdot)$ be a function from \mathbb{N} to \mathbb{R} . $\mu(\cdot)$ is a (negligible function) iff for all polynomial $p(\cdot)$ holds $\exists N$ be a natural number s.t., $\forall n$ be a natural number s.t. $N \leq n$ holds*

$$\mu(n) < \frac{1}{|p(n)|}$$

4 Formalization of polynomially bounded functions and polynomial functions

In this section, we introduce our formal definitions of polynomially bounded functions and polynomial functions.

4.1 Formalized definition of polynomially bounded functions

We propose a formal definition of polynomially bounded functions as follows:

Definition: 8 (ASYMPT_2:def 1)
 definition
 let p be Real_Sequence;
 attr p is polynomially-bounded means
 ex k be Nat st
 p in Big_Oh(seq_n^(k));
 end;

This definition may appear very similar to Def. 6 but is only a formalization of an attribute for real sequences. In the Mizar language, attributes are constructors of adjectives. We can define an attribute followed by the keyword “attr”. Def. 6 gives only a definition of an attribute for real-valued functions. However, formal methods never accept the declaration of new objects without an evidence of their existence in order to eliminate any errors in proofs. In the Mizar system, there are several ways to treat polynomially bounded functions. One is to prove that declared real sequences satisfy the definition of being polynomially bounded. Another way is to prove the existence of polynomially bounded functions and to define a new mode for polynomially bounded sequences as a sub mode of a “Real_Sequence”. It should be noted that in the Mizar language, “mode” is similar to “type” used in other languages. Furthermore, all modes are derived from the mode “set” because the Mizar system is based on the set theory. The third option is to define a set of specific functions and their algebraic structure as new terms. We will show an example of the third case in Sec. 6.

Let us now introduce some theorems about polynomially bounded functions.

Theorem: 1 (ASYMPT_2:25)
 theorem
 for a be Nat st 1 < a holds
 seq_a^(a,1,0) is non polynomially-bounded;

This theorem shows that the exponential $f(x) = a^x$ is not polynomially bounded if $1 < a$. We first prove Theorem 2 as a lemma for proving Theorem 1. The following theorem holds that 2^x is nonpolynomially bounded:

Theorem: 2 (ASYMPT_2:16)
 theorem
 for x be Nat st 1 < x holds

```

not ex N,c be Nat st
for n be Nat st N <= n holds
2 to_power n <= c * (n to_power x);

```

To prove Theorem 2, we assume that the proposition of Theorem 1 is not true and derive a contradiction against Theorem 2. The actual formal proof of Theorem 1 encoded in the Mizar language is described in [15].

4.2 Formalization of polynomial functions

Although polynomials have been formalized in the MML [17], we propose a formal definition of univariate polynomials as sequences for formalizing negligible functions. Because this formalization defines polynomials as an algebraic structure of multivariate polynomials, it is directly unsuitable for asymptotic notation.

Definition: 9 (ASYMPT_2:def 2)

```

definition
  let c be XFinSequence of REAL;
  func seq_p(c) -> Real_Sequence
  means
  for x be Nat holds
  it.x = Sum(c (#) seq_a^(x,1,0));
end;

```

“XFinSequence” is a finite sequence whose indices start from 0. In this definition, XFinSequence c is the given coefficient for the representation of a polynomial. For example, $(\text{seq_p}(\langle *a,b,c* \rangle)).x = a + bx + cx^2$. Note that this definition formalizes “func seq_p(c)”, where “func” is a keyword that shows that the term is defined as a FUNCTOR. In the Mizar language, FUNCTOR plays the role of a constructor of new terms. To define a FUNCTOR, the proof checker requires proofs of at least the existence and uniqueness of the new term. In this case, once we call “seq_p(c)”, the checker automatically recognizes this term as a Real_Sequence. We can then refer to the defining theorem, “for x be Nat holds it.x = Sum(c (#) seq_a^(x,1,0))” to prove another theorem. We will discuss the usage of FUNCTOR in Sec. 6.

Let us now introduce theorems about polynomials.

Theorem: 3 (ASYMPT_2:45)

```

theorem
  for k be Nat, c be XFinSequence of REAL
  st len c = k+1 & 0 < c.k
  holds seq_p(c) in Big_0h( seq_n^(k) );

```

This theorem holds that any k th-degree polynomial function whose most significant coefficient is positive is of $\mathcal{O}(n^k)$. Theorem 4 follows Theorem 3:

Theorem: 4 (ASYMPT_2:46)
theorem
for k be Nat, c be XFinSequence of REAL
st len c = k+1 & 0 < c.k holds
seq_p(c) is polynomially-bounded;

This theorem holds that all univariate polynomials are polynomially bounded.

5 Formalization of negligible functions

In this section, we introduce our formal definition of a negligible function.

Definition: 10 (ASYMPT_3:def 4)
definition
let f be Function of NAT,REAL;
attr f is negligible means
for c be non empty positive-yielding
XFinSequence of REAL holds
ex N be Nat st
for x be Nat st N <= x holds
|. f.x .| < 1/((seq_p(c)).x);
end;

We then introduce some theorems about negligible functions. The following theorem holds that 2^{-x} is a negligible function:

Theorem: 5 (ASYMPT_3:25)
theorem
for f be Function of NAT,REAL
st for x be Nat holds
f.x = 1/ (2 to_power x) holds
f is negligible;

Theorem 6 holds that negligible functions converge to zero.

Theorem: 6 (ASYMPT_3:24)
theorem
for f be Function of NAT,REAL
st f is negligible
holds f is convergent & lim f = 0;

By providing a formal proof of correctness, we can automate inference rules (i.e., namely clusters) in the MML as follows:

```

registration
let f be negligible Function of NAT,REAL,
a be Real;
cluster a(#f) -> negligible

```

```

                                for Function of NAT,REAL;
end;
registration
  let f,g be negligible Function of NAT,REAL;
  cluster f+g -> negligible
    for Function of NAT,REAL;
end;
registration
  let f,g be negligible Function of NAT,REAL;
  cluster f(#)g -> negligible
    for Function of NAT,REAL;
end;
registration
  let f be negligible Function of NAT,REAL,
  g be polynomially-abs-bounded
    Function of NAT,REAL;
  cluster g(#)f -> negligible
    for Function of NAT,REAL;
end

```

Once these clusters are registered, the checker automatically infers that addition and scalar multiplication between negligible functions yield negligible results. For example, the Mizar proof checker infers the following theorem automatically:

Theorem: 7
 for a be Real,
 e,f be negligible Real_Sequence,
 g be polynomially-abs-bounded
 Real_Sequence holds
 (a (#) f)(#) g + e is negligible;

In general, a cryptosystem is secure if the probability of a successful attack against the cryptosystem is negligible. To prove this, we evaluate the total sum of the probability of success of any possible attack. Adversaries are allowed to attack in parallel or iteratively. In most cases, it is sufficient to evaluate the linear combination of probabilities of successful attacks if the probability of all attacks is negligible. We have formalized definitions and theorems concerning probability and probability distributions that can treat concrete probabilistic events for describing attacks against cryptosystems [2–7]. In combination with these formal descriptions, the above-mentioned clusters would be useful for formal verification of the security of cryptosystems.

6 Formalization of algebra of polynomially bounded functions

In Secs. 4 and 5, we introduced our formalizations of polynomially bounded functions and negligible functions. In this section, we propose a formal definition

of the algebra of polynomially bounded functions and present some theorems about negligible functions and this algebra.

6.1 Algebra of polynomially bounded functions

Set of polynomially bounded functions First, we define a relaxed attribute of polynomially bounded functions namely “polynomially-abs-bounded”, as follows:

Definition: 11 (ASYMPT_3:def 1)

definition

```

    let p be Real_Sequence;
    attr p is polynomially-abs-bounded means
    ex k be Nat st |. p .| in Big_Oh(seq_n^(k));
end;
```

Here, $|. p .|$ is the absolute value of the given function p . Although we define the attribute polynomially-bounded in Sec. 8, we formalize this definition for technical reasons, i.e., in order to introduce the multiplicative inverse of addition between polynomially bounded functions. Naturally, polynomially bounded functions are polynomially-abs-bounded. We then register the following clusters:

registration

```

    cluster polynomially-bounded ->
    polynomially-abs-bounded for Real_Sequence;
end;
```

Next, we define the set of polynomially-abs-bounded functions as a new term, namely, “Big_Oh_poly”, as follows:

Definition: 12 (ASYMPT_3:def 2)

definition

```

func Big_Oh_poly -> Subset of RAlgebra (NAT)
    means for x being object holds x in it
    iff x is polynomially-abs-bounded
    Function of NAT,REAL;
end;
```

We then register the following cluster:

registration

```

    cluster Big_Oh_poly -> non empty;
end;
```

In other words, we made the proof-checker recognize the existence of polynomially-abs-bounded functions.

Algebraic structure of polynomially bounded functions In this section, we define the algebraic structure of polynomially bounded functions as follows:

Definition: 13 (ASYMPT_3:def 3)

```

definition
  func R_Algebra_of_Big_Oh_poly
    -> strict AlgebraStr means
  the carrier of it = Big_Oh_poly
  & the multF of it
    = (RealFuncMult(NAT)) || Big_Oh_poly
  & the addF of it
    = (RealFuncAdd(NAT)) || Big_Oh_poly
  & the Mult of it = (RealFuncExtMult(NAT))
    | [:REAL,Big_Oh_poly:]
  & the OneF of it
    = RealFuncUnit(NAT)
  & the ZeroF of it
    = RealFuncZero(NAT);
end;

```

In this formalization, we define the carrier, multiplication, addition, scalar multiplication, multiplicative unit, and additive unit. The carrier of `R_Algebra_of_Big_Oh_poly` (i.e., `Big_Oh_poly`) is included in the set of all real-valued functions. Instead of defining a new operation, we use predefined operations on real-valued functions by restricting their domain according to the carrier of this structure. We define the additive and multiplicative units of this structure as being the same as those of the algebra of real-valued functions.

We then prove the attributes that are held by the algebraic structure `R_Algebra_of_Big_Oh_poly`, and register them as the following cluster:

```

registration
  cluster R_Algebra_of_Big_Oh_poly
  -> strict Abelian add-associative
    right_zeroed right_complementable
    commutative associative right_unital
    right-distributive vector-associative
    scalar-associative vector-distributive
    scalar-distributive;
end;

```

As a result, we proved the following theorem:

Theorem: 8 (ASYMPT_3:18)

```

theorem
  R_Algebra_of_Big_Oh_poly is Algebra;

```

6.2 Set of negligible functions

In this section, we define the set of negligible functions as a new term, namely, “negligibleFuncs”, as follows:

Definition: 14 (ASYMPT_3:def 5)
 definition
 func negligibleFuncs ->
 Subset of Big_Oh_poly
 means
 for x being object holds x in it
 iff x is negligible Function of NAT,REAL;
 end;

We then register the following cluster:

```
registration
  cluster negligibleFuncs -> non empty;
end;
```

Note that we define “negligibleFuncs” as a nonempty subset of R_Algebra_of_Big_Oh_poly.

In the rest of this section, we describe particularly useful theorems about polynomially bounded functions and negligible functions.

Equivalence between polynomially bounded functions We define the following predicate:

Definition: 15 (ASYMPT_3:def 6)
 definition
 let f,g be Function of NAT,REAL;
 pred f negligibleEQ g means
 ex h be Function of NAT,REAL st
 h is negligible &
 for x be Nat holds
 |. f.x - g.x .| <= |.h.x.|;
 reflexivity;
 symmetry;
 end;

Here “f negligibleEQ g” means that there exists a negligible function, h, which is the upper bound of the difference between two functions, f and g. In other words, we introduce an equivalence relationship between polynomially bounded functions. This equivalence would be useful not only for cryptography and computer science but also for approximating and analyzing functions.

Moreover, we prove the following theorem about multiplication between polynomially bounded and negligible functions:

Theorem: 9 (ASYMPT_3:39)

```

theorem
  for v,w be
    VECTOR of R_Algebra_of_Big_0h_poly
    st w in negligibleFuncs
    holds v * w in negligibleFuncs;

```

where VECTOR is a synonym for an element of the structure, and “ $v * w$ ” is the product of two VECTORS v and w : $(v * w).x = (v.x) * (w.x)$. This theorem shows that the product “ $v * w$ ” of two functions is negligible if v is a polynomially bounded function and w is a negligible function. Recall the above mentioned theorems and clusters concerning negligible functions. The set of negligible functions is a nonempty subset of the set of polynomially bounded functions. Addition and scalar multiplication between negligible functions also yield negligible functions. These theorems mean that the set of negligible functions is an ideal of the set of polynomially bounded functions.

7 Case study: Evaluating the computational cost of algorithms

In this section, we show an example of evaluating the computational cost of algorithms. In [8], we did not formalize about computational cost of the Euclidean algorithm, although we proved its correctness. Therefore we formalize Lamé’s theorem to prove that the Euclidean algorithm is a polynomial-time algorithm. The Euclidean algorithm is an efficient algorithm that computes the greatest common divisor (GCD) of two given integers. Lamé’s theorem [18] is known as the first application of Fibonacci numbers and it shows that the Euclidean algorithm terminates after a maximum of $5X$ repetitions, where X represents the decimal digits of the smaller one of the two given integers. The Euclidean algorithm terminates after n steps such that $\text{Fib}(n - 1) \leq b$, where b is the smaller of the two given integers. By evaluating Fibonacci numbers, it was found that n is not larger than $5X$ repetitions, where X represents the decimal digits of b . Consequently, we conclude that n is not increasing faster than the polynomials. In other words, we can evaluate that the computational cost of the Euclidean algorithm is being polynomially bounded.

Theorem: 10

```

theorem
  for a,b be Element of INT st
    |.a.| > |.b.| & b > 1 holds
    ex A,B be sequence of NAT,
      C be Real_Sequence,
      n be Element of NAT st
    A.0 = |.a.| & B.0 = |.b.| &
    (for i be Nat holds
    A.(i+1) = B.i &

```

```

B.(i+1) = A.i mod B.i) &
n = (min*{i where i is Nat: B.i = 0} ) &
a gcd b = A.n &
Fib(n+1) <= |. b .| &
n <= 5*[/ log(10,|. b .|) \] &
n <= C.(|. b .|) & C is polynomially-bounded;

```

In future works, we will formalize complex algorithms that include iterations or subroutine calls even though we were able to formalize the Euclidean algorithm using simple sequences.

8 Overview of verifying security of cryptologic systems

In this section, we briefly describe our future plans for formalizing the security of cryptologic systems. It is well known that the security of cryptosystems is dependent on the complexity of computational problems. Polynomially bounded functions play an essential role in evaluating and classifying the complexity of computational problems. We are attempting to formalize computational complexity using our libraries mentioned in Sec. 4.

Probability distribution is a fundamental principal of cryptology. We can define other cryptologic topics using our formalization of probability distribution. Currently, we are attempting to formalize the indistinguishability of probability distributions. In modern cryptology, the security of cryptologic systems is essentially defined by indistinguishability. For example, various cryptologic topics, such as pseudo-random number generators and hash functions, are described using the concept of indistinguishability. We often design a cryptographic scheme by employing ideal functions. However, we must replace such ideal functions with feasible functions when we implement the schemes. Thus, an implemented scheme is not always secure even if its ideal functionality has been proven to be secure in design. Indistinguishability of functions means the indistinguishability of the distributions of the outputs of the functions between an ideal function (e.g., an ideal random function) and a feasible function (e.g., a hash function) to prove the security of a cryptographic scheme. Two probability distributions are statistically indistinguishable if the distance between them is negligible. We intend to define a formalization of indistinguishability using our formalization of negligible functions mentioned in Sec. 5 and probability distributions[2].

9 Conclusion

In this paper, we introduced a formalized definition of polynomially bounded and negligible functions in the Mizar language. We then showed formalized theorems related to these definitions. All formalized definitions and theorems in this paper are described in the Mizar language, and the proofs of these theorems have been verified using the Mizar proof checker. Our formal descriptions have been stored in the current version of the MML and are available online on the Mizar Project

official website. We prepared these formalized articles in order to achieve our aim of constructing an automated formal verification tool for cryptology. We will now attempt to formalize one of the most important concepts of cryptology indistinguishability using the formalization of negligible functions introduced in this paper.

ACKNOWLEDGEMENT

This study is partly supported by JSPS KAKENHI 15K00183. The authors would like to express their gratitude to Prof. Yasunari Shidama for his support and encouragement.

References

1. Mizar System: *Available at* <http://mizar.org/>.
2. H. Okazaki, Y. Futa, Y. Shidama, “Formal definition of probability on finite and discrete sample space for proving security of cryptographic systems using Mizar”, *Artificial Intelligence Research*, 2(4), pp.37-48, 2013. DOI : 10.5430/air.v2n4p37
3. H. Okazaki, Y. Shidama, “Random Variables and Product of Probability Spaces”, *Formalized Mathematics*, 21(1), pp.33-39, 2013. DOI : 10.2478/forma-2013-0003
4. H. Okazaki, “Posterior Probability on Finite Set”, *Formalized Mathematics*, 20(4), pp.257-263, 2013. DOI : 10.2478/v10037-012-0030-0
5. H. Okazaki, Y. Shidama, “Probability Measure on Discrete Spaces and Algebra of Real Valued Random Variables”, *Formalized Mathematics*, 18(4), pp.213-217, 2010. DOI : 10.2478/v10037-010-0026-6
6. H. Okazaki, “Probability on Finite and Discrete Set and Uniform Distribution”, *Formalized Mathematics*, 17(2), pp.173-178, 2009. DOI : 10.2478/v10037-009-0020-z
7. H. Okazaki, Y. Shidama, “Probability on Finite Set and Real-Valued Random Variables”, *Formalized Mathematics*, 17(2), pp.129-136, 2009. DOI : 10.2478/v10037-009-0014-x
8. H. Okazaki, Y. Aoki, Y. Shidama, “Extended Euclidean Algorithm and CRT Algorithm”, *Formalized Mathematics* 20(2), pp-175-179, 2012. DOI : 10.2478/v10037-012-0020-2
9. Y. Futa, D. Mizushima, H. Okazaki, “ Formalization of Gaussian integers, Gaussian rational numbers, and their algebraic structures with Mizar”, *ISITA 2012*: pp.591-595, 2012.
10. Y. Futa, H. Okazaki, Y. Shidama, “Formalization of Definitions and Theorems Related to an Elliptic Curve Over a Finite Prime Field by Using Mizar”, *J. Autom. Reasoning* 50(2), pp.161-172 (2013). DOI : 10.1007/s10817-012-9265-2
11. Ronald L. Rivest, Adi Shamir, Len M. Adelman, “A Method for Obtaining Digital Signature and Public-key Cryptosystems”, *MIT-LCS-TM-082*, (1977).
12. Taher Elgamal, “A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, *IEEE Transactions on Information Theory*, IT-31(4), pp. 469?-472, (1985).
13. Jon Kleinberg, Eva Tardos, “Algorithm Design”, Addison-Wesley, 2005.
14. Goldreich, “Foundations of Cryptography : Volume: 1 Basic Tools”, Cambridge University Press, 2001.

15. H. Okazaki, Y. Futa, “Polynomially Bounded Sequences and Polynomial Sequences”, *Formalized Mathematics*, 23(3), pp.205-213, 2015. DOI: 10.1515/forma-2015-0017
16. H. Okazaki, “Algebra of Polynomially Bounded Sequences and Negligible Functions”, *Formalized Mathematics*, 23(4), pp.371-378, 2015.
17. P. Rudnicki, A. Trybulec, “Multivariate Polynomials with Arbitrary Number of Variables”, *Formalized Mathematics*, 9(1), pp. 95–110, 2001.
18. G. Lamé’, “Note sur la limite du nombre des divisions dans la recherche du plus grand commun diviseur entre deux nombres entiers”, *Comptes Rendus Acad. Sci*, 19, pp.867–870, 1844.

A Smooth Transition to Modern mathoid-based Math Rendering in Wikipedia with Automatic Visual Regression Testing

Moritz Schubotz¹ and Alan P. Sexton²

¹ Database Systems and Information Management Group,
Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany

`schubotz@tu-berlin.de`

² School of Computer Science

University of Birmingham

Edgbaston, Birmingham, U.K.

`a.p.sexton@cs.bham.ac.uk`

`http://png.formulasearchengine.com`

Abstract. Pixelated images of mathematical formulae, which are inaccessible to screen readers and computer algebra systems, disappeared recently from Wikipedia. In this paper, we describe our efforts in maturing **mathoid**, the new services that provides better math rendering to Wikipedia, from a research prototype to a production service and a novel visual similarity image comparison tool designed for regression testing mathematical formulae rendering engines.

Currently, updates to Math rendering engines that are used in production are infrequent. Due to their high complexity and large variety of special cases, developers are intimidated by the dangers involved in introducing new features and resolving non critical problems. Today's hardware is capable of rendering large collections of mathematical contents in a reasonable amount of time. Thus, developers can run their new algorithms before using them in production. However, until now they could not identify the most significant changes in rendering due to the large data volume and necessity for human inspection of the results.

The novel image comparison tool we are proposing, will help to identify critical changes in the images and thus lower the bar for improving production level mathematical rendering engines.

Keywords: Wikipedia, Math rendering, Mathematical formulae, Image comparison

1 Introduction

In [11], we analysed different ways to improve math rendering in Wikipedia and presented our solution, which we coined **mathoid**. However, two years later, only a small group of registered users benefit from the improvements in rendering. In the past two years, many bugs were reported and fixed. Those bugs can be

categorized into two main concerns; performance and layout. While improving performance is relatively straightforward to measure, improving the layout is still an open problem [1, 9, 10, 17]. In particular, any work in this area is hindered by the issue of ensuring that improving the layout of some aspect of math rendering does not negatively impact other aspects. When such regression testing requires humans to visually inspect and compare tens of thousands of images, progress on layout can be slow and error prone.

In this paper, we present a method to automatically compare images of mathematical formulae generated by different rendering engines and thus automate visual regression testing in this domain.

Our paper is structured as follows: We begin by presenting an overview of the improvements over the old rendering, then we describe the request flow of the new rendering process in detail and analyse its performance.

Thereafter, we describe `mathpipe`, the tool we built to compare different rendering mechanisms at scale and finally present the current state of our image comparison tool. Since this is work in progress, we encourage the reader to visit <http://png.formulasearchengine.com> for the latest updates.

2 mathoid’s Improvements Over texvc Rendering

As described in [11], `mathoid` provides “Robust, Scalable, Fast and Accessible Math Rendering for Wikipedia.” In 2014 `mathoid` was one of the first Wikimedia nodeJS services, supporting the MediaWiki extension `Math`, which takes care of the handling of mathematical expressions used in Wikipedia and other websites running MediaWiki. Today `mathoid` is constructed from a huge number of services such as `citoid` and `graphoid`, which support the MediaWiki extensions `Cite` and `Graph` respectively. All of these service-supported extensions add complex functionality to MediaWiki, which would be hard to reimplement with native and efficient PHP code. Moreover, they all share the goal to be “Robust, Scalable, Fast and Accessible”. The so called `service-template` provides a common ground for robustness and scalability and reduces the maintenance effort.

For the `Math` extension, that means no binaries need to be installed on the MediaWiki servers and no files in the file-system are created. Thus *robustness* is improved with respect to the old approach that relies on the `texvc` binary and creates local files [11], which uses `MathJax` [2] rather than `LATEX` for the rendering. Note, that the set of supported “`LATEX` like macros“ is exactly the same for the old and the new system.

With the service template scaling the `mathoid` service is simple. As an improvement over the original version of `mathoid` presented in [11], now all formulae of a page are processed in parallel (cf. Section 3) instead of sequential. That way the page loading time is determined by the formula that takes the longest time to render and no longer by the sum of all the rendering times. See Section 4 for the time measurements of individual formulae.

However, most significant to the user are the accessibility component and the change in the layout itself. After the new rendering was tested on beta clusters

De Broglie übertrag dies 1923 auf beliebige Teilchen:

$$p = \frac{h}{\lambda}$$

mit dem relativistischen Impuls:

$$p = \frac{m v}{\sqrt{1 - \frac{v^2}{c^2}}}$$

ergibt sich daraus die sogenannte De-Broglie-Wellenlänge:

$$\lambda = \frac{h \cdot \sqrt{1 - \frac{v^2}{c^2}}}{m v}$$

In[8]:= $p == \frac{h}{\lambda}$

Out[8]= $p == \frac{h}{\lambda}$

In[9]:= $p = \frac{m v}{\sqrt{1 - \frac{v^2}{c^2}}}$

Out[9]= $\frac{m v}{\sqrt{1 - \frac{v^2}{c^2}}}$

In[10]:= `Solve[%8, λ]`

Out[10]= $\left\{ \left\{ \lambda \rightarrow \frac{h \sqrt{1 - \frac{v^2}{c^2}}}{m v} \right\} \right\}$

Fig. 1. Copying MathML expressions from the German version of Wikibooks to Wolfram Mathematica: On the left, there is a screen-shot, displaying a section of the book on quantum mechanics taken on May'16 2016 (the day MathML rendering became the default rendering mode for mathematical Formulae on Wikibooks) as non registered visitor using Firefox version 45, with the Native MathML plugin enabled. The photon momentum was selected, to demonstrate the bounding boxes of the symbols, and to copy and paste it to the computer Algebra System Mathematica (screen-shot on the right) as In[8]. The same step was performed for the relativistic momentum In[9]. Thereafter, an additional = sign has been manually inserted to In[8] and Solve[%8, λ] was typed to compute the De-Broglie wavelength.

it has been enabled on Wikidata and the German version of Wikibooks on May 16th, 2016, and was enabled globally on May 31th, 2016.

As visualized in Figure 1, the MathML code is available to the browser. This allows screen readers to verbalize the formulae from the additional information that is neither available from the new SVG image nor from the old PNG image. The documentation page of the Math extension contains links to examples of this feature and [3, 4, 8, 12, 14] provide further information on that topic.

However, since MathML requires certain fonts that are not available on all operating systems, MathML will only be provided to people that have installed a browser extension that indicates that their browser actually provides good MathML rendering. This could be either Mozilla Firefox with the Native MathML plugin [16] or Internet Explorer with the MathPlayer plugin [13]. The majority will see SVG images, which is already an improvement, especially for high resolution displays, or in situations where the formulae are printed.

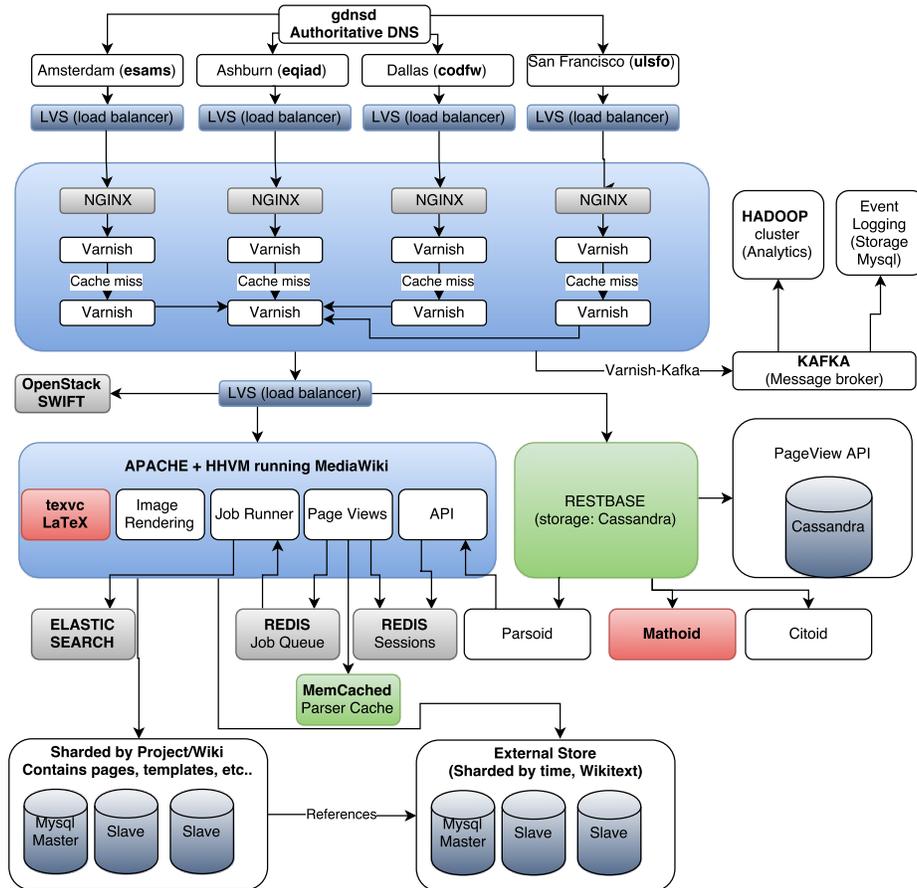


Fig. 2. Overview of the Wikimedia infrastructure (based on an image from Luca Toscano, which was released under the Creative Commons (CC3) license.)

3 Internals of the New mathoid Rendering

While the Wikipedia request flow as visualized in Figure 2 is quite complicated, we summarize a typical visit of a page that contains mathematical formulae below.

1. A user requests a page
2. If that page is not cached (neither physically close to the user or abroad), MediaWiki renders the page from the internal representation (wikitext) to HTML. Note that this is only true for a regular visit. If the user wants to edit the page using the visual editor the page would be rendered by parsoid (cf. 3).
3. All math elements on a page are collected

4. A bundle of requests (the size of this bundle is a performance tuning parameter) is sent to `restbase` (check request)
5. `restbase` checks the requests; either the result is cached in the internal cassandra data store
6. or it contacts `mathoid` which calls `texvcinfo` which calls `texvcjs` (this will be explained later in detail).
7. `restbase` returns the results of the check and other information about the formulae i.e. the sanitized tex, a hash and the SVG.
8. After a bundle of check requests is returned to the MediaWiki, the extension evaluates each check response, and either replaces the Math tag with an error message in case the check request was not successful or collects the hash of the MathML and SVG.
9. Now, the math extension has collected the hashes of all valid input formulae and a second request bundle is sent to `restbase` requesting the MathML rendering.
10. `restbase` responds with the MathML rendering from storage. In the header of the request response, the SVG dimensions are stored.
11. Thereafter, the math extension replaces the math tags with MathML and a link to the SVG fallback image. The style-sheet is configured in a way that the fallback image is visible by default and the MathML element is invisible.
12. Finally, MediaWiki returns the HTML of the page, including the links to the SVG fallback images.
13. If the browser of the user does not overwrite the visibility information of the MathML and SVG, the browser sends a request for each individual formula.
14. While most of the image requests will be served from either the browser cache or varnish (a caching http reverse proxy [15]), the rest will go directly to `restbase` without contacting MediaWiki.

4 Measuring mathoid’s performance

To evaluate the absolute performance of the new rendering, we performed measurements with the following set-up: We use two identical work stations (Intel(R) Core(tm) i7-3770 CPU @ 3.40GHz 16 GB RAM, Gigabit Ethernet, 2x1 TB HDD, Ubuntu 14 LTS). One system (hereafter referred to as the server) has the Cassandra storage engine, the `restbase` Client and `mathoid`. The other system (client) has MediaWiki with the extension Math and MathSearch, it uses as many parallel workers as CPUs are present (8) to call to get the data as would be done in production. Our performance tests script can be downloaded from github script.

For each formula, the requests are sent in as described in 3 and the time is measured. Note that, for this test, we did not use the complete endpoint but the MathML or SVG endpoint respectively. For each formula we randomly chose if it was rendered first in SVG or MathML mode. The measurement results are visualized in Figure 3. The figures indicate an almost linear relationship between the formula length and the rendering time. It should be noted that formulae whose request could be answered by `restbase` from the cassandra storage without

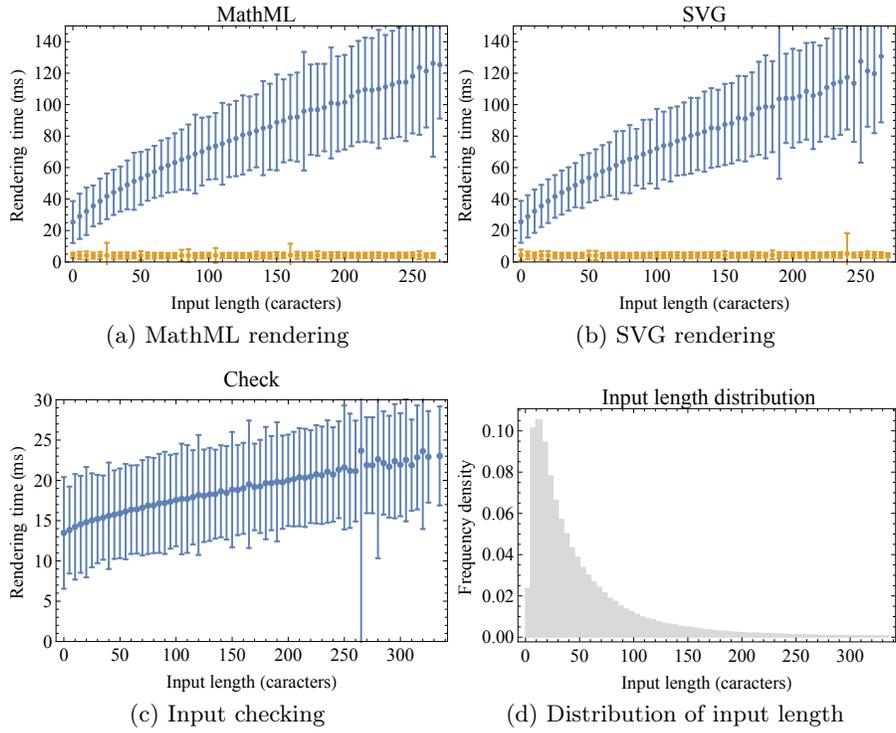


Fig. 3. Rendering and verification time versus input length: Measurements for the Test Collection, with different rendering modes. (a-c) shows the rendering, times and standard deviations respectively. (a-b) Include two series, where the yellow (lower) series is the cached result.

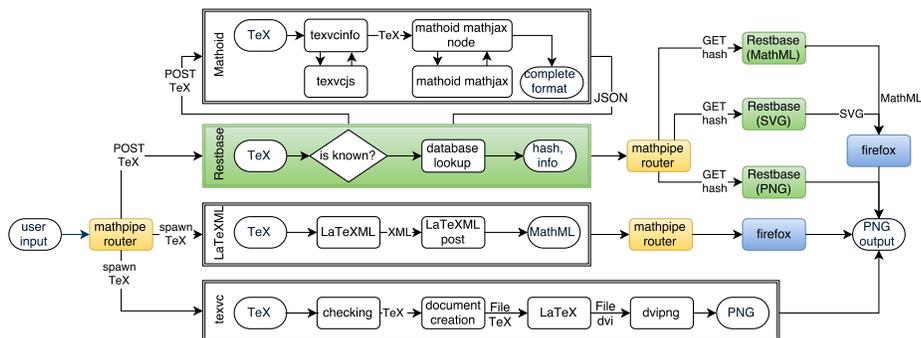


Fig. 4. mathpipe processing chain

contacting mathoid were fast, independent of the length of the input tex string. The average was around 4ms. These measurements indicate that, in the average case, where already rendered formulae get requested again, most of the time is spend at the checking phase (>20 ms). Therefore, the most recent version of restbase, which has not been tested yet, also caches the check responses from mathoid. This is supposed to reduce the checking significantly.

Compared to the old \LaTeX based rendering, which created a \LaTeX document for each formulae, rendered that document and thereafter converted it to a PNG image, this is a significant performance improvement.

5 Comparing Different Rendering Engines with mathpipe

mathpipe is a program specifically designed for testing the different options for rendering mathematics on the web. It starts with user input (currently limited to texvc dialects) and goes via different routes to PNG output. All current routes are visualized in Figure 4. From there on, we perform the analysis of different generated png images and calculate the similarity scores as described below. mathpipe has two modes of operation. A command-line mode that can be used for batch jobs and a web interface. The command-line will be used to identify the formulae that have the biggest divergence. The web interface will provide a quick comparison for humans to manually investigate the difference outputs and check the derived results. We encourage the reader to visit <http://png.formulasearchengine.com> and test an instance of our service.

6 Image comparison

The lack of practical automatic regression testing on mathpipe-generated renderings of the huge collection of mathematical formulae in Wikipedia has been a serious hinderance to speedy improvement of rendering quality and performance. Currently, any change made to the rendering pipeline can only be checked visually by humans. Even then, it is very easy for a human inspector to overlook

some small but important error that has arisen in the rendering. The motivation for automating this procedure with a suitably high precision machine-based approach is clear.

The problem to be solved involves comparing two PNG images rendered using different methods from the same input source. These methods vary in their choice of fonts (and therefore also the thickness of the character strokes), the resolutions of rendered images, the approaches to anti-aliasing and background transparency rendering, spacing between characters and relative sizes of characters. Some of the methods also generate extra empty padding around the images and most generate some form of greyscale or colour images, the details of which can vary, rather than monochrome.

In comparing these images, our ideal is to make judgements about the relative validity of these renderings as an extremely careful human inspector would. Differences insignificant to the readability and correct interpretation of the underlying mathematical formulas should be overlooked. For example:

- changes of font where corresponding characters from the two images are visually very similar.
- differences in character spacing in the two images which are within aesthetically reasonable bounds.
- characters that are spatially discrete but close together in one rendering may touch in another. This is often due to the rendering resolution chosen together with the antialiasing approach used. If the resulting touching characters are perfectly readable, then this should not cause significant concern or trigger reporting of errors unless the user is explicitly looking for such problems.
- with symptoms very similar to the previous case, a single character in one image may be broken into two or more characters in the other. This arises not from characters touching, but by the renderer building a character, often an extendable fence or integral character, out of separate character components. The renderer is supposed to make such characters touch, but may fail to do so completely. Such a case rarely interferes with the readability of the image, but should be overlooked or highlighted as the user requires.

Conversely, significant issues should be identified and reported:

- characters in one image but missing from the other.
- characters that have significantly different shapes, for example if the character is missing or mis-indexed in the font used in one of the images.
- significant differences in spacing that may confuse the reader, e.g. a superscript that is rendered on the same baseline as a superscript.
- touching characters that are unreadably overlapped.
- broken characters that are so spatially separated that it confuses the reader.

The tolerances used to determine many of the fuzzy quality issues (i.e. “*visually similar*”, “*aesthetically reasonable*”, etc.) described above should be choosable by the user to correspond to the user’s purpose at the time, although defaults should be set to practical values for general purpose regression testing.

In general, the system should be biased to not overlook serious issues, even if that means that more insignificant differences are reported as significant (false positives).

Finally, a practical and robust way of reporting issues found is necessary to assist developers in quickly identifying problems and their sources.

6.1 Approach

The constraints of a solution as described above mean that a purely image-focused approach is unlikely to be successful. Instead we have taken an approach based on a structural analysis of the image into a form where we can deal with the image components and their relationships more abstractly. This form is based on *connected components*. A connected component in a monochrome image is a maximal set of foreground pixels that are horizontally, vertically or diagonally connected. Thus an equals symbol, “=” and the letter “i” both have two connected components, while an equivalence symbol “≡” and a capital greek letter xi, “Ξ”, both have three.

In brief, the general approach we have chosen involves binarising the images, decomposing them into sets of connected components, scaling the meta-data (bounding-box information) about the connected components to make them comparable between the images, identifying viable pairings of corresponding connected components in the different images based on relative positions, aspect ratios, and size, verifying the pairings using features of the underlying connected component shape and pixel distributions, and finally treating connected components not successfully paired as possible candidates for touching/broken character analysis.

It should be noted that, though we use techniques borrowed from the areas of document analysis and optical character recognition [5], we do not attempt to classify characters or interpret the mathematical formulae in any way, and hence we avoid the problems of classifier training, mis-classifications and incompleteness of a model for the structure of mathematical formulae. More precisely, we borrow only methods of binarisation, connected component analysis and some simple shape feature extraction methods that are commonly used in document analysis for character classification purposes but here are used only to provide a basis for metric shape similarity measurements between corresponding components of the two images.

We shall now discuss each part of the approach in detail.

Binarisation and Connected Component Analysis Connected component analysis requires binary choices between foreground and background images. Since these images are generated rather than scanned, a very simple binarisation method selecting the pixel class based only on the RGBA (red-green-blue-alpha) value of the pixel itself is sufficient. The only issue of (minor) concern is the different approaches to background colours and anti-aliasing. Background pixels and anti-aliasing in an effectively monochrome image is best implemented by

using the same (foreground) colour for all pixels but varying the transparency of the anti-aliased pixels down to fully transparent for background pixels. However, some images do change the grey-scale/colour as well as the transparency while others do not use transparency at all but merely blend the foreground into a different background colour. Binarisation must be aware of these variations and manage them all.

Connected component analysis is accomplished using the algorithms described in [6, 7]

Cropping and Scaling Removing extraneous background padding is a simple matter of cropping to the size of the rectangle union of the bounding boxes of the identified connected components.

Scaling is slightly more subtle; Since these images can be of relatively low resolution, the size of a pixel relative to the whole connected component can be significant. Hence scaling the image to a common size can introduce discretisation artifacts that impede the analysis. Hence the images themselves are not scaled but a scaled version of the connected component bounding box information is added. The scale factors are chosen so that one of the images' information is scaled to correspond to an image of width 1.0 (using floating point rather than integer numbers of pixels) and a height that maintains the original aspect ratio. The other image is scaled so that the image is exactly the same size, even if that distorts its aspect ratio. This results in relative positions of connected components in the scale spaces being directly comparable. The underlying image pixels are not changed so shape analysis is not affected by the scaling.

Simple Component Pairing At this point we attempt to pair off connected components in one image with the corresponding ones in another. A simple matching of corresponding positions does not work for a number of reasons:

- While scaling ensures that the left-most and rightmost characters are in very close to the same horizontal position, variations in spacing may mean that characters in the centre of the image are significantly out of horizontal alignment with the corresponding character of the other image, and, indeed, may be in exactly the same relative position as a non-corresponding character. Ditto for vertical alignments.
- Multiple characters in one image may be touching and therefore occur as a single connected component while they are separated in the other. Therefore the correct pairing should be of at least one connected component with a set of connected components.
- Depending on the actual parameters used, a component of one image may viably, but incorrectly, be paired with any one of a number of different components from the other. We call such a case a “*multi-match*”
- Because of variations of character fonts, positioning and sizing, any choice of discrimination parameters that correctly accept/reject a pairing in one part of the image tends to be wrong for another part of the image. An adaptive

approach that varies the parameters over different parts of the image might be possible but it is not clear what criterion can be used to accomplish it without more in-depth classification or recognition.

For these reasons we chose an iterative approach, where we start with very tight constraints on acceptable parameters to pair components based on their position, aspect ratio and size (i.e. area of the bounding boxes), with a verification element based on a metric shape similarity measure, to guarantee reliability of the pairing. This ensures that any pairings found are robust and can be removed from consideration. Repeating the process with slightly relaxed parameters on the thinned out set of remaining components allows robust pairings to be made where, with the full set of components, an unambiguous pairing would not have been possible. This cycle continues until one of the following holds:

- no unpaired components remain (in which case the two images can be considered to have passed the comparison check) or
- any further pairings require relaxing the parameters beyond their upper limits or
- there is a component which, at the current parameter setting, could viably be paired with more than one component (a multi-match).

In the latter two cases further analysis is necessary.

Touching Component Analysis At this point, assuming there are multi-matches or unpaired components remaining, there is a set of components from each image that could not be paired with components from the others. The only allowable remaining situation that would not justify reporting this as an error is if components are touching in one image but not in the other, and there may be multiple separate cases involved. Simply trying all possible combinations of ways that components could touch is computationally infeasible for our purposes.

To find such cases, note that a touching component in one image that corresponds to a group of components from the other is necessarily larger than the individual sub-components. Therefore we work iteratively starting at the largest remaining unpaired component of **both** images, the *target* component, and select the set of unpaired components (the *candidate* components) from the other image that overlap with an expanded version of the bounding box of the target component. This excludes from consideration components that should not realistically be considered as candidates, but the expansion allows for some distortion of the spacing between the different images.

Even if the target does correspond to some of the candidate components, it may not correspond to all. For example, consider the following expressions where the left is from one image and the right from another:

$$\sqrt{x^K} \quad \sqrt{x^K}$$

Here the target would be the touching $\sqrt{x^K}$ component from the right and the candidate set would include all three components from the left, because all three

are within the appropriate space. However, the x should not be included in the pairing or it will cause the shape matching to fail.

For these reasons, all combinations of the candidate components are considered by calculating the comparison attributes of the union of each combination, where the attributes involved are; centre of the bounding box, aspect ratio, area of bounding box and, only if the checking of those attributes passes, the more expensive shape similarity test. The final shape similarity test ensures that the resulting merged shape is still readable. The limitation of the set of candidate characters to those that overlap the expanded target component bounding box ensures computational feasibility.

If a pairing is found, the components are removed and the process repeats on any remaining unpaired components until exhaustion of unpaired components or failure to find a pairing.

Reporting of Results The results of the above analysis is four-fold:

1. *Simple matches*: A set of pairs of single compatible components that are within appropriate matching parameters.
2. *Touching matches*: A set of pairs of sets of components corresponding to target/candidate set matching pairs for touching component cases.
3. *First image unpaired*: A set of components from the first image that could not be paired with corresponding components from the second.
4. *Second image unpaired*: A set of components from the second image that could not be paired with corresponding components from the first.

The test passes if all except *simple matches* is empty. It can be considered to pass if *touching matches* is also non-empty and the user chooses that option. A short narrative report is generated of the results to a log file or to the standard output stream. However, a textual description of the problems when errors occur is frustratingly difficult to interpret. Hence we also generate two error images, These are cropped binarised images but with unpaired components drawn in one colour, target components of touching matches in another and the corresponding matching candidate components in a third. This is usually sufficient for immediate identification of the problem to the user. However, sometimes it is necessary to investigate more directly the relative positioning or sizing issues that triggered the comparison failure so we provide a python plugin for the GIMP image processing tool that allows the two error images to be loaded as separate layers, scaled and positioned to exactly the same size and position so that, by varying the transparency of the layers with the GIMP's layer tool, one can precisely see the issues involved.

An example of the error results when touching matches are found are shown in Figure 5

7 Further work

As a work-in-progress, there is still much work to do in refining and improving the image comparison tool, testing and evaluating it on the various `mathpipe`

$$P = 3B_0 \left(\frac{1 - \eta}{\eta^2} \right) e^{\frac{3}{2}(B'_0 - 1)(1 - \eta)}$$

$$P = 3B_0 \left(\frac{1 - \eta}{\eta^2} \right) e^{\frac{3}{2}(B'_0 - 1)(1 - \eta)}$$

$$P = 3B_0 \left(\frac{1 - \eta}{\eta^2} \right) e^{\frac{3}{2}(\cancel{B}'_0 - 1)(1 - \eta)}$$

Fig. 5. Error images from two different renderings of the same formula. The original of the second image has been artificially edited to force the B' characters to touch, as indicated by the green colour. In the first image the two characters that form the matching candidates are in blue. The third image shows the result when the two images are scaled and overlaid using the GIMP plugin to demonstrate differences in the spacing and character shapes between the two images.

rendering pipelines and, eventually, building it into the `mathpipe` construction toolchain.

8 Conclusion

We have presented Wikipedia's new approach to higher performance, higher quality, scalable, accessible mathematical formula rendering and delivery and the work we carried out in performance analysis of its results that demonstrates its huge performance improvement over the previous approach.

We have also presented our work on addressing a critical need for speedier and more robust development of further improvement in mathematical formula rendering; namely an image comparison program suited for use in automatic regression testing of mathematical formula rendering software. While this program is still under heavy development, it is already showing promise in providing support for more aggressive development of new `mathoid`-based rendering methods.

Acknowledgments. The authors would like to thank the Wikimedia Foundation employees, especially Marko Obrovac and Gabriel Wicke for their code reviews, guidance, contributions, and discussions in the context of `mathoid`.

Bibliography

- [1] Börjesson, E. and Feldt, R. (2012). Automated system testing using visual GUI testing tools: A comparative study in industry. In Antoniol, G., Bertolino, A., and Labiche, Y., editors, *5th IEEE Int. Conf. on Software Testing, Verification and Validation, ICST 2012*, pages 350–359. IEEE Computer Society.
- [2] Cervone, D. (2012). Mathjax: A Platform for Mathematics on the Web. *Notices of the American Mathematical Society*, 59(2):312–316.
- [3] Chisholm, W., Vanderheiden, G., and Jacobs, I. (2001). Web content accessibility guidelines 1.0. *Interactions*, 8(4):35–54.
- [4] Cooper, M., Lowe, T., and Taylor, M. (2008). Access to mathematics in web resources for people with a visual impairment. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 5105 of *LNCS*, pages 926–933. Springer.
- [5] Doermann, D. and Tombre, K., editors (2014). *Handbook of Document Image Processing and Recognition*. Springer.
- [6] He, L., Chao, Y., and Suzuki, K. (2008). A run-based two-scan labeling algorithm. *IEEE Transactions on Image Processing*, 17(5):749–756.
- [7] He, L., Chao, Y., Suzuki, K., and Wu, K. (2009). Fast connected-component labeling. *Pattern Recognition*, 42(9):1977–1987.
- [8] Maddox, S. (2007). Mathematical equations in Braille. *Maths, Stats and Operations Research (MSOR) Connections*, 7(2):45–48.
- [9] Memon, A., Nagarajan, A., and Xie, Q. (2005). Automating regression testing for evolving GUI software. *J. of Software Maintenance*, 17(1):27–64.
- [10] Memon, A. M. (2008). Automatically repairing event sequence-based GUI test suites for regression testing. *ACM Trans. Softw. Eng. Methodol.*, 18(2).
- [11] Schubotz, M. and Wicke, G. (2014). Mathoid: robust, scalable, fast and accessible math rendering for wikipedia. In Watt, S. M., Davenport, J. H., Sexton, A. P., Sojka, P., and Urban, J., editors, *Proc. Int. Conf. on Intelligent Computer Mathematics (CICM 2014)*, pages 224–235. Springer.
- [12] Soiffer, N. (2005a). MathPlayer. In *Proc. 7th Int. ACM Conf. on Computers and Accessibility – ASSETS 2005*, page 204, New York. ACM Press.
- [13] Soiffer, N. (2005b). MathPlayer. In *Proc. 7th Int. ACM Conf. on Computers and Accessibility – ASSETS 2005*, page 204, New York, New York, USA. ACM Press.
- [14] Sorge, V., Chen, V., Raman, T., and Tseng, D. (2014). Towards making mathematics a first class citizen in general screen readers. In *11th Web for All Conference*, Seoul, Korea, 6–9 April 2014. ACM.
- [15] Varnish (2016). Varnish HTTP cache. <https://varnish-cache.org/>. Accessed: 25 June 2016.
- [16] Wang, F. (2016). Native MathML. <https://addons.mozilla.org/en-US/firefox/addon/native-mathml>. seen May, 2016.
- [17] Yoo, S. and Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. *Softw. Test., Verif. Reliab.*, 22(2):67–120.

Getting the units right

Moritz Schubotz¹, David Veenhuis¹, and Howard S. Cohl²

¹ Database Systems and Information Management Group,
Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany
schubotz@tu-berlin.de, david.veenhuis@campus.tu-berlin.de

² Applied and Computational Mathematics Division,
National Institute of Standards and Technology, Gaithersburg, Maryland, U.S.A.
howard.cohl@nist.gov
<http://units.formulasearchengine.com>

Abstract. To understand applied physics, and physical formulae in particular, the investigation of identifier units is beneficial. However, normally the units are not given explicitly in formulae and have to be inferred. In this paper, we investigate how this process can be automated. As an example application, we use physical formulae from Wikipedia together with information from the related knowledge-base Wikidata. We envision that this method can be generalized and describe how, in the future, hard logical constraints may be used as a feedback mechanism for statistical methods in the context of natural language processing.

Keywords: Wikidata, Wikipedia, Units, Natural Language Processing, Inference, Mathematical Language Processing, Constraint propagation

1 Introduction

Units play an essential role in the physical sciences. Especially, *dimensional analysis* is one of the most significant tools for comprehension and understanding of physical formulae. We claim that this technique is not only beneficial for humans on their way to physical understanding, but also to machines that are programmed to semantically enrich mathematical and especially, physical content.

In this paper, we consider physical units in Wikipedia, as a first step towards a general solution of the underlying problem. We aim to: (1) identify formulae that deal with physical relationships; (2) automatically derive the units of the identifiers used in those formulae; and (3) integrate and store the learned data in the central Wikimedia triple store Wikidata.

Our paper is structured as follows. First, we analyze related works that can be used to complete the task at hand. In that context, we recap how possible definitions can automatically be extracted from the text surrounding formulae. Thereafter, we describe our method to relate the identifier to dimensions using the Wikidata knowledge base and our approach to unit constraint propagation. This will be followed by a refinement of the definition candidates based on the

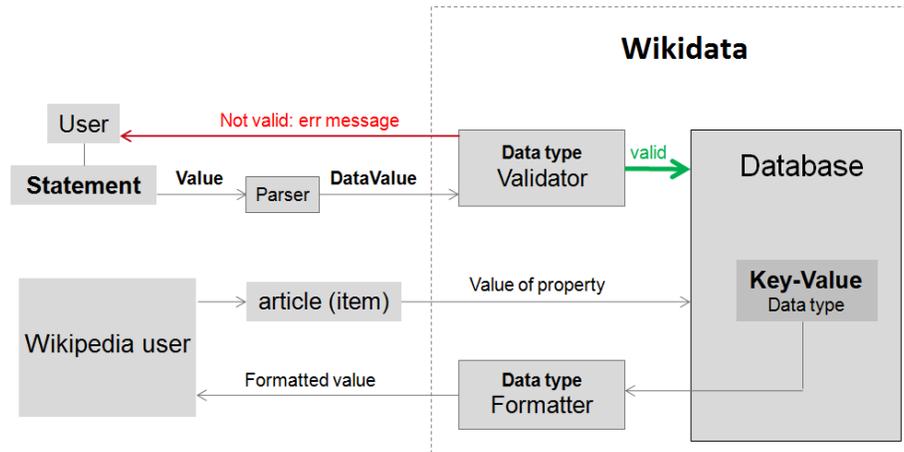


Fig. 1: As of January 2016, Wikidata users (visualized by the grey box ‘User’, top left in the figure) can store mathematical expressions in Wikidata. Thereafter, these expressions can be displayed in all language versions of Wikipedia (visualized by “Wikipedia user”, bottom left). Moreover, other use cases for this data are possible. For example, there is the article place-holder service which displays information regarding a topic in languages, for which human generated articles in that language do not exist yet. For more details see [6] (picture by Julian Hilbig and Duc Linh Tran [6]).

constraints. Lastly, we describe the reinsertion of learned units into the Wikidata knowledge base, which will be used for the formulae we processed. Finally, we provide an outlook on how this method can be used for feedback driven self-tuning of Mathematical Language Processing [11].

2 Related Work

A method to find *identifier definiens tuples* in natural language text is proposed in [11]. There, the authors use Natural Language Processing (part of speech tagging combined with word distance based scoring) to get the tuples. This approach shows advantages over the more static pattern-matching approaches, because it is able to also retrieve results that do not follow the pattern $\langle identifier \rangle$ is $\langle description \rangle$ (see also [8]).

This method has been applied to Wikipedia articles to enrich formulae with definitions for their included identifiers. It works for Wikipedia sites with different languages. The result may have more than a single possible definition for an identifier, each with a probability that expresses the likeliness for the definition to be the relevant. The selection of the correct definition is a problem addressed in our paper.

A method to map the meaning to identifiers is used in [12]. In that paper the namespace concept known from programming languages is used to assign documents to namespaces and thus the meaning of the identifiers are mapped to the meaning belonging to the chosen namespace.

In [9], an algorithm is proposed to use the need for dimensional homogeneity in physical formulae compared with constraint propagation to prove formulae that students gave as answers to physics problems. It aims on validating the formulae for known units.

Dimensional analysis is also a widely investigated field in the area of programming languages. In [4], the authors use the need for dimensional homogeneity in physical equations in conjunction with constraint solving to automatically infer unit types for programs handling scientific problems. Their approach infers a general set of unit types using constraints created over the variables and constants occurring in the program. The user can then annotate the inferred unit types with real units thereby cannot violate the dimensional correctness as it is proved for the inferred unit type system. In [1], constraint solving is used to prove the unit correctness of calculations in spreadsheets. More examples for validating dimensional correctness in programs can be found in [3, 7].

As preparation of this work, a new feature in Wikidata, the data-type mathematical expression, has been developed [6]. Properties with data-type mathematical expression with for instance, *defining formulae*, represent mathematical expressions. As of January 2016, these expressions are rendered by the software which runs Wikidata.

3 Our Method

3.1 Identify Physical Formulae

In this paper, we limit ourselves to the following definition.

Definition 1 (physical formula). *A physical formula is a binary mathematical relation of type equation or inequality containing one or more physical quantities.*

Consider the following examples:

$$E = mc^2, \tag{1}$$

$$\sin^2 \theta + \cos^2 \theta = 1, \tag{2}$$

$$m_{\text{moon}} < m_{\text{earth}} < m_{\text{sun}}, \tag{3}$$

$$\hbar = \frac{\lambda p}{2\pi}. \tag{4}$$

Expressions (1), (4) are physical formulae according to Definition 1. They contain the **physical quantities** $E, m, c, \lambda, p, \hbar$ of which c, \hbar are **physical constants**. Expressions (2), (3) are not physical formulae since (2) does not contain physical quantities, and (3) is not binary. Note that the above definition can be extended to non-binary relation chains without loss of generality.

This definition implies the following algorithm to identify physical formulae:

1. identify mathematical expressions;
2. check if they are binary relations of type equation or inequality;
3. extract identifiers; and
4. decide for each identifier, if it is a physical quantity or expression.

While we will apply the heuristics from [12] for steps 1 and 3, we need to develop new approaches for steps 2 and 4. A simple approach to 2, is to convert the mathematical expression to content MATHML using \LaTeX ML [10] and afterwards analyze the content MATHML tree using fixed rules. We thereby rely on \LaTeX ML. Possibly occurring problems and limitations of \LaTeX ML for our application will be listed in the final report. The main focus of our work will be on step 4. While we can find Wikidata items from the algorithm presented in [12], we might need to improve these algorithms.

Our main focus is on the development of an algorithm which decides if an item is a physical quantity or entity. Our approach to address this problem is to analyze the semantic properties of the relevant Wikidata item using the SPARQL [5]³ endpoint. This means all information from Wikidata that expose information on the units or dimensions respectively. More technically, we will develop a method to check the relatedness to Q107715 (physical quantity)⁴. This will be one of the key contributions of our research project.

3.2 Identifying units and dimension of physical quantities

Table 1: Dimension, base unit, and symbol according to the international system of units (SI)

Dimension	Unit	Symbol
Length	meter	L
Mass	kilogram	M
Time	second	T
Electric Current	ampere	I
Luminous Intensity	candela	J
Temperature	kelvin	θ
Amount of Substance	mole	N

The dimension of a physical quantity is an inherent property of each quantity. Dimensions are for example length L , mass M or time T . The derived quantity

³ SPARQL is used to query RDF (Resource Description Framework) triples. They consist of subject, predicate, object. Example: Find all subjects (items) in Wikidata that have predicate "subclass of" and object "physical quantity"

⁴ Wikidata stores information in form of triples like ("length", "subclass of", "physical quantity") where the unique id of "physical quantity" in Wikidata is Q107715. If a Item like "length" has a relation like "subclass of" or "instance of" to the Item for "physical quantity" we suppose it to be of kind physical quantity.

speed has the dimension LT^{-1} . For all physical quantities, we try to derive their dimension from Wikidata using SPARQL queries.

To obtain that, we also take unit information into account, since it is more prevalent in the Wikidata dataset compared to dimension information. Because there are multiple unit systems (e.g., imperial units using yard for length versus SI units using meter) more than one unit per dimension exists. However, physical laws are usually valid independent from the unit system that is used. In this context, it has to be noted that some adjustment needs to be done for units that disregard conceptually important physical properties. For instance the Carnot efficiency η_{Carnot} depends on the absolute temperature scale (e.g., kelvin) of the hot T_{H} and the cold T_{C} reservoir via

$$\eta_{\text{Carnot}} = 1 - \frac{T_{\text{C}}}{T_{\text{H}}}. \quad (5)$$

Note that the fact that those temperatures are based on an absolute temperature scale is essential. This requires that non-cardinal temperature units such as Celsius and Fahrenheit to be converted to prior to computation.

Given the extracted dimensions, the mathematical domain of physical quantities can be specified better. Approaches to formally describe this domain are presented in [2]. We introduce the following notation for a physical quantity x

$$x \equiv \begin{bmatrix} \mathfrak{x} \\ d \end{bmatrix},$$

where \mathfrak{x} is the *spatial part* of x as defined in [2] and d is the dimension of the unit of x . To simplify readability, we use the identifier for x and its spatial part. We write $x = \mathfrak{x}$ if $d = 1$. Thus, (1) can be written as

$$\begin{bmatrix} \mathbf{E} \\ ML^2T^{-2} \end{bmatrix} = \begin{bmatrix} \mathfrak{m} \\ M \end{bmatrix} \begin{bmatrix} \mathfrak{c} \\ LT^{-1} \end{bmatrix}^2,$$

and (5) reads

$$\eta_{\text{Carnot}} = 1 - \frac{\begin{bmatrix} T_{\text{C}} \\ \theta \end{bmatrix}}{\begin{bmatrix} T_{\text{H}} \\ \theta \end{bmatrix}} = 1 - \begin{bmatrix} \frac{T_{\text{C}}}{T_{\text{H}}} \\ \theta\theta^{-1} \end{bmatrix} = 1 - \frac{T_{\text{C}}}{T_{\text{H}}}. \quad (6)$$

3.3 Compatible operations

This notation leads to our next definition

Definition 2 (valid physical formula). *We call a physical formula valid, if the dimensions are compatible with the mathematical operators used in that formula.*

Table 2: Compatibility of Mathematical Operations with Physical units. Here, $|\cdot|_1$ denotes the 1-norm.

class	rule	constraint	operators
3: map	$o\left(\begin{bmatrix} \mathbf{a} \\ x \end{bmatrix}, \begin{bmatrix} \mathbf{b} \\ y \end{bmatrix}\right) \rightarrow \begin{bmatrix} \mathbf{o}(\mathbf{a}, \mathbf{b}) \\ \frac{\mathbf{o}(x, y)}{ \mathbf{o}(x, y) _1} \end{bmatrix}$		times, division, integration, differentiation
2: restrict	$o\left(\begin{bmatrix} \mathbf{a} \\ x \end{bmatrix}, \begin{bmatrix} \mathbf{b} \\ y \end{bmatrix}\right) \rightarrow \begin{bmatrix} \mathbf{o}(\mathbf{a}, \mathbf{b}) \\ x \end{bmatrix}$	$x = y$	plus, minus, equals
1: apply	$o\left(\begin{bmatrix} \mathbf{a} \\ x \end{bmatrix}, \begin{bmatrix} \mathbf{b} \\ y \end{bmatrix}\right) \rightarrow \begin{bmatrix} \mathbf{o}(\mathbf{a}, \mathbf{b}) \\ \mathbf{o}(x, \mathbf{b}) \end{bmatrix}$	$y = 1$	power, roots
0: unitless	$o\left(\begin{bmatrix} \mathbf{a} \\ y \end{bmatrix}\right) \rightarrow \begin{bmatrix} \mathbf{o}(\mathbf{a}) \\ y \end{bmatrix}$	$y = 1$	function application

We exemplify the meaning of compatible operations based on (1). This physical formula contains the mathematical operators equals (=), times (\cdot), and power (\wedge). For ‘equals’, the dimensions on the right-hand side and left-hand side must be the same, for ‘times’, no restrictions apply and for ‘power’ the unit of the exponent must be 1. If any of these constraints are violated (e.g., with the incorrect $E = mc$) then the units or the formula can not be correct. Note also that this method is not limited to scalar physical quantities and can be applied to physical quantities of higher dimensions such as vectors like

$$\begin{bmatrix} \mathbf{W} \\ ML^2T^{-2} \end{bmatrix} = \int_C \begin{bmatrix} \mathbf{F} \\ MLT^{-2} \end{bmatrix} d \begin{bmatrix} \mathbf{s} \\ L \end{bmatrix} = \int_{\begin{bmatrix} \mathbf{v} \\ T \end{bmatrix}}^{\begin{bmatrix} \mathbf{v}_2 \\ T \end{bmatrix}} \begin{bmatrix} \mathbf{F} \\ MLT^{-2} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ LT^{-1} \end{bmatrix} d \begin{bmatrix} \mathbf{t} \\ T \end{bmatrix}.$$

We will implement validation for the mathematical operations enumerated in Table 2, which we group into four classes. For some of those cases [9] defined detailed unit propagation rules.

3.4 Unit Inference

After having defined those fundamental concepts, we apply them to the actual data extracted by the Mathematical Language Processing Project and the Unit information fetched from Wikidata.

As an example, we demonstrate the work-flow for adding dimensional information to formulae extracted from Wikipedia. The result of the process, proposed in [11], is a probability distribution over identifiers and their possible definiens. An identifier can have more than one definition candidate. That may lead to more than one possible dimension for an identifier.

Example 1: Mass-energy equivalence

The relation between energy and mass is described by the mass-energy equivalence formula $E = mc^2$, where E is energy, m is mass, and c is the speed of light.

For Example 1 (from Wikipedia), which was also used in [12], the identifier-definiens pairs for E may have this form:

Id.	Definition	score	Id.	Definition	score	Id.	Definition	score
E	energy	0.42	m	energy	0.35	c	energy	0.30
E	mass	0.42	m	mass	0.35	c	mass	0.35
E	speed of light	0.16	m	speed of light	0.30	c	speed of light	0.35

All three definitions are found in the same sentence, but have different distances between identifier and definiens. That results in different scores. Note, that the actual scoring computation is more evolved and the score values have been made up for demonstration purposes. We now describe this more formally.

For a physical formula $f(x_1, \dots, x_n, o_{n+1}, \dots, o_m)$, with the identifiers (physical quantities and other identifiers) x_i and mathematical operators o_j , and a set of definiens candidates \mathcal{N} , the MLP Project returns a probability distribution for the identifiers $\chi(\underline{x}) = \sum_{N \in \mathcal{N}} \underline{w}_{0,n} N$, with $\sum_{N \in \mathcal{N}} w_{0,N} = 1$. From Wikidata we get the unit and implied dimension information denoted as $\rho_1(\underline{x}) = \sum_{N \in \mathcal{N}} \underline{w}_{0,n} \dim(N) = \sum_i \underline{w}_{1,i} d_i$, where $\underline{w}_{1,j}$ is the sum over all $\underline{w}_{0,N}$ with $\dim N = d_j$. Next, we apply the operator compatibility rules, which affects the probability distribution according to their constraints. The dimension probability distribution of an operator o , $\dim(o(a, b))$ implies operator dependent constraints to $\dim a$ and $\dim b$.

$$\text{class}(o) = 3 \implies \dim o(a, b) = o(\dim a, \dim b).$$

$$\text{class}(o) = 2 \implies \dim a = \dim b.$$

$$\text{class}(o) = 1 \implies \dim b = 1.$$

$$\text{class}(o) = 0 \implies \dim a = 1, o(a, b) = o(a).$$

To reflect that, we define a refined probability distribution $\rho(\underline{x}) = \sum_i \underline{w}_{2,i} d_i$, with $\sum_i \underline{w}_{2,i} d_i = 1$. Finally, we need to solve a system of linear equations describing the $\underline{w}_2 = \mathbf{A} \underline{w}_1$. We propose the following method to realize that. Assume that the mathematical notation is good enough to extract the operator tree, with tools such as L^AT_EX_ML. We convert this operator to the root form and call $\underline{Q}(f) \in \mathbb{N}^{m \times m}$ the adjacent matrix. Since the tree is not directed $\underline{Q}(f) = \underline{Q}(f)^T$, and since identifiers are always connected by an operator, the top left entries of the adjacent matrix are zero, i.e., $\forall i \leq n \wedge j \leq n \implies \underline{Q}(f)_{i,j} = 0$. Based on that, we will elaborate different strategies for the most efficient execution of the constraint propagation problem. Our first approach is the following sketch of an algorithm:

1. start with the identifiers (\underline{x}) as working set;

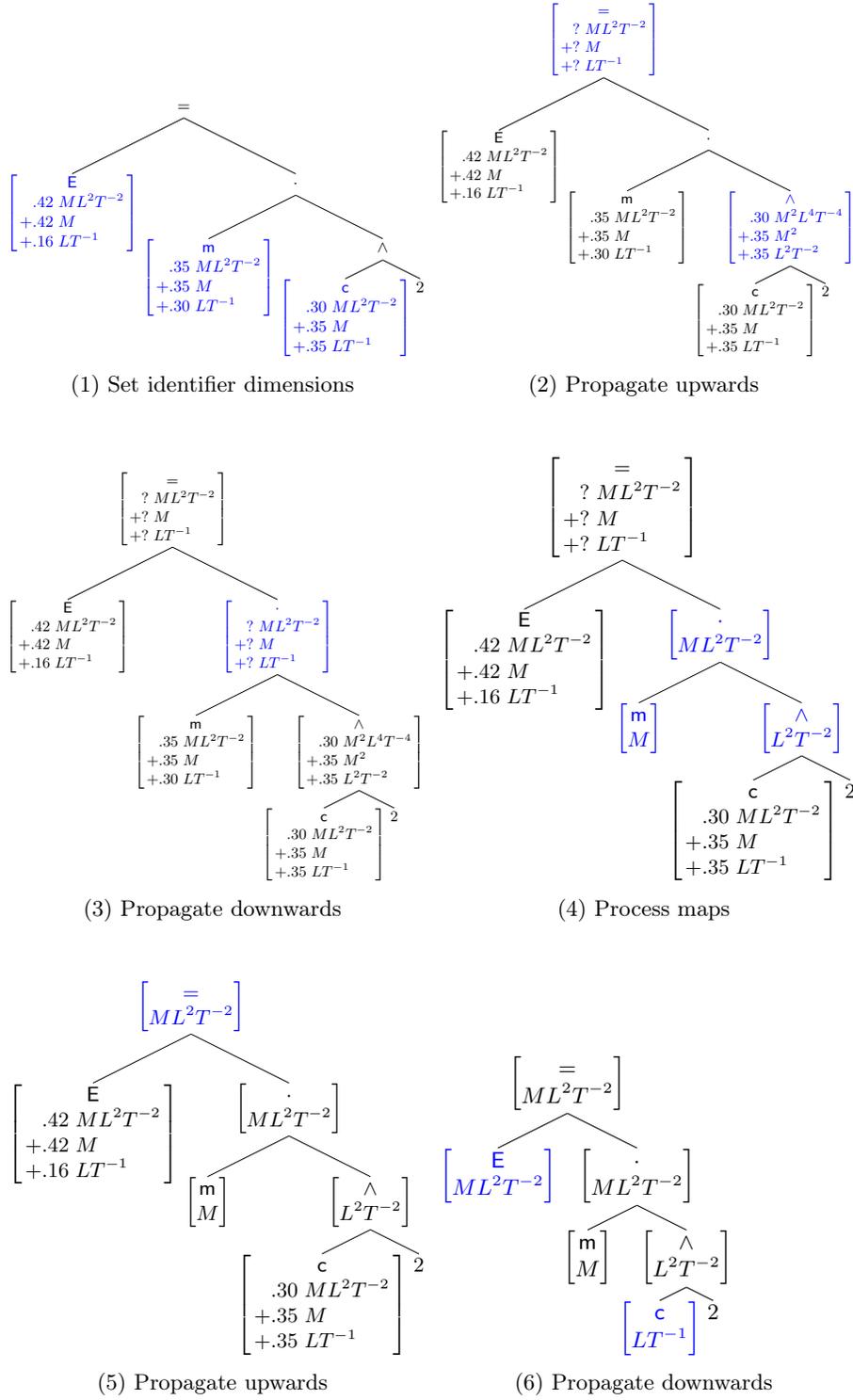


Fig. 2: Demonstration of our constraint propagation algorithm.

2. propagate the constraints to their parent operators of classes 0-2 and remember the parent operators of class 3 (maps);
3. update the working set to consist of the updated parents;
4. propagate the constraints downwards;
5. the changed children are now the new working set;
6. continue with upwards and downwards propagation until the working set is empty, and a steady state is reached;
7. update the working set with the class 3 operators;
8. resolve the class 3 operators, in appropriate order (note that this might lead to many possible dimensions for the class 3 operators);
9. propagate the new constraints by going back to step 2;
10. the algorithm terminates if the working set and the set of unprocessed class 3 operators is empty.

We demonstrate this algorithm in figure 2 based on Example 1. Formula (1) depends on $(E, m, c, 2, =, \cdot, \wedge)$ and the adjacent matrix reads

$$O("E = mc^2") = O \begin{pmatrix} = \\ E \widehat{\cdot} \\ m \widehat{\wedge} \\ \widehat{c} 2 \end{pmatrix} = \begin{matrix} E \\ m \\ c \\ = \\ \cdot \\ \wedge \end{matrix} \begin{matrix} E & m & c & 2 & = & \cdot & \wedge \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{matrix}.$$

One alternative to this approach is [9]. The evaluation of our approach must show the performance of the probabilistic approach.

3.5 Wikidata insert/update

After having propagated the units, we might have found a unique solution as demonstrated in Example 1. In cases where we found this unique solution, we will write back the dimension information to Wikidata. If the formula already has an item in Wikidata, we check the properties and update/insert the missing information. Since we use the Wikidata database for finding information for all the identifiers in a formula we have automatically a defining item to link to for every identifier.

3.6 Limitations

For level 3 operators, numerical artifacts (such as scalar factors from the integration) and probability density are mixed. For example in $\int s dt$ and a hypothetical $\rho_0(s) = .5M + .5T$ leads to a propagated probability of

$$\rho \left(\int s dt \right) = \frac{\int \rho_0(s) dt}{\left| \int \rho_0(s) dt \right|_1} = \frac{4}{3} \left(\frac{1}{2} MT + \frac{1}{4} T^2 \right) = \frac{2}{3} MT + \frac{1}{3} T^2.$$

The likelihood for MT is higher in comparison to T^2 , which does not seem plausible at the first place. Consequently, we will search for better suitable ways to re-scale the unit vector to length 1.

4 Future work

The finding of items in Wikidata can be improved by using methods like stemming to match the definition with the item caption. This is due to the fact that the definitions are extracted from free-text and may be conjugated. The identification of the units/dimensions can be done by the querying property, *has quality*. Due to the community-driven inserts of items, it is not guaranteed that every item has all of the possible properties. For example, the authors may omit a property such as, *has quality*. Instead the property, *quantity symbol*, may be queried.

Moreover, our algorithm can be used as a feedback mechanism for the MLP process. With the additional unit and dimension information, the algorithm will learn about mistakes from the unit checking. With this information, the algorithm will be able to tune itself.

5 Conclusion

The knowledge about units in physical formulae is fundamental. However, in most cases they are not marked up explicitly. We investigated how they can be determined automatically. We described a process to automatically infer the unit information for identifiers in physical formulae. This process is based on formulae and identifier-definition pairs that are extracted from text using Natural Language Processing techniques. The process may result in multiple definition candidates for identifiers. To infer a consistent identifier-definition mapping, we use an algorithm based on the principle of dimensional homogeneity in physical formulae. The unit/dimensional information for the definitions is retrieved from the structured data store Wikidata. The results are then used to extend Wikidata with the formulae and links Wikidata entries describing the identifiers.

Acknowledgements. We would like to thank Volker Markl, Michael Kohlhase and Abdou Youssef for their support of this research project.

Bibliography

- [1] T. Antoniu, P. A. Steckler, S. Krishnamurthi, E. Neuwirth, and M. Felleisen. Validating the unit correctness of spreadsheet programs. In A. Finkelstein, J. Estublier, and D. S. Rosenblum, editors, *26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom*, pages 439–448. IEEE Computer Society, 2004.
- [2] J. B. Collins. A mathematical type for physical variables. In S. Autexier, J. A. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics, 9th International Conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 - August 1, 2008. Proceedings*, volume 5144 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2008.

- [3] M. Contrastin, A. C. Rice, M. Danish, and D. A. Orchard. Units-of-measure correctness in fortran programs. *Computing in Science and Engineering*, 18(1):102–107, 2016.
- [4] P. Guo and S. McCamant. Annotation-less unit type inference for c. In *Final Project, 6.883: Program Analysis, CSAIL, MIT*, 2005.
- [5] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. <https://www.w3.org/TR/sparql11-query>. seen May, 2016.
- [6] J. Hilbig and D. L. Tran. Mathematical expression as new data type for WikiData - Database project - supervised by Moritz Schubotz. Technical Report Winter-term 2015/2016, Technische Universität Berlin, feb 2016. <https://github.com/TU-Berlin/WikidataMath/releases/download/v1.0.0/ReportWikiDataDBPRO.pdf>.
- [7] L. Jiang and Z. Su. Osprey: A practical type system for validating dimensional unit correctness of C programs. In *Proceedings of the International Conference on Software Engineering*, 2006.
- [8] G. Y. Kristianto, G. Topic, and A. Aizawa. Extracting textual descriptions of mathematical expressions in scientific papers. *D-Lib Magazine*, 20(11/12), 2014.
- [9] C. W. Liew. Checking for dimensional correctness in physics equations. In *In Proceedings of Fourteenth International Florida AI Research Society Conference*, 2002.
- [10] B. R. Miller. LaTeXML: A L^AT_EX to XML converter. <http://dlmf.nist.gov/LaTeXML>. seen May, 2016.
- [11] R. Pagel and M. Schubotz. Mathematical language processing project. In M. England, J. H. Davenport, A. Kohlhase, M. Kohlhase, P. Libbrecht, W. Neuper, P. Quaresma, A. P. Sexton, P. Sojka, J. Urban, and S. M. Watt, editors, *Joint Proceedings of the MathUI, OpenMath and ThEdu Workshops and Work in Progress track at CICM*, number 1186 in CEUR Workshop Proceedings, Aachen, 2014.
- [12] M. Schubotz, A. Grigorev, M. Leich, H. S. Cohl, N. Meuschke, B. Gipp, A. S. Youssef, and V. Markl. Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016.

MKM Work In Progress

Mathematical Knowledge Management is an interdisciplinary field of research in the intersection of mathematics, computer science, library science, and scientific publishing. The objective of MKM is to develop new and better ways of managing sophisticated mathematical knowledge, based on innovative technology of computer science, the Internet, and intelligent knowledge processing. MKM is expected to serve mathematicians, scientists, and engineers who produce and use mathematical knowledge; educators and students who teach and learn mathematics; publishers who offer mathematical textbooks and disseminate new mathematical results; and librarians and mathematicians who catalog and organize mathematical knowledge.

The interests of the MKM Track of CICM necessarily, and intentionally, overlap those of other CICM tracks but generally with a broader point of view. Authors of papers dealing more specifically with Theorem Proving or Mathematical Libraries may consider submitting to the CALCULEMUS or DML tracks, respectively. Work involving prototypes of or enhancements to systems may wish to submit to the Systems and Data track, possibly as a complementary paper.

July 2016, Bruce Miller

Lemma Extraction Criteria Based on Properties of Theorem Statements*

Karol Pałk

Institute of Informatics,
University of Białystok, Białystok, Poland
pakkarol@uwb.edu.pl

Abstract. Both easily readable and obscure proof scripts can be found in the Mizar Mathematical Library (MML). Many authors do not want to invest additional efforts in improving readability of their deductions, once the computer accepts their scripts. In their opinion, text of the proofs are ignored by most of the readers and the readability is essential only at the top level of scripts where statements of theorems are formulated. However, the analysis of such scripts is unavoidable if we have to rebuild some theorems to make them stronger or more easily applicable. Therefore, it is important to develop tools that can improve legibility of proofs, in particular those that shorten reasoning by removing technical sub-deductions, and this requires development of criteria that lead to extraction of statements which, in the opinion of human readers, describe well the extracted reasoning.

We propose characteristics of formula complexity that can be applied to determine which sub-deductions should be extracted so that resulting lemmas are more comprehensible. To better understand their significance we study the distribution of these characteristics on statements of theorems that are collected in the MML.

Key words: Lemma extraction, Complexity of formulas, Legibility of proofs

1 Introduction

1.1 Motivations

The legibility of proof scripts might be considered as one of the most important factors of formalization quality, but in practice the growth of proof databases is not always accompanied by the improvement of the formalization quality of the articles. Analyzing proof scripts developed with proof assistants, especially the longer and more complex deductions, leads to a conclusion that their legibility often seems to be of the secondary importance to their authors since computer assisted proof development frameworks can check the correctness of such deductions. According to the opinion of some proof authors any attempt to analyze

* The paper has been financed by the resources of the Polish National Science Center granted by decision n°DEC-2012/07/N/ST6/02147.

details of the proofs scripts created in this way is extremely difficult or even impossible. However, analysis of such proofs is unavoidable if we try to adapt or modify them [5].¹

This concerns especially systems such as Mizar [2, 10], where the proof script language is close to the natural language, which makes it possible to create legible deductions. Therefore, authors of Mizar proof scripts can manually try to improve the comprehensibility of their work spending a lot of time over their readability [13, 6], similar as it is done for informal mathematical proofs. However, many authors do not want to invest additional efforts in this process and assume that the task can be handled automatically for them, since having a digital form of structured formal proof, a computer can not only automatically verify the correctness, but also automatically enhance proof scripts. Therefore, it comes as no surprise that Mizar is being developed in many directions to meet these needs of users, similar to the way people work on informal mathematical proofs.

We can identify three main directions of development to improve legibility. The first one is based on improving the representation of proof scripts in HTML format [14] by adding selected information that is automatically generated by Mizar and also by bringing the formal mathematical language to the informal one by introduction to the formal language idioms that stem from informal mathematical practice [7–9]. The second direction is based on the simplification of deductions in proof scripts by finding and removing irrelevant parts of reasoning or by elimination of redundant premises from the justification of steps, preserving the correctness of the modified proof scripts. The third direction is based on rebuilding the deduction structure in proof scripts by changing the order of independent steps in reasoning and also by detecting reasoning passages (called *packets*) that are, e.g., technical and repeated many times in a reasoning, and extracting them as lemmas or encapsulating them on a deeper level of a proof in the form of a nested lemma.

The last direction is still the least developed. SMT solvers can be used to choose a better order of independent proof steps [12]. A method of extracting packets as external or nested lemmas so that the correctness of proof scripts is preserved has also been developed [11]. However, an additional challenge remains to formulate packet extraction criteria so that resulting new lemmas can be accepted as ones that deserve readers' attention and are worth extracting. In this paper we concentrate on this aspect.

1.2 Proposed approach

The ability to find such passages automatically is crucial, since the extraction of carelessly selected packets can drastically reduce the proof scripts legibility even if the modification reduces the length of the proof. Additionally, the statements associated with the reasoning in a packet has usually a more complicated form

¹ Actually this is mentioned in Page 3 of an unpublished preliminary version of the article [4].

than a plain implication *premises* \implies *conclusions* preceded by a sequence of universal quantifiers. It turns out that a mathematical statement can be perceptibly more complex than another one, even if both have the same number of assumptions and theses; or their prenex normal forms are at the same level of the same arithmetic hierarchy [1, 11].

In this paper we analyze the selected characteristic of sentence complexity of theorems and lemmas in the Mizar Mathematical Library (MML) to get the most typical values. and then apply them in the context of lemma extraction based upon the notion of proof graph. In Section 2 we introduce the notion of an abstract model of proofs and packets. In Section 3 we discuss selected indicators of the packet's statement complexity that describe a structure generated by premises and conclusion in a packet. We discuss also the impact of the positions of quantifiers in a formula for the complexity of a deduction that justifies the formula. Then in Section 4 we report the statistical results obtained on statements of theorems and lemmas occurring in the MML. Finally, Section 5 concludes the paper and discusses future work.

2 Packets in Abstract Proof Graph

To formulate the notion of a packet, we have to fix the terminology and notation.

Let $G = \langle V, E \rangle$ be a DAG, E_1 be a subset of E , V_1 be as subset of V . An arc is called E_1 -arc if it belongs to E_1 . A path $P = \langle u_1, u_2, \dots, u_n \rangle$ of G is called an E_1 -path if $\langle u_i, u_{i+1} \rangle$ is an E_1 -arc for $i = 1, 2, \dots, n - 1$. Additionally, we say that P passes V_1 if u_2, u_3, \dots, u_{n-1} belong to V_1 . For vertices u, v in V , the notation $u \xrightarrow{E_1} v$ means that $\langle u, v \rangle$ is an E_1 -arc. Moreover, the notation $u \xrightarrow{V_1}^* v$ means that there exists an E_1 -path that leads from u to v . Additionally, we say that vertices u, v are connected only by passing V_1 and denoted this by $u \rightsquigarrow_{V_1}^* v$, if there exists a path that leads from u to v and every path that leads from u to v passes V_1 .

An abstract model of proofs was considered in detail in an earlier work [11]. For our purposes we recall only the part of its definition that is the most relevant for our purposes. We illustrate it with an example on Fig. 1, where P, Q, R, S, T represent predicate symbols with two arguments and F represents a functor with one argument. Note also that the additional packet \mathcal{P} is analyzed in the further part of this article. Generally, we call a DAG $\mathfrak{P} = \langle \mathbb{V}, \mathbb{O} \cup \mathbb{M} \rangle$ an *abstract proof graph* if \mathbb{O} and \mathbb{M} are disjoint families of arcs, called *ordered arcs* and *meta-edges* respectively, $\langle \mathbb{V}, \mathbb{M} \rangle$ is a forest, and \mathbb{O} contains a distinguished set of arcs $\mathfrak{R}(\mathfrak{P}) \subseteq \mathbb{O}$, the elements of which are called *references*. Additionally, the contents of a nested reasoning are closed outside, i.e., introduced variables and formulated statements in a sub-reasoning cannot be used outside the sub-reasoning, i.e., from areas of the proof in which the sub-reasoning is nested (for each $u, v, w \in V$ if $u \xrightarrow{\mathbb{O}} v$, $u \xrightarrow{\mathbb{M}} w$ then $v \xrightarrow{\mathbb{M}}^* w$ and $v \neq w$). The vertices of \mathfrak{P} represent steps of the reasoning, \mathbb{O} -arcs represent the flow of information between different steps of the reasoning, and \mathbb{M} -arcs represent the dependence

α : let x be set;
 β : A1: $P[x]$;
 γ : consider y be set such that
 A2: $y=F(x)$ by A1;
 δ : A3: $Q[y]$ by A2;
 ϵ : consider z be Subset of y such that
 A4: $R[z]$ by A3;
 ζ : consider r be Relation of y, z such that
 A5: $S[r]$ by A4;
 η : A6: $T[r]$ by A5;

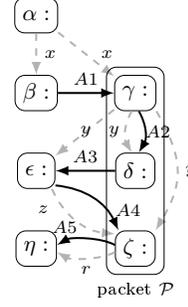


Fig. 1. The example of reasoning sequence written in the Mizar style and the corresponding fragment of the abstract proof graph that represents the reasoning.

between each step from areas of the proof and a proving fact. $\mathfrak{R}(\mathfrak{P})$ -arcs that correspond to solid arrows represent the information flow between a premise (e.g., the fact labeled by A1 in β) and a place of its use (γ). The other \mathbb{O} -arcs that correspond to dashed arrows represent all kinds of additional constraints that force one step to precede another one, e.g., the dependence between a step that introduces a variable (the variable v in α) into the reasoning and a step that uses this variable in its expression (β, γ). Note also that arcs and vertices of abstract proof graphs are not labeled (arcs and nodes in Fig. 1 are labeled only to simplify their identification).

Using the notion of meta-edges we can define formally the notion of packet introduced in an earlier study [11]. Let us fix the notation $\mathcal{D} = \langle V_{\mathcal{D}}, E_{\mathcal{D}} \rangle$ for a subgraph of \mathfrak{P} induced by a set of vertices. We call \mathcal{D} a *packet*, if \mathcal{D} is induced by the set of all roots in the forest $\langle \mathbb{V}, \mathbb{M} \rangle$ or every vertex of \mathcal{D} has a common successor in $\langle \mathbb{V}, \mathbb{M} \rangle$. Note that in an earlier definition (see [12]) the packet was a subset of steps in a one-level deduction, where we ignored each nested local lemma that was a justification of a step in the deduction. To consider the set of a packet steps together with steps of such supplementing lemmas we define an *area* of the packet by $\mathcal{A}(\mathcal{D}) := \{v \in \mathbb{V} : \exists_{u \in V_{\mathcal{D}}} v \xrightarrow{\mathbb{M}}^* u\}$. A vertex v of \mathfrak{P} is called a \mathcal{D} -premise, if $v \notin V_{\mathcal{D}}$ and $\langle v, u \rangle$ is a $\mathfrak{R}(\mathfrak{P})$ -arc for some $u \in \mathcal{A}(\mathcal{D})$. Similarly, we call a vertex v of \mathfrak{P} a \mathcal{D} -conclusion, if $v \in V_{\mathcal{D}}$ and $\langle v, u \rangle$ is a $\mathfrak{R}(\mathfrak{P})$ -arc for some $u \in V_{\mathcal{D}} \setminus \mathcal{A}(\mathcal{D})$. Let v be a vertex of \mathbb{V} . A \mathcal{D} -premise p is called v -necessary if there exists $\mathbb{O} \cup \mathbb{M}$ -path that passes $\mathcal{A}(\mathcal{D})$ and connects p with v . Note that to explore every dependency between a premise and a conclusion, we cannot be limited only to $\mathfrak{R}(\mathfrak{P})$ -paths, even if reference arcs are sufficient to define premises and conclusions. As an illustration note that the step β presented in Fig. 1 is ζ -necessary, even if the reference arcs $\langle \delta, \epsilon \rangle, \langle \epsilon, \zeta \rangle$ do not occur in the proof graph of the packet, since there exists ordered arc $\langle \gamma, \zeta \rangle$. Note that the packet \mathcal{P} has also ζ -necessary premises ϵ , and β is also δ -necessary.

In our research, we distinguished also a set of vertices $\mathcal{V}(\mathfrak{P})$ that correspond to steps that introduce variables into a reasoning in both cases of steps that introduce an universal or an existential quantifier and of steps that introduce

a new constant. A vertex v of $\mathcal{V}(\mathfrak{P})$ is called a \mathcal{D} -universal, if $v \notin V_{\mathcal{D}}$ and $\langle v, u \rangle$ is a $\mathbb{O} \setminus \mathfrak{R}(\mathfrak{P})$ -arc for some $u \in \mathcal{A}(\mathcal{D})$. Similarly, we call a vertex v of $\mathcal{V}(\mathfrak{P})$ \mathcal{D} -existential, if $v \in V_{\mathcal{D}}$ and $\langle v, u \rangle$ is a $\mathbb{O} \setminus \mathfrak{R}(\mathfrak{P})$ -arc for some $u \in V_{\mathcal{D}} \setminus \mathcal{A}(\mathcal{D})$. In Fig. 1, the packet \mathcal{P} has two universal vertices: α, ϵ and also two existential vertices: γ, ζ . Additionally, the order of these steps in graph suggests that the packet's statement must have the following form $\forall_x \exists_y \forall_z \exists_r \dots$.

3 Properties of the Packet's Statement

A method of extracting an arbitrary packet as an external or a nested lemma has been described in an earlier work [11]. However, the question of the packet's property which determines the choice of the extraction method is omitted. In this Section we describe the characteristic of the packet's statement complexity that describes a packet considered as a subgraph position in an abstract proof graph, i.e., the information flow between the packet and the rest of a reasoning that contains the packet.

Let us focus on the packet \mathcal{P} presented in Fig. 1, the reachability relation between packet's premises and conclusions, and also the reachability relation between ones that are connected only by passing the packet's area. Using the reasoning in a packet we can provide a packet's statement in the form "*quantifiers premises* \rightarrow *conclusions*" (e.g., $\dots (P(x) \wedge R(z)) \implies (Q(y) \wedge S(r))$). However, we cannot preserve the correctness of the modified reasoning, since $\delta(Q(y))$ is ϵ -necessary ($R(z)$) or more precisely, there exists an outgoing path $\langle \gamma, \epsilon, \zeta \rangle$ that generates a circle if we replace the packet by a single step (for more detail see the definition of lemma extraction procedure [11]). Since, the packet's statement has to preserve the *necessary* relation, we have at least two options for the formulation of the packet's statement:

$$\begin{aligned} & \dots (P[x] \text{ implies } Q[y]) \ \& \ (R[z] \text{ implies } S[r]) \\ & \dots P[x] \text{ implies } (Q[y] \ \& \ (R[z] \text{ implies } S[r])) \end{aligned} \tag{1}$$

Obviously, the first proposition is more general than the second one. However we can extract \mathcal{P} preserving the correctness in both cases. Analyzing the structure of implications we can observe that a deduction justifying the first statement should have a form of two disjoint proofs (corresponding to each implication) and the skeleton of a deduction justifying the second one has the form "assume $P[x]$; thus $Q[y]$; assume $R[z]$, thus $S[r]$ ", (see Fig. 3).

Analyzing the structure of implications we assume that the formula contains only negation, conjunction, implication; where negation can precede only a literal. Additionally, we eliminate every formula of the form $\alpha \implies (\beta \implies \gamma)$ using the *Exportation law* ($((\alpha \wedge \beta) \implies \gamma) \iff (\alpha \implies (\beta \implies \gamma))$) and also we eliminate repetitions of formulas as $(\alpha \implies \beta) \wedge (\alpha \implies \gamma)$. We say that such formula is *implicational*. Generally, we can transform a formula to several expected forms. However, this nondeterministic does not occur in majority statements of theorem occurring in the MML. Note that the equivalence in the Mizar system is a syntactic sugar for two implications, the disjunction is used

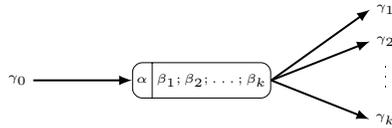
only in 2% of all theorem statements in the MML, and the negation precedes a non-literal formula only in justified cases where it facilitates the understanding of a theorem’s statement in the Mizar reviewer’s opinion.

Note also that when proving an implication, the most natural proof is the one where we first assume the antecedent. Obviously, we can conclude the consequent directly or using the *reductio ad absurdum* method. However, in both cases, the complexity of the antecedent has a negligible impact on the proof graph that describes the information flow in a reasoning. The only exception is when the antecedent is a conjunction of several facts, but even in that case the number of facts plays an important role than the complexity of each of them. Therefore, we omit the complexity of antecedents (accordingly, packet’s premises) in our structure that describes implicational formulas (see $P[x]$, $R[z]$ in Fig. 1).

Let us consider an implicational formula ϕ . Then to each sub-formula of ϕ that has the form:

$$(\gamma_0 \implies (\alpha \wedge (\beta_1 \implies \gamma_1) \wedge (\beta_2 \implies \gamma_2) \wedge (\beta_k \implies \gamma_k)))$$

we can associate a vertex of a directed rooted tree as follows



The height of obtained tree is called the *depth* of ϕ and the number of leafs there the *breadth* of ϕ . The first formula presented in (1) has depth 1 and breadth 2; and the second one has depth 2 and breadth 1 (see Fig. 2). We show that



Fig. 2. The structure of implicational formulas presented in (1)

depth and breadth represent the generality level of formula that distinguishes statements of lemmas and theorems. Note that in the most general packet’s statement, the formula is equivalent to the basic packet’s statement (for more detail see [11]), i.e., a conjunction of implications, where the consequent of a given implication is one of the packet’s conclusions c , and the antecedent is a conjunction of c -necessary the packet’s assumptions. In consequence, the breadth of the packet’s statement should be close to the number of packet’s assumptions and the depth of the packet’s statement should be close to one or even 0 if the packet has no assumptions. However, the modification of the reasoning part left after

the extraction of a packet with such statement is more complicated and a generated proof of the statement based upon the packet's reasoning is extremely long and has repeated passages that do not occur in existing proof scripts in the MML. Therefore, we analyze the depth and the breadth of the statement of a theorem that is collected in the MML in the context of a situation, where the statement is as general as possible and a generated proof does not have repeated passages. Obviously, the optimal values of depth and breadth are determined by properties of packets. Note also that we can indicate the formula that has breadth 1 (see (3)). However, the generality level of the formula is low. As an illustration, let us denote by $\mathcal{P}re(\mathcal{D})$ the set of \mathcal{D} -premises and similarly denote by $\mathcal{C}on(\mathcal{D})$ the set of \mathcal{D} -conclusions. We define two recursive families of vertices $\{\mathcal{P}re(\mathcal{D})^i\}_{i=1}^\infty$, $\{\mathcal{C}on(\mathcal{D})^i\}_{i=1}^\infty$ as follows:

$$\begin{aligned} \mathcal{P}re(\mathcal{D})^0 &= \{p \in \mathcal{P}re(\mathcal{D}) : \neg \exists_{u \in \mathcal{C}on(\mathcal{D})} u \xrightarrow{\text{M}\cup\text{O}}^* p\}, \\ \mathcal{C}on(\mathcal{D})^i &= \{c \in \mathcal{C}on(\mathcal{D}) : \exists_{p \in \mathcal{P}re(\mathcal{D})^i} p \rightsquigarrow_{\mathcal{A}(\mathcal{D})}^* c\}, \\ \mathcal{P}re(\mathcal{D})^{i+1} &= \{p \in \mathcal{P}re(\mathcal{D}) : \exists_{c \in \mathcal{C}on(\mathcal{D})^i} c \rightsquigarrow_{\mathcal{A}(\mathcal{D})}^* p\}. \end{aligned} \quad (2)$$

Note that only a finite number of elements in the families can be non-empty, since $\mathcal{P}re(\mathcal{D}) \cup \mathcal{C}on(\mathcal{D})$ is finite. Additionally, there exists a number d such that $\mathcal{C}on(\mathcal{D})^0 \neq \emptyset$, $\mathcal{P}re(\mathcal{D})^i \neq \emptyset$, $\mathcal{C}on(\mathcal{D})^i \neq \emptyset$, for $i = 1, \dots, d$, and $\mathcal{P}re(\mathcal{D})^i = \mathcal{C}on(\mathcal{D})^i = \emptyset$, for $i > d$, since for each \mathcal{D} -premise p there exists at least one \mathcal{D} -conclusion c that p is c -necessary. Then the following formula has breadth 1.

$$\begin{aligned} &\mathcal{P}re(\mathcal{D})^0 \text{ implies } (\mathcal{C}on(\mathcal{D})^0 \ \& \ (\\ &\quad \mathcal{P}re(\mathcal{D})^1 \text{ implies } (\mathcal{C}on(\mathcal{D})^1 \ \& \ (\\ &\quad \quad \dots \\ &\quad \quad (\mathcal{P}re(\mathcal{D})^m \text{ implies } \mathcal{C}on(\mathcal{D})^d) \dots))) \end{aligned} \quad (3)$$

Additionally, the formula has the minimal depth among these statements of packet \mathcal{D} that have the breadth equal 1.

So far in this section we ignore information about variables. Generally, the packet's statement has to be preceded by a sequence of quantifiers that bind occurring variables. The packet extraction methods that take into consideration variables has been described in an earlier paper [11]. Obviously, the number of universal and existential vertices of a packet suggest two natural characteristics that correspond to the number of variables, which are bound by universal and existential quantifiers in the packet's statement. The arithmetical hierarchy of a theorem's statements, considered in the work of Alama [1], takes into consideration the complexity of the antecedent that we omit in considerations. It is important to note that in [11] only packet's statements not in prenex normal form were considered. Moreover, there is an *a priori* assumption that the existential quantifier corresponding to a \mathcal{D} -existential vertex e has to be preceded by every corresponding statement of e -necessary \mathcal{D} -premises. In this section we present only a simple justification of this assumption, i.e., without this assumption modification of reasoning is extremely complicated and generates unnatural proof scripts.

```

α0  LemmaP: for x be set st P[x] ex y be set st Q[y] &
      for z be Subset of y st R[z] holds ex r be Relation of y, z st S[r]
      proof
α1 :   let x be set;
α2 :   assume A1: P[x];
γ' :   consider y be set such that A2:y=F(x) by A1;
α3 :   take y;
δ' :   thus Q[y] by A2;
α4 :   let z be Subset of y;
α5 :   assume A3: R[z];
ζ' :   consider r be Relation of y, z such that A4: S[r] by A3;
α6 :   take y;
ζ'' :  thus S[r] by A4;
      end;
α :   let x be set;
β :   A1: P[x];
θ1 : consider y be set such that B1: Q[y] & for z be Subset of y st R[z] holds
      ex r be Relation of y, z st S[r] by A1, LemmaP;
δ :   A3: Q[y] by B1;
ε :   consider z be Subset of y such that A4: R[z] by A3;
θ2 : consider r be Relation of y, z such that B2: S[r] by B1;
η :   A6: T[r] by B2;

```

Fig. 3. The modification of the proof script (presented in Fig. 1) which represents the extraction of the packet \mathcal{P} , where the packet's statement is presented in (4).

According to the *a priori* assumption, in every statement of the packet \mathcal{P} presented in Fig. 1, the premise $P[x]$ (β) has to precede both existential quantifiers $ex\ y\ be\ set$; $ex\ r\ be\ Relation\ of\ y, z$ (γ, ζ) and the premise $R[z]$ (ϵ) has to precede only the second one. In consequence, we obtain the following formula:

$$\text{for } x \text{ be set st } P[x] \text{ ex } y \text{ be set st } Q[y] \ \& \ \text{for } z \text{ be Subset of } y \text{ st } R[z] \text{ holds ex } r \text{ be Relation of } y, z \text{ st } S[r] \quad (4)$$

that corresponds to the second formula in (1). A generated deduction that demonstrates the formula and a modified part of reasoning remaining after the packet's extraction is presented in Fig. 3. Note that the generated deduction has to contain six skeleton steps that correspond to universal (α_1, α_4) and existential (α_3, α_6) quantifiers; and correspond to antecedents (α_2, α_5). Similarly, variables (y, r) that are introduced to a reasoning in the area of a packet have to be reintroduced in the remaining parts of the reasoning after the packet's extraction (θ_1, θ_2). It is important to note that the resulting proof script, see Fig. 3, is the shortest of all proofs where the packet \mathcal{P} is extracted as a lemma.

4 Statistical Results and Discussion

In our study we analyze 55160 theorems and 53839 lemmas that are collected in the MML version 5.32.1234. For us, a “*theorem*” is only this item in the MML that is explicitly called theorem in the Mizar syntax. Every other step that

has a nested deduction as a justification, not only on the top level of its proof script, is called a *preselected lemma*. We call a preselected lemma “*lemma*” if the statement does not constitute a correctness condition or a property, where the statement is imposed by the Mizar syntax (for more detail see the current overview of the Mizar system [2]), e.g., in every definition of a functor, we have to demonstrate the existence and the uniqueness conditions, if the functor value is not defined by a term.

4.1 Premises and Conclusions

As we have expected, the analysis of the number of explicitly formulated premises in theorems and lemmas shows that lemmas have on average 1.8 times less premises than theorems do. Note that the average number of premises in lemmas is equal 0.76. Additionally, more than half of lemmas does not have premises and only 6.25% do not have more than 2, whereas for comparison 19.82% of theorems have more than 2 (see Fig. 4). The results for the average number of conclusions are not significantly different for lemmas and theorems (1.15 and 1.38 respectively), and in both cases the percentage of statements dramatically decreases with the increase of the number of conclusions. Obviously, not all premises are visible in the statement of a lemma. Indeed, if the statement of a step (1) is used as a premise in one of the steps of the nested reasoning that is the justification of a step (2), and both steps (1) and (2) are in the same level of nesting, then the premise does not occur in the statement of the step (2). Such hidden premises (1) are called *implicit premises*, in contrast to these that are formulated in the statement of the step (2), and these are called *explicit premises*. However, a similar situation occurs in the case of theorems if we use the previously proven facts that are formulated on the top level of proof scripts. Obviously, these facts can also be used in lemmas. But if we sum up *explicit* and *implicit* premises, then the percent of formulas in the MML with the same number of such premises does not distinguish statements of lemmas and theorems (see the ratio of percents (solid line) at the diagram that represent number of all premises presented at Fig. 4). However, the set of *implicit* premises can be defined in terms of the notion of packet \mathcal{D} and is equal to $Pre(\mathcal{D})^0$ (see (2)). Therefore, as an appropriate numerical characteristic of a packet we choose the cardinality of $Pre(\mathcal{D}) \setminus Pre(\mathcal{D})^0$.

We can also analyze the arithmetical hierarchy of *explicit* and *implicit* premises. However, the main part of premises contains identifiers of local constants (mainly lemmas) or reserved variables (theorems). In consequence, the main part of *implicit* lemma’s premises is at 0-level on the arithmetic hierarchy. Therefore, in our study, we preceded by a sequence of necessary universal quantifiers every *implicit* premise that contains such identifiers. But then the percent of *implicit* premises on the level of the arithmetic hierarchy that occurs in lemmas and theorems is almost identical.

A bit different is the case of *implicit* premises, since according to the assumptions of Section 3, we should analyze *explicit* premises as they were originally formulated by the author of a proof script. In that case the average level of

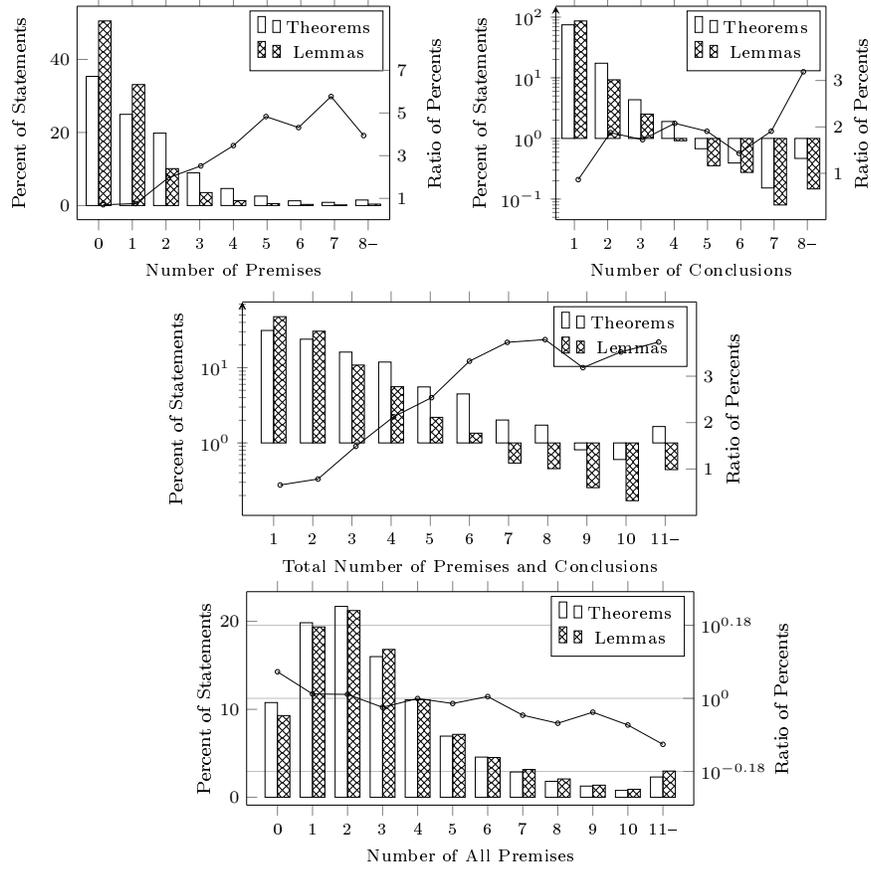


Fig. 4. The percent of statements (bars) with the same number of premises; conclusions; premises and conclusions together; *explicit* and *implicit* premises together; and also the ratio of these percents (solid lines) for theorems to lemmas.

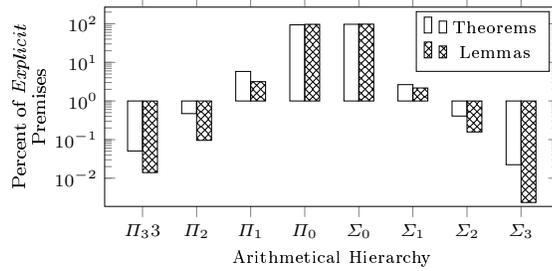


Fig. 5. The percent of *explicit* premises on the same level of the arithmetical hierarchy that occurs in lemmas and theorems.

arithmetic hierarchy is 0.116 and 0.027 for Π_* ; 0.0572 and 0.0196 for Σ_* in *explicit* premises occurring in lemmas and theorems, respectively, where for simplicity of notation a formula ϕ is Π_* (Σ_*) if ϕ is Π_i (Σ_i) formula for some i . Note also that non-literal premises, if they are Π_* (Σ_*), then they occur 1.92 (1.32) times more often in theorems than in lemmas. Additionally, the number of premises drastically decreases with increasing level of arithmetical hierarchy, on average 10.9 and 18.2 times in lemmas and theorems, respectively. Therefore, as a characteristic of a packet \mathcal{D} we should take into consideration not only the cardinality of $\mathcal{P}re(\mathcal{D}) \setminus \mathcal{P}re(\mathcal{D})^0$, but also the level of arithmetical hierarchy of statements formulated in vertices of $\mathcal{P}re(\mathcal{D}) \setminus \mathcal{P}re(\mathcal{D})^0$, to reduce the number of more complex one.

4.2 Variables bounded by universal and existential quantifiers

As in the previous section, we assume that every formula is preceded by a sequence of necessary universal quantifiers, which fix all local constants (mainly lemmas) or reserved variables, called together *parameters*. Note that the average number of parameters in lemmas and theorems is equal 3.40 and 1.54, respectively (see Fig. 6). Additionally, 54.26% of theorems have less than 2 parameters and only 4.64% have more than 4 parameters, whereas for comparison 48.36% of lemmas have 3 ± 1 parameters and 36.75% have more than 4 parameters.

For simplicity of notation, we called a variable *universally quantified* if it is a parameter or it is bounded by a universal quantifier and we call a variable *existentially quantified* if is bounded by an existential quantifier. Analyzing the number of variables bounded by universal quantifiers in formula, we obtain that percent of lemmas and theorems with the same value is almost identical, for the most popular values (3, 4, see Fig. 6). However, the ratios of these percents for theorems to lemmas is less than 1 for popular values (2–5). Note also that existentially quantified variables occur very rarely in formulas and the ratios of the statement percents with the same number of such variable for theorems to lemmas is less than 1 if and only if the number is less than 2.

4.3 The Depth and Breadth

As we have expected, the main part of statements have depth less than or equal to 1. Additionally only 7 theorems in the MML have the depth greater than 3 (3 of them describe differentiability in higher dimensions, for more detail see the relevant Mizar article [3]). Similarly, the main part of statements have breadth equal 1 and 42% of them are formulated without premises (37.25% theorems and 51.34% lemmas with breadth equal 1). Moreover, only 23.42% of formulas that have the depth greater than 0 or the breadth greater than 1 are used as lemmas (see Fig. 7).

We are aware that the described result is only a small step forward. However this is the first promising result concerning this questions. Analysis of human readers' opinions, especially the Mizar's users, does not make it possible to obtain more important results, which can be summarized by the statement: a lemma

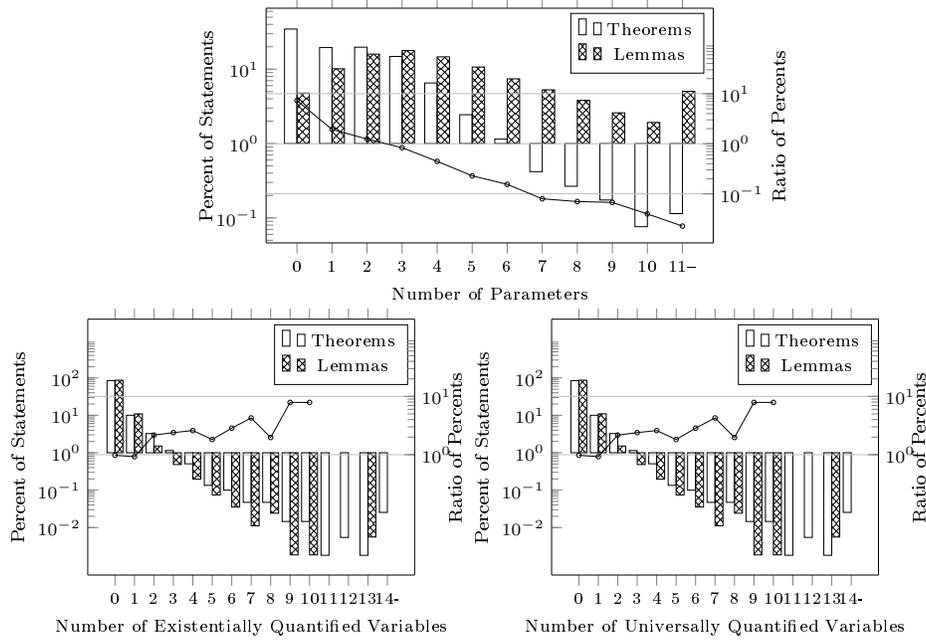


Fig. 6. The percent of statements (bars) with the same number of parameters; existentially quantified variables; universally quantified variables that occurs in lemmas and theorems, and also the ratio of these percents (solid lines) for theorems to lemmas.

corresponds to a separate fragment of reasoning between “two throats” in the proof that correspond to the premise and the thesis of the lemma (A. Trybulec). Additionally in [11] an artificial property of the package (the closeness of packets with respect to directed paths) has been described.

It is crucial in the process of packets’ extraction, but the determination of the influence of this property on the statement of existing lemmas was extremely counterintuitive. This impact is our “small step forward” and is described in Section 3 as a depth and breadth of formula. These two parameters seem to be also non-intuitive but they can be easily calculated for formulas and they will improve the result of a method that distinguishes formulas that occur in lemmas and theorems, and base only on the number of premises, conclusions and bound variables.

5 Conclusions

In this paper we describe a next stage in the research on methods that improve proof legibility realized in [11] that base on rebuilding the proof structure, either

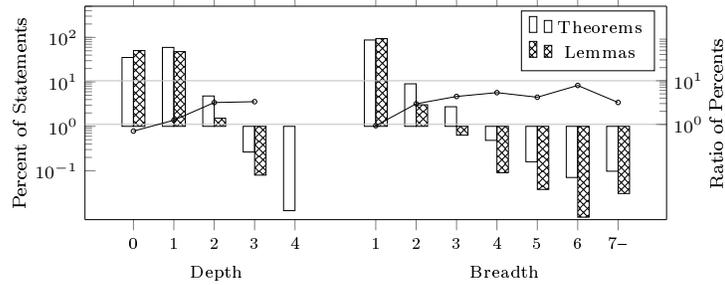


Fig. 7. The percent of statements (bars) with the same depth; breadth that occurs in lemmas and theorems, and also the ratio of these percents (solid lines) for theorems to lemmas.

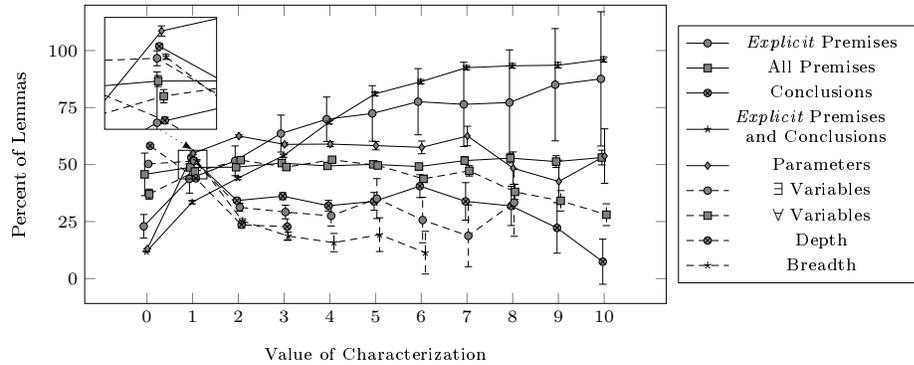


Fig. 8. The percent of statements with the same value of a characteristic that are used as lemmas and also 95% confidence intervals of the percent.

encapsulating sub-deductions (packets) in the form of a nested lemma or extracting them as lemma. We define characteristics of the packet’s statement and also we indicate the most appropriate values that are helpful in deciding whether to extract a packet as a lemma or not. Moreover, the proposed characteristics based on the packet’s statement determine a packet position in an abstract proof graph. This dependence is crucial, since in our approach we analyze the probability that in human readers’ opinions, a package should be extracted basing on the lemmas and theorems statement collected in the MML that are not generated from packets.

This research showed that for every packet we can calculate the probability of using in the MML the that the packet’ statement is used in the MML to formulate a theorem or a lemma; and also it indicates which of these two types is more probable, in respect to each proposed characteristic. The two non-intuitive characteristics of a statement have proved to be especially important. The depth and the breadth well improve distinction that based only on the num-

ber of premises, conclusions and bound variables. However, still an open problem to investigate is to determine impact of individual characteristics on the final qualitative assessment of a packet.

References

1. J. Alama. Sentence Complexity of Theorems in Mizar. *CoRR*, abs/1311.1915, 2013.
2. G. Bancerek, C. Bylinski, A. Grabowski, A. Kornilowicz, R. Matuszewski, A. Naumowicz, K. Pał, and J. Urban. Mizar: State-of-the-art and Beyond. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics - International Conference*, vol. 9150 of *LNCS*, 261–279, 2015.
3. N. Endou, H. Okazaki, and Y. Shidama. Higher-Order Partial Differentiation. *Formalized Mathematics*, 20(2):113–124, 2012.
4. G. Gonthier. A Computer-Checked Proof of the Four Colour Theorem. <http://research.microsoft.com/en-us/um/people/gonthier/4colproof.pdf>, 2005. [Online; accessed 19-May-2016].
5. G. Gonthier. Formal Proof—The Four-Color Theorem. *Notices of the AMS*, 55(11):1382—1393, 2008.
6. A. Grabowski. On the Computer-assisted Reasoning About Rough Sets. In B. Dunin-Kępicz, A. Jankowski, A. Skowron, and M. Szczuka, editors, *International Workshop on Monitoring, Security, and Rescue Techniques in Multiagent Systems Location*, vol. 28 of *Advances in Soft Computing*, 215–226, 2005.
7. A. Kornilowicz. Tentative Experiments with Ellipsis in Mizar. In J. Jeuring, J. A. Campbell, J. Carette, Gabriel G. Dos Reis, P. Sojka, M. Wenzel, and V. Sorge, editors, *Intelligent Computer Mathematics 11th International Conference*, vol. 7362 of *LNAI*, 453–457, 2012.
8. A. Kornilowicz. On Rewriting Rules in Mizar. *Journal of Automated Reasoning*, 50(2):203–201, 2013.
9. A. Naumowicz and C. Byliński. Improving Mizar Texts with Properties and Requirements. In A. Asperti, G. Bancerek, and A. Trybulec, editors, *Mathematical Knowledge Management, Third International Conference*, vol. 3119 of *LNCS*, 290–301, 2004.
10. A. Naumowicz and A. Kornilowicz. A Brief Overview of Mizar. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, vol. 5674 of *LNCS*, 67–72, 2009.
11. K. Pał. Methods of Lemma Extraction in Natural Deduction Proofs. *Journal of Automated Reasoning*, 50(2):217–228, 2013.
12. K. Pał. Automated Improving of Proof Legibility in the Mizar System. In S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics - International Conference*, vol. 9150 of *LNCS*, 373–387, 2014.
13. K. Pał. Readable Formalization of Euler's Partition Theorem in Mizar. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics - International Conference*, vol. 9150 of *LNCS*, 211–226, 2015.
14. J. Urban. XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. In M. P. Bonacina, editor, *4th International Conference Mathematical Knowledge Management 2005*, vol. 3863 of *LNCS*, 346–360, 2005.

The impact of proof steps sequence on proof readability — experimental setting^{*}

Karol Pałk¹ and Aleksy Schubert²

¹ Institute of Informatics, University of Białystok,
ul. K. Ciołkowskiego 1M, 15-245 Białystok, Poland
`pakkarol@uwb.edu.pl`

² Institute of Informatics, University of Warsaw
ul. S. Banacha 2, 02-097 Warsaw, Poland
`alx@mimuw.edu.pl`

Abstract. Different sequences of proof steps may result in different experiences of the overall formal proof clarity. There is a conjecture that sequence in which more steps refer to the content of the preceding statement (1) is more comprehensible than the one in which references span longer distances (2). We studied the claim in experimental setting where subjects indicated their reading preference with two versions of the same proof. The difference between their reports indicates that contrary to the conjecture, proofs of the kind (2) can give cognitive advantage in a statistically significant way.

1 Introduction

The readability of formal proofs, as in the case of computer programs, has significant impact on the process of proof formalisation development and maintenance. This phenomenon has already been observed in the programme of Nicolas Bourbaki—the resulting proofs written in formalist fashion were perceived as too obscure and the whole project gained the opinion of unwieldy [12]. Moreover, there are numerous situations in which proof scripts are indeed read, e.g. when a proof development is used as a library of facts [4], when users try to learn new proving techniques [8], or when users want to strengthen theorems [2].

There are many factors that have an impact on readability. In this paper we focus on the psychological readability factors associated with locality of reference. They can be summarised in a high-level statement due to Behaghel that *elements that belong close together intellectually should be placed close together* [1] and this statement concerns particularly systems such as Mizar [3] and Isabelle/Isar [13], which try to construct proofs close to natural language ones.

Discussions among Mizar Mathematical Library developers [9] led to the conclusion that proof steps that refer to the preceding step are perceived as more comprehensible. This led to construction of algorithms that rearranged

^{*} The paper has been supported by the resources of the Polish National Science Centre granted by decision n° DEC-2012/07/N/ST6/02147.

Mizar proof scripts so that they maximise the number of such references [9] and further analysis of the complexity of the rearrangement process [10,11].

One important step in understanding the applicability of the techniques is to assess their impact on real readers. This paper is an attempt in this direction. We investigate the influence of one of the employed techniques — changing the sequence of steps so that the number of **then** steps in continuous sequences is maximal. In our study we give two versions of the same proof to a substantial number of subjects and obtain their readability assessment.

The paper is constructed as follows. In Section 2 we present the design of our experiment. Next, we present the statistical methods we employ and present the way our experiment was executed. This is done in Section 3. The results of the experiments are presented in Section 4 and discussed in Section 5. Many people helped us in organising the experiment so we acknowledge them in Section 6.

2 The Design of the Experiment

2.1 The General Idea of the Experiment

It is not straightforward to devise an experiment in which two text structures are compared for readability. First of all, the feature of readability is subjective so a comparing judgement of a single person must be the basic unit of the measurement. Moreover, one must note that it is not directly valid to ask subjects to compare texts of different proofs for readability since then there may be many other structural factors that must be recognised and well understood to make such a comparison informative. Therefore, our first design choice is that one subject at one moment compares two proofs of the same mathematical statement and the result of the judgement is accounted in the study.

This assumption brings one major difficulty in. Comparison of essentially the same proofs requires reading two texts that convey the same content. The texts are by the nature of the reading process read in sequence. Clearly, the process of reading of the second text is strongly influenced by that of the first text. We can now expect two effects:

1. either the subjects will judge the second text as more readable due to the fact that it reflects something that is already known,
2. or the subjects will judge the second text as less readable since they already familiarised themselves with the structure of the first one and perceive the text of the second one as alien and so less favourable.

It turns out that the second effect is much stronger than the first one (we provide statistical evidence for this in Section 4.2). Therefore, we decided that meaningful responses are only those that counter this strong trend and that such responses indicate visible cognitive advantage to the subjects that behave in this way.

This assumption requires us to create two contrasting situations in which subjects can counter this trend. We decided here to distribute among subjects two variants of the test: one that presents a version *A* of the proof first and then a version *B*, and one that presents the version *B* first and *A* subsequently.

One more important issue to deal with here is making sure that the tests are not neglected by the subjects. That is why we decided that the test should have a form of an assignment in which the analysis of the proofs is required to give an answer. In the end we decided that a good way to achieve this is to introduce a mistake into the proof and instruct subjects to find it. The subjects were to obtain an award for fulfilling the task.

2.2 The Actual Tests

The assignments given to the students were based upon a faulty proof of the wrong set-theoretic formula

$$(X \cup Y) \setminus (X \cap Y) = (X \setminus Y) \cap (Y \setminus X).$$

It has a correct counterpart

$$(X \cup Y) \setminus (X \cap Y) = (X \setminus Y) \cup (Y \setminus X)$$

the proof of which can be formalised in Mizar as presented in Fig. 1 (version *A* of the proof). This proof was taken as the basis for the experiment. We devised another proof (see Fig. 2) of the same fact (version *B*) in which certain two steps were permuted, which is reflected by the changed line numbers (7, 8) on the left margins of the proofs. The actual internal structure of the two versions is the same and is presented in Fig. 3 where labels of the nodes correspond directly to the labels on the left margins of both the proof in version *A* and *B*.³ In this way we can see that the proofs indeed differ only in the sequence of steps and the necessary Mizar syntax reorganisations. We expected that the version *B* will be perceived by subjects in our experiments as less readable than the version *A* (and this turned out to be actually false).

Since we wanted to obtain judgements of many subjects and it is difficult to gather significantly many Mizar experts in one place, we had to rewrite the proof texts in natural language so that they are digestible by people who are not familiar with Mizar syntax. Moreover, the subjects in the experiment were Polish native speakers so the assignments had to be prepared in that language.⁴ We provide a faithful translation of the tests in Appendix A.

The two versions of the proof were printed on one page of an A4 leaf. Half of the tests had version *A* of the proof located on the left-hand side of the page and version *B* on the right-hand one and half of the tests had version *B* located on the left-hand side and version *A* on the right-hand one. To make distribution of tests quick we did not strictly keep the size of two resulting groups even. However, we made sure that the groups are of substantial size (see Section 4).

³ Graph representation of proofs was defined by Pałk [9]. We invite interested readers to consult his publication for details.

⁴ The original tests are available in the package with experiment data at the address <http://www.mimuw.edu.pl/~alx/proof-readability.zip>

```

theorem
1:  $(X \vee Y) \setminus (X \wedge Y) = (X \setminus Y) \vee (Y \setminus X)$ 
proof
2: for  $x$  holds  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$  iff  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$ 
proof
3: let  $x$ ;
4: thus  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$  implies  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$ 
proof
5: assume  $A1: x \text{ in } (X \vee Y) \setminus (X \wedge Y)$ ;
6: then not  $x \text{ in } (X \wedge Y)$  by  $XBOOLE\_0:\text{def } 5$ ;
7: then  $A2: \text{not } x \text{ in } X$  or not  $x \text{ in } Y$  by  $XBOOLE\_0:\text{def } 4$ ;
8:  $x \text{ in } X$  or  $x \text{ in } Y$  by  $A1, XBOOLE\_0:\text{def } 3$ ;
9: then  $x \text{ in } (X \setminus Y)$  or  $x \text{ in } (Y \setminus X)$  by  $A2, XBOOLE\_0:\text{def } 5$ ;
10: hence  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$  by  $XBOOLE\_0:\text{def } 3$ ;
end;
11: thus  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$  implies  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$ 
proof
12: assume  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$ ;
13: then  $x \text{ in } (X \setminus Y)$  or  $x \text{ in } (Y \setminus X)$  by  $XBOOLE\_0:\text{def } 3$ ;
14: then  $A3: x \text{ in } X \ \& \ \text{not } x \text{ in } Y$  or
 $x \text{ in } Y \ \& \ \text{not } x \text{ in } X$  by  $XBOOLE\_0:\text{def } 5$ ;
15: then  $A4: x \text{ in } (X \vee Y)$  by  $XBOOLE\_0:\text{def } 3$ ;
16: not  $x \text{ in } (X \wedge Y)$  by  $A3, XBOOLE\_0:\text{def } 4$ ;
17: hence  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$  by  $A4, XBOOLE\_0:\text{def } 5$ ;
end;
end;
hence thesis by  $TARSKI:2$ ;
end;

```

Fig. 1. The Mizar proof the experiments were based upon, version A.

```

theorem
1:  $(X \vee Y) \setminus (X \wedge Y) = (X \setminus Y) \vee (Y \setminus X)$ 
proof
2: for  $x$  holds  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$  iff  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$ 
proof
3: let  $x$ ;
4: thus  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$  implies  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$ 
proof
5: assume  $A1: x \text{ in } (X \vee Y) \setminus (X \wedge Y)$ ;
6: then  $A2: \text{not } x \text{ in } (X \wedge Y)$  by  $XBOOLE\_0:\text{def } 5$ ;
8:  $A3: x \text{ in } X$  or  $x \text{ in } Y$  by  $A1, XBOOLE\_0:\text{def } 3, \text{def } 4$ ;
7: not  $x \text{ in } X$  or not  $x \text{ in } Y$  by  $A3, A2, XBOOLE\_0:\text{def } 4$ ;
9: then  $x \text{ in } (X \setminus Y)$  or  $x \text{ in } (Y \setminus X)$  by  $A3, XBOOLE\_0:\text{def } 5$ ;
10: hence  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$  by  $XBOOLE\_0:\text{def } 3$ ;
end;
11: thus  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$  implies  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$ 
proof
12: assume  $x \text{ in } (X \setminus Y) \vee (Y \setminus X)$ ;
13: then  $x \text{ in } (X \setminus Y)$  or  $x \text{ in } (Y \setminus X)$  by  $XBOOLE\_0:\text{def } 3$ ;
14: then  $A4: x \text{ in } X \ \& \ \text{not } x \text{ in } Y$  or
 $x \text{ in } Y \ \& \ \text{not } x \text{ in } X$  by  $XBOOLE\_0:\text{def } 5$ ;
15: then  $A5: x \text{ in } (X \vee Y)$  by  $XBOOLE\_0:\text{def } 3$ ;
16: not  $x \text{ in } (X \wedge Y)$  by  $A4, XBOOLE\_0:\text{def } 4$ ;
17: hence  $x \text{ in } (X \vee Y) \setminus (X \wedge Y)$  by  $A5, XBOOLE\_0:\text{def } 5$ ;
end;
end;
18: hence thesis by  $TARSKI:2$ ;
end;

```

Fig. 2. A Mizar proof in which steps were permuted, version B.

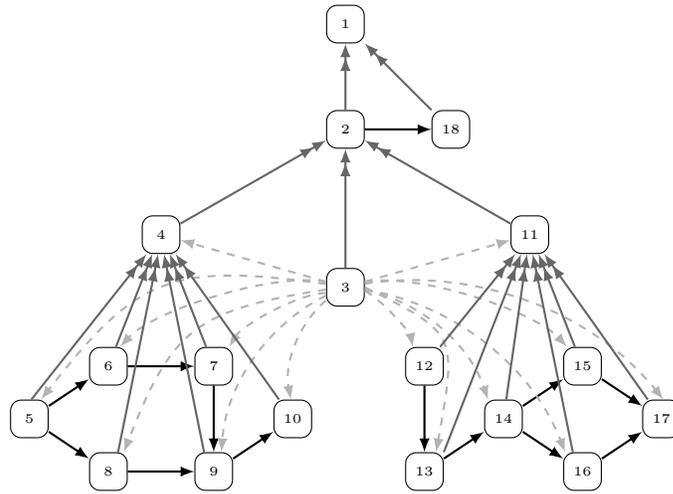


Fig. 3. The structures of the two versions of the proof.

3 The Execution of the Experiment

The experiments were conducted on students of informatics at Faculty of Mathematics, Informatics and Mechanics, University of Warsaw and Faculty of Mathematics and Computer Science, University of Białystok. They were executed in three rounds of two slightly different settings.

Round I took place in January 2015 and was conducted as a separate assignment in an exam with a slot of 10 minutes dedicated exclusively to the task of comparing the proofs. The award for students were additional points in the exam score. The topic of the exam was *Logic for Computer Scientists*. The subjects of the experiment were students of first year postgraduate studies. These students had studied the full curriculum of undergraduate informatics, including *Foundations of Mathematics* as taught on the first year of undergraduate studies. We can assume that their experience with formal systems, including programming languages, was good enough for this test.

The experiment was conducted in parallel in two lecture halls which contained groups of students of approximately 35 people each. The assignments were given in parallel and in the conditions of the exam, which guarantee the lack of communication between subjects. Therefore, we can assume that the results were independent.

The instruction of the assignment was as follows

You are presented two proofs of the same fact. Both have the same flaw. Choose the version of the reasoning for which you can easier perform the following task: please show the place where the flaw is located and describe in one sentence what is the reason of the mistake.

Therefore, the students were given information that the two texts are essentially equivalent and they were instructed to indicate which of the texts gave them cognitive advantage.

Both versions of the proof were formulated using the same natural language phrases so that one formulation corresponds to the same kind of inference step.

Round II took place in February 2015 and was conducted in Białystok. The procedure and tests were the same as in the case of Round I. The topic of the exam was *Logic and Set Theory*. The subjects of the experiment were students of first year undergraduate studies of mathematics and informatics. We can assume that their experience with formal systems was fair and enough for this test, since they were taught specifically to prove facts using natural deduction format that was close to the one used in Mizar. Still, we obtained significantly more answers than in Round I which indicated that subjects did not understand the subject matter of the proof. The exam was taken in two groups of approximately 20 and 40 people respectively

Round III took place in January 2016 in Warsaw and was conducted as a separate assignment during blackboard classes with a slot of 10 minutes dedicated exclusively to the task of comparing the proofs. The award for students were additional points in their score of the classes. The topic of the classes was *Foundations of Mathematics*. The subjects of the experiment were students of first year undergraduate studies. The experience of the students with formal systems was limited. The experiment was conducted at the end of the course on foundations of mathematics so their understanding of the notion of proof should be satisfactory for the purpose of our procedure.

The experiment was conducted in 4 different groups of classes at different times. The size of the groups was approximately 10 people. However, we can assume that the measurements were independent since the students of the first year of undergraduate studies rarely communicate between different groups (all other classes are given in groups of the same composition) and the details of the assignment necessary to complete it successfully are not very specific and so very difficult to explain without having the actual test at hand. Therefore, we can safely assume that despite the assignments were not given in parallel their results are independent. Of course, the assignments within each of the group were executed so that subjects did not communicate one with another.

The instruction of the assignment was as follows

You are presented two proofs of the same fact. Find in them as many flaws as you can. In case a mistake occurs in both proofs, mark with a star the version in which you found it first. Is the structure of some of the proofs more readable for you?

Therefore, the students were given information that the two texts are essentially equivalent and they were instructed to indicate which of the texts gave them cognitive advantage. Additionally, they were instructed to search for as many flaws as possible and mark places where the mistake was found first. We intro-

duced this change to get deeper confidence that the proofs were read thoroughly and check if this change has impact on the effects (E1) and (E2)

In this case the two versions of the proof were formulated in a different fashion. Moreover, as the proofs actually demonstrate equivalence, they divide into two parts: one for (\Rightarrow) and one for (\Leftarrow). We decided to have a different sequence of these parts in each version. The rationale behind these changes was to make line to line comparison more difficult and force subjects to perceive the structure of the proofs first and then judge their clarity.

We would like to point out that in Round I and II the instruction was less direct and referred to *easiness of the task* while in Round III the instruction appealed directly to the intuitive understanding of the term *readability*. However, we would like to stress that each of the formulations actually means that the subjects were instructed to indicate their cognitive preference for the two versions of the proof, i.e. which of the texts is easier to digest.

3.1 The Mann-Whitney Test

For our verification, we use the non-parametric Mann-Whitney test (also called Willcoxon test), as the use of standard parametric tests assumes the distributions are normal. This test is used to statistically refute a null hypothesis H_0 in favour of its negation, i.e. the alternative hypothesis H_A .

This kind of test can be applied when the following prerequisites are met:

Hypotheses The null hypothesis H_0 for the study should be formulated in the fashion such that the probability of an observation from the population G_1 exceeding an observation from the second population G_2 is less than or equal to the probability of an observation from G_2 exceeding an observation from G_1 , i.e. $P(\text{score}(G_1) > \text{score}(G_2)) \leq P(\text{score}(G_2) > \text{score}(G_1))$.

Consequently, the alternative hypothesis H_A is that the probability of an observation from the group G_1 exceeding an observation from the group G_2 is greater than the probability of an observation from G_2 exceeding an observation from G_1 , i.e. $P(\text{score}(G_1) > \text{score}(G_2)) > P(\text{score}(G_2) > \text{score}(G_1))$.

Uniformity of populations We should also be able to assume that in case the distribution of results from G_1 and G_2 is the same, the probability of an observation from G_1 exceeding one from G_2 is equal to the probability of an observation from G_1 exceeding one from G_2 . In other words the background of both groups is the same and their members were chosen randomly from the point of view of the experiment.

Independence We should be able to assume that the observations in G_1 and G_2 are independent.

Comparable responses The scores should be numeric so they can be easily compared one with another so a single ranking of the subjects can be formed.

The basic idea of the test is that we build a ranking list of the scores from the lowest to the highest (with possible ties). Then we try to get a numerical representation on which of the groups has more results close to the top one.

Consider the following pattern

$$U = n_1 n_2 + \frac{n_2(n_2 + 1)}{2} - \sum_{i=1}^{n_2} R_i$$

where U is the value of the Mann-Whitney U test, n_1 is the sample size of G_1 , n_2 is the sample size of G_2 and R_i is the ranking position of the i -th score in the group G_2 . We can see that the more scores of G_2 are closer to the top the number U is greater.

Contemporary statistics packages return a normalised value Z of the statistical test that is computed according to the pattern

$$Z = \frac{U - m_U}{\sigma_U}$$

where m_U is the mean of U and σ_U is the standard deviation of U , which are in turn computed using the formulas

$$m_U = \frac{n_1 n_2}{2}, \quad \sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}.$$

However, the formula for σ_U must be corrected in case ties (caused by equal scores of certain subjects) are present and it is then

$$\sigma_U = \sqrt{\frac{n_1 n_2}{12} \left((n + 1) - \sum_{i=1}^k \frac{t_i^3 - t_i}{n(n-1)} \right)}$$

where k is the number of tied positions, t_i is the number of subjects that attained the i -th tied rank, and $n = n_1 + n_2$.

3.2 Stouffer-Lipták Method

Since the tests given to the subjects were different and the students were of different maturity we cannot directly sum the groups that took part in the two rounds. Therefore we need a method to combine results of three experiments. For this we use a meta-analysis technique called Stouffer-Lipták method [7]. In this method we compute a combined Z value based upon Z -values obtained from the Mann-Whitney test using the pattern (tailored to the case where three experiments are combined):

$$Z = \frac{w_1 Z_1 + w_2 Z_2 + w_3 Z_3}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

where Z_1, Z_2, Z_3 are the cumulative normal distribution values that match the probabilities $1 - p_1, 1 - p_2$ and $1 - p_3$ with p_1, p_2, p_3 being p-values obtained in two sub-experiments while w_1, w_2, w_3 are weights of the sub-experiments that can be used to accommodate for different sample sizes (note that sub-experiment with

a bigger number of subjects should weigh more than the sub-experiment with the smaller one, according to Lipták square roots of sample sizes are optimal here).

The p-value for Z-value obtained in this way is $1 - v$ where v is the probability that a normally distributed random number will be less than that this Z-value.

3.3 The Binomial Test

The binomial test is used to check which of two outcomes has greater probability. We assume here that our process has n outcomes represented by random variables X_1, \dots, X_n . These outcomes are either 0 (failure) or 1 (success) and now $Y = \sum_{i=1}^n X_i$ represents the number of successes in the process. We observe now that

$$P(Y = k) = \binom{n}{k} q^k (1 - q)^{n-k} \text{ for } k \in \{0, \dots, n\}$$

where q is the probability of success (i.e. that $X_i = 1$). We can now use the value $Q_{n,q}(\alpha)$ of the quantile of order α for the distribution to reject H_0 telling that $q \geq p_0$ versus the alternate hypothesis H_A that $q < p_0$ if and only if $Y \leq Q_{n,p_0}(\alpha)$ where α is the desired level of confidence. Typically $p_0 = 0.5$ and we actually check that the probability of success is greater than the one of failure.

4 Results of the Experiment

Since we use the Mann-Whitney test, we have to make sure that the assumptions of the test are met. First of all we have to ensure that the scores are numeric. In our setting we give the score 1 to those assignments that clearly indicate the right-hand side of the test. The other score, 0, is given when the left-hand side is indicated as preferred, when there is no preference and also when the assignment was solved incorrectly by the subject.

We can now compare two populations depending on what was the content of the left-hand side (i.e. first side) of obtained assignment: (G_1) these who obtained version A there with (G_2) those who obtained version B . We understand that subjects in G_1 , by choosing the right-hand side in their test, judged that version B gives them cognitive advantage while subjects in G_2 , by choosing the same right-hand side, judged that version A gives them this. Our main statistical hypotheses can be formulated in the following way

- H_0^a : the probability that a subject from G_2 has the score 1 is greater or equal to the probability that a subject from G_1 does it;
- H_A^a : the probability that a subject from G_2 has the score 1 is smaller than the probability that a subject from G_1 does it.

We have two more populations, G'_1 with subjects that obtained 0 in the assessment of assignments and G'_2 with subjects that obtained 1 there. This time we assign score 1 to all subjects of the groups. We can now formulate two other hypotheses

- H_0^f : the probability that a subject from G_2' has the new score 1 is greater than the probability that a subject from G_1' does it;
- H_A^f : the probability that a subject from G_2' has the new score 1 is smaller or equal to the probability that a subject from G_1' does it.⁵

Let us take a look at the assumptions of the Mann-Whitney test. This formulation of the two pairs of hypotheses directly falls under the format prescribed in Section 3.1. In each case the leaves with assignments were distributed in sequence without any taking into account the subjects' history (in particular scores in earlier exams) so we can assume that the populations of all the groups had the same background. The assignments in each case were worked out by the subjects in the exam-like conditions, i.e. without any communication with their colleagues solving the same assignment. Therefore, we may assume that the measures for each of the assignment were taken independently. At last, the final scores were numeric so they fulfil the assumption of comparability of responses.

Round I

Version	size	mean	sd
V. A first (G_1)	33	0	0
V. B first (G_2)	43	0.1627907	0.3735437
Total	76	0.09210526	0.2910959

Round II

Version	size	mean	sd
V. A first (G_1)	34	0.08823529	0.2879022
V. B first (G_2)	31	0.09677419	0.3005372
Total	65	0.09230769	0.2917125

Round III

Version	size	mean	sd
V. A first (G_1)	17	0.2941176	0.4696682
V. B first (G_2)	17	0.1764706	0.3929526
Total	34	0.2352941	0.4305615

Fig. 4. The basic statistics of the experiment.

4.1 General Overview of Results

The overall number of subjects who took part in Round I of the experiment was 76. Out of the number, 33 people got the assignment leaf with version *A* of the proof on the left-hand side and 43 people got the assignment with version *B* on the left-hand side. In the case of Round II the total number of subjects was 65 with 34 assignments where version *A* of the proof was on the left-hand side and 31 where version *B* was there. In Round III the total number of subjects was 34 and each of the groups had 17 people. We have to note here that the uneven number of subjects in the two groups does not prevent applicability

Round I

Version	score 0	score 1
V. A first (G_1)	36	7
V. B first (G_2)	33	0

Round II

Version	score 0	score 1
V. A first (G_1)	28	3
V. B first (G_2)	31	3

Round III

Version	score 0	score 1
V. A first (G_1)	14	3
V. B first (G_2)	12	5

Fig. 5. The distribution of the assigned scores.

⁵ The small difference in the formulation of the hypotheses H_0^a, H_A^a and H_0^f, H_A^f is probabilistically negligible, but it is easier to handle in computations by R.

of the Mann-Whitney statistical test, it only impairs its accuracy.

The basic statistics of the groups are presented in Fig. 4. We can see the sizes of the groups (column size) as well as the mean score (column mean) and standard deviation of the scores (column sd). We immediately see that the mean of the score in the group G_1 during Round I is 0. This is due to the fact that none of the subjects returned the assignment with indicated version A on the right-hand side. We can confirm this by looking in the table displayed in Fig. 5 where the numerical scores of all the six subgroups are presented.

4.2 First Text Is Favoured

We evaluate first the hypotheses H_0^f versus H_A^f as they give the background for the whole experiment. For this we use a more direct *binomial test* based upon the Bernoulli distribution. The p-values for this test are $3.208984 \cdot 10^{-14}$, $2.482325 \cdot 10^{-12}$, and 0.001467528 for the samples taken in Round I, Round II, and Round III respectively. The combined p-value obtained with the Stouffer-Lipták method is so small that it is below the precision range of our tools. Therefore the overall result is 0.⁶ All the figures are gathered in the table presented in Fig. 6. Summing up, we obtained a prevalent evidence that the left hand side is favoured in such comparison tests.

Round	p-value
Round I	$3.208984 \cdot 10^{-14}$
Round II	$2.482325 \cdot 10^{-12}$
Round III	0.001467528
Stouffer-Lipták	0

Fig. 6. The p-values for the test on which side is favoured.

4.3 Readability Assessment

The picture in the readability assessment is more complicated. Already a look at the table in Fig. 5 reveals that in Round I the preference for version B was strong, in Round II the scores were even and in Round III turned in the opposite

Round	Z-value	p-value
Round I	2.41645	0.007836335
Round II	0.117872	0.4530845
Round III	-0.7966275	0.7871663
Stouffer-Lipták	1.31315	0.0945662

Fig. 7. The statistics of the readability experiment.

direction. This is reflected in obtained Z-values, which are 2.41645, 0.117872 and -0.7966275 respectively. The negative value of the second statistics reflects the fact that we are closer to rejecting H_A^a . The p-values obtained in the tests are 0.007836335, 0.4530845 and 0.7871663 respectively.⁷ This shows that the first test gives a strong statistically significant result, which supports our claim that the proofs in version A format can give cognitive advantage at the statistically significant level, actually the significance here is higher than 99%. This evidence is so strong that even when we combine the three rounds with Stouffer-Lipták

⁶ The results were obtained with help of the standard R function `binom.test`.

⁷ The results were obtained with help of the R package `coin` [5,6] with its heuristics `mid-ranks` to handle ties in input data.

test we obtain result of reasonable significance at the level over 90%. These results are summarised in the table presented in Fig. 7.

5 Discussion

The fact that the left-hand side of tests devised in our experiment is strongly favoured has very strong support in our data. The case of readability is more curious. We can see that Round I resulted in strong preference for version *B* of the proof, which is more convoluted. However slight changes to the conditions in Rounds II and III gave a picture that tends to support the opposite claim. We can observe that Round I (during an exam) was likely more stressful than Round III, and this could make the short-term memory less effective. As a result, the bigger number of labels present in version *B* could turn out to be advantageous for the understanding process. Some of the subjects shared with us this opinion on their assessments. However, this line of argument is not supported by the result of Round II, which was also taken under exam conditions.

One more phenomenon that can also play role here is that the proof in version *A* indeed more often referred to the previous step. However, each step depends not only on the directly preceding one, but also on some other ones. In the case of version *A* we have a step (line 8. in Fig. 1) which required subjects to refer 3 steps back, while the proof in version *B* required subjects to refer at most 2 steps back. Further investigations are necessary to check which of the reasons actually holds. However, this investigation shows that the tools to improve readability based upon the principle used here should be used cautiously. Readability is a feature that depends highly on individual abilities of proof readers so the tools to improve it should not enforce any fixed method of proof improvement, but rather offer a number possibilities that can be chosen by users.

The subjects in this experiment actually were not trained in extensive reading of mathematical proofs. We can say that they were accidental proof readers. This experiments showed that they prefer proofs in version *B*. Still, experienced users, as implied by our initial expectations, seem to prefer proofs structured as in version *A*. It may be that the experienced users learn this preference as they gain experience. This conjecture is also worth further investigation.

6 Acknowledgements

A number of people helped us to execute the experiment. We would like to thank professors Jerzy Tyszkiewicz from University of Warsaw and Krzysztof Prażmowski from University of Białystok for allowing us to take the test during exams of their classes. The tests were also part of classes conducted by Daria Walukiewicz-Chrząszcz, Jacek Chrząszcz, and Piotr Wasilewski so we are also grateful for their help. These people were also very kind to discuss with us the design of the experiment which helped us in its better organisation.

References

1. Otto Behaghel. Beziehungen zwischen umfang und reihenfolge von satzgliedern. *Indogermanische Forschungen*, (29):110–142, 1909.
2. Georges Gonthier. Formal proof the four-color theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.
3. Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Four decades of Mizar. *Journal of Automated Reasoning*, 55(3):191–198, October 2015.
4. Adam Grabowski and Christoph Schwarzweller. Revisions as an essential tool to maintain mathematical repositories. In *Proceedings of the 14th Symposium on Towards Mechanized Mathematical Assistants: 6th International Conference, Calculemus '07 / MKM '07*, pages 235–249, Berlin, Heidelberg, 2007. Springer-Verlag.
5. Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, and Achim Zeileis. Implementing a class of permutation tests: The coin package. *Journal of Statistical Software*, 28(8), 2008.
6. Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel, and Achim Zeileis. *Package ‘coin’: Conditional Inference Procedures in a Permutation Test Framework*. CRAN, 2013.
7. T. Lipták. On the combination of independent tests. *Magyar Tud Akad Mat Kutato Int Közl.*, 3:171–196, 1958.
8. Adam Naumowicz and Czesław Byliński. Improving Mizar texts with properties and requirements. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Mathematical Knowledge Management, Third International Conference, MKM 2004 Proceedings*, volume 3119 of *MKM'04, Lecture Notes in Computer Science*, pages 290–301, 2004.
9. Karol Pał. The Algorithms for Improving and Reorganizing Natural Deduction Proofs. *Studies in Logic, Grammar and Rhetoric*, 22(35):95–112, 2010.
10. Karol Pał. Improving Legibility of Natural Deduction Proofs is Not Trivial. *Logical Methods in Computer Science*, 10(3), 2014.
11. Karol Pał. Improving Legibility of Formal Proofs Based on the Close Reference Principle is NP-Hard. *Journal of Automated Reasoning*, 55(3):295–306, 2015.
12. Ernst Snapper. The three crises in mathematics: Logicism, intuitionism and formalism. *Mathematics Magazine*, 52(4):207–216, 1979.
13. Makarius Wenzel. *The Isabelle/Isar Reference Manual*. University of Cambridge, 2013.

A English Translations of the Tests Used in the Experiment

In each of the two versions of the experiment the assignment text and the formulation of the statement repeat on each side of the page. Only the proof part is different. We give the repeating parts only once in this translation.

A.1 Version Used in January 2015

Assignment You are presented two proofs of the same fact. Both have the same flaw. Choose the version of the reasoning for which you can easier perform the following task: please show the place where the flaw is located and describe in one sentence what is the reason of the mistake.

Statement $(X \cup Y) \setminus (X \cap Y) = (X \setminus Y) \cap (Y \setminus X)$

Proof (version A) We show first that for each x the equivalence holds

$$x \in (X \cup Y) \setminus (X \cap Y) \text{ if and only if } x \in (X \setminus Y) \cap (Y \setminus X).$$

For the proof from left to right:

- *A1*: Let us take any $x \in (X \cup Y) \setminus (X \cap Y)$.
- We immediately obtain that $x \notin (X \cap Y)$ (def. of set difference),
- *A2*: and further that $x \notin X$ or $x \notin Y$ (def. of set intersection).
- Additionally $x \in X$ or $x \in Y$ (see *A1*, def. of set sum)
- therefore $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (see *A2*, def. of set difference),
- which gives the expected $x \in (X \setminus Y) \cap (Y \setminus X)$ (def. of set intersection).

For the proof from right to left:

- Let us take any $x \in (X \setminus Y) \cap (Y \setminus X)$.
- We immediately obtain that $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (def. of set intersection),
- *A4*: and further $x \in X$ and $x \notin Y$ or $x \in Y$ and $x \notin X$ (def. of set difference),
- and further $x \in (X \cup Y)$ (def. of set sum).
- Additionally $x \notin (X \cap Y)$ (see *A4*, def. of set intersection)
- which gives the expected $x \in (X \cup Y) \setminus (X \cap Y)$ (def. of set difference).

From the proved equivalence and definition of set equality we immediately obtain the goal statement.

Proof (version B) We show first that for each x the equivalence holds

$$x \in (X \cup Y) \setminus (X \cap Y) \text{ if and only if } x \in (X \setminus Y) \cap (Y \setminus X).$$

For the proof from left to right:

- *A1*: Let us take any $x \in (X \cup Y) \setminus (X \cap Y)$.
- *A2*: We immediately obtain that $x \notin (X \cap Y)$ (def. of set difference),
- *A3*: Additionally $x \in X$ or $x \in Y$ (see *A1*, def. of set sum)
- Observe that $x \notin X$ or $x \notin Y$ (see *A2*, def. of set intersection)
- Consequently $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (see *A3*, def. of set difference),
- which gives the expected $x \in (X \setminus Y) \cap (Y \setminus X)$ (def. of set intersection).

For the proof from right to left:

- Let us take any $x \in (X \setminus Y) \cap (Y \setminus X)$.
- We immediately obtain that $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (def. of set intersection),
- *A3*: and further $x \in X$ and $x \notin Y$ or $x \in Y$ and $x \notin X$ (def. of set difference),
- and further $x \in (X \cup Y)$ (def. of set sum).
- Additionally $x \notin (X \cap Y)$ (see *A3*, def. of set intersection)
- which gives the expected $x \in (X \cup Y) \setminus (X \cap Y)$ (def. of set difference).

From the proved equivalence and definition of set equality we immediately obtain the goal statement.

A.2 Version Used in January 2016

Assignment You are presented two proofs of the same fact. Find in them as many flaws as you can. In case a mistake occurs in both proofs, mark with a star

the version in which you found it first. Is the structure of some of the proofs more readable for you?

Statement $(X \cup Y) \setminus (X \cap Y) = (X \setminus Y) \cap (Y \setminus X)$

Proof (version A) We show first that for each x the equivalence holds

$$x \in (X \cup Y) \setminus (X \cap Y) \text{ if and only if } x \in (X \setminus Y) \cap (Y \setminus X).$$

For the proof from left to right:

- [1] Let us take any $x \in (X \cup Y) \setminus (X \cap Y)$.
- We immediately obtain that $x \notin (X \cap Y)$ (def. of set difference),
- [2] and further that $x \notin X$ or $x \notin Y$ (def. of set intersection).
- Additionally $x \in X$ or $x \in Y$ (see [1], def. of set sum)
- therefore $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (see [2], def. of set difference),
- summing up, we obtain in the end that $x \in (X \setminus Y) \cap (Y \setminus X)$ (def. of set intersection).

For the proof from right to left:

- Let us take any $x \in (X \setminus Y) \cap (Y \setminus X)$.
- We immediately obtain that $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (def. of set intersection),
- [3] and then $x \in X \wedge x \notin Y$ or $x \in Y \wedge x \notin X$ (def. of set difference),
- [4] and then $x \in (X \cup Y)$ (def. of set sum).
- Additionally $x \notin (X \cap Y)$ (see [3], def. of set intersection)
- summing up, we obtain in the end that $x \in (X \cup Y) \setminus (X \cap Y)$ (see [4] and def. of set difference).

From the proved equivalence and definition of set equality we immediately obtain the goal statement.

Proof (version B) We show first that for each x the equivalence holds

$$x \in (X \cup Y) \setminus (X \cap Y) \text{ if and only if } x \in (X \setminus Y) \cap (Y \setminus X).$$

For the proof from right to left:

- Let us fix arbitrary $x \in (X \setminus Y) \cap (Y \setminus X)$.
- From this, we obtain that $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (def. of set intersection),
- [1] and then $x \in X \wedge x \notin Y$ or $x \in Y \wedge x \notin X$ (def. of set difference),
- [2] and then $x \in (X \cup Y)$ (def. of set sum).
- Except from that $x \notin (X \cap Y)$ (see [1], def. of set intersection)
- which gives the expected $x \in (X \cup Y) \setminus (X \cap Y)$ (see [2] and def. of set difference).

For the proof from left to right:

- [3] Let us fix arbitrary $x \in (X \cup Y) \setminus (X \cap Y)$.
- [4] From this we obtain that $x \notin (X \cap Y)$ (def. of set difference),
- [5] Additionally $x \in X$ or $x \in Y$ (see [3], def. of set sum)
- Observe that $x \notin X$ or $x \notin Y$ (see [4], def. of set intersection).
- therefore $x \in (X \setminus Y)$ or $x \in (Y \setminus X)$ (see [5], def. of set difference),
- which gives the expected $x \in (X \setminus Y) \cap (Y \setminus X)$ (def. of set intersection).

From the proved equivalence and definition of set equality we immediately obtain the goal statement.

Models for Metamath

Mario Carneiro

The Ohio State University, Columbus OH, USA

Abstract. Although some work has been done on the metamathematics of Metamath, there has not been a clear definition of a model for a Metamath formal system. We define the collection of models of an arbitrary Metamath formal system, both for tree-based and string-based representations. This definition is demonstrated with examples for propositional calculus, ZFC set theory with classes, and Hofstadter’s MIU system, with applications for proving that statements are not provable, showing consistency of the main Metamath database (assuming ZFC has a model), developing new independence proofs, and proving a form of Gödel’s completeness theorem.

Keywords: Metamath · Model theory · formal proof · consistency · ZFC · Mathematical logic

1 Introduction

Metamath is a proof language, developed in 1992, on the principle of minimizing the foundational logic to as little as possible [1]. An expression in Metamath is a string of constants and variables headed by a constant called the expression’s “typecode”. The variables are typed and can be substituted for expressions with the same typecode. See § 2.1 for a precise definition of a formal system, which mirrors the specification of the `.mm` file format itself.

The logic on which Metamath is based was originally defined by Tarski in [2]. Notably, this involves a notion of “direct” or “non-capturing” substitution, which means that no α -renaming occurs during a substitution for a variable. Instead, this is replaced by a “distinct variable” condition saying that certain substitutions are not valid if they contain a certain variable (regardless of whether the variable is free or not—Metamath doesn’t know what a free variable is). For instance, the expression $\forall x \varphi$ contains a variable φ inside a binding expression “ $\forall x \square$ ”. (Metamath also does not have a concept of “binding expression”, but it is safe to say that under a usual interpretation this would be considered a binding expression.) If there is a distinct variable condition between x and φ , then the substitution $\varphi \mapsto x = y$ is invalid, because x is present in the substitution to φ . This is stricter than the usual first-order logic statement “ x is not free in φ ”, because $\varphi \mapsto \forall x x = y$ is also invalid. If there is no such distinct variable condition between x and φ , these substitutions would be allowed, and applying them to $\forall x \varphi$ would result in $\forall x x = y$ and $\forall x \forall x x = y$, respectively.

In this paper, we will develop a definition for models of Metamath-style formal systems, which will operate by associating a function to each syntactical

construct according to its type. For example, the forall symbol is defined by the axiom “wff $\forall x \varphi$ ”, which is to say it takes as input a set variable and a wff variable, and produces a wff expression. This construct is associated to an interpretation function $\pi_{\forall} : U_{\text{set}} \times U_{\text{wff}} \rightarrow U_{\text{wff}}$, where U_{set} is the universe of set variables and U_{wff} is the universe of wff variables, which are each provided as part of the definition of a model.

Note the difference from the usual signature of the forall, $\pi'_{\forall} : (M \rightarrow \text{Bool}) \rightarrow \text{Bool}$, which maps functions from the model universe M to boolean values, to a boolean value. In order to make our definition work, we need the set U_{wff} to be more complicated than just Bool . Instead, it is effectively $(V \rightarrow M) \rightarrow \text{Bool}$, that is, a function from assignments of variables to elements of the model, to a boolean value. In other words, a wff can be thought of as an infinite-place predicate $\varphi(v_0, v_1, v_2, \dots)$ (although the value can only depend on finitely many of the provided variables).

1.1 Grammars and trees

Unfortunately, although it is possible to define what it means to be a model for any Metamath formal system, we can't quite reduce it to a collection of interpretation functions, like it is normally done, without a way to parse the strings which are used in the proof. This leads to the idea of grammatical parsing, which we take up in earnest in § 2.3. By separating all axioms into “syntax axioms” and “logical axioms”, we can find an isomorphism to a representation of statements as trees, with syntax axioms forming the nodes of the tree. Most interesting Metamath systems are grammatical, but for example Hofstadter's MIU system [3], formalized in Metamath as `miu.mm`, is a valid formal system which is not grammatical (see § 3.2).

The main work is presented in § 2. A short recap of Metamath's formalism as it will be used in this work is in § 2.1, followed by the definition of a model in § 2.2. Then we define the subset of “grammatical” formal systems, which are those for which parsing is possible, in § 2.3, and rebuild the theory for a tree representation of formal systems in § 2.4. The model theory of tree formal systems is developed in § 2.5. A selection of examples is provided in § 3, and in particular we prove that Metamath's ZFC formalization, `set.mm`, has a model in Theorem 4. Some applications of model theory are developed in § 4, finishing with a proof of Gödel's completeness theorem in § 4.2.

2 Formal definition

2.1 Metamath recap

We recall the definitions from Appendix C of the Metamath book [1], but with a slight modification for a global type function.

1. Let CN, VR be disjoint sets, called the set of *constants* and *variables* respectively.

CN constants	§ 2.1	VR variables	§ 2.1	Type type of expr	§ 2.1
EX expressions	§ 2.1	DV distinct variables	§ 2.1	\mathcal{V} variables in expr	§ 2.1
σ substitution	§ 2.1	VH variable hypotheses	§ 2.1	TC typecodes	§ 2.1
VT variable typecodes	§ 2.1	U universe	§ 2.2	VL, μ valuations	§ 2.2
$\#$ freshness relation	§ 2.2	η interpretation	§ 2.2	SA syntax axioms	§ 2.3
Syn syntax for expr	§ 2.4	ST syntax trees	§ 2.4	π interpretation (tree)	§ 2.5

Table 1. Definition cheat sheet

2. Let $\text{Type} : VR \rightarrow CN$ be a function, understood to map a variable to its typecode constant.
3. Let $VT = \{\text{Type}(v) \mid v \in VR\}$ be the set of typecodes of variables.
4. $EX = \{e \in \bigcup_{n \in \omega} {}^n(CN \cup VR) \mid (|e| > 0 \wedge e_0 \in CN)\}$ (the set of expressions),
5. $DV = \{x \subseteq VR \mid |x| = 2\}$ (the set of distinct variable specifications), and
6. $\mathcal{V}(e) = VR \cap \{e_n \mid 0 \leq n < |e|\}$ (the set of variables in an expression).
7. We also write $\text{Type}(e) = e_0$ for $e \in EX$.
8. A *substitution* is a function $\sigma : EX \rightarrow EX$ such that $\sigma(\langle c \rangle) = \langle c \rangle$ for $c \in CN$ and $\sigma(gh) = \sigma(g)\sigma(h)$, where adjacency denotes concatenation of sequences. (Such a function is determined by its values on $\{\langle v \rangle \mid v \in VR\}$.)
9. Define $VH_v = \langle \text{Type}(v), v \rangle$, for $v \in VR$ (a *variable hypothesis*).
10. A *pre-statement* is a tuple $\langle D, H, A \rangle$ where $D \subseteq DV$, $H \subseteq EX$ is finite, and $A \in EX$.
11. The *reduct* of $\langle D, H, A \rangle$ is $\langle D_M, H, A \rangle$ where $D_M = D \cap \mathcal{P}(\mathcal{V}(H \cup \{A\}))$, and a statement is defined as the reduct of some pre-statement.
12. A *formal system* is a tuple $\langle CN, VR, \text{Type}, \Gamma \rangle$ where CN, VR, Type are as above and Γ is a set of statements.
13. The *closure* of a set $H \subseteq EX$ relative to D is the smallest set C such that:
 - $H \cup \{VH_v \mid v \in VR\} \subseteq C$
 - For every $\langle D', H', A' \rangle \in \Gamma$ and every substitution σ , if
 - For all $e \in H' \cup \{VH_v \mid v \in VR\}$, $\sigma(e) \in C$, and
 - For all $\{\alpha, \beta\} \in D'$, if $\gamma \in \mathcal{V}(\sigma(VH_\alpha))$ and $\delta \in \mathcal{V}(\sigma(VH_\beta))$, then $\{\gamma, \delta\} \in D$,
 then $\sigma(A') \in C$.
14. A pre-statement $\langle D, H, A \rangle$ is *provable* if A is in the closure of H relative to D , and a theorem is a statement that is the reduct of a provable pre-statement.
15. Let TC be the set of typecodes of theorems. (Explicitly, this is $TC = VT \cup \{\text{Type}(A) \mid \langle D, H, A \rangle \in \Gamma\}$.)
16. Two formal systems $\langle CN, VR, \text{Type}, \Gamma \rangle$ and $\langle CN, VR, \text{Type}, \Gamma' \rangle$ are *equivalent* if they generate the same set of theorems (or equivalently, if every axiom in one is a theorem of the other).

Why a global type function? A careful comparison with Appendix C of the Metamath book [1] shows that in the original definition a variable only has a type locally (inside a statement), while we require all variables to have a unique and globally defined type, provided by the Type function. In practice, variables are never reintroduced with a different type, so this is not a strong requirement.

Additionally, there is some ongoing work to amend the specification to disallow such multi-typed variables.

Nevertheless, it is a simple fix to convert a formal system with multi-typed variables to one with a global type function: Take the set of variables to be $\mathcal{VT} \times \mathcal{VR}$, and define $\text{Type}(c, v) = c$. Then whenever a variable v appears in a statement with type c , use the variable $\langle c, v \rangle$ instead. This is equivalent to just prepending the type of the variable to its name, so that uses of the same variable with a different type are distinguished.

2.2 Models of formal systems

Fix a collection of sets U_c for $c \in \mathcal{TC}$, which will represent the “universe” of objects of each typecode.

Definition 1. A valuation is a function μ on \mathcal{VR} such that $\mu(v) \in U_{\text{Type}(v)}$ for all $v \in \mathcal{VR}$. The set of all valuations is denoted by \mathcal{VL} .

Definition 2. A freshness relation $\#$ is a symmetric relation on the disjoint union $\bigsqcup U = \bigsqcup_{c \in \mathcal{TC}} U_c$ such that for any $c \in \mathcal{VT}$ and any finite set $W \subseteq \bigsqcup U$, there is a $v \in U_c$ with $v \# w$ for all $w \in W$.

Definition 3. A model of the formal system $\langle \mathcal{CN}, \mathcal{VR}, \text{Type}, \Gamma \rangle$ is a tuple $\langle U, \#, \eta \rangle$ where U is a function on \mathcal{TC} and $\#$ is a freshness relation, and for each $\mu \in \mathcal{VL}$, η_μ is a partial function on \mathcal{EX} such that:

- (Type correctness) For all $e \in \mathcal{EX}$, if $\eta_\mu(e)$ is defined then $\eta_\mu(e) \in U_{\text{Type}(e)}$.
- (Variable application) For all $v \in \mathcal{VR}$, $\eta_\mu(\mathcal{V}H_v) = \mu(v)$.
- (Axiom application) For each $\langle D, H, A \rangle \in \Gamma$, if
 - $\mu(\alpha) \# \mu(\beta)$ for all $\{\alpha, \beta\} \in D$, and
 - $\eta_\mu(h)$ is defined for all $h \in H$,
 then $\eta_\mu(A)$ is defined.
- (Substitution property) For each substitution σ and $e \in \mathcal{EX}$, $\eta_\mu(\sigma(e)) = \eta_{\sigma(\mu)}(e)$, where $\sigma(\mu) \in \mathcal{VL}$ is defined by $\sigma(\mu)(v) = \eta_\mu(\sigma(\mathcal{V}H_v))$.
- (Dependence on present variables) For all $\nu \in \mathcal{VL}$, $e \in \mathcal{EX}$, If $\mu(v) = \nu(v)$ for all $v \in \mathcal{V}(e)$, then $\eta_\mu(e) = \eta_\nu(e)$.
- (Freshness substitution) For all $v \in \bigsqcup U$, $e \in \mathcal{EX}$, if $\eta_\mu(e)$ is defined and $v \# \mu(w)$ for all $w \in \mathcal{V}(e)$, then $v \# \eta_\mu(e)$.

Here equality means that one side is defined iff the other is and they have the same value. We say that $e \in \mathcal{EX}$ is true in the model if $\eta_\mu(e)$ is defined for all $\mu \in \mathcal{VL}$.

The key property of a model is *soundness*, the fact that the axiom application law applies also to theorems.

Theorem 1. For any theorem $\langle D, H, A \rangle$, if $\mu(\alpha) \# \mu(\beta)$ for all $\{\alpha, \beta\} \in D$ and $\eta_\mu(h)$ is defined for all $h \in H$, then $\eta_\mu(A)$ is defined.

Proof. By dependence on present variables, we may replace μ by any other μ' such that $\mu(v) = \mu'(v)$ for all $v \in \mathcal{V}(H \cup \{A\})$ without affecting the truth of the hypotheses or conclusion. If $\langle D, H, A \rangle$ is the reduct of $\langle D', H, A \rangle$ where D' refers to finitely many additional variables (i.e. $D' \subseteq \mathcal{P}(V)$ for some finite set $V \supseteq \mathcal{V}(H \cup \{A\})$), order these as $V = \{v_1, \dots, v_n\}$ with $\mathcal{V}(H \cup \{A\}) = \{v_1, \dots, v_k\}$ for some $k \leq n$. Then use the freshness constraint to recursively select values $\mu'(v_i)$ for each $k < i \leq n$ such that $\mu'(v_i) \# \mu'(v_j)$ for all $j < i$. Then this new μ' will satisfy the hypothesis $\mu'(\alpha) \# \mu'(\beta)$ for all $\{\alpha, \beta\} \in D'$, so that it suffices to prove the theorem for provable pre-statements.

We prove by induction that whenever A is in the closure of H relative to D , $\eta_\mu(A)$ is defined. If $A \in H$, it is true by assumption, and if $A = \mathbf{V}H_v$ for some $v \in \mathbf{V}R$, then it is true by the variable application law. Otherwise we are given $\langle D', H', A' \rangle \in \Gamma$ and a substitution σ , such that for all $e \in H' \cup \{\mathbf{V}H_v \mid v \in \mathbf{V}R\}$, $\eta_\mu(\sigma(e))$ is defined (by the induction hypothesis), and for all $\{\alpha, \beta\} \in D'$, if $\gamma \in \mathcal{V}(\sigma(\mathbf{V}H_\alpha))$ and $\delta \in \mathcal{V}(\sigma(\mathbf{V}H_\beta))$, then $\{\gamma, \delta\} \in D$, and we wish to show that $\eta_\mu(A)$, with $A = \sigma(A')$, is defined.

For each $\gamma \in \mathcal{V}(\sigma(\mathbf{V}H_\alpha))$ and $\delta \in \mathcal{V}(\sigma(\mathbf{V}H_\beta))$, $\{\gamma, \delta\} \in D$ implies $\mu(\gamma) \# \mu(\delta)$ from the theorem hypothesis, hence by freshness substitution on the left and the right, $\mu(\gamma) \# \eta_\mu(\sigma(\mathbf{V}H_\beta))$, and then $\eta_\mu(\sigma(\mathbf{V}H_\alpha)) \# \eta_\mu(\sigma(\mathbf{V}H_\beta))$, or equivalently, $\sigma(\mu)(\alpha) \# \sigma(\mu)(\beta)$ for each $\{\alpha, \beta\} \in D'$.

Apply the axiom application law to $\sigma(\mu)$ and $\langle D', H', A' \rangle$. The substitution property reduces $\eta_\mu(\sigma(e))$ to $\eta_{\sigma(\mu)}(e)$ in the hypotheses, and $\eta_{\sigma(\mu)}(A')$ to $\eta_\mu(\sigma(A'))$ in the conclusion, hence $\eta_\mu(\sigma(A'))$ is defined, as we wished to show. \square

In particular, if $\langle \emptyset, \emptyset, A \rangle$ is provable, then $\eta_\mu(A)$ is defined for all $\mu \in \mathbf{V}L$, which makes it a useful technique for showing that certain strings are not provable (see § 4.1).

For any formal system, there is a model, where $U_c = \{*\}$ for each c , $* \# *$ is true, and $\eta_\mu(e) = *$ for all μ, e . Thus statements like “formal system X has a model” are not as useful here as they are in first-order logic. To marginalize this kind of model, we will call a model where each η_μ is a total function *trivial*. (We will have a slightly wider definition of trivial model given grammatical information, cf. Definition 6.)

Although this defines the property of being a model under the full generality of Metamath formal systems, the process simplifies considerably when expressions can be parsed according to a grammar.

2.3 Grammatical parsing

Definition 4. A formal system is said to be weakly grammatical if for every $\langle D, H, A \rangle \in \Gamma$, if $\text{Type}(A) \in \mathbf{V}T$, then there is some axiom $\langle \emptyset, \emptyset, A' \rangle \in \Gamma$ such that $\sigma(A') = A$ for some substitution σ and no variable occurs more than once in A' .

For these systems we will define

$$\mathbf{S}A = \{A \mid \langle \emptyset, \emptyset, A \rangle \in \Gamma \wedge \text{Type}(A) \in \mathbf{V}T \wedge \forall mn, (A_m = A_n \in \mathbf{V}R \rightarrow m = n)\},$$

the set of syntax axioms. (We will identify the expression A with its statement $\langle \emptyset, \emptyset, A \rangle$ when discussing syntax axioms.)

For example, any context-free grammar is a weakly grammatical formal system, where each production translates to a syntax axiom, and each nonterminal translates to a variable typecode. Most recursive definitions of a well-formed formula will fit this bill, although we can't capture the notion of bound variables with this alone.

Conversely, a weakly grammatical formal system yields a context-free grammar, where the terminals are $CN \setminus VT$, the non-terminals are VT , and for each $A \in SA$ there is a production $\text{Type}(A) \rightarrow \alpha$, where $\alpha_n = A_{n+1} \in CN$ if $A_{n+1} \in CN$ or $\alpha_n = \text{Type}(A_{n+1}) \in VT$ if $A_{n+1} \in VR$. (This assumes that $A_{n+1} \in VT$ is always false, but the two sets can be disjointified if this is not the case.)

Definition 5. A grammatical formal system is a weakly grammatical formal system augmented with a function $\text{Syn} : TC \rightarrow VT$ such that $\text{Syn}(c) = c$ for all $c \in VT$ and, defining $\text{Syn}(e)$ for $e \in EX$ such that $\text{Syn}(e)_0 = \text{Syn}(e_0)$ and $\text{Syn}(e)_n = e_n$ for $n > 0$, $\langle \emptyset, \emptyset, \text{Syn}(e) \rangle$ is a provable statement for all $\langle D, H, A \rangle \in \Gamma$ and $e \in H \cup \{A\}$.

Remark 1. Of course, this notion is of interest primarily because it is satisfied by all major Metamath databases; in particular, `set.mm` is a grammatical formal system, with $VT = \{\text{set, class, wff}\}$, $TC = VT \cup \{\vdash\}$, and $\text{Syn}(\vdash) = \text{wff}$.

Definition 6. A model of a grammatical formal system is a model in the sense of Definition 3 which additionally satisfies $U_c \subseteq U_{\text{Syn}(c)}$, $v \# w_c \leftrightarrow v \# w_{\text{Syn}(c)}$ when $w \in U_c$ (and $w_c, w_{\text{Syn}(c)}$ are its copies in the disjoint union), and $\eta_\mu(e) = \eta_\mu(\text{Syn}(e))$ if the latter is in U_c , otherwise undefined. Such a model is trivial if $U_c = U_{\text{Syn}(c)}$ for all $c \in TC$.

2.4 Tree representation of formal systems

The inductive definition of the closure of a set of statements immediately leads to a tree representation of proofs. A proof tree is a tree with nodes labeled by statements and edges labeled by expressions.

Definition 7. We inductively define the statement “ T is a proof tree for A ” (relative to D, H) as follows:

- For each $e \in H \cup \{VH_v \mid v \in VR\}$, the single-node tree labeled by the reduct of $\langle D, H, e \rangle$ is a proof tree for e .
- For every $\langle D', H', A' \rangle \in \Gamma$ and every substitution σ satisfying the conditions for $\sigma(A') \in C$ in § 2.1.13, the tree labeled by $\langle D', H', A' \rangle$ with edges for each $e \in H' \cup \mathcal{V}(H \cup \{A\})$ leading to a proof tree for $\sigma(e)$, is a proof tree for $\sigma(A')$.

The definition of closure ensures that there is a proof tree for A relative to D, H iff $\langle D, H, A \rangle$ is provable pre-statement. (The branches for variables outside $\mathcal{V}(H \cup \{A\})$ are discarded because they can always be replaced by the trivial substitution $\sigma(\langle v \rangle) = \langle v \rangle$ without affecting the closure deduction.) Additionally, we can prove by induction that every proof tree T encodes a unique expression $\text{Expr}(T)$.

Definition 8. *An unambiguous formal system is a grammatical formal system whose associated context-free grammar is unambiguous.*

Remark 2. Note that for this to make sense we need SA to contain only axioms and not theorems, i.e. this property is not preserved by equivalence of formal systems. For such systems every Expr is an injection when restricted to the set ST of *syntax trees*, trees T relative to \emptyset, \emptyset such that $\text{Type}(T) := \text{Type}(\text{Expr}(T)) \in VT$ (or equivalently, $\text{Type}(T) = \text{Type}(A) \in VT$ where $\langle D, H, A \rangle$ is the root of T). The subtrees of a syntax tree are also syntax trees, and the nodes are syntax axioms, with variables (of the form VH_v) at the leaves.

With this, we can “rebuild” the whole theory using trees instead of strings, because the all valid substitutions have unique proof tree representations. The expressions in this new language will be trees, whose nodes are syntax axioms such as $\mathbf{wa} = \text{“wff } (\varphi \wedge \psi)\text{”}$, representing the “and” function, with variables at the leaves. However, we no longer need to know that \mathbf{wa} has any structure of its own, besides the fact that it takes two wff variables and produces a wff. Thus we can discard the set CN entirely. (That is, the constant “(” has no meaning of its own here.)

Instead, we take as inputs to the construction the set TC' of typecodes, a set VR' of variables, a set SA' of things we call syntax axioms, although they have no internal structure, the function $\text{Type}' : VR' \cup SA' \rightarrow VT'$, as well as $\text{Syn}' : TC' \rightarrow VT'$. A tree $T \in ST'$ is either a variable from VR' or a syntax axiom $a \in SA'$ connecting to more subtrees; each syntax axiom has a set v_i^a of variables labeling the edges, and a type $\text{Type}'(a)$; $\text{Type}'(T)$ is defined as the type of the root of T .

We replace EX in the string representation with EX' , which consists of tuples $\langle c, T \rangle$ where $c \in TC'$, $T \in ST'$, and $\text{Syn}'(c) = \text{Type}'(T)$. We extend Type' to EX' by $\text{Type}'(\langle c, T \rangle) = c$. $\mathcal{V}'(T)$ is defined by induction such that $\mathcal{V}'(v) = \{v\}$ and $\mathcal{V}'(T)$ at a syntax axiom is the union of $\mathcal{V}'(T_i)$ over the child subtrees T_i . A substitution σ is a function $ST' \rightarrow ST'$ such that $\sigma(a[T_1, \dots, T_n]) = a[\sigma(T_1), \dots, \sigma(T_n)]$ for each syntax axiom a , with the value at variables left undetermined, extended to $EX' \rightarrow EX'$ by $\sigma(\langle c, T \rangle) = \langle c, \sigma(T) \rangle$.

Pre-statements and statements are defined exactly as before: A pre-statement is a tuple $\langle D, H, A \rangle$ where $D \subseteq DV'$, $H \subseteq EX'$ is finite, and $A \in EX'$. The reduct of $\langle D, H, A \rangle$ is $\langle D_M, H, A \rangle$ where $D_M = D \cap \mathcal{P}(\mathcal{V}'(H \cup \{A\}))$. A tree formal system (this time unambiguous by definition) is a tuple $\langle TC', VR', SA', \text{Type}', \text{Syn}', \Gamma' \rangle$ where Γ' is a set of statements. The closure of a set $H \subseteq EX'$ relative to D is defined as in the string case, but the base case instead takes $H \subseteq C$ and $\langle \text{Type}'(T), T \rangle \in C$ for every $T \in ST'$.

To map an unambiguous formal system $\langle CN, VR, \text{Type}, \Gamma, \text{Syn} \rangle$ to a tree formal system $\langle \mathcal{TC}', VR', SA', \text{Type}', \text{Syn}', \Gamma' \rangle$, one takes $\mathcal{TC}' = \mathcal{TC}$, VR' to be the set of VH_v singleton trees for $v \in VR$, $SA' = SA$, $\text{Type}'(T) = \text{Type}(T)$, $\text{Syn}' = \text{Syn}$, and $\Gamma' = \{\langle D, \{t(h) \mid h \in H\}, t(A) \rangle \mid \langle D, H, A \rangle \in \Gamma \setminus ST\}$, where $t(e) = \{\text{Type}(e), \text{Expr}^{-1}(\text{Syn}(e))\}$.

These two formal systems are isomorphic, in the sense that expressions and statements can be mapped freely, respecting the definitions of theorems and axioms, variables and typecodes.

2.5 Models of tree formal systems

Given the isomorphism of the previous section, the model theory of unambiguous formal systems can be mapped to models of tree formal systems, with $\langle U, \#, \eta \rangle$ satisfying an exactly equivalent set of properties. But the major advantage of the tree formulation is that the substitution property implies that η is completely determined except at syntax axioms, so for trees we will replace η with a new function π .

Definition 9. *Given a function U on \mathcal{TC}' satisfying $U_c \subseteq U_{\text{Syn}(c)}$, and π a SA -indexed family of functions, where $\pi_a : \prod_i U_{\text{Type}(v_i^a)} \rightarrow U_{\text{Type}(a)}$ for each $a \in SA$, define η_μ for $\mu \in \text{VL}$ recursively such that:*

- For all $v \in VR$, $\eta_\mu(v) = \mu(v)$.
- For each $T \in ST$ with $a \in SA$ at the root, $\eta_\mu(T) = \pi_a(\{\eta_\mu(T_i)\}_i)$
- For $e = \langle c, T \rangle \in EX'$, $\eta_\mu(e) = \eta_\mu(T)$ if $\eta_\mu(T) \in U_c$, otherwise undefined.

Definition 10. *A model of a tree formal system is a tuple $\langle U, \#, \pi \rangle$ where U is a function on \mathcal{TC}' satisfying $U_c \subseteq U_{\text{Syn}(c)}$, and $\#$ is a freshness relation (which is extended from $\bigsqcup_{v \in VT} U_c$ to $\bigsqcup_{v \in \mathcal{TC}} U_c$ by setting $v \# w_c$ iff $v \# w_{\text{Syn}(c)}$ for the copies of w in the disjoint union), and π is a SA -indexed family of functions, where $\pi_a : \prod_i U_{\text{Type}(v_i^a)} \rightarrow U_{\text{Type}(a)}$ for each $a \in SA$, such that for each $\mu \in \text{VL}$, and defining η as above:*

- For each $\langle D, H, A \rangle \in \Gamma'$, if
 - $\mu(\alpha) \# \mu(\beta)$ for all $\{\alpha, \beta\} \in D$, and
 - $\eta_\mu(h)$ is defined for all $h \in H$,
 then $\eta_\mu(A)$ is defined.
- For all $v \in \bigsqcup U$, $a \in SA$, and $f \in \prod_i U_{\text{Type}(v_i^a)}$, if $v \# f_i$ for all i , then $v \# \pi_a(f)$.

Theorem 2. *Let $\langle U, \#, \pi \rangle$ be a model for the tree formal system. Then the associated $\langle U, \#, \eta \rangle$ is a model in the sense of Definition 6.*

Proof.

- The variable application law is true by definition of η , and the axiom application law is true by assumption.

- For the substitution law, suppose σ and $e \in EX'$ are given. We want to show that $\eta_\mu(\sigma(e)) = \eta_{\sigma(\mu)}(e)$, where $\sigma(\mu) \in \mathbf{VL}$ is defined by $\sigma(\mu)(v) = \eta_\mu(\sigma(v))$. We show this for $\text{Syn}(e) = \langle \text{Type}(T), T \rangle$ by induction on T . In the base case, $T = v$ for $v \in \mathbf{VT}$, so $\eta_{\sigma(\mu)}(v) = \sigma(\mu)(v) = \eta_\mu(\sigma(v))$. Otherwise let $a \in \mathbf{SA}$ be the root of T , so

$$\eta_{\sigma(\mu)}(T) = \pi_a(\{\eta_{\sigma(\mu)}(T_i)\}_i) = \pi_a(\{\eta_\mu(\sigma(T_i))\}_i) = \eta_\mu(\sigma(T)).$$

Then for $e = \langle c, T \rangle$, if $\eta_{\sigma(\mu)}(T) = \eta_\mu(\sigma(T))$ is in U_c , then both are defined and equal, otherwise both are undefined.

- Dependence on present variables is also provable by induction; in the base case $\eta_\mu(v) = \mu(v)$ only depends on $\mathcal{V}(v) = \{v\}$; and for a tree with $a \in \mathbf{SA}$ at the root, $\eta_\mu(T) = \pi_a(\{\eta_\mu(T_i)\}_i)$ only depends on $\mathcal{V}(T) = \bigcup_i \mathcal{V}(T_i)$. For $e = \langle c, T \rangle$ this property is maintained since $\mathcal{V}(e) = \mathcal{V}(T)$ and $\eta_\mu(e) = \eta_\mu(T)$ or undefined.
- For the freshness substitution law, in the base case $v \# \mu(v) = \eta_\mu(v)$; and for a tree with $a \in \mathbf{SA}$ at the root, if $v \# \mathcal{V}(T)$ then $v \# \mathcal{V}(T_i)$ so $v \# \eta_\mu(T_i)$ for each i , and then by definition $v \# \pi_a(\{\eta_\mu(T_i)\}_i) = \eta_\mu(T)$. For $e = \langle c, T \rangle$ this property is maintained since $\mathcal{V}(e) = \mathcal{V}(T)$ and $\eta_\mu(e) = \eta_\mu(T)$ or undefined. \square

By the isomorphism this approach also transfers to models of unambiguous formal systems. So we are left with the conclusion that a model can be specified by its functions π_a , for each $a \in \mathbf{SA}$; this is known in conventional model theory as the interpretation function.

3 Examples of models

3.1 Propositional logic

We start with a model for classical propositional logic. We define:

$$CN = \{(\cdot), \rightarrow, \neg, \text{wff}, \vdash\}$$

$$VR = \{\varphi, \psi, \chi, \dots\}$$

$$\Gamma = \{\text{wn}, \text{wi}, \text{ax-1}, \text{ax-2}, \text{ax-3}, \text{ax-mp}\},$$

where the axioms are

- **wn**: wff $\neg\varphi$
- **wi**: wff $(\varphi \rightarrow \psi)$
- **ax-1**: $\vdash (\varphi \rightarrow (\psi \rightarrow \varphi))$
- **ax-2**: $\vdash ((\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi)))$
- **ax-3**: $\vdash ((\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \varphi))$
- **ax-mp**: $\{\vdash \varphi, \vdash (\varphi \rightarrow \psi)\}$ implies $\vdash \psi$

Additionally, $TC = \{\text{wff}, \vdash\}$ and $VT = \{\text{wff}\}$ are implied by the preceding definition.

Although the axiom strings appear structured with infix notation, this is not required; we could just as easily have an axiom with string $\vdash \varphi(\rightarrow)$. That this is not the case is what makes this a grammatical formal system, with $\text{Syn}(\text{wff}) = \text{Syn}(\vdash) = \text{wff}$. The syntax axioms are $SA = \{\text{wn}, \text{wi}\}$. In fact this formal system is unambiguous, but we will not prove this here.

This formal system has a nontrivial model:

$$\begin{aligned} U_{\text{wff}} = \text{Bool} &:= \{\mathbf{T}, \mathbf{F}\} & U_{\vdash} &= \{\mathbf{T}\} \\ x \# y &\text{ is always true} \\ \pi_{\neg}(\mathbf{F}) &= \mathbf{T}, & \pi_{\neg}(\mathbf{T}) &= \mathbf{F} \\ \pi_{\rightarrow}(\mathbf{F}, \mathbf{F}) &= \mathbf{T}, & \pi_{\rightarrow}(\mathbf{F}, \mathbf{T}) &= \mathbf{T}, & \pi_{\rightarrow}(\mathbf{T}, \mathbf{F}) &= \mathbf{F}, & \pi_{\rightarrow}(\mathbf{T}, \mathbf{T}) &= \mathbf{T} \end{aligned}$$

We will write “ $\alpha = \mathbf{T}$ ” simply as “ α is true” or “ α ”, treating the elements of Bool as actual truth values in the metalogic. The π_a functions generate η as previously described, so that, for example, if $\mu(\varphi) = \mathbf{T}$, $\mu(\psi) = \mathbf{F}$, then $\eta_{\mu}(\text{wff } (\neg\varphi \rightarrow (\varphi \rightarrow \psi))) = \pi_{\rightarrow}(\pi_{\neg}(\mathbf{T}), \pi_{\rightarrow}(\mathbf{T}, \mathbf{F})) = \pi_{\rightarrow}(\mathbf{F}, \mathbf{F}) = \mathbf{T}$.

In order to verify that this indeed yields a model, we must check the axiom application law for each non-syntax axiom (usually called “logical axioms” in this context). By our definition of η given π , $\eta_{\mu}(\langle c, T \rangle)$ is defined iff $\eta_{\mu}(\langle \text{Syn}(c), T \rangle) \in U_{\text{Syn}(c)}$ (where $\text{Syn}(c) = \text{Type}(T)$); in this case this translates to $\eta_{\mu}(T) = \mathbf{T}$ since $\text{Syn}(c) = \{\vdash\}$.

- **ax-1**: $\eta_{\mu}(\vdash (\varphi \rightarrow (\psi \rightarrow \varphi))) = \pi_{\rightarrow}(\mu_{\varphi}, \pi_{\rightarrow}(\mu_{\psi}, \mu_{\varphi})) = \mathbf{T}$
- **ax-2**: $\pi_{\rightarrow}(\pi_{\rightarrow}(\mu_{\varphi}, \pi_{\rightarrow}(\mu_{\psi}, \mu_{\chi})), \pi_{\rightarrow}(\pi_{\rightarrow}(\mu_{\varphi}, \mu_{\psi}), \pi_{\rightarrow}(\mu_{\varphi}, \mu_{\chi}))) = \mathbf{T}$
- **ax-3**: $\pi_{\rightarrow}(\pi_{\rightarrow}(\pi_{\neg}(\mu_{\varphi}), \pi_{\neg}(\mu_{\psi})), \pi_{\rightarrow}(\mu_{\psi}, \mu_{\varphi})) = \mathbf{T}$
- **ax-mp**: If μ_{φ} and $\pi_{\rightarrow}(\mu_{\varphi}, \mu_{\psi})$ are true, then $\mu_{\psi} = \mathbf{T}$.

In each case, there are a finite number of variables that range over $\{\mathbf{T}, \mathbf{F}\}$ (such as $\mu_{\varphi}, \mu_{\psi}, \mu_{\chi}$ in the case of **ax-3**), so it suffices to verify that they are true under all combinations of assignments to the variables, i.e. truth table verification.

3.2 MIU system

Let us solve the MU puzzle by using a model. Hofstadter’s MIU system [3] is defined in Appendix D of the Metamath book [1], so we will just define the model itself. We have $TC = \{\text{wff}, \vdash\}$ and $VT = \{\text{wff}\}$, and let $\text{Syn}(\vdash) = \text{wff}$ with x, y variables of type wff. The axioms are:

- **we**: wff
- **wM**: wff xM
- **wI**: wff xI
- **wU**: wff xU
- **ax**: $\vdash MI$

- I.: $\vdash xI$ implies $\vdash xIU$
- II: $\vdash Mx$ implies $\vdash Mxx$
- III: $\vdash xIIIy$ implies $\vdash xUy$
- IV: $\vdash xUUy$ implies $\vdash xy$

The syntax axioms are $\{\mathbf{we}, \mathbf{wM}, \mathbf{wI}, \mathbf{wU}\}$. Note that in \mathbf{we} , there are no symbols after the typecode, so this says that the empty string is a valid wff. This formal system is weakly grammatical, but not grammatical, because wffs are built from the right, while axiom II contains the string $\vdash Mx$, which cannot be parsed as a wff. If there was a syntax axiom “wff xy ”, then it would be grammatical, but not unambiguous.

In order to solve the MU puzzle, we build the relevant invariant, which is the number of I’s modulo 3, into the model. Let $U_{\text{wff}} = \{0, 1, 2\}$, $U_{\vdash} = \{1, 2\}$, let $x \# y$ be always true, and define $\eta_{\mu}(\text{wff } e)$ to be $\sum_i f(e_i) \bmod 3$, where $f(v) = \mu(v)$ if v is a variable, $f(I) = 1$, and $f(c) = 0$ for other constants. Let $\eta_{\mu}(\vdash e) = \eta_{\mu}(\text{wff } e)$ when $\eta_{\mu}(\text{wff } e) \in \{1, 2\}$.

Verifying that this yields a model is then equivalent to verifying that the axioms preserve the invariant, and we can deduce that MU is not provable because $\eta_{\mu}(\vdash \text{MU})$ is not defined for any μ .

3.3 Set theory

Of course, the more interesting case is the verification that the full structure of `set.mm` has a nontrivial model. As the background, we need a model of ZFC set theory; let this be $\langle M, \varepsilon \rangle$.

The typecodes are $TC = \{\text{set}, \text{class}, \text{wff}, \vdash\}$, with the only non-variable typecode being \vdash and $\text{Syn}(\vdash) = \text{wff}$. With these definitions `set.mm` becomes an unambiguous formal system. (Again, the proof of unambiguity is complex and not undertaken here.) Let $V =: U_{\text{set}}$ be any infinite set. This is the set of variables of the “object language” over which Metamath is understood to sit; they are customarily labeled $V = \{v_0, v_1, v_2, \dots\}$. Note that these are *not* actually variables in our sense, they are constants, elements of the set V . Set variables such as x in Metamath range over elements of V .

We will need a few preliminary sets before properly defining U_{wff} and U_{class} . Take $U'_{\text{wff}} = (V \rightarrow M) \rightarrow \mathbf{Bool}$, that is, the set of functions from $V \rightarrow M$ to \mathbf{Bool} . The subset of U_{wff} corresponding to true formulas, U_{\vdash} , is the singleton $\{\lambda f \mathbf{T}\}$ (i.e. the constant function true). Similarly, $U'_{\text{class}} = (V \rightarrow M) \rightarrow \mathcal{P}(M)$.

Definition 11. *Given $A : (V \rightarrow M) \rightarrow B$ (A is a wff or class variable) and $W \subseteq V$, we say A is constant outside W if for all $f, g : V \rightarrow M$, if $f(v) = g(v)$ for all $v \in W$, then $A(f) = A(g)$.*

We define a relation $\#$ on the disjoint union $V \sqcup U'_{\text{wff}} \sqcup U'_{\text{class}}$:

- If $x, y \in V$, then define $x \# y$ iff $x \neq y$.
- If $x \in V$ and $A \in U'_{\text{class}} \sqcup U'_{\text{wff}}$, then define $x \# A$ if A is constant outside $V \setminus \{x\}$.

- The case $A \# x$ when $x \in V$ and $A \in \mathcal{U}_{\text{class}} \sqcup \mathcal{U}_{\text{wff}}$ is covered by symmetry; $x \# y$ is true for any other combination.

To define the real set \mathcal{U}_{wff} , we take the set of $A \in \mathcal{U}'_{\text{wff}}$ such that A is “effectively finite-dimensional”, that is, A is constant outside some finite $V' \subseteq V$. Similarly, $\mathcal{U}_{\text{class}}$ is the set of effectively finite-dimensional $A \in \mathcal{U}'_{\text{class}}$. There is a minimal set V' outside which A is constant; this set is called $\text{Free}(A)$. It immediately follows from the definition that $A \# x$ for $x \notin \text{Free}(A)$. We can extend the definition slightly to set variables by taking $\text{Free}(x) = \{x\}$ when $x \in V$.

Theorem 3. *# as defined above is a freshness relation.*

Proof. Clearly $\#$ is a symmetric relation, so we need only verify that for every $c \in \{\text{set}, \text{wff}, \text{class}\}$ and every finite set $W \subseteq \mathcal{VR}$, there is a $v \in \mathcal{U}_c$ with $v \# W$. If $c = \text{wff}$ then take $v = \lambda f \mathbf{T}$, and if $c = \text{class}$ then take $v = \lambda f M$; in each case $v \# w$ for any $w \in \sqcup \mathcal{U}$, so the condition is satisfied.

If $c = \text{set}$, then for each $w \in \mathcal{U}_c$ the set $\text{Free}(w)$ is finite, as is the union $\bigcup_{w \in W} \text{Free}(w)$. Since V is infinite by assumption, choose some $v \in V \setminus \bigcup_{w \in W} \text{Free}(w)$; then for $w \in W$, $v \notin \text{Free}(w)$ implies $v \# w$. \square

Definition 12. *For $f : V \rightarrow M$, $x \in V$ and $m \in M$, the function $f[x \rightarrow m] : V \rightarrow M$ is defined by $f[x \rightarrow m](y) = f(y)$ for $y \neq x$ and $f[x \rightarrow m](x) = m$.*

Finally, we define the π_a functions. By definitional elimination, we can ignore definitional syntax axioms without loss of generality.

- Take $\pi_{\rightarrow}(f) = f \circ \pi'_{\rightarrow}$, where π'_{\rightarrow} is the function called π_{\rightarrow} in § 3.1, and similarly for $\pi_{\leftarrow}(f) = f \circ \pi'_{\leftarrow}$.
- $\pi_{\forall}(x, \varphi)$ is the wff corresponding to $\forall x, \varphi$ and is defined so that $\pi_{\forall}(x, \varphi)(f)$ iff $\varphi(f[x \rightarrow m])$ for all $m \in M$. (Since it comes up often, the restricted quantifier $\forall m \in M$ will be abbreviated $\forall_M m$.)
- The class abstraction, **cab**: $\text{class } \{x \mid \varphi\}$, is defined so that $\pi_{\text{cab}}(x, \varphi)(f) = \{m \in M \mid \varphi(f[x \rightarrow m])\}$.
- The set-to-class type conversion is a syntax axiom called **cv**: $\text{class } x$. The function for this is defined so that $\pi_{\text{cv}}(x)(f) = \{m \in M \mid m \varepsilon f(x)\}$.
- Equality of classes is defined by **wceq**: $\text{wff } A = B$, and is defined so that $\pi_{=} (A, B)(f)$ is true iff $A(f) = B(f)$.
- We define $\pi_{\in} (A, B)(f)$ true if $\exists_M m (A(f) = \{n \in M \mid n \varepsilon m\} \wedge m \in B(f))$.

For the common case where one or both of the arguments to $=, \in$ are sets, we note that $\pi_{\in}(\pi_{\text{cv}}(x), A)(f)$ iff $f(x) \in A(f)$, $\pi_{\in}(\pi_{\text{cv}}(x), \pi_{\text{cv}}(y))(f)$ iff $f(x) \varepsilon f(y)$, and $\pi_{=}(\pi_{\text{cv}}(x), \pi_{\text{cv}}(y))(f)$ iff $f(x) = f(y)$. We need $\langle M, \varepsilon \rangle$ to satisfy the extensionality axiom for this to work.

Lemma 1 (The deduction theorem). *$\pi_{\rightarrow}(\varphi, \psi)(f)$ if and only if $\varphi(f)$ implies $\psi(f)$.*

Proof. Suppose not. Then $\pi_{\rightarrow}(\varphi, \psi)(f) = \mathbf{F}$, which by definition implies $\varphi(f) = \mathbf{T}$ and $\psi(f) = \mathbf{F}$, a contradiction. The converse is just **ax-mp** (verified by truth tables). \square

Lemma 2 (The non-free predicate). $x \# \varphi$ iff $\pi_{\rightarrow}(\varphi, \pi_{\forall}(x, \varphi))(f)$ is true for all f (which is also equivalent to $\eta_{\mu}(\vdash (\varphi' \rightarrow \forall x' \varphi'))$ being defined, where $\mu(\varphi') = \varphi$ and $\mu(x') = x$).

Proof. By definition, $x \# \varphi$ iff for all $f, g : V \rightarrow M$, $f(v) = g(v)$ for all $v \neq x$ implies $\varphi(f) = \varphi(g)$. In this case, given f , and using the deduction theorem, if $\varphi(f)$, then since $f[x \rightarrow m]$ differs from f only at x , $\varphi(f[x \rightarrow m]) = \varphi(f)$ is true for each m , so $\pi_{\forall}(x, \varphi)(f)$. Thus $\pi_{\rightarrow}(\varphi, \pi_{\forall}(x, \varphi))(f)$ by Lemma 1. Conversely, if f, g differ only for $v = x$, either $\varphi(f) = \varphi(g) = \mathbf{F}$, or one of them (say $\varphi(f)$) is true. Then by **ax-mp**, $\pi_{\forall}(x, \varphi)(f)$, so taking $m = g(x)$, $\varphi(f[x \rightarrow g(x)]) = \varphi(g)$ is true. Hence $\varphi(f) = \varphi(g)$, so $x \# \varphi$. \square

Theorem 4. The tuple $\langle U, \#, \eta \rangle$ defined via the above construction is a model for the **set.mm** formal system.

Proof. As in the baby example for propositional calculus, we must verify that η honors all the logical axioms. The tricky ones are the predicate calculus axioms:

- **ax-1, ax-2, ax-3, ax-mp:** Verification by truth tables, as in the propositional calculus example
- **ax-5:** $\vdash (\forall x(\varphi \rightarrow \psi) \rightarrow (\forall x\varphi \rightarrow \forall x\psi))$
Assume $\pi_{\forall}(x, \pi_{\rightarrow}(\varphi, \psi))(f)$ and $\pi_{\forall}(x, \varphi)(f)$; then $\varphi(f[x \rightarrow m])$ and $\pi_{\rightarrow}(\varphi, \psi)(f[x \rightarrow m])$, so by **ax-mp**, $\psi(f[x \rightarrow m])$. Conclude by Lemma 1.
- **ax-6:** $\vdash (\neg\forall x\varphi \rightarrow \forall x\neg\forall x\varphi)$
By Lemma 2, this is equivalent to $x \# \pi_{\rightarrow}(\pi_{\forall}(x, \varphi))$. If f, g differ only at x , then $\pi_{\forall}(x, \varphi)(f)$ if $\varphi(f[x \rightarrow m])$ for all m ; but $f[x \rightarrow m] = g[x \rightarrow m]$, so $\pi_{\forall}(x, \varphi)(f) = \pi_{\forall}(x, \varphi)(g)$ and $\pi_{\rightarrow}(\pi_{\forall}(x, \varphi))(f) = \pi_{\rightarrow}(\pi_{\forall}(x, \varphi))(g)$.
- **ax-7:** $\vdash (\forall x\forall y\varphi \rightarrow \forall y\forall x\varphi)$
By Lemma 1, assume $\pi_{\forall}(x, \pi_{\forall}(y, \varphi))(f)$. If $x = y$ then this is the same as $\pi_{\forall}(y, \pi_{\forall}(x, \varphi))(f)$, otherwise it reduces to $\varphi(f[x \rightarrow m][y \rightarrow n])$ for all $m, n \in M$, and note that $f[x \rightarrow m][y \rightarrow n] = f[y \rightarrow n][x \rightarrow m]$.
- **ax-gen:** $\vdash \varphi$ implies $\vdash \forall x\varphi$
If $\varphi(f)$ for all f , then set $f := g[x \rightarrow m]$ to deduce $\varphi(g[x \rightarrow m])$ for all m , hence $\pi_{\forall}(x, \varphi)(g)$.
- **ax-8:** $\vdash (x = y \rightarrow (x = z \rightarrow y = z))$
By Lemma 1, assume $\pi_{=} (x, y)$ and $\pi_{=} (y, z)$. Then $f(x) = f(y) = f(z)$, so $\pi_{=} (x, z)$.
- **ax-9:** $\vdash \neg\forall x\neg x = y$
This is equivalent to $\exists m \in M, \pi_{=} (x, y)(f[x \rightarrow m])$, or $\exists m \in M, m = f[x \rightarrow m](y)$. If $x = y$, then any $m \in M$ will do (M is assumed nonempty because it is a model of ZFC), and if $x \neq y$ then take $m = f(y)$.
- **ax-11:** $\vdash (x = y \rightarrow (\forall y\varphi \rightarrow \forall x(x = y \rightarrow \varphi)))$
Assume $f(x) = f(y)$ and $\forall_M n, \varphi(f[y \rightarrow n])$, take some $m \in M$, and assume

- $f[x \rightarrow m](x) = f[x \rightarrow m](y)$. We want to show $\varphi(f[x \rightarrow m])$. If $x = y$, then the second hypothesis implies $\varphi(f[y \rightarrow m]) = \varphi(f[x \rightarrow m])$. Otherwise, $m = f(y) = f(x)$, so $\varphi(f[x \rightarrow m]) = \varphi(f) = \varphi(f[y \rightarrow m])$.
- **ax-12**: $\vdash (\neg x = y \rightarrow (y = z \rightarrow \forall x y = z))$
Assume $f(x) \neq f(y) = f(z)$, and take $m \in M$. We want to show $f[x \rightarrow m](y) = f[x \rightarrow m](z)$. If $x = y$ or $x = z$ then $f(x) = f(y)$ or $f(x) = f(z)$, a contradiction, so $f[x \rightarrow m](y) = f(y) = f(z) = f[x \rightarrow m](z)$.
 - **ax-13**: $\vdash (x = y \rightarrow (x \in z \rightarrow y \in z))$
Assume $f(x) = f(y)$ and $f(x) \varepsilon f(z)$; then $f(y) \varepsilon f(z)$.
 - **ax-14**: $\vdash (x = y \rightarrow (z \in x \rightarrow z \in y))$
Assume $f(x) = f(y)$ and $f(z) \varepsilon f(x)$; then $f(z) \varepsilon f(y)$.
 - **ax-17**: x, φ distinct implies $\vdash (\varphi \rightarrow \forall x \varphi)$
This is just the forward direction of Lemma 2.

We can also verify the class axioms:

- **df-clab**: The left hand expression $x \in \{y \mid \varphi\}$ expands to $f(x) \in \{m \in M \mid \varphi(f[y \rightarrow m])\}$, that is, $\varphi(f[y \rightarrow f(x)])$, while the right side says $f(y) = f(x) \rightarrow \varphi(f)$ and $\exists_M m, (f[y \rightarrow m](x) = m \wedge \varphi(f[y \rightarrow m]))$. If $x = y$, the left conjunct becomes $\varphi(f)$ and the right becomes $\exists_M m, \varphi(f[x \rightarrow m])$, which is provable from the other conjunct by setting $m = f(x)$ so that $f[x \rightarrow m] = f$. At the same time the left expression also reduces to $\varphi(f[x \rightarrow f(x)]) = \varphi(f)$. If $x \neq y$, then $f[y \rightarrow m](x) = f(x)$ and the right conjunct becomes $\varphi(f[y \rightarrow f(x)])$, and the left conjunct is provable from this since $f(y) = f(x)$ implies $f[y \rightarrow f(x)] = f[y \rightarrow f(y)] = f$, so both sides are equivalent to $\varphi(f[y \rightarrow f(x)])$.
- **df-clel**: We want to show that $\pi_{\in}(A, B)(f)$ iff there is an m such that $\{n \mid n \varepsilon m\} = A(f[x \rightarrow m])$ and $m \in B(f[x \rightarrow m])$, which matches our definition after replacing $A(f[x \rightarrow m]) = A(f)$ and $B(f[x \rightarrow m]) = B(f)$, since $x \# A$ and $x \# B$.
- **df-clleq**: (This has an extra hypothesis **ax-ext** which is already built into our model.) We want to show that $\pi_{=}(A, B)(f)$ iff for all $m, m \in A(f[x \rightarrow m]) \leftrightarrow m \in B(f[x \rightarrow m])$. We are also assuming $x \# A$ and $x \# B$ in this axiom, so this reduces to $m \in A(f) \leftrightarrow m \in B(f)$, or (using extensionality in the metalanguage) $A(f) = B(f)$, which is the definition of $\pi_{=}(A, B)(f)$.

The “true” axioms of set theory are all phrased in terms of only $=, \varepsilon$, and so factor straight through to axioms in $\langle M, \varepsilon \rangle$:

- **ax-ext** (Axiom of Extensionality): The original expression is

$$\pi_{\rightarrow}(\pi_{\forall}(z, \pi_{\mathbf{wb}}(\pi_{\in}(z, x), \pi_{\in}(z, y))), \pi_{=}(x, y))(f),$$

which simplifies, according to the definitions, to $\forall_M m (m \varepsilon f(x) \leftrightarrow m \varepsilon f(y)) \rightarrow f(x) = f(y)$, for all f . With a change of variables this is equivalent to

$$\forall_M x \forall_M y \forall_M z (z \varepsilon x \leftrightarrow z \varepsilon y) \rightarrow x = y,$$

exactly the same as the original universally quantified Metamath formula, but with ε in place of \in and \forall_M instead of \forall . Since the other axiom expressions are long and the process is similar, I will only quote the final equivalent form after reduction.

- **ax-pow** (Axiom of Power sets): Equivalent to:

$$\forall_M x \exists_M y \forall_M z (\forall_M w (w \varepsilon z \rightarrow w \varepsilon x) \rightarrow z \varepsilon y)$$

- **ax-un** (Axiom of Union): Equivalent to:

$$\forall_M x \exists_M y \forall_M z (\exists_M w (z \varepsilon w \wedge w \varepsilon x) \rightarrow z \varepsilon y)$$

- **ax-reg** (Axiom of Regularity): Equivalent to:

$$\forall_M x (\exists_M y (y \varepsilon x \rightarrow \exists_M y (y \varepsilon x \wedge \forall_M z (z \varepsilon y \rightarrow \neg z \varepsilon x))))$$

- **ax-inf** (Axiom of Infinity): Equivalent to:

$$\forall_M x \exists_M y (x \varepsilon y \wedge \forall_M z (z \varepsilon y \rightarrow \exists_M w (z \varepsilon w \wedge w \varepsilon y)))$$

- **ax-ac** (Axiom of Choice): Equivalent to:

$$\begin{aligned} \forall_M x \exists_M y \forall_M z \forall_M w ((z \varepsilon w \wedge w \varepsilon x) \rightarrow \\ \exists_M v \forall_M u (\exists_M t (u \varepsilon w \wedge w \varepsilon t \wedge u \varepsilon t \wedge t \varepsilon y) \leftrightarrow u = v)) \end{aligned}$$

- **ax-rep**: This one is more complicated than the others because it contains a wff metavariable. It is equivalent to: for all binary relations $\varphi \subseteq M^2$:

$$\forall_M w \exists_M y \forall_M z (\varphi(w, z) \rightarrow z = y) \rightarrow \exists_M y \forall_M z (z \varepsilon y \leftrightarrow \exists_M w (w \varepsilon x \wedge \varphi(w, z))).$$

To show the Metamath form of the axiom from this one, given $\varphi' \in \mathcal{U}_{\text{wff}}$ and $f : V \rightarrow M$, define $\varphi(w, z) \leftrightarrow \forall_M t, \varphi'(f[w' \rightarrow w][y' \rightarrow t][z' \rightarrow z])$, and apply the stated form of the axiom.

□

Thus if ZFC has a model, so does `set.mm`.

4 Applications of Metamath models

4.1 Independence proofs

We conclude with a few applications of the “model” concept. The primary application of a model is for showing that statements are not provable, because any provable statement must be true in the model. (The converse is not usually true.) This extends to showing that a system is consistent, because any nontrivial model has unprovable statements (and in a logic containing the principle of explosion $\vdash (\varphi \rightarrow (\neg\varphi \rightarrow \psi))$, this implies that there is no provable statement whose negation is also provable). But it can also be applied for independence

proofs, by constructing a (necessarily nonstandard) model of all statements except the target statement.

For an easy example, if we change the definition of our model of propositional calculus so that instead $\pi_{\neg}(\mathbf{T}) = \pi_{\neg}(\mathbf{F}) = \mathbf{T}$, we would have a new model that still satisfies **ax-1**, **ax-2**, and **ax-mp** (because they do not involve \neg), but violates **ax-3**. If we take $\mu_{\varphi} = \mathbf{F}$ and $\mu_{\psi} = \mathbf{T}$, we get

$$\begin{aligned} \eta_{\mu}(\text{wff } ((\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \phi))) &= \pi_{\rightarrow}(\pi_{\rightarrow}(\pi_{\neg}(\mu_{\varphi}), \pi_{\neg}(\mu_{\psi})), \pi_{\rightarrow}(\mu_{\psi}, \mu_{\varphi})) \\ &= \pi_{\rightarrow}(\pi_{\rightarrow}(\mathbf{T}, \mathbf{T}), \pi_{\rightarrow}(\mathbf{T}, \mathbf{F})) \\ &= \pi_{\rightarrow}(\mathbf{T}, \mathbf{F}) = \mathbf{F}, \end{aligned}$$

so $\eta_{\mu}(\vdash ((\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \phi)))$ is not defined. Thus this shows that **ax-3** is not provable from **ax-1**, **ax-2**, **ax-mp** and the syntax axioms alone (although this should not come as a surprise since none of the other axioms use the \neg symbol).

4.2 Gödel's completeness theorem

One important construction that can be done for any arbitrary (string-based) model is to use a formal system as a model of itself. This model will have the property that the theorems (with no hypotheses) are the only statements that are true in the model, leading to an analogue of Gödel's completeness theorem for statements with no hypotheses and all variables distinct. It is also the “original” model of Metamath, from which the terminology “disjoint variable condition” and the “meta” in Metamath are derived.

Given a formal system $\langle \mathbf{CN}, \mathbf{VR}, \text{Type}, \Gamma \rangle$, choose some set \mathbf{VR}' , with types for each variable, such that $\{v \in \mathbf{VR}' \mid \text{Type}(v) = c\}$ is infinite for each $c \in \mathbf{VT}$. (It is possible to use $\mathbf{VR}' = \mathbf{VR}$ here, provided that \mathbf{VR} satisfies this condition, but it is also helpful to distinguish the two “levels” of variable in the construction.) Using $\mathbf{CN}' = \mathbf{CN}$, define \mathbf{EX}' analogously with the new sets. We will call formulas in \mathbf{EX}' the “object level” and those in \mathbf{EX} the “meta level”.

We also define a substitution $\sigma : \mathbf{EX} \rightarrow \mathbf{EX}'$ in the same way as § 2.1.8. Here variables of the meta level are substituted with expressions in the object level.

We can build another formal system at the object level, where $\langle \mathbf{D}', \mathbf{H}', \mathbf{A}' \rangle \in \Gamma'$ if there is some $\langle \mathbf{D}, \mathbf{H}, \mathbf{A} \rangle \in \Gamma$ and a substitution $\sigma : \mathbf{EX} \rightarrow \mathbf{EX}'$ such that $\forall v \in \mathbf{VR}, \sigma(v) \in \mathbf{VR}'$ (σ substitutes variables for variables) and $\sigma(v) \neq \sigma(w)$ for each $\{v, w\} \in \mathbf{D}$, and $\mathbf{D}' = \mathbf{DV}'$ (all variables are distinct), $\mathbf{H}' = \sigma(\mathbf{H})$ and $\mathbf{A}' = \sigma(\mathbf{A})$. This new formal system differs from the original one only in having a different set of variables.

Theorem 5. *Let $A \in \mathbf{U}_c$ if there is some theorem $\langle \mathbf{D}, \emptyset, \mathbf{A} \rangle$ in the object level formal system with $\text{Type}(\mathbf{A}) = c$, define $e \# e'$ when $\mathcal{V}(e) \cap \mathcal{V}(e') = \emptyset$, and let η_{μ} be the unique substitution $\mathbf{EX} \rightarrow \mathbf{EX}'$ satisfying $\eta_{\mu}(\mathbf{VH}_v) = \mu(v)$, restricted to the e such that $\eta_{\mu}(e) \in \mathbf{U}_{\text{Type}(e)}$. Then $\langle \mathbf{U}, \#, \eta \rangle$ is a model for the meta level formal system $\langle \mathbf{CN}, \mathbf{VR}, \text{Type}, \Gamma \rangle$.*

Proof.

- The type correctness and variable application laws are true by definition, and substitution and dependence on present variables are a consequence of properties of substitutions.
- The relation $\#$ is a freshness relation because the finite set $\bigcup_{e \in W} \mathcal{V}(e)$ misses some variable in each type.
- The freshness substitution rule says that if $\mathcal{V}(w) \cap \mathcal{V}(\mu(w)) = \emptyset$ for all $w \in \mathcal{V}(e)$, then $\mathcal{V}(w) \cap \mathcal{V}(\eta_\mu(e)) = \emptyset$, which follows from $\mathcal{V}(\eta_\mu(e)) \subseteq \bigcup_{w \in \mathcal{V}(e)} \mathcal{V}(\mu(w))$ which is a basic property of variables in a substitution.
- The axiom application law translates directly to the induction step of closure in § 2.1.13 for the object level formal system.

□

Corollary 1 (Gödel’s completeness theorem). *A statement $\langle DV, \emptyset, A \rangle$ of a formal system is a theorem iff it is true in every model.*

Proof. The forward direction is trivial by the definition of a model. For the converse, a statement true in the model of Theorem 5, with $VR' \supseteq VR$ extended to contain infinitely many variables of each type, is derivable by definition. □

Acknowledgments. The author wishes to thank Norman Megill and the anonymous reviewers for pointing out some minor and major omissions in early drafts of this work.

References

1. Megill, N.: Metamath: A Computer Language for Pure Mathematics. Lulu Publishing, Morrisville, North Carolina (2007), <http://us.metamath.org/downloads/metamath.pdf>
2. Tarski, A.: “A Simplified Formalization of Predicate Logic with Identity,” *Archiv für Mathematische Logik und Grundlagenforschung*, 7:61-79 (1965).
3. Hofstadter, D.: *Gödel, Escher, Bach*. Basic Books, Inc., New York (1979).

A first step towards automated conjecture-making in higher arithmetic geometry

Andreas Holmstrom

Uppsala University, Sweden
andreas.holmstrom@gmail.com
<http://andreasholmstrom.org/>

Abstract. We present a framework for encoding information about objects from higher arithmetic geometry. This framework is built around a new kind of data type called a Tannakian symbol. The arithmetic objects we have in mind include modular forms (and more general automorphic representations), elliptic curves (and more general schemes, motives and algebraic stacks), finite graphs, group representations, and multiplicative functions (like the Euler totient function). The language of Tannakian symbols not only allows for representations of individual objects, but also representations of classes of objects, relations between objects, and various important unary and binary operations on objects. The development of this framework is the first small step in a long-term project aiming to apply machine-learning algorithms to some problems of current interest in modern arithmetic geometry.

Keywords: Tannakian categories, arithmetic geometry, zeta functions, motives, modular forms, lambda-rings, automated conjecture-making

1 Introduction

Arithmetic geometry is one of the most vibrant and abstract areas of modern pure mathematics. Out of the seven Millennium Problems, four come from pure mathematics, and of these four, one is solved and the remaining three belong to arithmetic geometry.

The prospect of artificially intelligent programs making new and deep discoveries in this area of mathematics is a tantalizing one. However, most of the concepts encountered in modern arithmetic geometry are not easily stored or manipulated by a computer. In the very long term, one may hope that advances in mathematical linguistics, as developed in Ganesalingam's thesis [9], may lead to new computer-generated discoveries and proofs in arithmetic geometry. In the short term however, it is natural to look for other, less ambitious routes to making partial progress on selected problems.

In this project paper, we present a framework for encoding data about objects from arithmetic geometry, with the aim of laying the foundation for future applications of machine-learning techniques in the field. We emphasize that this

is a first brief survey of a long-term project, focussing on examples. A more detailed discussion of potential applications will be provided in future publications and in discussions at the CICM conference, but as a first application, we propose experiments with automated conjecture-making on invariants of motives and schemes, using the language of Tannakian symbols presented here, together with the SAGE package for domain-independent automated conjecture-making developed recently by Larson and Van Cleemput [14].

1.1 Arithmetic objects

Somewhat informally, we shall use the term “arithmetic object” to refer to any kind of object that is of central importance in arithmetic geometry. Some classes of such objects are: (1) Geometric objects (e.g. a scheme). The reader unfamiliar with the theory of schemes may think of a scheme simply as a system of polynomial equations. (2) Algebraic objects (a group, a ring, a Hopf algebra, etc.). (3) Homotopical objects (like an algebraic stack or a ring spectrum). (4) Combinatorial objects (for example a graph). (5) Analytic objects (e.g. a zeta function). (6) Objects in a Tannakian category. Examples of the latter include representations of finite groups, representations of Lie groups, Galois representations, automorphic representations, motives, Hodge structures, and F-isocrystals.

Technical definitions of all the above terms (schemes, Tannakian categories, etc) can be found in the online Encyclopedia of Mathematics [7]. Rather than giving all of these definitions here, we shall present explicit examples from most of these classes and explain how our proposed encoding framework applies to each example.

In addition to seeking encodings of single objects, a central goal of our work is to also encode information about *classes* of objects (for example the class of objects in some given Tannakian category), *operations* on objects (such as tensor product of group representations, or Tate twist of motives, or Dirichlet convolution of multiplicative functions), *relations* between objects (such as a representation being a direct summand of another), and *invariants* of objects (like the Euler characteristic of a scheme).

1.2 What would be required of a good encoding framework?

Let \mathcal{C} be some class of arithmetic objects, for examples the class of all elliptic curves over the rational numbers, or the class of all finite undirected graphs, or the class of all complex representations of the Monster group.

We seek an encoding framework for objects in \mathcal{C} satisfying the following properties:

1. To every object X in the class \mathcal{C} we can assign a finite amount of structured data $\mathbf{E}(X)$. (We think of $\mathbf{E}(X)$ as an *elementary* or *electronic* “shadow” of the object X .)
2. Given a description of X , there should be an explicit algorithm computing $\mathbf{E}(X)$.

3. Many important invariants of X should be computable from $\mathbf{E}(X)$ only.
4. Many important operations on objects in \mathcal{C} should correspond to explicit manipulations of the corresponding structured data.
5. Given two objects X and X' from different classes (say one graph and one elliptic curve), the two associated pieces of data $\mathbf{E}(X)$ and $\mathbf{E}(X')$ should "be of a similar form" (to facilitate the discovery of connections between different kinds of structures).
6. Many of the deepest theorems and conjectures about objects X in modern arithmetic geometry should have a formulation in terms of the associated data $\mathbf{E}(X)$ only.

Any encoding satisfying these requirements will have the property that a computer could in principle discover (or guess) interesting mathematical statements by searching for patterns in the structured data of many arithmetic objects.

We have found an approach that satisfies all of the above criteria for many classes of arithmetic objects. The framework is built around the notion of a *Tannakian symbol*. In many cases, the information contained in the Tannakian symbol is the same as the information contained in the "zeta function" of the arithmetic object, but Tannakian symbols are more flexible than zeta functions, and there are also cases where it makes sense to speak of Tannakian symbols even though there are no zeta functions around.

1.3 Previous work

We are not aware of any previous work with the explicit ambition of applying machine-learning algorithms to geometric, Tannakian and homotopical categories in higher arithmetic geometry. However, we have drawn inspiration from many places. Due to algorithmic breakthroughs over the past decade by Kedlaya [13], Harvey [11], [12], Costa and Tschinkel [5] and others, it is now possible to compute zeta functions of schemes in much higher dimensions and higher cohomological complexity than before, and these computations generate huge amounts of data, that can be interpreted in the language of Tannakian symbols. A project with the aim of collecting this kind of data has been launched under the name *the L-functions and Modular Forms Database* (LMFDB) [15]. From another direction, we have been inspired by the now classical work of Zeilberger on holonomic sequences [20], the PhD thesis and articles of Colton [2], [3], [4] on automated conjecture-making in number theory, and of course the Online Encyclopedia of Integer Sequences (OEIS) [17]. For a more comprehensive overview of previous work on automated conjecture-making, we refer to Larson and Van Cleemput [14].

2 Summary of algebraic theory

The aim of this section is to define what Tannakian symbols are, and to summarize their most important algebraic properties. Proofs of these statements will be given elsewhere.

2.1 Algebraic structures

We begin by recalling some definitions from abstract algebra. A *monoid* is a set equipped with a binary operation that is associative and has an identity element. A *group* is a monoid in which each element has an inverse. A monoid is called *commutative* if its binary operation is commutative. An *abelian* group is the same thing as a commutative group.

Example 1. The set of positive integers \mathbb{N} is a monoid under addition, and it is also a monoid under multiplication. The set of all complex roots of unity is a monoid under multiplication.

A *commutative ring* is a set R with two binary operations, called addition (+) and multiplication (\cdot), with the requirements that R is an abelian group under addition, a commutative monoid under multiplication, and multiplication distributes over addition. The identity element for addition is denoted by 0, and the identity element for multiplication is denoted by 1.

Example 2. The set of integers \mathbb{Z} is a commutative ring. The set \mathbb{Z}/m , identified with $\{0, 1, \dots, m-1\}$ is a commutative ring for any integer $m \geq 2$, in which addition and multiplication are carried out modulo m . Whenever R is a commutative ring, the set $R[x]$ of polynomials in x with coefficients in R is also a commutative ring.

A *field* is a commutative ring in which the nonzero elements under multiplication form a group (and not just a monoid).

Example 3. The set \mathbb{Q} of rational numbers is a field, and so is the set \mathbb{R} of real numbers, and the set \mathbb{C} of complex numbers.

A *monoid homomorphism* from one monoid to another monoid is a function which commutes with the binary operation and sends the identity element to the identity element. A *ring homomorphism* from a ring to another ring is a function which is a monoid homomorphism both with respect to addition and with respect to multiplication. An *isomorphism* (of rings or of monoids) is a homomorphism which admits a two-sided inverse.

Example 4. Let q be a positive integer. It is known that there exists a field with exactly q elements if and only if q is a power of a prime number (i.e. $q = p^e$ for some prime p and some positive integer e). Two such finite fields with the same number of elements are always isomorphic (i.e. there exists a ring isomorphism between them), and we write \mathbb{F}_q for any finite field with exactly q elements.

It is possible to describe all finite fields in a very concrete way. First of all, when q is a prime number, the ring \mathbb{Z}/q is a field with q elements. A more interesting example is the field with four elements \mathbb{F}_4 , which can be described as the set $\{0, 1, \alpha, \alpha + 1\}$ where addition is carried out modulo 2, and multiplication is carried out modulo 2 and modulo the relation $\alpha^2 = \alpha + 1$. Similar models of finite fields exist (but are not in general unique) for any prime power q .

A *lambda-ring* is, informally, a commutative ring R “equipped with all possible symmetric operations”. The precise definition of “all possible symmetric operations” is expressed in the notion of a *lambda-structure* on a commutative ring. The general definition of “lambda-structure” is given in terms of an infinite sequence $\lambda^0, \lambda^1, \lambda^2, \dots$ of functions (not ring homomorphisms!) from R to R , satisfying axioms that are a bit complicated. However, when the ring R is torsion-free (meaning that finite sums $x + x + \dots + x$ are never zero unless x itself is zero), there is a simpler equivalent definition which we give here. All lambda-rings in this paper will be torsion-free, so this definition is enough for our purposes.

Definition 1. *Let R be a torsion-free commutative ring. A lambda-structure on R is an infinite sequence of ring homomorphisms ψ^1, ψ^2, \dots from R to R satisfying the following axioms:*

1. $\psi^1(x) = x$ for all $x \in R$.
2. $\psi^m(\psi^n(x)) = \psi^{mn}(x)$ for all m, n and all $x \in R$.
3. $\psi^p(x) \equiv x^p \pmod{pR}$ for all prime numbers p and all $x \in R$.

The last condition means that the difference $\psi^p(x) - x^p$ can be written as a multiple of p , in the ring R . The homomorphisms ψ^m are called *Adams operations*.

2.2 Tannakian symbols

The kind of “structured data” we shall construct (denoted by $\mathbf{E}(X)$ in the introduction) will be called a *U -indexed M -valued Tannakian symbol*, and we now turn to the explanation of what this means.

Recall that a *multiset* is a unordered collection of elements, in which elements are allowed to be equal. For example, $\{2, 2, 2, 5\}$ is a multiset with four elements taken from the set of integers.

Definition 2. *Let M be a monoid. An M -valued Tannakian symbol is an ordered pair (A, B) of disjoint finite multisets with elements taken from M . We write $\mathbf{TS}(M)$ for the set of all M -valued Tannakian symbols.*

Conventions: We shall use the notation A/B or $\frac{A}{B}$ for the ordered pair (A, B) , and will refer to A as the *upstairs* multiset and to B as the *downstairs* multiset. Also, if M happens to be a ring and we write $\mathbf{TS}(M)$, we always think of M as a *multiplicative* monoid (in other words, we forget the additive structure).

Example 5. The symbol $\{1, 1, i, -1, -i\}/\emptyset$ is an example of a \mathbb{C} -valued Tannakian symbol. Here i is a complex square root of -1 and \emptyset is the empty multiset.

Definition 3. *Let U be a set. A U -indexed M -valued Tannakian symbol is a function from U to $\mathbf{TS}(M)$. The set of U -indexed M -valued Tannakian symbols will be denoted by $\mathbf{TS}_U(M)$.*

Example 6. Let \mathbb{P} be the set of prime numbers, and let p denote a variable element of \mathbb{P} . Then $\{p^2, 1\}/\{p, p\}$ is a \mathbb{P} -indexed \mathbb{N} -valued Tannakian symbol.

Consider multisets $A = \{a_1, a_2, \dots\}$, $B = \{b_1, b_2, \dots\}$, $C = \{c_1, c_2, \dots\}$ and $D = \{d_1, d_2, \dots\}$ where all the elements are taken from the same monoid M . We define operations on Tannakian symbols by the following formulas:

$$\text{Addition: } \frac{A}{B} \oplus \frac{C}{D} = \frac{A \uplus C}{B \uplus D}$$

(Here \uplus denotes disjoint union of multisets.)

$$\text{Multiplication: } \frac{A}{B} \otimes \frac{C}{D} = \frac{A \cdot C \uplus B \cdot D}{A \cdot D \uplus B \cdot C}$$

(Here $A \cdot C$ denotes the monoid product of A and C , i.e. the multiset of all possible products $a \cdot c$ with $a \in A$ and $c \in C$, listed with repetition.)

$$\text{Adams operations: } \psi^n \left(\frac{A}{B} \right) = \frac{\{a^n \mid a \in A\}}{\{b^n \mid b \in B\}}$$

(Here, if the element a is repeated several times in A , the element a^n is also repeated the same number of times on the right hand side.) In each of these operations, it is understood that if the operation results in a symbol in which the upstairs and the downstairs multisets are not disjoint, then we remove pairs of identical elements until the multisets are disjoint. A few examples will illustrate what this means.

Example 7. Computations in $\mathbf{TS}(\mathbb{Z})$:

$$\begin{aligned} \frac{\{5\}}{\{1, -1\}} \oplus \frac{\{1, 1, 1\}}{\{-1\}} &= \frac{\{5, \cancel{1}, 1, 1\}}{\{\cancel{1}, -1, -1\}} = \frac{\{5, 1, 1\}}{\{-1, -1\}} \\ \frac{\{5\}}{\{1, -1\}} \otimes \frac{\{10\}}{\{3, 7\}} &= \frac{\{50, 3, 7, -3, -7\}}{\{15, 35, 10, -10\}} \\ \psi^2 \left(\frac{\{-1, -1, 2, 5\}}{\{1, -2, 7\}} \right) &= \frac{\{\cancel{(-1)^2}, (-1)^2, \cancel{2^2}, 5^2\}}{\{\cancel{1^2}, \cancel{(-2)^2}, 7^2\}} = \frac{\{1, 25\}}{\{49\}} \end{aligned}$$

The main theorem about Tannakian symbols is the following:

Theorem 1. *For any monoid M , the set $\mathbf{TS}(M)$ is a lambda-ring under the operations \oplus , \otimes and ψ^n . The same is true for $\mathbf{TS}_U(M)$ for any set U . Furthermore, $\mathbf{TS}_U(M)$ is functorial in M as well as in U .*

One can go on and give explicit definitions of other structural features and invariants of Tannakian symbols, such as exterior powers, symmetric powers, virtual dimension, super-dimension, supertrace and superdeterminant. All this terminology comes from the setting of lambda-rings obtained by decategorifying Tannakian categories, but is retained also in situations where there is no Tannakian category involved. *The point of all this structure is that whenever elements of the monoid M can be stored and manipulated by a computer, the same is true for elements of $\mathbf{TS}_U(M)$, and the latter capture huge amounts of structure relevant for higher arithmetic geometry.*

2.3 Assignments and fibers

Now we can be a bit more precise about the picture we would like to paint of structured data assigned to arithmetic objects. Return to the situation where \mathcal{C} is some class of arithmetic objects. Given such a class, we can in many cases choose a monoid M and a set U and construct a map

$$\mathbf{E} : \mathcal{C} \rightarrow \mathbf{TS}_U(M)$$

which satisfies most of the requirements in the introduction.

In such a setup, we are interested in the following general goals.

1. Understand how much information is lost when we pass from X to $\mathbf{E}(X)$. A way of making this more precise is to define the *fiber* of a Tannakian symbol S as the set of all arithmetic objects X in \mathcal{C} with $\mathbf{E}(X) = S$. In many cases one can either prove that each fiber consists of at most one element, or give a bound on the size of the fiber.
2. Describe the image of \mathbf{E} .
3. Set up a correspondence between operations on arithmetic objects in \mathcal{C} and operations on Tannakian symbols.

2.4 An elementary example: Linearly recursive sequences

Let a_0, a_1, a_2, \dots be a linearly recursive sequence in \mathbb{C} , with $a_0 = 1$. It is well-known that it is then possible to rewrite the power series $a_0 + a_1t + a_2t^2 + \dots$ as a rational expression of the form $\prod_{j=1}^n (1 - \beta_j t) / \prod_{i=1}^m (1 - \alpha_i t)$, and we define the Tannakian symbol attached to the linearly recursive sequence to be A/B , with the multisets $A = \{\alpha_i\}$ and $B = \{\beta_j\}$. With this definition, taking the product of power series corresponds to adding Tannakian symbols.

3 Schemes

3.1 General theory

For the purposes of this article, we define an *affine scheme* to be a finite set of variables x_1, x_2, \dots, x_d together with a finite list of polynomial equations (with integer coefficients) in these variables. Given an affine scheme X and a field K , we write $X(K)$ for the set of solutions to the equations of X with values in the field K ; elements of this set are called K -valued points of X .

We also define a *projective scheme* to be a finite set of variables x_0, x_1, \dots, x_d together with a finite list of *homogeneous* polynomial equations (with integer coefficients) in these variables. The equations are not required to be of the same degree. In this setting, we let $X(K)$ denote the set of *equivalence classes* of solutions with values in K , where two solutions are called equivalent if one is a scalar multiple of the other. For projective schemes, we never count the trivial solution in which all variables take the value zero.

As a special case we may take K to be the finite field \mathbb{F}_q , and the set $X(\mathbb{F}_q)$ is then automatically a finite set. We write $\#X(\mathbb{F}_q)$ for the cardinality of this set.

There is a general construction which associates a projective scheme to any affine scheme. If the affine scheme X is defined by equations

$$f_i(x_1, x_2, \dots, x_d) = 0 \quad i = 1, 2, \dots, m$$

then the associated projective scheme is defined by corresponding equations

$$F_i(x_0, x_1, x_2, \dots, x_d) = 0 \quad i = 1, 2, \dots, m$$

where F_i is obtained from f_i by multiplying each term by a suitable power of x_0 so that F_i becomes homogenous of degree $\deg(f_i)$.

Example 8. The equation $x^2 + 1 = 0$ defines an affine scheme X (in one variable). It is easy to see that in this case, we have $X(\mathbb{Q}) = X(\mathbb{R}) = \emptyset$ (the empty set), but $X(\mathbb{C}) = \{i, -i\}$. Using modular arithmetic, we compute $X(\mathbb{F}_2) = X(\mathbb{F}_3) = \emptyset$ and $X(\mathbb{F}_5) = \{2, 3\}$. In general, for an odd prime p , the cardinality of $X(\mathbb{F}_p)$ is 2 or 0 depending on whether p is congruent to 1 or 3 modulo 4. This pattern is a special case of Gauss' famous quadratic reciprocity law, and quadratic reciprocity is an example of a pattern that can be expressed purely in terms of Tannakian symbols.

Theorem 2 (Dwork). *Let X be a scheme (affine or projective) and let p be a prime. There exists unique multisets $A = \{\alpha_1, \alpha_2, \dots, \alpha_m\}$ and $B = \{\beta_1, \dots, \beta_n\}$ of complex numbers such that for all $k \geq 1$, we have*

$$\#X(\mathbb{F}_{p^k}) = \beta_1^k + \beta_2^k + \dots + \beta_n^k - \alpha_1^k - \alpha_2^k - \dots - \alpha_m^k$$

Definition 4. *Let X be a scheme and let p be a prime. We define the Tannakian symbol of X at p to be A/B , where A and B are the multisets in Dwork's theorem.*

Example 9. Since Wiles proved Fermat's Last Theorem using the Modularity theorem for elliptic curves, the class of elliptic curves has probably become the most famous class of schemes. As a simple example of an elliptic curve, take the scheme X defined by the equation $y^2 + y = x^3 - x^2$. At the prime $p = 2$, the symbol becomes:

$$\text{Affine case: } \frac{\{-1+i, -1-i\}}{\{2\}} \quad \text{Projective case: } \frac{\{-1+i, -1-i\}}{\{1, 2\}}$$

The projective case is often the most interesting. In this example, deleting all numbers in the symbol except those with absolute value $\sqrt{2}$ corresponds to cutting out the "motive" $h^1(X)$ from X . One can also associate Tannakian symbols to modular forms, and the Modularity theorem can be formulated as saying that for every elliptic curve X , there exists a modular form which at all primes has the same Tannakian symbol as the motive $h^1(X)$.

Combining Tannakian symbols from all primes gives rise to a map from the class of elliptic curves to $\mathbf{TS}_{\mathbb{P}}(\mathbb{C})$. The fibers of this assignment are called *isogeny classes* of elliptic curves; it is known that these fibers are finite. Furthermore, elliptic curves come with a natural complexity measure N called the conductor (the above example has conductor 11), and by restricting attention to elliptic curves of, say, conductor less than 100000, we may restrict the set of indexing primes to a finite set without losing any information.

In general, the symbol attached to a projective scheme without singularities yields easy recipes for computing the Betti numbers and Euler characteristic of a scheme. In the above example the Betti numbers are 1, 2 and 1; these numbers are obtained by counting symbol elements with absolute value 1, $\sqrt{2}$, and 2, respectively. The Euler characteristic is computed by subtracting the number of elements upstairs from the number of elements downstairs; for this elliptic curve we get $2 - 2 = 0$. Plotting the numbers appearing in the symbol as points in the complex plane reveals symmetries related to Poincaré duality and patterns related to the Riemann hypothesis over finite fields (proved by Deligne, Fields medal 1978). All of this was originally formulated as the famous Weil conjectures in the 1950s.

After computing the Tannakian symbols for several primes (up to size \sqrt{N} approximately) one can easily compute what's called values of L-functions - these are the values appearing in the two Millennium Problems called the (global) Riemann hypothesis and the Birch and Swinnerton-Dyer conjecture. These Tannakian symbols also allows for explicit formulations of many other deep questions of current interest to arithmetic geometers, such as the Sato-Tate conjecture, and various conjectures on Galois representations. The operations on Tannakian symbols correspond to operations in the so-called Grothendieck ring of motives, which is of interest not only in arithmetic geometry, but also in physics, where they are directly related to Feynman integral calculations in perturbative quantum field theory [16].

3.2 Case study: Arithmetic mirror symmetry

Let X_{κ} be the "quartic Dwork family", i.e. the projective scheme defined by the equation

$$x^4 + y^4 + z^4 + w^4 = 4\kappa xyzw$$

where κ is a integer-valued parameter (so that by varying κ we get a *family* of schemes). The scheme X_{κ} comes with a natural action of the group $\mathbb{Z}/4 \times \mathbb{Z}/4$. Taking the quotient scheme by this group action and resolving singularities yields a new scheme Y_{κ} , called the mirror of X_{κ} .

For concreteness, let's look at the prime $p = 41$. For $\kappa = 2$, we get¹ the following symbols:

$$\mathbf{E}(X_2) = \{1, 41, 41, 41, 41, -41, -41, \dots, -41, \frac{25 - 8\sqrt{66}i}{2}, \frac{25 + 8\sqrt{66}i}{2}, 1681\}/\emptyset$$

¹ The examples here are adapted from the presentation of Ursula Whitcher [19], and were computed using computer code by Edgar Costa [5].

Here there are 4 copies of the number 41 and 16 copies of the number -41. For the mirror variety, which *a priori* might be expected to have a completely different symbol, we get

$$\mathbf{E}(Y_2) = \{1, 41, 41, \dots, 41, -41, \frac{25 - 8\sqrt{66}i}{2}, \frac{25 + 8\sqrt{66}i}{2}, 1681\}/\emptyset$$

with 19 copies of the number 41, a single copy of the number -41, and an otherwise identical symbol!

Still working with $p = 41$, for the case $\kappa = 3$ we get:

$$\mathbf{E}(X_3) = \{1, 41, 41, \dots, 41, -39 + 4\sqrt{10}i, -39 - 4\sqrt{10}i, 1681\}/\emptyset$$

with 20 copies of the number 41. And this time, the Tannakian symbol for the mirror variety Y_3 is

$$\mathbf{E}(Y_3) = \{1, 41, 41, \dots, 41, -39 + 4\sqrt{10}i, -39 - 4\sqrt{10}i, 1681\}/\emptyset$$

with 20 copies of 41, which means... that the symbols are absolutely identical!!

Patterns of this kind is the subject of arithmetic mirror symmetry, a relatively recent field inspired by the physics of string theory. It is conceivable that a computer searching for patterns in Tannakian symbols could have identified the schemes X_κ and Y_κ as "similar", even if no human had ever thought of mirror symmetry.

4 More examples

4.1 Multiplicative functions

Much of elementary number theory (questions about primes, divisibility, etc.), can be formulated in terms of *multiplicative functions* from \mathbb{N} to \mathbb{C} . In this context a function f is multiplicative if $f(1) = 1$ and $f(mn) = f(m)f(n)$ whenever m and n are coprime.

Let p be a prime. For all multiplicative functions appearing naturally in number theory, it turns out that the sequence of function values

$$f(1), f(p), f(p^2), f(p^3), \dots$$

is linearly recursive, and hence we can associate a Tannakian symbol to the pair (f, p) . Letting p vary over the set \mathbb{P} of all prime numbers, we get a \mathbb{P} -indexed \mathbb{C} -valued Tannakian symbol attached to the multiplicative function f . For example, the Euler totient function has symbol $\{p\}/\{1\}$, the characteristic function of the square numbers has symbol $\{1, -1\}/\emptyset$, and the sum-of-divisors function has symbol $\{1, p\}/\emptyset$.

This assignment is injective on multiplicative functions, and for many classical classes of functions it stays injective even when U is reduced to a finite set of primes. Furthermore, Dirichlet convolution of functions correspond to addition of symbols, product of function corresponds to product of symbols (under a certain hypothesis), and norm operators on multiplicative functions correspond to certain Adams operations.

4.2 Graphs

There are at least three interesting ways of associating a Tannakian symbol to a (finite) graph. Firstly, given a graph X , we could define the Tannakian symbol of X to be A/\emptyset , where A is the spectrum of X , i.e. the multiset of eigenvalues of the adjacency matrix of X . With this definition, taking the disjoint union of graphs would correspond to addition of Tannakian symbols, and taking tensor product of graphs would correspond to multiplication of Tannakian symbols. Graphs with the same spectrum are called *isospectral*, so the fibers of this assignment would be classes of isospectral graphs.

Example 10. With this definition, the Tannakian symbol of the complete graph on 4 vertices would be $\{-1, -1, -1, 3\}/\emptyset$

It is conjectured that almost all graphs are determined by their spectra. However, there are many cases of non-isomorphic graphs with identical spectrum. For example, the number of simple graphs on 9 vertices is 274668 (see OEIS: Sequence A000088), while the number of such graphs isospectral to at least one other graph is 51039 (OEIS: Sequence A099883).

A second approach would be to define the Tannakian symbol of a graph X to be A/\emptyset , where $A = \{\alpha_1, \alpha_2, \dots, \alpha_m\}$ is the finite multiset of complex numbers appearing in the expression

$$\zeta_X(T) = \frac{1}{(1 - \alpha_1 T)(1 - \alpha_2 T) \cdots (1 - \alpha_m T)}$$

where $\zeta_X(T)$ is the Ihara zeta function of the graph X .

Example 11. With this alternative definition, the Tannakian symbol of the complete graph on 4 vertices would be

$$\{-1, -1, 1, 1, 1, 2, \frac{-1+\sqrt{7}i}{2}, \frac{-1+\sqrt{7}i}{2}, \frac{-1+\sqrt{7}i}{2}, \frac{-1-\sqrt{7}i}{2}, \frac{-1-\sqrt{7}i}{2}, \frac{-1-\sqrt{7}i}{2}\}/\emptyset$$

In a recent paper [6], Durfee and Martin conjecture that almost all graphs which are not determined by their spectrum are determined by their zeta function.

As a third possibility, one can associate a certain polynomial (the “graph polynomial”) to any graph X . This polynomial defines a scheme, called the graph hypersurface of X , and we could associate Tannakian symbols to the graph X by counting points of its graph hypersurface, like we did in the previous section for an elliptic curve and the quartic Dwork family. We refer to Brown and Schnetz [1] for background, definitions, and extensive calculations motivated by applications to quantum field theory. One of their conclusions can be reformulated by saying that for many graphs, the Tannakian symbols constructed by this method seem to also come from modular forms.

4.3 Representations of finite groups

Let G be a finite group. Associated to G is its representation ring $R(G)$, a lambda-ring generated by irreducible complex representations under the operations of direct sum, tensor product, and exterior powers. Any element $g \in G$

gives rise to a lambda-ring homomorphism from $R(G)$ to $\mathbf{TS}(M)$, where M is the monoid of complex roots of unity. Combining several such maps, one obtains a lambda-ring homomorphism \mathbf{E} from $R(G)$ into $\mathbf{TS}_U(M)$, where U is a subset of G . The most interesting choice of U , which we will use in the remainder of this section, is to pick one representative of each conjugacy class of G ; this choice guarantees the injectivity of \mathbf{E} .

Many interesting patterns and unsolved problems about representations can be reformulated in terms of the map $\mathbf{E} : R(G) \rightarrow \mathbf{TS}_U(M)$. This is due to the fact that both the character table of G and the lambda-ring structure of $R(G)$ can be recovered from the values of \mathbf{E} .

Example 12. Taking G to be the Monster group, the character table is a 194 by 194 matrix, whose rank is 163. The number 163 also appears in the study of imaginary quadratic number fields; it is in fact the largest possible integer D such that the number field $\mathbb{Q}(\sqrt{-D})$ has class number 1 (meaning that its ring of integers is a unique factorization domain). The study of such number fields goes back to Gauss and is the simplest instance of Gauss' famous class number problem. As explained for example in the popular book of Mark Ronan [18], the appearance of the number 163 in both places might well be a coincidence, but it could also be a hint that there is some mysterious connection between the Monster group and algebraic number theory that is yet to be understood.

An even more spectacular pattern connected with the Monster group is the Monstrous Moonshine Conjecture, formulated by Conway and Norton and proved by Richard Borcherds (Fields medal 1998). The starting point of this wonderful story was the observation that a certain number obtained from the character table (the dimension of the smallest nontrivial irreducible representation) is (almost) equal to the coefficient of the linear term in the Fourier expansion of Klein's j -function.

Let's now turn to a much simpler group, for which everything can be worked out by hand.

Example 13. One of the simplest non-trivial examples of a finite group is the symmetric group S_3 of permutations on three objects. This group has 6 elements in total, partitioned into 3 conjugacy classes. Let e be the identity element, let t be any transposition (an element of order 2), and let r be one of the two "rotations" (an element of order 3). These three elements represent the three conjugacy classes of S_3 . The number of irreducible representations of a finite group is the same as the number of conjugacy classes, and in our example, the irreducible representations are:

- \mathbb{C}_+ : The trivial representation, sending every permutation to 1.
- \mathbb{C}_- : The sign representation, sending a permutation to its sign.
- \mathbb{C}^2 : A two-dimensional representation, visualized as a matrix action of S_3 on a triangle with vertices at the three cube roots of unity in the complex plane.

In this simple case, it is easy to compute the function \mathbf{E} by hand. We get, for the three different choices of group element g :

$$\begin{array}{lll} \text{Case } g = e : & \mathbf{E}(\mathbb{C}_+) = \{1\}/\emptyset & \mathbf{E}(\mathbb{C}_-) = \{1\}/\emptyset & \mathbf{E}(\mathbb{C}^2) = \{1, 1\}/\emptyset \\ \text{Case } g = t : & \mathbf{E}(\mathbb{C}_+) = \{1\}/\emptyset & \mathbf{E}(\mathbb{C}_-) = \{-1\}/\emptyset & \mathbf{E}(\mathbb{C}^2) = \{1, -1\}/\emptyset \\ \text{Case } g = r : & \mathbf{E}(\mathbb{C}_+) = \{1\}/\emptyset & \mathbf{E}(\mathbb{C}_-) = \{1\}/\emptyset & \mathbf{E}(\mathbb{C}^2) = \{\omega, \omega^2\}/\emptyset \end{array}$$

Here ω is a primitive 3rd root of unity. Any element of $R(G)$ can be written as a formal difference $V - W$, where V and W are representations built as direct sums of irreducible ones, and we may compute in $R(G)$ by identifying such a formal difference with an ordered triple of symbols (using the injective map \mathbf{E} and applying the rules for computing with Tannakian symbols). For example, we get (suppressing \mathbf{E} from the notation): $\mathbb{C}^2 = (\{1, 1\}/\emptyset, \{1, -1\}/\emptyset, \{\omega, \omega^2\}/\emptyset)$ and $\mathbb{C}_+ \oplus \mathbb{C}_- - \mathbb{C}^2 = (\emptyset/\emptyset, \emptyset/\emptyset, \{1, 1\}/\{\omega, \omega^2\})$. A similar computation shows that $\mathbb{C}^2 \otimes \mathbb{C}^2$ equals $\mathbb{C}_+ \oplus \mathbb{C}_- \oplus \mathbb{C}^2$, and in general any tensor product of representations can be expressed as a direct sum of irreducibles, using only Tannakian symbols.

4.4 Algebraic stacks

The arithmetic objects discussed so far in this paper were key players in many of the greatest arithmetic discoveries of the 20th century. However, in 21st century research, new classes of objects are becoming increasingly important, and these objects come from homotopy theory and higher category theory. The most beautiful application so far is probably the recent proof of the Tamagawa number conjecture by Gaitsgory and Lurie [8], in which homotopical objects called *stacks* play a prominent role. Stacks are a generalization of schemes, for which $X(\mathbb{F}_q)$ is no longer a set, but a *groupoid* or a *simplicial set*. There are several different ways of assigning Tannakian symbols to (certain classes of) stacks. The method used for schemes will work provided we accept infinite multisets. For example, the stack $B\mathbb{G}_m$ (the classifying stack of the multiplicative group) would then at the prime p have the Tannakian symbol $\{p^{-1}, p^{-2}, p^{-3}, \dots\}/\emptyset$.

5 Final remarks on arithmetic pattern-detection

What constitutes an important discovery or a deep conjecture in arithmetic geometry? Looking at many examples in the literature, a few of which we have seen in the present survey, it is reasonable to say that *deep arithmetic statements are often observations of patterns that can be expressed in terms of Tannakian symbols*.

But is it conceivable that automated algorithms really could have detected some of these patterns? And is it reasonable to expect machines to discover new patterns, maybe even of comparable interest to the Weil conjectures, Monstrous Moonshine, or arithmetic mirror symmetry? Although we do not claim to know the answer to these questions, we would like to end by suggesting two necessary

features that such algorithms would have to incorporate in order to have any chance of making such discoveries.

5.1 Exotic metrics

In traditional data analysis, data points are given by vectors of real numbers (or rather floating point approximations), visualized as points in n -dimensional Euclidean space \mathbb{R}^n . Two data points are then considered to have similar features if they are close with respect to some metric on \mathbb{R}^n derived from the standard metric $(x, y) \mapsto |x - y|$ on the set of real numbers.

Similarly, in a traditional neural network based on perceptrons or sigmoid neurons, the output of an individual neuron is a function of the size of an incoming real number, and “size” here refers to a measurement made using the standard metric on the real numbers.

In arithmetic geometry, many patterns and relations can be expressed in terms of a metric, *but* it is not enough to work with the standard metric on real numbers. Instead, the standard metric needs to be complemented by others, most importantly the p -adic metrics. For any prime number p , the p -adic metric on the set of rational numbers is defined as follows. For two distinct rational numbers x and y , there is a unique integer k such that $x - y$ can be written on the form $\pm p^k \cdot a/b$, where a and b are positive integers coprime to p . The p -adic distance $|x - y|_p$ between x and y is defined to be p^{-k} . This definition can be extended to irrational algebraic numbers, but unlike the standard metric, it is not defined for transcendental numbers.

Unravelling the definition, it is easy to see that as a special case, two integers x and y are congruent modulo p if and only if they are close in the sense that their p -adic distance is less than or equal to $1/2$. Combining different primes and using the Chinese remainder theorem, any arithmetic pattern involving congruences can be expressed (and hence potentially discovered) using p -adic metrics².

5.2 Symmetry detection

Another class of arithmetic patterns can be collected under the umbrella of symmetry. While the lambda-ring structure on Tannakian symbols in itself captures certain kinds of symmetry, there are also many other kinds, including Poincaré duality (seen in the symbol of a projective scheme without singularities), various symmetries in Hodge diamonds, symmetries of modular forms, and the fact that the number of rows in a character table equals the number of columns.

A mathematical treatment of symmetry invariably involves group theory, and in situations where the symmetry group is both finite and known, it is easy to implement algorithms for detecting symmetry. However, in cases where the symmetry group is infinite and/or unknown, the symmetry detection problem

² It might also be interesting to build algorithms based on other kinds of metrics, like the I -adic metric on a polynomial ring or a Grothendieck ring (where I is an ideal of the ring), or the Granville-Soundararajan metric on multiplicative functions [10].

is more challenging. It would be interesting to explore the image recognition and computer vision literature on symmetry detection and think about whether known algorithms can be applied to arithmetic settings, for example to plots of complex numbers taken from a Tannakian symbol.

References

1. Brown, F., Schnetz, O.: Modular forms in quantum field theory. *Communications in Number Theory and Physics* 7(2), 293–325 (2013)
2. Colton, S.: Automated Theory Formation in Pure Mathematics. PhD. Thesis, Department of Artificial Intelligence, University of Edinburgh (2001)
3. Colton, S.: Automated Conjecture Making in Number Theory using HR, Otter and Maple. *Journal of Symbolic Computation* 39(5), 593–615 (2005)
4. Colton, S.: Refactorable Numbers - A Machine Invention. *Journal of Integer Sequences* 2, 99.1.2 (1999)
5. Costa, E., Tschinkel, Y.: Variation of Neron-Severi ranks of reductions of K3 surfaces. *Experimental Mathematics* 23, pp. 475–481 (2014)
6. Durfee, C., Martin, K.: Distinguishing graphs with zeta functions and generalized spectra. *Linear Algebra Appl.* 481, pp. 54–82 (2015)
7. Encyclopedia of Mathematics, <http://www.encyclopediaofmath.org/>
8. Gaitsgory, D., Lurie, J.: Weil’s Conjecture for Function Fields. Preprint available at <http://www.math.harvard.edu/~lurie/papers/tamagawa.pdf>
9. Ganesalingam, M.: The Language of Mathematics - A Linguistic and Philosophical Investigation. *Theoretical Computer Science and General Issues*, Vol. 7805. Springer-Verlag Berlin Heidelberg (2013)
10. Granville, A., Soundararajan, K.: Pretentious multiplicative functions and an inequality for the zeta-function. In: De Koninck, J., Granville, A., Luca, F. (eds.): *Anatomy of Integers*, CRM Proceedings and Lecture Notes, Vol. 46 (2008)
11. Harvey, D.: Counting points on hyperelliptic curves in average polynomial time. *Ann. of Math. (2)* 179(2), 783–803 (2014)
12. Harvey, D.: Computing zeta functions of arithmetic schemes. *Proc. Lond. Math. Soc.* 111, no. 6, 1379–1401 (2015)
13. Kedlaya, K.S.: Computing Zeta Functions via p-adic Cohomology. In: Buell (ed.); *Algorithmic Number Theory*. LNCS, vol. 3076, pp. 1–17. Springer Berlin Heidelberg (2004)
14. Larson, C. E., Van Cleemput, N.: Automated Conjecturing I: Fajtlowicz’s Dalmatian Heuristic Revisited. *Artificial Intelligence* 231, 17–38 (2016)
15. The LMFDB Collaboration: The L-functions and Modular Forms Database. <http://www.lmfdb.org>
16. Marcolli, M.: *Feynman motives*. World Scientific, Hackensack, NJ (2010)
17. Online Encyclopedia of Integer Sequences, <http://oeis.org/>
18. Ronan, M.: *Symmetry and the Monster: One of the greatest quests of mathematics*. Oxford University Press, UK (2006)
19. Whitcher, U.” Mirror symmetry and K3 surface zeta functions. Presentation given at ICERM, Oct 2015. Slides available at <https://icerm.brown.edu/sp-f15-w2/>
20. Zeilberger, D.: A holonomic systems approach to special functions identities. *J. Comput. Appl. Math.* 32(3), 321–368 (1990)

Initial Experiments with Statistical Conjecturing over Large Formal Corpora

Thibault Gauthier¹, Cezary Kaliszyk¹, and Josef Urban²

¹ University of Innsbruck, Austria

{thibault.gauthier,cezary.kaliszyk}@uibk.ac.at

² Czech Technical University in Prague

josef.urban@gmail.com

Abstract. A critical part of mathematicians’ work is the process of conjecture-making. This involves observing patterns in and between mathematical objects, and transforming such patterns into conjectures that describe or better explain the behavior of the objects. Computer scientists have since long tried to reproduce this process automatically, but most of the methods were typically restricted to specific domains or based on brute-force enumeration methods. In this work we propose and implement methods for generating conjectures by using statistical analogies extracted from large formal libraries, and provide their initial evaluation.

1 Introduction

In the past decade, there has been considerable progress in proving conjectures over large formal corpora such as Flyspeck [7], Isabelle/HOL [12], the Mizar Mathematical Library (MML) [1] and others. This has been achieved by combining high-level learning or heuristic fact-selection mechanisms with a number of methods and strategies for guiding the strongest (typically superposition-based) automated theorem provers (ATPs). While this approach has not reached its limits [2], and its results are already very useful, it still seems quite far from the way humans do mathematics. In particular, even with very precise premise (fact) selection, today’s ATPs have trouble finding many longer Mizar and Flyspeck proofs of the toplevel lemmas in their libraries. In fact, only a few of such lemmas would be called a “theorem” by a mathematician. Often the “theorem” would be just the final proposition in a formal article, and the toplevel lemmas preceding the theorem would be classified as simple technical steps that sometimes are not even mentioned in informal proofs.

An important inductive method that mathematicians use for proving hard problems is *conjecturing*, i.e., proposing plausible lemmas that could be useful for proving a hard problem.³ There are likely a number of complicated inductive-deductive feedback loops involved in this process that will need to be explored, however it seems that a major inductive method is *analogy*. In a very high-level

³ A famous example is the Taniyama-Shimura conjecture whose proof by Wiles finished the proof of Fermat’s Last Theorem.

way this could be stated as: “*Problems and theories with similar properties are likely to have similar proof ideas and lemmas.*”

Again, analogies can likely be very abstract and advanced. Two analogue objects may be related through a morphism. They can also be two instances of the same algebraic structure. The organization of such structures was recently addressed by the MMT framework [13] in order to represent different logics in a single consistent library. In this work we start experimenting with the analogies provided by *statistical concept matching* [4]. This method has been recently developed in order to align formal libraries of different systems and to transfer lemmas between them [5]. Here we use statistical concept matching to find the *most similar sets of concepts* inside one large library. The discovered sets of matching concepts can then be used to translate a hard conjecture C into a “related” conjecture C' , whose (possible) proof might provide a further guidance for proving C . The remaining components that we use for this first experiment are standard large-theory reasoning components, such as fast machine-learners that learn from the thousands of proofs and propose the most plausible lemmas for proving the related conjectures, and first-order ATPs – in this case we use Vampire 4.0 [10].

2 Matching concepts

In order to apply some analogies, we first need to discover what they are by finding similar concepts. For our initial evaluation, our similarity matching will be limited to concepts represented by constants and ground sub-terms. Later this can be extended to more complex term structures. We describe here a general concept matching algorithm inspired and improved upon [4] and discuss how this algorithm can be adapted to find analogies within one library.

2.1 Matching concepts between two libraries

Given two theorem libraries, the first step is to normalize all statements, as this increases the likelihood of finding similar theorem shapes. If two theorems have the same shape (that we will call such abstract shapes a property), then the concrete constants appearing in this two theorems at the same positions are more likely to be similar. We will say that such pairs of constants are derived from the theorem pair.

Example 1. Given the theorems T_1 and T_2 with the statements, their respective normalizations, and the properties extracted from their statements:

$$T_1 : \forall x : num. x = x + 0 \quad T_2 : \forall x : real. x = x \times 1$$

$$P_1 : \lambda num, +, 0. \forall x : num x = x + 0 \quad P_2 : \lambda real, \times, 1. \forall x : real. x = x \times 1$$

The properties P_1 and P_2 are α -equivalent, therefore the theorems T_1 and T_2 form a matching pair of theorems, and the following three matching pairs of constants are derived:

$$num \leftrightarrow real, + \leftrightarrow \times, 0 \leftrightarrow 1$$

We further observe that the matchings $(+, \times)$ and $(0, 1)$ are in relation through the theorem pair (T_1, T_2) . The strength of this relation – *correlation* – is given by the number and accuracy of the theorem pairs these matchings are jointly derived from. We will call the graph representing the correlations between different matchings the *dependency graph*. The *similarity score* of each matching (i.e., pair of concepts) is then initialized with a fixed starting value and computed by a dynamical process that iterates over the dependency graph until a fixpoint is reached.

The principal advantage of this method is that the algorithm is not greedy, allowing a concept to match multiple other concepts in the other libraries. This is a desirable property, since we want to be able to create multiple analogues for one theorem. Matchings are then combined into *substitutions*, which are in turn applied to the existing theorem statements yielding plausible conjectures.

Very few adjustments are needed to adapt this method to a single library. We create a duplicate of the library and match it with its copy. Since we are not interested in identity substitutions, we prevent theorems from matching with their copies. However, we keep self matches between constants in the dependency graph since good analogies can be often found by using partial substitutions.

3 Context-dependent substitutions

Constant matchings by themselves create special one-element substitutions that transport the properties of one constant to another. In general, substitutions are created from translating more than one constant. Suppose, that we know that addition is similar to union and multiplication to intersection. We can hope to transport the distributivity of multiplication over addition to the distributivity of intersection over union. Therefore, the correlations between the matchings are crucial for building the substitutions.

We now present two methods for creating substitutions from a theorem. These methods are based on the correlations between the concept pairs and the similarity score of each concept pair.

We want to construct substitutions that are most relevant in the local context, i.e., for the symbols that are present in the theorem we want to translate (*target theorem*).

The first method starts by reducing the dependency graph to the subgraph of all concept pairs whose first element is contained in the target theorem. We first select a starting node in this subgraph which is not an identity matching, and recursively find nodes (possibly identities) that are connected by the strongest edges of the subgraph, under the constraint that no two selected nodes have the same first element. The algorithm stops when no new node can be added. The final set of nodes obtained in this way forms a partial substitution. We run this algorithm either for all starting nodes, or for those with similarity scores above a certain threshold. This produces a set of substitutions, which are effectively the most relevant substitutions for the target theorem. This process seems to

produce many substitutions, however in practice, many of them are identical, which limits their total number.

The second method is a brute-force approach where we first find the set of concepts ($Matched(T)$) that match at least one concept in the set of concepts ($Concepts(T)$) present in the target theorem T . We would like to create all possible substitutions between $Concepts(T)$ and $Matched(T)$ and rank them, however this would often blow up the generation phase. To limit the number of possible substitutions, we remove possible matches by iterating the following process until the number of substitutions is below a chosen threshold (1000): We select the constant C in T with most remaining matchings, remove its worst match, recompute the number of remaining matches for all constants in T , and check if the number of substitutions is already below the threshold. If so, the process terminates, otherwise we continue the iteration.

Next, we select the 200 substitutions with the highest *combined score*. The combined score of a substitution S is computed by multiplying the average correlation and similarity in S , i.e. formally as follows:

$$CombinedScore(S) = AverageCorrelation(S) \times AverageSimilarity(S)$$

$$AverageCorrelation(S) = \frac{1}{|S|^2} \times \sum_{M \in S, M' \in S} Correlation(M, M')$$

$$AverageSimilarity(S) = \frac{1}{|S|} \times \sum_{M \in S} SimilarityScore(M)$$

On top of these combined scores, the diversity of substitutions can be maximized by the following shuffling. The process iteratively chooses a (not yet selected) substitution with the best *diversity score* and increases the set of selected substitutions \mathfrak{T} . These scores are then updated to penalize the substitutions that have more matchings in common with the already selected ones.

$$CommonMatchings(S, T) = |S \cap T|$$

$$DiversityScore(S) = \frac{CombinedScore(S)}{(1 + \sum_{T \in \mathfrak{T}} CommonMatchings(S, T))^3}$$

These substitutions will eventually be applied to the initial theorem to create new conjectures. We hope that if such conjectures can be proved, they will improve the AI/ATP methods by enriching the theory. We consider in Section 4 two possible scenarios where the combination of conjecturing by analogies and premise selection could be useful.

4 Scenarios

We will consider two scenarios for the use of conjecturing: without and with a user given goal.

In the first scenario no goal set by the user, and our system should decide what is the next step in the development of the whole library. In this context, our algorithm produces the most likely conjectures by applying the first substitution generation method on all theorems. We then evaluate the conjectures by trying to prove them. We estimate the difficulty of a proved conjecture by the number of lemmas that were used in its proof. We estimate the relevance of a conjecture by how much adding this conjecture as a new theorem in the library helps to automatically prove formalized statements appearing after it in the library.

In the second scenario, a goal is given and the system should figure a way how to prove it. The typical AI-ATP method is to search through all lemmas (in an evaluation this means the lemmas which occur before the goal), and select the most relevant ones using machine learning techniques. If however an important lemma was not proven or proven after the goal, the premise selection fails to produce it. Therefore, we propose a way to produce some of such relevant missing lemmas. This method is depicted in Fig 1. We first select the 20 best scoring substitutions for this goal, which creates 20 new goals. We then try to prove them using the AI-ATP system. If some of them are successfully proved, we obtain small sets of lemmas which were sufficient to prove the translated goals. These lemmas are then translated back to the original concepts. We run our AI-ATP system again and remove those it cannot prove. The final step is to try to prove the user goal from those additional lemmas. This strategy simulates the following thought process a human could have: “Can I prove this conjecture for a similar concept”?, “If so, what where the necessary lemmas?” “If some of this lemmas holds for my original concept, they would likely help me in my original proof”.

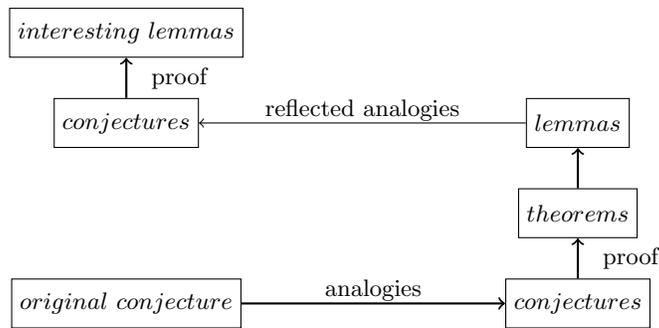


Fig. 1. Creating additional relevant lemmas for a conjecture through analogies.

5 Experiments

5.1 Untargeted Conjecture Generation

In the first scenario, we apply⁴ the 20 “most plausible” substitutions to all MML theorems, and attempt to prove a small part (73535) of those, each time using the whole MML. We want to see how complicated and interesting the conjectures can be. After premise selection Vampire can prove 10% (7346) of them in 10 s, which is reduced to 4464 after pseudo-minimization [8] and removing tautologies and simple consequences of single lemmas. An example of a reasonably interesting conjecture (and analogy) with a new nontrivial proof is the convexity of empty subsets of real linear spaces, induced from a similar claim about them being “circled”.⁵

```
registration
  let X be non empty RLSStruct;
  cluster empty -> circled for Element of bool the carrier of X;
```

Here “circled” is defined as⁶

```
definition
  let X be non empty RLSStruct; let A be Subset of X;
  attr A is circled means :Def6: :: RLTOPSP1:def 6
  for r being Real st abs r <= 1 holds r * A c= A;
```

and “convex” as⁷

```
definition
  let V be non empty RLSStruct; let M be Subset of V;
  attr M is convex means :Def2: :: CONVEX1:def 2
  for u, v being VECTOR of V
  for r being Real st 0 < r & r < 1 & u in M & v in M holds
    (r * u) + ((1 - r) * v) in M;
```

For example the following properties of circled⁸ and convex⁹ subsets are quite similar, leading the conjecturer into conjecturing further properties like the one stated above.

```
registration
  let X be RealLinearSpace;
  let A, B be circled Subset of X;
  cluster A + B -> circled ;
```

⁴ The experimental version of our conjecturer used here is available at:

http://147.32.69.25/~mptp/conj/conjecturing_standalone.tar.gz .

⁵ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rltopsp1.html#CC1

⁶ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rltopsp1.html#V3

⁷ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/convex1.html#V1

⁸ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rltopsp1.html#FC3

⁹ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/convex1.html#T2

```

theorem :: CONVEX1:2
  for V being non empty Abelian add-associative vector-distributive
    scalar-distributive scalar-associative scalar-unital RLSStruct
  for M, N being Subset of V st M is convex & N is convex holds
    M + N is convex

```

5.2 Targeted Conjecture Generation

In the second experiment, we have used as our target the set of 22069 *ATP-hard* Mizar toplevel lemmas (theorems). These are the theorems that could not be proved in any way (neither with human-advised premises, nor with learned premise selection) in our recent extensive experiments with state-of-the-art AI/ATP methods over the MML [9]. For the current experiment, those experiments are very thorough. They used high ATP time limits, many ATPs, their targeted strategies, a number of learning methods and their combinations, and a number of iterations of the learning/proving loop, approaching in total a month of a server time. Proving these problems with a low time limit and a single AI/ATP method is thus quite unlikely.

To each such hard theorem T we apply the 20 best (according to the diversity score) substitutions. Such substitutions are additionally constrained to target only the part of the MML that existed before T was stated. This gives rise to 441242 conjectures, which we attempt to prove – again only with the use of the MML that precedes T . Because of resource limits, we use only one AI/ATP method: k-NN with 128 best premises, followed by Vampire using 8 s. This takes about 14 hours on a 64-CPU server, proving 9747 (i.e. 2.2%) of the conjectures. We do two rounds of pseudo-minimization and remove tautologies and simple consequences of single lemmas. This leaves 3414 proved conjectures, originating from 1650 hard theorems, i.e., each such conjecture C is a translation of some hard theorem T under some plausible substitution σ ($C = \sigma(T)$). We translate the MML lemmas L_C^i needed for the proof of C “back” into the “terminology of T ” by applying to them the reverse substitution σ^{-1} .

This results in 26770 back-translated conjectures. For each of them we hope that (i) it will be provable from the MML preceding T , and (ii) it will be useful for proving its T , since its image under σ was useful for proving $\sigma(T)$. We use again only 8 s to select those that satisfy (i), yielding after the minimization and removal of trivial ones 2170 proved back-translated lemmas, originating from 500 hard theorems. For each of these 500 theorems T we do standard premise selection (using slices of size 128 and 64) over the preceding MML, and add these lemmas (obviously only those that “belong” to T) to the premises of T . Then we run Vampire for 30 s on the standard and lemma-augmented problems. While there is no difference when using 128 lemmas, for 64 lemmas we obtain (in addition to the 6 proofs that both the standard and augmented methods find) an interesting new proof of the theorem `MATHMORP:25`¹⁰, which cannot be found by Vampire using the standard method even with a time limit of 3600 s.

¹⁰ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/mathmorp.html#T25

```

theorem :: MATHMORP:25
for T being non empty right_complementable Abelian
      add-associative right_zeroed RLSstruct
for X, Y, Z being Subset of T
  holds X (+) (Y (-) Z) c= (X (+) Y) (-) Z

```

To find this proof, our concept matcher first used the statistical analogy between addition¹¹ and subtraction¹² in additive structures (`addMagma`). By doing that, it inadvertently constructed as a conjecture the theorem `MATHMORP:26`¹³, that actually immediately succeeds `MATHMORP:25` in MML. This alone is remarkable, because this theorem was not known at the point when `MATHMORP:25` was being proved. Using premise selection and Vampire, `MATHMORP:26` was proved in 4 s, and a particular back-translated lemma from its proof turned out to be provable and crucial for proving `MATHMORP:25` automatically. This lemma is actually “trivial” for Mizar, since it follows by climbing Mizar’s extensive type hierarchy [6] from an existing typing of the ‘‘(-)’’ function. However, as mentioned above, we were not able to get this proof without this lemma even with much higher time limits.

6 Conclusion and Future Work

We have investigated the application of a concept matching algorithm to formulate conjectures through analogies. We have described a way to combine it with premise selection methods and ATPs. This was designed to create potential intermediate lemmas that help an ATP to find complex proofs.

While these are just first experiments, it seems that statistical concept matching can occasionally already come up with plausible conjectures without resorting to the (in large libraries rather impossible) brute-force term enumeration methods. So far we do not even use any of the manually invented heuristic methods such as those pioneered by Lenat [11] and Fajtlowicz [3], and rather rely on a data-driven approach. Such heuristics and other methods could be combined with the statistical ones.

We can likely improve the matching algorithm by allowing the concepts to be represented by more complex term structures [14]. This may help us to connect concepts from more different domains. In the same direction, we could also relax our concept of properties to allow matching with errors. A more generic solution would be to try different shapes of theorems using substitutions trees or genetic programming, but this might need efficient implementation.

We can also modify how the matching algorithm and the AI-ATP system are combined. A simple approach is to enhance the premise selection algorithm of the AI-ATP system with the discovered similarities between concepts. In our experiments we also observe an increasing number of conjectures given by the

¹¹ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rusub_4.html#K6

¹² http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/mathmorp.html#K3

¹³ http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/mathmorp.html#T26

number of possible substitutions. A heuristic semantic evaluation could complement the substitutions scores to estimate the likely of a conjecture to be true.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF) grant P26201 and by the European Research Council (ERC) grant AI4REASON.

References

1. Grzegorz Bancerek, Czeslaw Bylinski, Adam Grabowski, Artur Kornilowicz, Roman Matuszewski, Adam Naumowicz, Karol Pak, and Josef Urban. Mizar: State-of-the-art and beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer, 2015.
2. Jasmin C. Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.
3. Siemion Fajtlowicz. On conjectures of graffiti. *Discrete Mathematics*, 72(1-3):113–118, 1988.
4. Thibault Gauthier and Cezary Kaliszyk. Matching concepts across HOL libraries. In Stephen Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors, *Proc. of the 7th Conference on Intelligent Computer Mathematics (CICM’14)*, volume 8543 of *LNCS*, pages 267–281. Springer Verlag, 2014.
5. Thibault Gauthier and Cezary Kaliszyk. Sharing HOL4 and HOL light proof knowledge. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2015.
6. Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
7. Thomas Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2012.
8. Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.
9. Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reasoning*, 55(3):245–256, 2015.
10. Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Proceedings of the 25th International Conference on Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.
11. Douglas Lenat. *An Artificial Intelligence Approach to Discovery in Mathematics*. PhD thesis, Stanford University, Stanford, USA, 1976.
12. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

13. Florian Rabe. The MMT API: A generic MKM system. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Proc. of the 6th Conference on Intelligent Computer Mathematics (CICM'13)*, volume 7961 of *LNCS*, pages 339–343. Springer, 2013.
14. Jiří Vyskočil, David Stanovský, and Josef Urban. Automated Proof Compression by Invention of New Definitions. In Edmund M. Clarke and Andrei Voronkov, editors, *LPAR (Dakar)*, volume 6355 of *LNCS*, pages 447–462. Springer, 2010.

A Standard for Aligning Mathematical Concepts

Cezary Kaliszyk¹, Michael Kohlhase², Dennis Müller², Florian Rabe²

¹ University of Innsbruck

² Jacobs University

Abstract. Mathematical knowledge is publicly available in dozens of different formats and languages, ranging from informal (e.g. Wikipedia) to formal corpora (e.g., Mizar). Despite an enormous amount of overlap between these corpora, few machine-actionable connections exist. We speak of *alignment* if the same concept occurs in different libraries, possibly with slightly different names, notations, or formal definitions. Leveraging these alignments would create a huge potential for knowledge sharing and transfer, e.g., integrating theorem provers or reusing services across systems.

Formally describing and verifying alignments is extremely expensive, as it typically requires not only aligning two concepts but whole libraries together with their foundations, or may even be impossible in case of alignments between semi-formal or informal concepts. Therefore, we introduce a lightweight approach that focuses on identifying the alignments while abstracting from formal definitions. Notably, this is already sufficient for many practically valuable applications.

We present a classification of alignments and design a simple format for describing alignments as well as an infrastructure for sharing them. We propose these as a centralized standard for the community to collect and curate alignments from the different kinds of mathematical corpora, including proof assistant libraries, computer algebra and programming language algorithms, and semi-formal libraries.

1 Introduction

Motivation and Related Work The sciences are increasingly collecting and curating their knowledge systematically in machine-processable corpora. For example, in biology many important corpora take the form of ontologies, e.g., as collected on BioPortal. These corpora typically overlap substantially, and much recent work has focused on integrating them. A central problem here is to find *alignments*: pairs (a_1, a_2) of identifiers from different corpora that describe the same concept.

For ontologies, this problem has been relatively well-studied under the heading *ontology matching* [ESC07]. The situation in mathematics is somewhat special because mathematical knowledge involves rigorous notations, definitions, and properties, and attempts to capture it fully lead to very diverse corpora. Logical corpora have been developed in proof assistants and feature machine-understandable theorems and proofs. Computational corpora have been devel-

oped in computer algebra systems and feature executable definitions and constructions. And narrative corpora have been developed in wikis and related tools featuring human-oriented semi-formal descriptions. For each kind, there are multiple large corpora, often the result of dozens of person-years of investment.

Alignments between computational corpora occur in bridges between the run time systems of programming languages. Alignments between logical and computational corpora are used in proof assistants with code generation such as Isabelle [WPN08] and Coq [Coq15]. Here functions defined in the logic are aligned with their implementations in the programming language in order to generate fast executable code from formalizations.

Wiedijk [Wie06] explored a single theorem (and its proof) across 17 proof assistants implicitly providing alignments between the concepts present in the theorem’s statement and proof. However, finding alignments has proved extremely difficult in general. There are three reasons for this: the conceptual differences between the three kinds of corpora; the differences between the underlying formal languages and tools; and the differences between the organization of the knowledge in the corpora.

The dominant methods for integrating logical corpora so far have focused on truth-preserving translations between the underlying knowledge representation languages. For example, [KS10] translates from Isabelle/HOL to Isabelle/ZF. [KW10] translates from HOL Light to Coq, [OS06] to Isabelle/HOL, and [NSM01] to Nuprl. Older versions of Matita [ACTZ06] were able to read Coq compiled theory files. [CHK⁺11] build a library of translations between different logics.

However, most translations are not alignment-aware, i.e., it is not guaranteed that a_1 will be translated to a_2 even if the alignment is known. This is because a_1 and a_2 may be subtly incompatible so that a direct translation may even lead to inconsistency or ill-typed results. [OS06] was — to the authors knowledge — the first that could be parametrized by a set of alignments. The OpenTheory framework [Hur09] provides a number of higher-order logic concept alignments. In [KR16], the second and fourth author discuss the corpus integration problem and conclude that alignments are of utmost practical importance. Indeed, corpus integration can succeed with only alignment data even if no logic translation is possible. Conversely, logic translations contribute little to corpus integration without alignment data.

An alignment does not need to be perfect to be useful. In some HOL proof assistants real numbers are defined using Cauchy sequences, while in some Dedekind cuts are used. The two structures undoubtedly share all the real number properties. However, if we look at the implications of the definitions themselves, the former implies that there is a canonical Cauchy sequence for each number. Despite this minor difference, we can use such an alignment in a logical translation [KK13]. In fact there is a whole spectrum of perfectness of alignments. An alignment can indeed be perfect when the definitions of two concepts are the same in a logical corpus. Aligning functions that have the same specifications, but differ only in the complexity is particularly useful in computational corpora, for example interfacing languages. Finally, concepts may be just similar.

In set theory zero is defined as the empty set, while in many other foundations is it a constructor of an inductive type. Only a quarter of the properties of the empty set are the same as those of explicitly defined zero.

Many practical services are enabled even by such imperfect alignments. Statistical analogies extracted from large formal libraries combined with imperfect alignments can be used to create new conjectures to automatically explore a logical corpus [GKU16]. Automated reasoning services can use the libraries of alignments to provide more precise proof recommendations [GK15]. Further ideas include searching for a single query expression in multiple corpora at once [AGC⁺04,KR14].

Due to the size of the involved corpora, it is desirable to find alignments automatically. Recently, the first author has developed heuristic methods for automatically finding alignments [GK14] targeted at integrating logical corpora [KK13] including HOL Light, HOL4, and Isabelle/HOL discovering 398 pairs of isomorphic concepts. Consistent name hashing combined with statement normalization was used to discover 39 symbols with equivalent definitions [KU15] in the Flyspeck development [H⁺15]. Ginev built a library of about 50,000 alignments between narrative corpora including Wikipedia, Wolfram Mathworld, PlanetMath and SMGloM [GC14].

Contribution and Overview Our contribution is two-fold. First, we present a phenomenological study of alignments between mathematical corpora in Section 2. Most importantly, we collect the various subtle reasons why an alignment may be imperfect because not all properties of the aligned symbols transfer exactly.

Our standardization includes a standardization of forming MMT URIs [RK13] for a number of a major logical corpora in Section 3. These assign a canonical URI to every symbol in a way that is unique across corpora and across logics. We use these URIs to give several examples from logical corpora in Section 4. The examples focus on logical corpora, but our results carry over to other kinds of corpora as well.

Second, we propose a standard for storing and sharing alignments. Most corpora are developed and maintained by separate, often disjoint communities. That makes it difficult for researchers to utilize alignments because no central repository exists for jointly building a large collection of alignments. We have started such a central repository — it is public, and we invite all researchers to contribute their alignments. We seeded our repository with the alignment sets mentioned above. Moreover, we are hosting a web-server that allows for conveniently querying for all symbols aligned with a given symbol. We describe this standard and infrastructure in Section 5.

2 Types of Alignments

Let us assume two corpora C_1, C_2 with underlying foundational logics F_1, F_2 . We examine examples for how two concepts a_i from C_i can be aligned.

Perfect Alignment If a_1 and a_2 are logically equivalent modulo a translation φ that is fixed in the context, we speak of a perfect alignment. More precisely, all formal properties (type, definition, axioms) of a_1 carry over to a_2 and vice versa. Typical examples are primitive types and their associated operations. Consider:

$$\text{Nat}_1 : \text{Type} \quad \text{Nat}_2 : \text{Type}$$

then translations between C_1 and C_2 can simply interchange a_1 and a_2 .

The above example is deceptively simple for two reasons. Firstly, it hides the problem that F_1 and F_2 do not necessarily share the symbol **Type**. Therefore, we need to assume that there are symbols **Type**₁ and **Type**₂, which have been already aligned (perfectly). Such alignments are crucial for all fundamental constructors that occur in the types and characteristic theorems of the symbols we want to align such as **Type**, \rightarrow , **bool**, \wedge , etc. These alignments can be handled with the same methodology as discussed here. Therefore, here and below, we assume we have such alignments and simply use the same fundamental constructors for F_1 and F_2 .

Secondly, it ignores that we usually only want certain formal properties to carry over, namely those in the *interface theory* in the sense of [KR16]. For example, in Section 4 we give many perfect alignments between symbols that use different but interface-equivalent definitions.

Alignment up to Argument Order Two function symbols can be perfectly aligned except that their arguments must be reordered when translating.

The most common example is function composition, whose arguments may be given in application order ($f \circ g$) or in diagram order ($f;g$). Another example is given

$$\begin{aligned} \text{contains}_1 &: (T : \text{Type}) \rightarrow \text{SubSet } T \rightarrow T \rightarrow \text{bool} \\ \text{in}_2 &: (T : \text{Type}) \rightarrow T \rightarrow \text{SubSet } T \rightarrow \text{bool} \end{aligned}$$

Here the expressions $\text{contains}_1(T, A, x)$ and $\text{in}_2(T, x, A)$ are aligned.

Alignment up to Determined Arguments The perfect alignment of two function symbols may be broken because they have different types even though they agree in most of their properties. This often occurs when F_1 uses a more fine-granular type system than F_2 , which requires additional arguments.

Examples are untyped and typed (polymorphic, homogeneous) equality: The former is binary, while the latter is ternary

$$\text{eq}_1 : \text{Set} \rightarrow \text{Set} \rightarrow \text{bool} \quad \text{eq}_2 : (T : \text{Type}) \rightarrow T \rightarrow T \rightarrow \text{bool}.$$

The types can be aligned, if we apply $\varphi(\text{Set})$ to eq_2 . Similar examples arise between simply- and dependently-typed foundations, where symbols in the latter take additional arguments.

These additional arguments are uniquely determined by the values of the other arguments, and a translation from C_1 to C_2 can drop them, whereas the reverse translations must infer them – but F_1 usually has functionality for that.

The additional arguments can also be proofs, used for example to represent partial functions as total functions, such as a binary and a ternary division operator

$$\text{div}_1 : \text{Real} \rightarrow \text{Real} \rightarrow \text{Real} \quad \text{div}_2 : \text{Real} \rightarrow (d : \text{Real}) \rightarrow \vdash d \neq 0 \rightarrow \text{Real}$$

Here inferring the third argument is undecidable, and it is unique only in the presence of proof irrelevance.

Alignment up to Totality of Functions The functions a_1 and a_2 can be aligned everywhere where both are defined. This often happens since it is often convenient to represent partial functions as total ones by assigning values to all arguments. The most common example is division. div_1 might both have the type $\text{Real} \rightarrow \text{Real} \rightarrow \text{Real}$ with $x \text{div}_1 0$ undefined and $x \text{div}_2 0 = 0$.

Here a translation from C_1 to C_2 can always replace div_1 with div_2 . The reverse translation can usually replace div_2 with div_1 but not always. In translation-worthy data-expressions, it is typically sound; in formulas, it can easily be unsound because theorems about div_2 might not require the restriction to non-zero denominators.

Alignment for Certain Arguments Two function symbols may be aligned only for certain arguments. This occurs if a_1 has a smaller domain than a_2 .

The most fundamental case is the function type constructor \rightarrow itself. For example, \rightarrow_1 may be first-order in F_1 and \rightarrow_2 higher-order in F_2 . Thus, a translation from C_1 to C_2 can replace \rightarrow_1 with \rightarrow_2 , whereas the reverse translation must be partial.

Another important class of examples is given by subtyping (or the lack thereof). For example, we could have

$$\text{plus}_1 : \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat} \quad \text{plus}_2 : \text{Real} \rightarrow \text{Real} \rightarrow \text{Real}.$$

Another way for a_1 to have a smaller domain is to take less arguments. For example, we might have

$$\text{ln}_1 : \text{Real} \rightarrow \text{Real} \quad \text{log}_2 : \text{Real} \rightarrow \text{Real} \rightarrow \text{Real}$$

where $\text{ln}_2(x)$ can be translated to $\text{log}_2(e, x)$.

Alignment up to Associativity An associative binary function (either logically associative or notationally right- or left-associative) can be defined as a flexary function, i.e., a function taking an arbitrarily long sequence of arguments. In this case, translations must fold or unfold the argument sequence. For example

$$\text{plus}_1 : \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat} \quad \text{plus}_2 : \text{List Nat} \rightarrow \text{Nat}.$$

Contextual alignments Two symbols may be aligned only in certain contexts. For example, the complex numbers are represented as pairs of real numbers in some proof assistant libraries and as an inductive data type in others. Then only selected occurrences of pairs of real numbers can be aligned with the complex numbers.

Alignment with a Set of Declarations Here a single declaration in C_1 is aligned with a set of declarations in C_2 . An example is a conjunction a_1 in C_1 of axioms aligned with a set of single axioms in C_2 . More generally, the conjunction of a set of C_1 -statements may be equivalent to the conjunction of a set of C_2 -statements.

Here translations are much more involved and may require aggregation or projection operators.

Alignment between the Internal and External Perspective on Theories Logical theories can be represented in two ways. We define them only by example. We speak of the internal perspective if we use a theory like

$$\text{theory Magma}_1 = \{u_1 : \text{Type}, \circ_1 : u_1 \rightarrow u_1 \rightarrow u_1\}$$

and of the external perspective if we use operations like

$$\text{Magma}_2 : \text{Type}, u_2 : \text{Magma}_2 \rightarrow \text{Type}, \circ_2 : (G : \text{Magma}) \rightarrow u_2 G \rightarrow u_2 G \rightarrow u_2 G$$

Here we have non-trivial, systematical translation from C_1 to C_2 ; a reverse may also be possible, depending on the details of F_1 .

Corpus-Foundation Alignment Orthogonal to all of the above, we have to consider alignments, where a symbol is primitive in one system but defined in another. More concretely, a_1 can be built-into F_1 whereas a_2 is defined in F_2 . This is common for corpora based on significantly different foundations, as each foundation is likely to select different primitives. Therefore, it mostly occurs for the most basic concepts. For example, the boolean connectives, integers and strings are defined in some systems but primitive in others, as in some foundations they may not be easy to define.

The corpus-foundation alignments can be reduced to previously considered cases if we follow the “foundations-as-theories” approach [KR16], where the foundations themselves are represented in an appropriate logical framework. Then a_1 is simply an identifier in the corpus of foundations of the framework F_1 .

Opaque Alignments The above alignments focused on logical corpora, partially because logical corpora allow for precise and mechanizable treatment of logical equivalence. Indeed, alignments from a logical into a computational or narrative corpus tend to be opaque: Whether and in what way the aligned symbols correspond to each other is not (or not easily) machine-understandable. For example, if a_2 refers to a function in a programming language library, that functions specification may be implicit or given only informally. Even worse, if a_2 is a wiki article, it may be subject to constant revision.

Nonetheless, such alignments are immensely useful in practice and should not be discarded. Therefore, we speak of opaque alignments if a_2 refers to a symbol whose semantics is unclear to machines.

3 Global Identifiers

An essential requirement for relating logical corpora is standardizing the identifiers so that each identifier in the corpus can be uniquely referenced. It is desirable to use a uniform naming schema so that the syntax and semantics of identifiers can be understood and implemented as generically as possible. Therefore, we use MMT URIs [RK13], which have been specifically designed for that purpose.

3.1 General Structure

Syntax MMT URIs are triples of the form

NAMESPACE ? MODULE ? SYMBOL

The namespace part is a URI that serve as globally unique root identifiers of corpora (e.g., `http://mathhub.info/MyLogic/MyLibray`). It is not necessary (although often useful) for namespaces to also be URLs, i.e., a reference to a physical location. But even if they are URLs, we do not specify what resource dereferencing should return. Note that because MMT URIs use `?` as a separator, `MODULE ? SYMBOL` is the query part of the URI, which makes it easy to implement dereferencing in practice.

The module and symbol parts of an MMT URI are logically meaningful names defined in the corpus: The module is the container (e.g., a signature, functor, theory, class, etc.) and the symbol is a name inside the module (of a type, constant, axiom, theorem etc.). Both module and symbol name may consist of multiple `/`-separated segments to allow for nested modules and qualified symbol names.

MMT URIs allow arbitrary Unicode characters. However, `?` and `/`, which MMT URIs use as delimiters, as well as any character not legal in URIs must be escaped using the `%`-encoding. We refer to RFC 3986/7 for details.

Both the corpus itself and the system with which it was processed may be subject to change. Therefore, it may be useful to record a version in an MMT URI. However, most developers take care to avoid semantically critical changes at least to the widely used parts of their corpora. Since those parts are also the most interesting ones for integration, we omit issues of versioning here and simply remark that the version can be recorded as a part of the root namespace.

Formation Principles The precise formation of MMT URIs may depend subtly on the foundational logic underlying the corpus. In the sequel, we identify some general principles that allow stating the formation rules concisely.

The most important physical structure of a corpus is usually a directory tree, whose leaves are files containing modules. In this case, the following principles are typical options to define namespaces:

- **flat** structure: All modules use the same namespace as the root namespace of the corpus regardless of their physical location in the corpus. This naming schema is most well-known from SML.

- **directory-based** structure: The namespace of a module is formed by concatenating the root namespace with the path to the directory containing it. There are two subcases regarding the treatment of the file name:
 - **files-as-modules**: The file contains exactly one module. The name of the module may be given *explicitly* in the file or may be *implicit*. Either way, the name of the module must be the same as the file name without the file name extension. Files as explicitly named modules is most well-known as the convention of Java.
 - **irrelevant file names**: The file name is irrelevant, i.e., the grouping of modules into files within the same directory is arbitrary. In particular, a file can contain multiple modules.
- **file-based** structure: The namespace of a module is formed by concatenating the root namespace of the corpus with the path to the file containing it.

3.2 URIs for Selected Proof Assistants

Using the principles defined above, we describe the MMT URI formation principles for some important proof assistants. In all cases, we also assign MMT URIs for the underlying foundations in order to refer to built-in concepts.

PVS [ORS92] uses directory-based namespaces with irrelevant file names. We propose the following root URIs for some important PVS-related corpora:

- the PVS foundation: <http://pvs.csl.sri.com/foundation>
- the standard library that is shipped with PVS: <http://pvs.csl.sri.com/Prelude>
- the NASA corpus: <http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library>

Within a PVS corpus, the top-level modules are theories and (co)datatype declarations. The only possible nesting between them is that theories may contain (co)datatype declarations. Consequently, the module names have at most two segments. If a module is a (co)datatype, its symbols are the constructors, testers, etc. These are not hierarchical. If a module is a theory, its symbols are all declarations declared in it. These are not hierarchical. However, if a symbol name N is declared multiple times in the same module (due to overloading), we use two-level names of the form N/i where i numbers all declarations of N in that module (starting from 1).

Coq [Coq15] uses directory-based namespaces with files as implicitly named modules. We propose the following root URIs:

- the Coq foundation: <https://coq.inria.fr/foundation>
- the standard library shipped with Coq: <https://coq.inria.fr/theories>
- the Coq contributions: <https://coq.inria.fr/contribs>
- the Mathematical Components corpus (including SSReflect):
<http://ssr.msr-inria.inria.fr/math-comp>

Coq modules can be nested. Besides the module name given implicitly by the file, Coq files can contain modules and module types. Symbols are all declarations inside a module. Their names can be hierarchic due to generative functor instantiation.

Matita uses the same URIs as Coq except for not allowing nested modules. We suggest the following root URIs:

- the Matita foundation: <http://matita.cs.unibo.it/foundation>
- the Matita standard library: <http://matita.cs.unibo.it/library>

Mizar uses a flat namespace. We propose the following root URIs:

- the Mizar foundation: <http://mizar.org/foundation>
- the Mizar Mathematical Library (MML): <http://mizar.org/library>

The Mizar modules are the articles. The name of a module is the name of the article without the file name extension. There is no nesting of modules. The Mizar symbols are all the declarations inside an article. Their names are obtained through a heavily idiosyncratic naming schema that includes generating unique names by numbering the declarations of the same kind in each article. For example, the MMT URI of conjunction is <http://mizar.org/corpus?XBOOLEANO?K4>

HOL Light does not have an obvious MMT URI formation principle because it does not maintain all its identifiers itself — instead it relies on the OCaml toplevel to store the assigned values. There are three kinds of HOL Light symbols: types, constants, and theorems; only the former two are visible to the HOL Light kernel. For each type or constant, we use the name visible to the HOL Light kernel. For each theorem, we use the OCaml binding name. OCaml toplevel symbols may be grouped into OCaml modules. This feature is seldom used, as it only affects theorem names: the kernel is also not aware of the OCaml module in which the definitions of constants or types are introduced.

This has the effect that different HOL Light files can be incompatible with each other. There are two reasons for this incompatibility: First, toplevel symbols may be overwritten by others, which means that the original ones are no longer accessible and a formalization might fail. This happens for example in case of the OCaml basic output function `open_in` which gets overwritten by a theorem with the same name. Second, the loading of a file may change the state of HOL Light kernel or packages to one that is no longer compatible with another file [Wie09]. For example the theories `complexnumbers.ml` and `complexes.ml` both introduce the type `complex` but with different definitions.

Thus, HOL Light’s names do not uniquely identify symbols. Therefore, we use directory-based namespaces with files-as-modules. For constants and types introduced by a module we add the prefixes `const/` and `type/` respectively. If a file contains OCaml modules, we use their names to form multi-segment module names. Accordingly, if symbols result from OCaml structures, we form multi-segment symbol names. This has the effect that HOL Light URIs are formed in exactly the same way as for Coq.

We propose the following root URIs:

- the HOL Light foundation and the library shipped with it:
<http://github.com/jrh13/hol-light>
- the Formal Proof of Kepler formalization [H⁺15]:
<http://github.com/flyspeck/flyspeck>

For example, the Flyspeck theorem `well_defined_unordered_pair` is assigned the URI: http://github.com/flyspeck/flyspeck?text_formalization/packing/marchal3?Marchal_cells_3.well_defined_unordered_pair.

HOL4 internal module names correspond to names of files, however the names of types and constants are not associated with the modules. Furthermore, the names of constants and types are separate. To avoid ambiguity we use the same module names as for HOL Light. We propose the following root URIs:

- the HOL4 foundation and the library shipped with it:
<https://hol-theorem-prover.org>

Isabelle is a logical framework: Its distribution includes a number of object logics, and each Isabelle theory uses an object logic (or declares a new one). Isabelle uses files as explicitly named modules. However, it disregards the directory structure: The system makes sure that two modules with same name cannot be loaded in the same session even if they are stored in different directories. But as different object logics and different developments often declare incompatible notions, we still use directory-based namespaces to make sure all theories have unique namespaces.

Isabelle allows several module mechanisms including locales and type classes [HW06]. Therefore, we form nested module names by concatenating theory and locale/type class names. We propose the following root URIs:

- Isabelle foundation and distributed libraries: <http://isabelle.in.tum.de/>
- the Archive of Formal Proofs: <http://afp.sf.net/>
- the TLA+ logic: <http://tla.msr-inria.inria.fr/>

For example the type of streams is represented by the URI:
<http://isabelle.in.tum.de/?HOL/corpus/Stream?stream>

4 Examples of Alignments

Using the MMT URIs defined in Sect. 3, we give a detailed presentation of alignments across proof assistants for three representative concepts. We also include some alignments to programming languages, which are relevant for code generation. In all cases, we will see how big the differences between the details are across the logical corpora even though most of the alignments are in fact perfect.

Cartesian Product In constructive type theory, there are two common ways of expressing the non-dependent Cartesian product. First, if the foundation has inductive types such as the Calculus of Inductive Constructions, it can be an inductive type with one binary constructor. Second, if the foundation has a dependent sum type, Cartesian products can be the non-dependent special case. The first two symbols below use the former, the last one the latter approach:

- <http://coq.inria.fr/theories?Init/Datatypes?prod.ind>
- <http://matita.cs.unibo.it/?datatypes/constructors?Prod.ind>
- <http://isabelle.in.tum.de/?CTT/CTT?times>

In higher-order logic, the only way to introduce types is by using the `typedef` construction, which constructs a new type that is isomorphic to a certain subtype of an existing type. In particular, most HOL systems introduce the Cartesian product $A \times B$ by using an appropriate unary predicate on $A \rightarrow B \rightarrow \text{bool}$:

- <http://github.com/jrh13/hol-light?pair/type?prod>
- <http://hol-theorem-prover.org/?pair/type?prod>
- <http://isabelle.in.tum.de/?HOL/Product?prod>

In PVS, the product type constructor is part of the system itself:

- http://pvs.csl.sri.com/foundation?PVS?tuple_tp

In set theory, it is also possible to restrict dependent sum types to obtain the Cartesian product. This is used in Isabelle/ZF. In Mizar the Cartesian product is defined implicitly as a first order functor, which involves discharging the well-definedness condition. Therefore, we give the URIs of both the definition and the generated functor.

- http://isabelle.in.tum.de/?ZF/ZF?cart_prod
- http://mizar.org/library/?ZFMISC_1?K2
(defined by http://mizar.org/library/?ZFMISC_1?def_2)

Finally, Cartesian products appear in most programming languages and we list here a number of constructions that can be aligned:

- <http://caml.inria.fr/?core?>*
- <http://haskell.org/?core?>,
- <http://scala-lang.org/?core?>,
- <http://cppreference.com/?std?pair>

Informal sources that can be aligned are e.g.:

- https://en.wikipedia.org/wiki/Cartesian_product
- https://en.wikipedia.org/wiki/Product_type
- <http://mathworld.wolfram.com/CartesianProduct.html>

Addition In constructive type theory, addition of natural numbers is typically defined as a fixed points of certain equations.

- <http://coq.inria.fr/theories?Init/Nat?add>
- <http://matita.cs.unibo.it/?nat/plus?plus>

Higher-order logic proof assistants usually use high-level constructions for primitive recursion. Interestingly, here the alignment is very obvious at this high-level even though the elaboration into core language features may result in very different-looking, but logically equivalent definitions. This is how addition is implemented in HOL Light and HOL4. Isabelle/HOL uses a similar construction but inside a type class for commutative monoids with difference.

- <http://github.com/jrh13/hol-light?arith/const?+>
- <http://hol-theorem-prover.org/?arithmetic/const?+>
- http://isabelle.in.tum.de/?HOL/Nat?plus_nat_inst.plus_nat

In set theory, defining addition is surprisingly the least straightforward: In Isabelle/ZF, addition uses the `primrec` construction together with a coercion from non naturals to naturals. PVS defines a general addition on number fields, which all the concrete number spaces (reals, integers, etc.) inherit. In Mizar, it is the restriction of complex addition to natural numbers. Mizar's complex

addition itself is built from the real number addition, which in turn is built from positive real addition, rational addition, and ordinal addition. So in principle it would be possible to align with Mizar’s ordinal addition.

- http://pvs.csl.sri.com/Prelude?number_fields?+
- <http://isabelle.in.tum.de/?ZF/Arith?add>
- http://mizar.org/library/?NAT_1?K2

Most programming languages do not implement arbitrary precision natural numbers. However, it is possible to find partial mappings, which are in fact already used by efficient code generation [HN10] for Isabelle/HOL natural numbers and the representation of Coq natural numbers by its extended bytecode machine [AGST10].

- http://caml.inria.fr/?Big_Int?add_big_int
- <http://haskell.org/?core?+>
- <http://www.smlnj.org/?IntInf?+>

Concatenation of Lists In constructive type theory (e.g. for Matita, Coq), the append operation on lists can again be defined as a fixed point. In higher-order logic, append for polymorphic lists can be defined by primitive recursion, as done by HOL Light and HOL4. Isabelle/HOL slightly differs from these two because it uses lists that were built with the co-datatype package [B⁺14].

- <http://coq.inria.fr/theories?Init/Datatypes?app>
- <http://github.com/jrh13/hol-light?lists/const?APPEND>
- <http://hol-theorem-prover.org/?list/const?APPEND>
- <http://isabelle.in.tum.de/?HOL/List?append>

In set theory, PVS and Isabelle/ZF also use primitive recursion for monomorphic lists. In Mizar, lists are represented by finite sequences, which are functions from a finite subset of natural numbers (one-based FINSEQ and zero-based XFINSEQ) with append provided.

- http://pvs.csl.sri.com/Prelude?list_props?append
- http://isabelle.in.tum.de/?ZF/List_ZF?app
- <http://mizar.org/library/?ORDINAL4/K1>

Concatenation of lists is also common in programming languages.

- <http://caml.inria.fr/?core?@> <http://haskell.org/?core?++>
- <http://scala-lang.org/?core?++>

5 A Standard and Database for Alignments

Based on the observations of the previous sections, we now define a standard for alignments that covers the practically relevant examples. We use the following formal grammar for collections of alignments:

```

Collection ::= (NSDef | Alignment | Comment)*
NSDef      ::= namespace String URI
Alignment  ::= URI URI (String = "String")*
Comment    ::= // String

```

Here NSDef defines abbreviations for CURIEs (as defined by the W3C), which allows shortening URIs with the same long namespaces. An alignment is just a pair of URIs with a list of key-value pairs, which allows adding author/source, certainty scores, translation instructions, etc.

We also standardize some special keys and possible values that are important for practical applications. As a guiding criterion for defining these keys, we use how and in which directions expressions with head symbols s_1 or s_2 can be translated.

The simplest case is the following:

Definition 1. A *simple alignment* between symbols s_1 and s_2 uses the key *direction* with the possible values **forward**, **backward**, and **both**. It induces the translation that replaces every occurrence of s_1 with s_2 , or of s_2 with s_1 according to the value of the key.

This subsumes perfect alignments (where the direction is **both**) and several unidirectional cases: alignment up to totality of functions or up to associativity, and alignment for certain arguments. The absence of this key indicates alignments where no translation is possible, in particular opaque alignments.

The following case covers alignments up to argument order or determined arguments:

Definition 2. An *argument alignment* uses the key **arguments** whose value A is of the form $(i, j)^*$ where i and j are natural numbers.

It induces the translation of $s_1(x_1, \dots, x_m)$ to $s_2(y_1, \dots, y_n)$ where y_j is the recursive translation of x_i if (i, j) is among the pairs in A and inferred from the context if there is no i for which (i, j) is in A .

Example 1. We obtain the following argument alignments for some of the examples from Section 2:

```

Nat1 Nat2 direction = "both"
eq1 eq2 arguments = "(1, 2)(2, 3)"
contains1 in2 arguments = "(1, 1)(2, 3)(3, 2)"

```

We have implemented alignments in the MMT system [Rab13]. Moreover, we have created a public repository³ and seeded it with a number of alignments including the ones mentioned in this paper. The MMT system can be used to read and serve all these alignments, implement the transitive closure, and (if possible) translate expressions according to alignments.

As an example service, we have added alignment support to the MMT web browser: Figure 1 shows a screenshot from browsing the HOL Light library, in particular a snapshot of the `pairs` module. The symbol `prod` is aligned with several formal and informal sources, which can be shown and navigated to by right-clicking the symbol in an expression (here in the type of the symbol `ABS_prod`).

³ <https://gl.mathhub.info/alignments/Public>

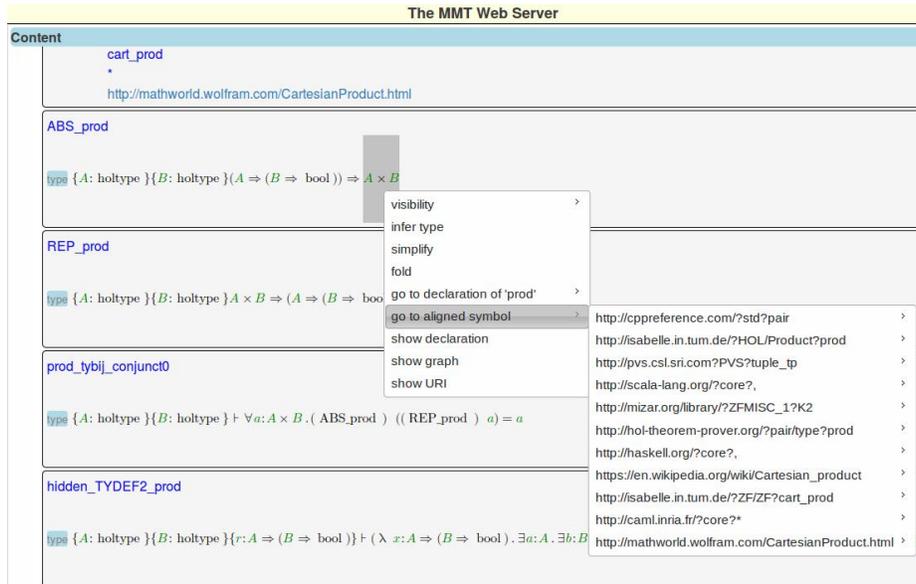


Fig. 1. The MMT Server Showing Formal and Informal Alignments

6 Conclusion

We have motivated and proposed a standard for aligning mathematical corpora. We presented examples of alignments between logical, computational, and semi-formal corpora and classified the different examples. The presented MMT-based system for sharing such alignments has been preloaded with thousands of alignments between the various kinds of concepts, including proof assistant types and constants, programming language (including computer algebra) algorithms, and semi-formal descriptions.

Future work includes extending the automated discovery of alignments [GK14] to foundations other than HOL. Our main focus was on the logical corpora, but we expect to be able to find much more opaque alignments. We invite the community to use the service. Finally we plan to integrate the use of the alignments database in the various mathematical knowledge management systems.

Acknowledgements We acknowledge financial support from the German Science Foundation (DFG) under grants KO 2428/13-1 and RA-1872/3-1 and the Austrian Science Fund (FWF) grant P26201.

References

- ACTZ06. A. Asperti, C. Sacerdoti Coen, E. Tassi, and S. Zacchirolì. Crafting a Proof Assistant. In T. Altenkirch and C. McBride, editors, *TYPES*, pages 18–32. Springer, 2006.

- AGC⁺04. Andrea Asperti, Ferruccio Guidi, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. A content based mathematical search engine: Whelp. In Jean-Christophe Filliâtre, Christine Paulin-Mohring, and Benjamin Werner, editors, *Types for Proofs and Programs, International Workshop, TYPES 2004, Jouy-en-Josas, France, December 15-18, 2004, Revised Selected Papers*, volume 3839 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2004.
- AGST10. Michaël Armand, Benjamin Grégoire, Arnaud Spiwack, and Laurent Théry. Extending Coq with imperative features and its application to SAT verification. In Matt Kaufmann and Lawrence C. Paulson, editors, *ITP*, volume 6172 of *LNCS*, pages 83–98. Springer, 2010.
- B⁺14. Jasmin Christian Blanchette et al. Truly modular (co)datatypes for Isabelle/HOL. In Gerwin Klein and Ruben Gamboa, editors, *ITP*, volume 8558 of *LNCS*, pages 93–110. Springer, 2014.
- CHK⁺11. M. Codescu, F. Horozal, M. Kohlhase, T. Mossakowski, and F. Rabe. Project Abstract: Logic Atlas and Integrator (LATIN). In J. Davenport, W. Farmer, F. Rabe, and J. Urban, editors, *Intelligent Computer Mathematics*, pages 289–291. Springer, 2011.
- Coq15. Coq Development Team. The Coq Proof Assistant: Reference Manual. Technical report, INRIA, 2015.
- ESC07. J. Euzenat, P. Shvaiko, and Ebooks Corporation. *Ontology matching*. Springer, 2007.
- GC14. Deyan Ginev and Joseph Corneli. Nnexus reloaded. In Watt et al. [WDS⁺14], pages 423–426.
- GK14. Thibault Gauthier and Cezary Kaliszyk. Matching concepts across HOL libraries. In Stephen Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors, *CICM*, volume 8543 of *LNCS*, pages 267–281. Springer Verlag, 2014.
- GK15. Thibault Gauthier and Cezary Kaliszyk. Sharing HOL4 and HOL Light proof knowledge. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *LPAR*, volume 9450 of *LNCS*, pages 372–386. Springer, 2015.
- GKU16. Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. Initial experiments with statistical conjecturing over large formal corpora. In *Work in Progress at CICM*, 2016. to appear.
- H⁺15. Thomas C. Hales et al. A formal proof of the kepler conjecture. *CoRR*, abs/1501.02155, 2015.
- HN10. Florian Haftmann and Tobias Nipkow. Code generation via higher-order rewrite systems. In Matthias Blume, Naoki Kobayashi, and Germán Vidal, editors, *FLOPS*, volume 6009 of *LNCS*, pages 103–117. Springer, 2010.
- Hur09. J. Hurd. OpenTheory: Package Management for Higher Order Logic Theories. In G. Dos Reis and L. Théry, editors, *Programming Languages for Mechanized Mathematics Systems*, pages 31–37. ACM, 2009.
- HW06. Florian Haftmann and Makarius Wenzel. Constructive type classes in Isabelle. In Thorsten Altenkirch and Conor McBride, editors, *Types for Proofs and Programs, International Workshop, TYPES 2006*, volume 4502 of *LNCS*, pages 160–174. Springer, 2006.
- KK13. Cezary Kaliszyk and Alexander Krauss. Scalable LCF-style proof translation. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP*, volume 7998 of *LNCS*, pages 51–66. Springer Verlag, 2013.
- KR14. Cezary Kaliszyk and Florian Rabe. Towards knowledge management for HOL Light. In Watt et al. [WDS⁺14], pages 357–372.

- KR16. M. Kohlhase and F. Rabe. QED Reloaded: Towards a Pluralistic Formal Library of Mathematical Knowledge. *Journal of Formalized Reasoning*, 9(1):201–234, 2016.
- KS10. A. Krauss and A. Schropp. A Mechanized Translation from Higher-Order Logic to Set Theory. In M. Kaufmann and L. Paulson, editors, *Interactive Theorem Proving*, pages 323–338. Springer, 2010.
- KU15. Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.
- KW10. C. Keller and B. Werner. Importing HOL Light into Coq. In M. Kaufmann and L. Paulson, editors, *Interactive Theorem Proving*, pages 307–322. Springer, 2010.
- NSM01. P. Naumov, M. Stehr, and J. Meseguer. The HOL/NuPRL proof translator - a practical approach to formal interoperability. In R. Boulton and P. Jackson, editors, *14th International Conference on Theorem Proving in Higher Order Logics*. Springer, 2001.
- ORS92. S. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, pages 748–752. Springer, 1992.
- OS06. S. Obua and S. Skalberg. Importing HOL into Isabelle/HOL. In N. Shankar and U. Furbach, editors, *Automated Reasoning*, volume 4130. Springer, 2006.
- Rab13. F. Rabe. The MMT API: A Generic MKM System. In J. Carette, D. Aspinall, C. Lange, P. Sojka, and W. Windsteiger, editors, *Intelligent Computer Mathematics*, pages 339–343. Springer, 2013.
- RK13. F. Rabe and M. Kohlhase. A Scalable Module System. *Information and Computation*, 230(1):1–54, 2013.
- WDS⁺14. Stephan Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors. *Intelligent Computer Mathematics*, number 8543 in LNCS. Springer, 2014.
- Wie06. Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of LNCS. Springer, 2006.
- Wie09. Freek Wiedijk. Stateless HOL. In Tom Hirschowitz, editor, *TYPES*, volume 53 of *EPTCS*, pages 47–61, 2009.
- WPN08. Makarius Wenzel, Lawrence C. Paulson, and Tobias Nipkow. The Isabelle framework. In Ait Mohamed, Munoz, and Tahar, editors, *Theorem Proving in Higher Order Logics (TPHOLs 2008)*, number 5170 in LNCS, pages 33–38. Springer, 2008.

FramelT Reloaded: Serious Math Games from Modular Math Ontologies

Denis Rochau, Michael Kohlhase, and Dennis Müller

Computer Science, Jacobs University, Bremen, Germany

Abstract. Serious games are an attempt to leverage the inherent motivation in game-like scenarios for an educational application and to transpose the learning goals into real-world applications. Unfortunately, serious games are also very costly to develop and deploy. For very abstract domains like mathematics, already the representation of the knowledge involved becomes a problem.

We propose the **FramelT** method that uses OMDoc/MMT theory graphs to represent and track the underlying knowledge in serious games. In this paper we report on an implementation and experiment that tests the method. We obtain a simple serious game by representing a “word problem” in OMDoc/MMT and connect the MMT API with a state-of-the-art game engine that “plays” the problem/knowledge exploration process.

1 Introduction

Serious games could be a potential solution to the often-diagnosed problem that traditional education via personal instruction and educational documents has serious scalability, subject specificity, and motivation limitations. A serious game is *“a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives”* [Zyd05]. Serious games have the power to effectively supplement technical documents and online courses and thereby allow students to learn how to apply their knowledge to real world scenarios. Moreover, serious games very elegantly solve the motivation problem many people experience when studying technical subjects. Through gamification [Det+11] a serious game can be very entertaining while at the same time providing educational value to the user. Unfortunately, serious games are currently either very complex, domain specific, and expensive applications or lack the necessary complexity to effectively facilitate learning of complex subjects like mathematics.

To alleviate this, we propose the **FramelT** method [KK12] which uses mathematical knowledge management techniques (MKM) to map a real world scenario to its theoretical basis. In this paper we report on an implementation and experiment that tests the method. We obtain a very simple serious game by representing a “word problem” in OMDoc/MMT and connect the MMT API with a state-of-the-art game engine. This paper is a short version of [Roc16], to which we refer for details.

2 Preliminaries

Before we can explain the **FramelT** method in detail we need to establish a clear and common understanding of the **OMDoc/MMT** format and **Learning Object Graphs** as they form the basis of the method.

3 Learning Object Graphs as OMDoc/MMT Theories

To realize our implementation we are going to use the OMDoc/MMT language [RK13], which is implemented MMT system (Meta-Meta-Tool) [Rab13], as a foundation independent logic framework that allows us to create logic powered applications. OMDoc/MMT consists of theories, symbols, and objects which are related to each other by typing and equality. A theory in OMDoc/MMT is defined as a list of symbol declarations, where each symbol declaration is of the form $c[: t][= d][\#N]$, where c is the symbol identifier, the objects t and d are its type and definiens, and N its notation. MMT objects over a theory are formed from the symbols available to the theory. Moreover, theories can represent, via the Curry-Howard correspondence, judgments, inference rules, axioms, and theorems. A theory morphism for two theories A and B is a morphism $m : A \rightarrow B$ that relates both theories to each other by mapping every symbol from the theory A to a symbol in the theory B .

Theories and theory morphisms together form multigraphs, which we call **theory graphs**, that relate different theories to each other. In such a theory graph we have two different kinds of edges: **views** and **inclusions**. Inclusions allow us to combine and reuse multiple theories by including them in an inheritance style fashion while views allow us "to link two pre-existing theories" given that "all the source axioms are theorems of the target theory" [Koh14]. Furthermore, inclusions cannot form a cycle in a theory graph, while views can.

We will use theory graphs as **learning object graphs** in the FramelT context. They form the fundamental basis for the FramelT method as they allow us to relate different learning objects with each other in a machine understandable and logical way [KK12].

As theories and theory morphisms form a category with certain colimits, MMT is able to derive a pushout from two theories A and B it exists. A pushout takes two theory morphisms $f : C \rightarrow A$ and $g : C \rightarrow B$ and produces a theory P and two morphisms $i_1 : A \rightarrow P$ and $i_2 : B \rightarrow P$ such that the square commutes (Figure 1). Intuitively, the pushout P is formed as the union of A and B so that they share exactly C . [Rab15]

In the current version of MMT the result of a pushout is a new MMT theory that contains a set of simplified declarations. Lastly, MMT provides us with several ways of developing services and applications on top of it, we can either use the RESTful interface, develop MMT plugins in Java/Scala [Rab13] or directly execute a Scala script via the MMT shell.

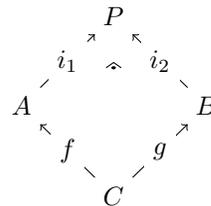


Fig. 1. Pushout

3.1 Unreal Engine

The Unreal Engine is a game engine that provides game developers with the necessary tools to design and build games, simulations, and visualizations [Inc16]. The provided tools include a rich game editor that lets the developer to manage game assets, create the game user interface and build up the game environment.

The functionality of a game can be developed in C++ and/or by blueprint visual scripting. Another important feature of the Unreal Engine 4 is its platform independence, which allows us to deploy our project to nearly all platforms. This is crucial for future development as it enables us to target a larger audience. The engine supports plugins and the complete source code of the engine is available on GitHub, which could allow us in the future to include support for MMT natively.

4 The FramelT Method

The central idea of this new approach to serious games is to use the **FramelT** method to frame real world problems in our serious game environment as abstract problems [KK12]. We have extended and refined this method to address the shortcomings revealed during its implementation.

The primary objective of this method is to provide students with a new way of applying their domain knowledge to real problems. It is not intended as the primary way of obtaining new knowledge. Students are still expected to acquire

new knowledge through personal instruction, by reading technical documents or by taking online courses. Nevertheless, our method helps the user to retain and understand the learned material. To exemplify the FramelT method we want to illustrate its application on a concrete example problem – finding the height of a tree (see Figure 2).

In this example, we assume that the user has recently learned about trigonometry. Thus, the user might realize, after thinking about the problem, that they can map the real world problem to a simple trigonometry problem. This process of discovering a mapping between real world problems and abstract problems is exactly what the user should learn/train by playing our serious game.

This problem is similar to what many students might find in their mathematics textbook when they first learn about trigonometry. In addition, many textbooks might even provide small diagrams to visualize the problem. While this

Word Problem	Game Problem
How can you measure the height of a tree you cannot climb, when you only have a laser angle finder and a tape measure at hand?	

Fig. 2. Example Problem

method of presenting problems to students is acceptable for elementary problems, it is an inadequate approach for more complicated compound problems which are harder for students to understand and solve. Additionally, traditional mathematics textbook problems often focus on the computational aspects of solving problems, rather than on the more conceptual aspects that are underlying these computational solutions.

Our method allows students to explore a problem in its entirety in the game world. We will see below that being able to explore the game world and our method itself allows students to understand and solve even more complex problems in a step by step fashion. Furthermore, the focus of our method is not to train the ability of students to evaluate mathematical expressions, but instead their ability to apply their knowledge to real world problems. Thus, in our game any computational results are provided by MMT or the game itself.

Figure 3 shows the complete learning object graph¹ associated with the problem from Figure 2. It is partitioned into a game world side and MMT side. The game world side shows the different parts of the serious game the user is able to interact with, while the MMT side contains the learning object graph that powers the serious game and allows our approach to work. Additionally, the MMT side is subdivided into two separate sections, where the first section represents the current user knowledge and the second section represents the new knowledge that should be obtained by the learner. Actually, this separation is made for explanatory purposes only. In practice these sections are part of the same theory graph.

4.1 Game World Side (User Perspective)

On the game world side, the user is prompted to solve a given problem, here to find the height of a tree. Before trying to solve the problem at hand the user is able to explore his or her surroundings to accustom himself or herself with the problem and to think about possible approaches. In particular the user has to register facts about the world they are in. In the beginning, a user knows nothing about the world and so their list of registered facts is empty. However, the user can obtain new facts about the world by using scrolls or the a set of gadgets provided.

In our method scrolls are a mechanism to obtain new facts about the world from existing ones. A **scroll** is a game object that contains mathematical explanations and conditions that need to be satisfied in order for the given scroll to produce a new fact about the world. The name “scroll” is meant to evoke the fact that the knowledge contained in it is a valuable commodity in the game. Eventually scrolls are objects that can be found, traded, and earned. From a logical perspective a scroll acts like a complex inference rule that encapsulates a theorem or definition. To apply a given scroll, the user has to map a subset of the facts they obtained about the world to the contents of the specific scroll.

¹ We have glossed the contents of the theories and partially visualized them by diagrams, for the actual content see Figure 4.

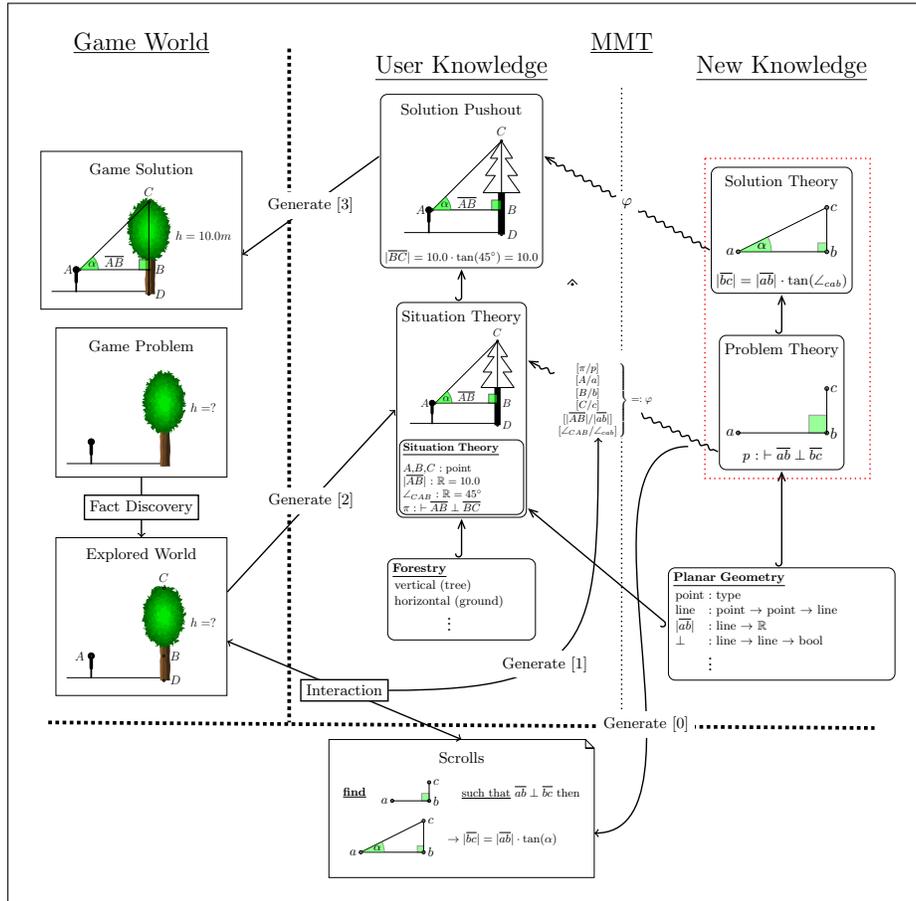


Fig. 3. Learning Object Graph Example

As scrolls require facts as their input, the user has to first use some of the provided gadgets to determine properties of the world and the objects within it. A **gadget** is a game object that can either produce new facts solely by interacting with the world itself or by relating existing facts to each other and thereby providing additional information. For instance, the **pointer gadget** marks a point in the game world and produces a new fact, which simply declares the existence of a point with the specified name. Based on this the user can relate two different point facts to each other by measuring the distance between them with a **measuring tape gadget**. The **laser angle finder gadget** to measure angles between three different marked points. The result of the exploration is visualized on the left hand side of Figure 3.

Didactically, the game world situation is rigged so that the user cannot solely rely on the provided gadgets to solve the given problem. Instead they need to

use scrolls to discover new facts about the world. In our example, the user cannot climb the tree and therefore not measure its height directly with the provided measuring tape gadget. In the problem at hand, the user could chose the trigonometry scroll at the bottom Figure 3 which provides the length of the opposite side of a right angle triangle given the angle and the length of the adjacent side.

In this situation, the **FramelT** methods works as follow: The user uses the marking gadget to mark multiple points in the game world, such as a point A at his or her current location, a point B at the base of the tree and a point C at the top of the tree. The user proceeds by using the laser angle finder gadget to measure the angle between the points C , A and B and the measuring tape is used to determine the distance between the point A and the B . Lastly, the user maps the obtained facts to the contents of the scroll by assigning points A , B , C to the required point a , b , and c in the scroll. If the perpendicularity condition is met, the scroll is applicable and will produce a new fact, otherwise it will provide the user with an explanation of why the scroll was not applicable² Finally

Lastly, the user has to assign a given fact as the solution of the problem. In our case they have to declare that the newly obtained fact about the distance between the points B and C represent the height of the tree. The game is then going to check this answer and if the user solved the problem correctly it is going to congratulate the user on the correct solution. Otherwise, the game will show them the flaws in their approach in order to facilitate learning.

4.2 OMDoc/MMT: Framing Situations and Pushing Out Solutions

The concrete knowledge registered by the user and abstract mathematical knowledge are jointly represented in a modular theory graph in OMDoc/MMT – see the right side of Figure 3.

We formalize the current list of registered facts into the **Situation Theory**, which is extended after any exploratory action of the user: Each discovered fact is translated from its representation in the game to a symbol declaration in MMT. For instance, the marked point A in the game world has underlying data structures associated with it and through this generation step we distill the elaborate game data structures into one symbol declaration, that declares a new constant symbol A of type *point*.

To fix the meaning of declarations in the situation theory, it includes several other theories by using **inclusion theory morphisms**. These imported theories are either theories that contain additional user knowledge about the world or base theories that provide us with the essential declarations to build up more sophisticated theories. In our example one of the included user knowledge theories could be a **forestry theory** which specifies properties of forests and trees, known to the user, such as that a tree is vertical and the ground is normally

² Generating helpful explanations in case of failure was left to future work. It is clear that scrolls can be extended suitably, e.g. via the techniques put forward in [GM08].

horizontal. A common base theory could be a **planar geometry** theory that provides us with declarations for points and lines.

Thus a situation theory and its included theories only represent the current user knowledge about the world. To discover new facts about the world we need to frame the current user knowledge to a **problem/solution pair** [KK12] that produces new facts about the world. A problem/solution pair consists of two theories, the **problem theory** and the **solution theory** and an inclusion morphism between them. A problem theory contains a formal definition of an abstract (mathematical) problem, while the solution theory contains the corresponding solution to the problem. In other words, the problem theory specifies the required facts that are needed as an input for a scroll and the solution theory specifies the new facts that will be obtained as an output. Therefore, a problem/solution pair acts similarly to a mathematical function. In Figure 3 one can see an example for a problem/solution pair in the red box on the right hand side. Similar to the situation theory, the problem theory is supported by the inclusion of several other theories such as the planar geometry theory. In fact these are important for the application of the mathematics to the situation.

In order to compute a new fact from a given problem/solution pair, we need to create an assignment (φ) that maps the given information from the situation theory to the correct problem theory. Moreover, this assignment can be understood as specifying the input parameters to a mathematical function. As described in subsection 4.1 the user specifies this assignment with his or her interaction between the chosen scroll and the explored world. On the MMT side, this interaction is used to generate a view between the situation theory and the problem theory. In our example, this view needs to assign a symbol declaration from the situation theory to the corresponding symbol declaration in the problem theory. For instance, the points (A, B, C) in the situation theory are assigned to the respective points (a, b, c) in the problem theory. Obviously, the view between a problem theory and a situation theory is going to depend purely on which scroll a user chooses, as the assignment will be generated from this interaction.

From the generated situation theory, the appropriate problem/solution pair and a view between the situation theory and the problem theory, MMT is able to produce a corresponding **solution pushout**. A pushout is only successfully produced if the view between the situation theory and the problem theory is total and appropriate. For instance, if we generate the same view but the lines AB and BC are not perpendicular in our situation theory, then the problem/solution pair is not applicable, because the assignment fails to be a total view. Fortunately, the solution pushout is successful in our example and the resulting theory contains a new symbol declaration that defines the length of the line segment \overline{BC} . Lastly, the solution pushout theory is used to add the newly obtained facts to the list of facts in the game itself. These newly obtained facts can then be used by the user to either solve the problem at hand or as the inputs for further scroll applications.

4.3 Scrolls

Scrolls allow the user to discover new facts about the world. Thus, they form the primary link between the Game World and the logic engine. Given the need for scrolls, we would like to generate them from the predefined problem/solution pairs.

While the current form of problem/solution pairs works perfectly for our present implementation, for future implementations and for the generation of scrolls they need to be extended and refined. The goal behind this extension would be to include visualization theories that allow us to display the abstract problem and its solution effectively in the game as well as on the scroll itself. Such an extension of problem/solution pairs could look similar to Figure 4.

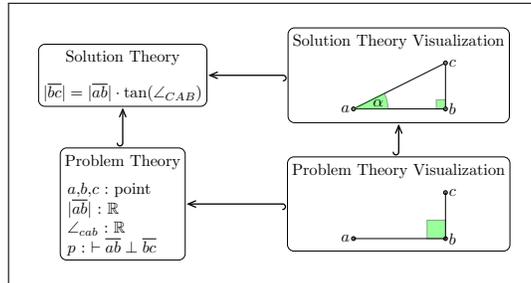


Fig. 4. Problem Solution Pair Extension

A scroll consists of two major parts. The first part of a scroll is a document that contains mathematical explanations in natural language in addition to diagrams that illustrate the concepts. This document could be generated non-dynamically by an extension of \LaTeX [Koh08] we tentatively call **ScrollTeX**. In the game itself this document should be displayed each time the user inspects the scroll. The second part of a scroll is a set of machine readable information that specifies exactly what kind of facts a scroll requires for its application and what kind of facts it is going to produce as an output. The OMDoc format [Koh06] might be the ideal format here as it allows us to store formal and informal information in a highly interrelated form and generate active mathematical documents [Koh+11].

The FramelT method supports compound problems directly. Each subproblem of a compound problem can be solved by discovering the required facts for the solution of the subproblems through successive scroll application and tool usage. The facts representing the solution of the subproblems can then be related to each other in exactly the same way, which in turn allows the user to solve the problem in its entirety. In this section we presented a simplified example, but it is not difficult to think of more advanced examples that require multiple scroll applications.

5 Design and Implementation

To evaluate the refined FramelT method, we designed and implemented a proof of concept serious math game that demonstrates our method. Moreover, this

first implementation allows us to critically assess the method and to discover any shortcomings.

5.1 MMT

On the MMT side we created the respective theories for the theory graph seen in Figure 3. The basis (meta-theory) of all of our theories is formed by a higher-order logic theory, which allows us to declare multiple different types and provides us with the basic logical quantifiers and semantics.

Fundamental Theories For the FramelT method we require a set of fundamental theories on which we can base the higher level theories such as the situation theory or the problem/solution pairs. Thus, we implemented a real number theory and an Euclidean geometry theory for our proof of concept implementation.

The geometry theory makes use of the real number theory to implement other types such as vectors and lines in similar fashion. Given these two theories we are able to form precise propositions about the world and the problems within it. Obviously, these fundamental theories are incomplete and in order for them to handle more complicated scenarios they need to be extended in the future.

Problem Solution Pairs The problem theory simply (see Figure 3) contains a list of propositions that are needed by the solution theory to solve the problem. The solution theory includes the problem theory and contains additional declarations that specify the solution to the abstract problem. In other words the problem theory specifies the "input" of a function while the solution theory specifies the "output". The problem and solution theory implementations can be seen in Figure 6. Although our implementation is based on the example presented in section 4, we deviate slightly from it as we are actually requiring the user to proof explicitly that the scroll is applicable. Hence, our problem/solution pair contains an additional declaration (an axiom), the angle between the ground and the tree, which needs to be provided by the user.

Situation Theories and Framing In order to determine the requirements for the situation theory and the view between the problem theory and the situation theory we implemented an example situation theory and view. The situation theory contains the observed facts and their values and the example view assigns every declaration in the problem theory a declaration from the situation theory.

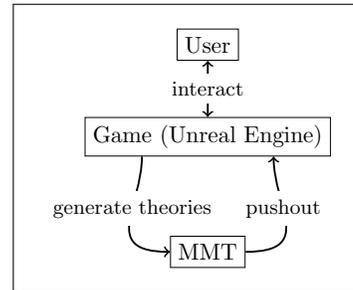


Fig. 5. System Architecture

```

namespace http://cds.omdoc.org/FramelT M

theory problem_theory : ?geometry =
  Pa : Vec3D D
  Pb : Vec3D D
  Pc : Vec3D D
  anglePaPbPc_value : ℝ D
  anglePaPbPc : ⊢ (∠ Pa Pb Pc) ≐ anglePaPbPc_value D
  anglePcPaPb_value : ℝ D
  anglePcPaPb : ⊢ (∠ Pc Pa Pb) ≐ anglePcPaPb_value D
  lineSegPaPb_value : ℝ D
  lineSegPaPb : ⊢ (lineSegmentLength Pa Pb) ≐ lineSegPaPb_value D
  proof : ⊢ ((⟨Pa, Pb⟩) ⊥ (⟨Pb, Pc⟩)) D M

theory solution_theory : ?geometry
  include = ?problem_theory D
  lineSegPbPc_value : ℝ O = (tan anglePcPaPb_value) * lineSegPaPb_value D
  lineSegPbPc : ⊢ (lineSegmentLength Pb Pc) ≐ lineSegPbPc_value D M

```

Fig. 6. MMT Problem and Solutions Theories

5.2 Game (Unreal Engine)

We only focus on the parts of the implementation relevant to the FramelT method and not on the auxiliary parts such as the creation of the landscape.

The problem exploration gadgets are implemented classes in the game engines based on as C++ classes for the different kinds of facts a user can discover. Each of these classes contains methods that can serialize that fact into the OMDoc format. Moreover, the game stores each of these facts in a list which we call the **fact list**. Whenever the game needs to produce a situation theory it iterates through this list to serialize each fact to the OMDoc format.

The most important of those tools is the *point marking tool* as the player can use it to create a point fact by simply marking a location. This has two different marking modes. The first mode allows the user to simply mark any location he or she points to, while the second mode can be used to mark specific spots on what we call semantic actors. Figure 7 shows the *point marking tool* in action: Semantic Actors are game objects that contain additional information and methods that allow them to interact in a more specific way with the user and his or her available tools. To illustrate, the tree in our example problem is a semantic actor, which allows the user to mark exactly the bottom and top of the tree with his or her *point marking tool*. If the tree would not be a semantic actor then marking exactly the top and bottom of the tree would be incredible

difficult. Fortunately, it is a semantic actor which allows the user to select the points by simply marking a point close to the actual top or bottom of the tree.

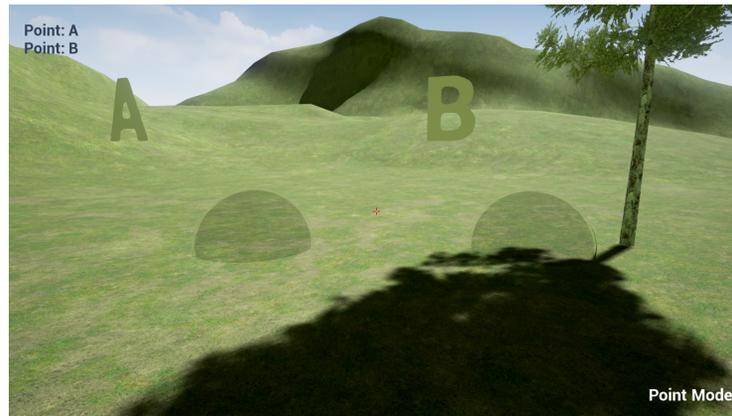


Fig. 7. Point Marking Tool

The measurements obtained with the *distance measuring tool* are in Unreal Units, which is the basic unit of length in the Unreal Engine. One Unreal Unit is equivalent to one centimeter. All of the facts the user discovers by using all of his or her tools are displayed as a list in the top left hand corner of the game.

As one can see in Figure 8 by measuring facts about the world the user's fact list grows gradually.

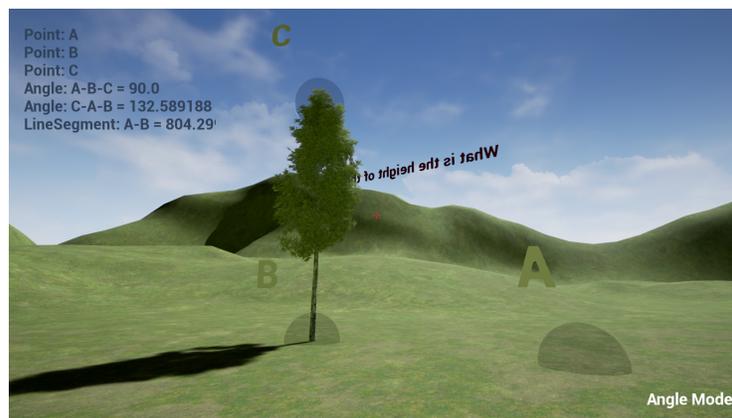


Fig. 8. Measuring Facts about the World

After measuring all facts the user can enter the *View Mode* in which they assign for each fact required by the scroll a fact from the fact list. During this assignment process a mapping between scroll facts and facts from the user's fact list is created. After the assignment is completed the mapping is serialized into OMDoc. At the same time the user's fact list is serialized into an OMDoc situation theory. Next, the generated OMDoc is passed on to MMT, which takes these theories and uses them to compute a solution pushout. If the pushout was successful then MMT produces a new theory that is subsequently parsed by the game. Through parsing this theory the necessary information needed to create the resulting facts is extracted and used to populate the fact list with the newly obtained facts. In case the pushout was unsuccessful an error message is displayed. The result of a successful pushout can be seen in Figure 9. In this specific case the user has now determined that the height of this tree is 875 centimeters and thus the user has successfully solved the problem.

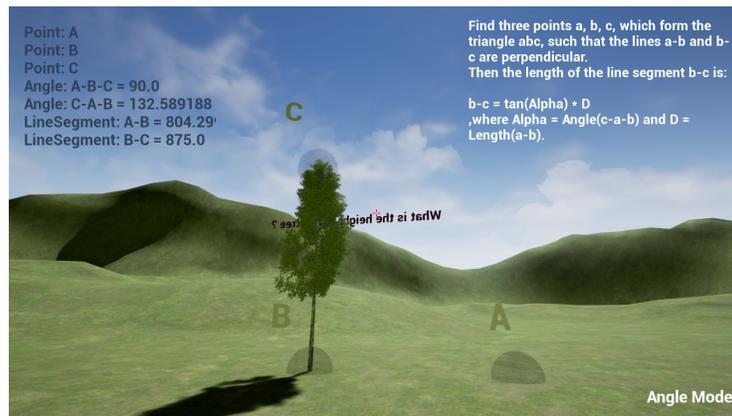


Fig. 9. Scroll Result

6 Conclusion and Future Work

The proof of concept implementation and our experiment the method shows that it is indeed possible to use mathematical knowledge management techniques (MKM) to map a real world scenario to its theoretical basis. This suggests that the FramelT method can be used as a framework for creating full-fledged serious math games.

Separation of Necessary Skills Crucially, the framework keeps many parts of a serious game generic. As a consequence, the development of new games (and extensions to existing ones) naturally separates into three independent tasks: the development of *i*) didactic game situations, *ii*) learning objects (real-world

problems, problem-solutions pairs and base theories) as mathematical theories in OMDoc/MMT, and finally *iii*) semantic actors and exploration gadgets. The first is a high-level, transdisciplinary design task that results in the storyboard for a serious game. For the implementation *ii*) only requires the skills of math educators trained in OMDoc/MMT formalization, while *iii* only needs regular game realization skills. In particular, the mathematics didactics and games realisation tasks are well-separated by the FramelT framework, which promises to greatly simplify the realization phase. Based on our experience, realizing another “word problem-type” game problem is the work of one or two days for an individual trained in OMDoc/MMT formalization and we estimate the development of a new, non-trivial semantic actor or gadget to be in the same range. Of course this greatly depends on the availability of foundational theories and game components; fortunately, these are developments that can be shared and accumulated by an open community.

Limitations While we can already demonstrate the FramelT method, we are still missing some integral parts such as the automatic generation of scrolls as discussed in section 4.3. Furthermore, we are currently not generating detailed explanatory texts in case MMT is unable to produce a pushout or the user was unsuccessful in solving the problem. Future implementations should address this by providing the user with helpful hints and error messages that allow the user to learn from his or her mistakes. Lastly, we would like to allow the user to be able to use his or her knowledge directly. For instance, a user should be able to create his or her own scrolls in the game by providing a proof that his or her envisioned scroll is mathematically sound.

Finally, the current lack of content does not yet allow us to demonstrate the ability of the FramelT method to support compound problems. But the fact that scroll-based fact extension is essentially the same as theorem proving – which is one of the activities the MMT tool was designed for suggests that this should be very easy to achieve.

Availability & Future work Future work should also focus on the gamification aspects and the user interface of the game itself, as the current interaction between the user and the game is not ideal. For instance, the interface for assigning views is not optimal as we are selecting facts from a list of facts instead of selecting them directly in the game world. Hence, we would like to improve the user interface to make the game more intuitive and attractive for the user. Furthermore, we are currently not visualizing facts about angles or distances in the game world. Utilizing the results of human computer interaction research could allow us to resolve these issues effectively and thereby increase the usability of our serious math game.

While our implementation successfully connects MMT and the Unreal Engine, there still exists a plethora of software engineering challenges to make transferring knowledge between the system smoother and more efficient. Integrating MMT directly into the Unreal Engine could solve many of these challenges and

improve the performance of the system. Additionally, it would allow other serious game developers to use the services provided by MMT directly. Such an integration could use the Semantic Alliance framework [Dav+12] as its basis and adapt it for the Unreal Engine.

The current implementation is released under the MIT License and is available at <https://github.com/KWARC/FrameIT>, while the OMDoc/MMT content can be found at <https://gl.mathhub.info/groups/FrameIT>. As a first step we want to stabilize the system and extend the content (game situations and OMDoc/MMT formalizations) to a point, where we can start designing coherent serious games from the collection of problems.

No Evaluation Yet The work reported on in this paper only constitutes of proof-of-concept for the FrameIT method. In particular, the system is not currently in a shape that we could qualitatively and quantitatively analyze the usefulness and usability of the developed serious math game, which was the motivation of the FrameIT method in the first place.

Acknowledgements The development of the FrameIT method has profited from discussions in the KWARC group at Jacobs University, in particular from contributions by Mihnea Iancu and Andrea Kohlhase.

References

- [Dav+12] Catalin David et al. “Semantic Alliance: A Framework for Semantic Allies”. In: *Intelligent Computer Mathematics*. Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 49–64. URL: <http://kwarc.info/kohlhase/papers/mkm12-SAlly.pdf>.
- [Det+11] Sebastian Deterding et al. “From Game Design Elements to Gamefulness: Defining ”Gamification””. In: *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. MindTrek ’11. Tampere, Finland: ACM, 2011, pp. 9–15. DOI: [10.1145/2181037.2181040](https://doi.org/10.1145/2181037.2181040).
- [GM08] George Gogvadze and Erica Melis. “Feedback in ActiveMath Exercises”. In: *International Conference on Mathematics Education (ICME)*. 2008.
- [Inc16] Epic Games Inc. *Unreal Engine 4*. <https://www.unrealengine.com/>. 2016. URL: <https://www.unrealengine.com/> (visited on 02/27/2016).
- [KK12] Andrea Kohlhase and Michael Kohlhase. “Frames: Active Examples for Technical Documents”. 2012. URL: <http://kwarc.info/kohlhase/submit/activeex-2012.pdf>.

- [Koh+11] Michael Kohlhase et al. “The Planetary System: Web 3.0 & Active Documents for STEM”. In: *Procedia Computer Science* 4 (2011): *Special issue: Proceedings of the International Conference on Computational Science (ICCS)*. Ed. by Mitsuhsa Sato et al. Finalist at the Executable Paper Grand Challenge, pp. 598–607. DOI: [10.1016/j.procs.2011.04.063](https://doi.org/10.1016/j.procs.2011.04.063).
- [Koh06] Michael Kohlhase. *OMDOC – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [Koh08] Michael Kohlhase. “Using L^AT_EX as a Semantic Markup Format”. In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: <https://svn.kwarc.info/repos/stex/doc/mcs08/stex.pdf>.
- [Koh14] Michael Kohlhase. “Mathematical Knowledge Management: Transcending the One-Brain-Barrier with Theory Graphs”. In: *EMS Newsletter* (June 2014), pp. 22–27. URL: <http://www.ems-ph.org/journals/newsletter/pdf/2014-06-92.pdf>.
- [Rab13] Florian Rabe. “The MMT API: A Generic MKM System”. In: *CoRR* abs/1306.3199 (2013). URL: <http://arxiv.org/abs/1306.3199>.
- [Rab15] Florian Rabe. “Theory Expressions (a Survey)”. submitted. 2015.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <http://kwarc.info/frabe/Research/mmt.pdf>.
- [Roc16] Denis Rochau. “FrameIT Reloaded: Serious Math Games from Logic”. B. Sc. Thesis. Jacobs University Bremen, 2016.
- [Zyd05] M. Zyda. “From visual simulation to virtual reality to games”. In: *Computer* 38.9 (2005), pp. 25–32. DOI: [10.1109/MC.2005.297](https://doi.org/10.1109/MC.2005.297).