T2Ku: Building a Semantic Wiki of Mathematics

Minqi Pan[*]

Undergraduate Student at School of Mathematical Sciences,
Capital Normal University, 100048 Beijing, PR China
`pmq2001@gmail.com`

Abstract. We introduce T2Ku, an open source project that aims at building a semantic wiki of mathematics featuring automated reasoning(AR) techniques. We want to utilize AR techniques in a way that truly helps mathematical researchers solve problems in the real world, instead of building another ambitious yet useless system. By setting this as our objective, we exploit pragmatic design decisions that have proven feasible in other projects, while still employs a loosely coupled architecture to allow better inference programs to be integrated in the future. In this paper, we state the motivations and examine state-of-the-art systems, why we are not satisfied with those systems and how we are going to improve. We then describe our architecture and the way we implemented the system. We present examples showing how to use its facilities. T2Ku is an on-going project. We conclude this paper by summarizing the development progress and encouraging the reader to join the project.

Keywords: semantic wiki, automated deduction systems, mathematical knowledge management

1   Motivations

The proliferation of mathematical knowledge is literally exploding, following its own version of Moore's law[8, Preface]. Nowadays, when doing researches in a particular mathematical field, we are often faced with the following questions. Is there in the existing mathematical publications a proof of the proposition that I'm working on? Can this proposition be easily deduced from the work already done by other mathematicians? How do I find pertinent theory to my research at hand in order to raise the initial height of my work?

Take an exercise from an algebra textbook [17, Sec 4.1] as an example.

Proposition 1. *Suppose that F is a perfect field with characteristic $p > 0$, E/F is an algebraic extension. Prove that E is also a perfect field.*

If the student were given this exercise to work out without any context of the book, it would be a very difficult proposition to prove. However, if the student can observe the following proven theorem from the book,

---

Theorem 1. *If F is a perfect field with characteristic $p > 0$, $E = F(\alpha)$ is a simple algebraic extension. Then E is a perfect field.*

Then the student could take this as a lemma, and work out the exercise with very little effort.

Proof. Pick $\alpha \in E$, then $F(\alpha)$ is a simple algebraic extension since E/F is an algebraic extension. Thus by lemma, $F(\alpha)$ is a perfect field and therefore α is a $p-$th power. By definition, we conclude that E is a perfect field.

Therefore, answering the pre-mentioned questions are very important, especially in mathematical problem solving. Of course, a solid mathematical education background could ensure the researcher of a nice grasp of the common knowledge of his/her researching field, but only to a limited extent. With the ongoing emergence of great quantity of latest mathematical knowledge, the education cost and time span could be huge.

We find that a digitized way to manage and query the current mathematical knowledge to be indispensable. We wish to employ the current information technologies to foster a common system for mathematical researchers to easily seize the latest proven mathematical facts, and use them to boost their own research. With that in mind, we started the T2Ku project.

2   The Goal

In one word, we want to build a semantic mathematical wiki, with a user-friendly Web interface, that supports the following inquiry. When the user gives an input describing a particular mathematical proposition $\mathcal{P}$, the system searches for pertinent mathematical facts, and try to use them to deduce $\mathcal{P}$. If the deduction failed, the system gives out pertinent mathematical facts for the user to consult. The system will also inform the user when $\mathcal{P}$ is found inconsistent with the known facts. Otherwise, the system gives out the outline of the proof.

By setting this as our goal, we found ourself dipping into two academic fields simultaneously. One field is automated reasoning, we have to find a way to take use of the existing automatic inference power to best implement the proof-searching process. The other is mathematical knowledge management. We have to find an effective way to construct and manage the knowledge base.

3   State of the Art

It couldn't be us alone who have come up with this idea. Before we commence our work, we must investigate existing systems that has similar goals. We find it useful to group those systems into two categories.

3.1   Non-Semantic Systems

We have observed that, from the people we met, most mathematicians use web platforms like Google Scholar, SpringerLink and CNKI[7] as their daily tools to look for relavent mathematical publications. These platforms do help researchers get what they want by presenting them with textual contents that match the keyword combinations that they have invented. Yet we believe that, this is far from the perfect way to query mathematical knowledges.

The main drawback of this category of systems is that they rely solely on plain-text search. It can be observed that, there exists an intrinsic logical relationship embodied in every mathematical proposition, which is immaterial to the actual text that presents the relationship. What those system do is to simply match against the textual presentation of this intrinsic relationship. However, the way of presentation varies. The first variation occurs when one chooses a particular natural language to write down the proposition sentence, which dictates different grammers and syntaxes. Every natural language additionally has a completely different set of mathematical terminologies. Even in the same natural language, we see different terminologies of the same mathematical concepts used in different literatures. Also, different authors have their distinct ways to utter the final sentence. Further more, when put on the web, mathematical formula has different ways to present. By embedded pictures, by MathML, by LaTeX, just to name a few.

Also, the non-semantic approach only support one-level-depth inference search. Suppose that we have a proposition $\mathcal{P}$ at hand, and we want to know if there exist any existing facts that imply $\mathcal{P}$. We would have to peel some keywords off $\mathcal{P}$ and search for it. In this way, we can only find propositions that has $\mathcal{P}$ as the direct conclusion. Deeper inquiry requires further human deliberations.

3.2   Semantic Systems

Seeing all the disadvantages of the non-semantic systems, we tend to believe that, it would be perfect if all mathematical queries are done at the semantic level, eliminating all the vagueness and insecurities. However, this idea entails a mathematical library to be built at the semantic level too. Thus the year 1994 have seen a publication of the QED manifesto [1], where a proposal for a computer-based database of all mathematical knowledge have been made. Also, several semantic based systems emerge, like Mizar Mathematical Library(MML) [12], MoWGLI[9], C-CoRN[3], etc.

It is the Mizar Mathematical Library[12] that draws most of our attention. MML record formal mathematics using a formal language called Mizar, by which the library achieves the formalization of 10013 definitions and 51223 theorems upon the release of version 4.166.1132 (28 Jun 2011)[11]. We believe that it would be irresponsible not to take use of a formal library of this size, abandoning all the human hours previous researchers have spent to make it available. We therefore begin studying and experiementing with this library.

We found that MML has a online query interface, the usage of which dictates mastering a query language called MML Query[2]. Yet we find most queries are considered with the system per se. For example,

```
list of article ordered by processing order select 0-29
```

queries for the latest 30 MML articles. Yet this is not what we want since it lacks inference abilities. It was then when we discovered another project called MPTP[13] that is built upon MML, which tried combining the power of automated theorem provers(ATP) with the library.

We then decided to build our system also on top of MML, letting the user to enter propositions in the Mizar language, and utilize the MPTP to translate the Mizar proposition into the TPTP format[10], which is a third-party language that can be easily translated into specific ATP input formats. Finally we do the translation and feed the input into multiple ATP programs to try getting the proposition proved.

As an example of our initial experiments:

Proposition 2. *Let G be a group. Suppose that $x * x = e$ for all $x \in G$. Prove that G is commutative.*

The corresponding Mizar-language version of this proposition is:

```
for G being Group holds
(for x being Element of G holds x * x= 1_G) implies G is commutative;
```

We then prepare the minimal header references:

```
environ
  vocabularies GROUP_1,SUBSET_1,BINOP_1,RELAT_1;
  notations STRUCT_0, ALGSTR_0,GROUP_1;
  constructors STRUCT_0, ALGSTR_0,GROUP_1;
```

Together the two combined could results in the "mizf" command of MML to only return "*4" errors, which means only the proof part is absent (cf.[4][2.2.2]). This is the exact moment when MPTP could translate it into an ATP problem:

```
fof(t1_mtest1, conjecture, (! [A] :
  ( ( ~ (v2_struct_0(A)) & (v2_group_1(A) &
      (v3_group_1(A) & l3_algstr_0(A)) ) ) =>
      ( (! [B] : (m1_subset_1(B, u1_struct_0(A)) =>
      k6_algstr_0(A, B, B)=k1_group_1(A)) ) => v5_group_1(A)) ) ) ).
...
```

We omit the rest of the whole output, since it is tedious and inaccessible to human readers. We now feed it to the ATP to solve with a 20s time limit:

```
Time Out
```

However, is this problem really that hard for ATP's? Here is another presentation of the same problem(c.f. TPTP Problem[10] GRP001-1):

```
include('Axioms/GRP003-0.ax').
cnf(square_element,hypothesis, ( product(X,X,identity) )).
cnf(a_times_b_is_c,negated_conjecture, ( product(a,b,c) )).
cnf(prove_b_times_a_is_c,negated_conjecture, ( ~ product(b,a,c) )).
```

The ATP could terminate with a proof in no time.

```
----> UNIT CONFLICT at 0.00 sec
----> 44 [binary,43.1,4.1] $F. Length of proof is 4. Level of proof is 3.
```

The reason why our translated ATP problem timed out is simple, the translator simply adds all the relavent mathematical facts into the problem from the MML, resulting in an explosion of inference results when the ATP tried solving it using refutation procedures. So we start optimizing this translator, making it more sophisticated to produce more solvable ATP problems.

It was then when we realized that MML is not for us.

4   Reflections

MML is over-designed for proof searching. MML emphasizes greatly on the logical soundness of its formalized content, resulting in a greatly complicated structure of the library, containing constructs that has no correspondence in ordinary mathematics ('multMagma', for instance). Also, the relations between Mizar articles are complicated, the header preparing process is no easy task.

As [14] have pointed, those formalizations make mathematical proofs more like computer programs, less like mathematics, which is unfriendly to most mathematicians. As a result, Mizar is popular only in the academia. Another reason why it did not gain its popularity is that its content lacks connection with real-world mathematical publications and thus is inaccessible to average users.

We also have to admit that, despite the gratifying development of automated theorem proving techniques in the last half century, most real world mathematical problems are still too difficult for a computer program to solve. If we were to make a servicable system, we have to put our expectations at a realistic level. The lack of creativity makes computer programs only possibly proficient at routine problems, where only a simple reference or brute-force search is required to obtain a solution.

5   The T2Ku Archetecture

Instead of recording mathematical knowledge directly using a formal language, we on the other hand record them with real-world mathematical literatures $\mathcal{L}$, and then *annotate* them with formal contents $\mathcal{F}$. Most users interact with the system using $\mathcal{L}$, the system works internally using $\mathcal{F}$, and present the results to the user using $\mathcal{L}$ again. Average users never interact with $\mathcal{F}$.

$\mathcal{L}$ includes books, articles, theses, etc. They are organized according to real-world mathematical publications, recorded with metadata like authors that correspond to real people, also with data that records their full-text contents.

5.1   The Annotation

With the $\mathcal{L}$ part alone, T2Ku would be very much like an amalgam of Wikipedia and Google Scholar. It's the $\mathcal{F}$ that distinguishes T2Ku, which adds semantic flavors to $\mathcal{L}$. We picked $\mathcal{F}lora - 2$ as the infrastructure for $\mathcal{F}$, which is a object-oriented knowledge representation language that is based upon XSB-implemented Prolog[16]. For example,

Example 1. The proposition "Let P be a nonabelian group of order 8. Then P is isomorphic either to the dihedral group D_8 or to the quaternion group Q_8." can be annotated with

```
either_true(isomorphic(?P,D_8),isomorphic(?P,Q_8)) :-
                                ?P:nonabelian_group[order->8].
```

All predicates and constants live in the same namespace. Cautions have to be made when creating new annotations, which is not to clash and effectively reference existing predicates and constants in order to construct a well connected knowledge graph. We provide useful query tools for editors to aid this process.

5.2   The Bridge

In order not to expose $\mathcal{F}$ to average users, we need a bridge to connect real-world mathematical expressions with the underlying $\mathcal{F}lora - 2$ expressions. Inspired by Cucumber[6], which is a framework that enables acceptance tests be written in natural languages and has proven useful in production projects, we use regular expressions and Ruby code as such a bridge.

Example 2. `Let /\d+ be an equivalence relation on \d+/ do |it,set|`
   `flora2("#{it}:EquivalenceRelation[base_set->#{set}].")`
`end`
   enables the parse of user input

```
Let $\sim$ be an equivalence relation on $S$.
```

to (with $...$ replaced by integers and then $\mathcal{F}lora - 2$ variables)

```
var_sim:EquivalenceRelation[base_set->var_S].
```

The reverse bridge is similar. Cautions have to be made not to generate parsing ambiguities when adding new bridges. The editor is responsible for providing parsing examples for his bridges. And when submitted, the system would try parsing those examples to look for ambiguities. The examples are crucial as it also serves as documents for average users to quickly find out expressions that the system can understand to prepare his input.

Yet ambiguities are hard to completely eliminate at edit-time. At run-time, the system would also warn the user when different ways of parsing is found and let the user to choose the intended one.
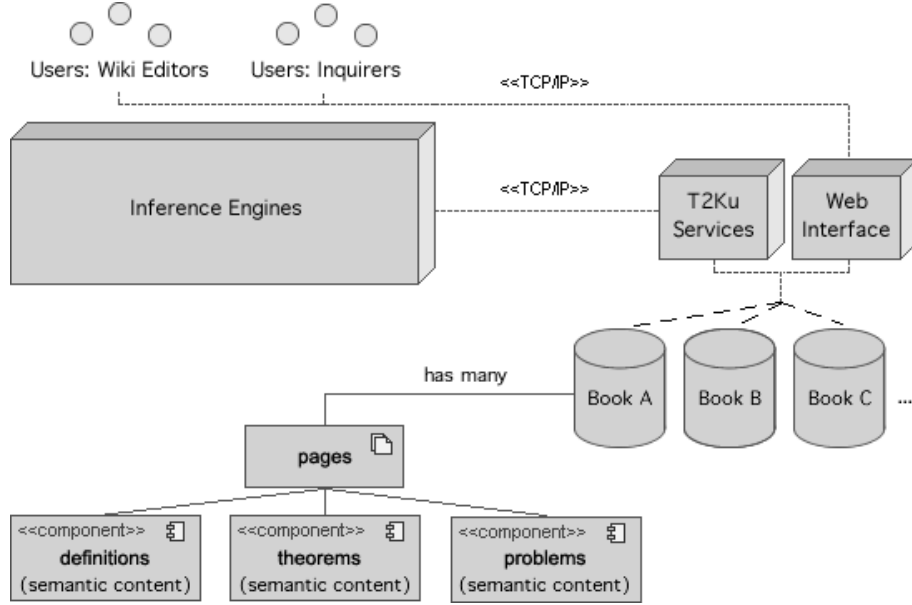
Fig. 1. The T2Ku Archetecture

5.3   T2Math

We designed a simple language for users to present propositions to the system. It is based on the following observation: a mathematical proposition contain no more than three parts, namely variable declarations, premises and conclusions[1].

Example 3. Let G be a group,
    e be the identity of G,
    $*$ be the binary operation of G.
Suppose that
    $x*x=e$ for all $x\in G$.
Prove that
    G is commutative.

We call this simple format T2Math, and have developed auto-hightlighting javascripts to boost the user experience when presenting propositions with it. Mathematical variables are required to be surrounded by dollar signs.

---

[1] Though in a more simplistic view, variable declarations can be viewed also part of premises, in which case a proposition is only composed of premises and conclusions, we are not going that far.

5.4   Inference Engines

Inference Engines are defined to be programs that reads the user input of T2Ku and tries proving the proposition and outputing other useful results.

As the reflection section mentioned, one simple program could hardly handle all the inference tasks. We decided to make T2Ku an "engine yard", making inference engines loosely coupled with the main system to allow combined powers of inference. As Figure 1 shows, inference engines live outside the T2Ku system and contacts with it via the TCP/IP protocol. T2Ku exposes the inference tasks and the knowledge base through a RESTful web service.

The inference engines are potentially remote machines that checks for new tasks with heartbeat requests. When a proving problem is created, potentially several inference engines are working on it at the same time. Yet rest assured, at least one inference engine live on the same intranet with T2Ku that is guaranteed to provide fast responses to user inputs. This is another important development task of the T2Ku project. At the current moment, we are working on utilizing XSB inference engine to provide a search engine for relevant mathematical facts.

This open archetecture allows professional users to register his inference engine with the system. It thus allows the latest development of the automated reasoning techniques to be integrated into T2Ku, making T2Ku an common experimental platform for automated reasoning programs.

5.5   Wrap It Up

Figure 1 depicts the overall structure of our design. We used the Ruby on Rails[5] framework to develop the Web layer, which unites all the above mentioned parts.

For the $\mathcal{L}$ part, we utilize git to handle the underlying version control of the books, fostering a wiki system that anyone could edit. When creating books and other publications, the editor can import meta-data easily from other web services. After that the editor could add pages. When creating pages, the user could specify a page's father page, creating a tree-structure of the book, after which the system generates the table of contents automatically.

For the $\mathcal{F}$ part, we provide code auto-highlighting, query tools and syntax checking facilities to make the editing process more convenient.

5.6   Copyright Issues

We shall only record publications that has written permissions of the copyright holders, yet T2Ku itself never owns the copyright of its content. For example, the copyright of the Graduate Studies in Mathematics textbook series is held by AMS, thus we have to contact AMS to gain permissions in order to reuse its contents, but AMS retains all rights that it previously held.

We have not yet succeeded in doing this, but hope exists as the T2Ku project itself is non-profit and helps popularize the publication and expands its reader groups. However, if this continued to fail, we would take another strategy that resembles Wikipedia[15], which goes by CC-BY-SA and GFDL license and allows

users to use any contents that are compatible with the licenses. Non-compatible contents would have to be reconstructed in order to be used.

6   Ongoing Development

The current status of this project can be inspected via

<div align="center"><code>http://www.t2ku.org</code>.</div>

The source code is hosted on Github and is available at

<div align="center"><code>https://github.com/t2ku/t2ku</code>.</div>

As of writing of this paper, we have only 1 people working on the code. Volunteers are highly solicited. Contributions to the source code are welcome and greatly appreciated.

References

1. Anonymous. The qed manifesto. *Lecture Notes in Artificial Intelligence*, 814:238–251, 1994. `http://www.cs.chalmers.se/Cs/Research/Logic/TypesSS05/Extra/wiedijk_2.pdf`.
2. Grzegorz Bancerek. Information retrieval and rendering with mml query. *Lecture Notes in Computer Science*, 4108:266–279, 2006.
3. C-CoRN. the coq constructive repository at nijmegen. `https://github.com/c-corn/corn`, 2011.
4. Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
5. David Heinemeier Hansson. Ruby on rails. `http://rubyonrails.org/`.
6. Aslak Hellesy. Cucumber - making bdd fun. `cukes.info/`, 2011.
7. China Academic Journals Electronic Publishing House. China knowledge resource integrated database. `http://www.cnki.net`.
8. Michael Kohlhase. *OMDoc – an open markup format for mathematical documents : (version 1.2)*. Berlin; New York; Springer, 2006.
9. MoWGLI. Mathematics on the web: Get it by logic and interfaces. `http://mowgli.cs.unibo.it/html_no_frames/home.html`, 2011.
10. Geoff Sutcliffe. The tptp world - infrastructure for automated reasoning. *Logic for Programming, Artificial Intelligence, and Reasoning*, 6355:1–12, 2010.
11. MML Query Transformation. Mml query. `http://mmlquery.mizar.org/`.
12. Andrzej Trybulec. Mizar mathematial library. `http://www.mizar.org`, 1973.
13. Josef Urban. Mptp 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 37:21–43, 2006. 10.1007/s10817-006-9032-3.
14. Freek Wiedijk. The qed manifesto revisited. *STUDIES IN LOGIC, GRAMMAR AND RHETORIC*, 10(23), 2007. `http://mizar.org/trybulec65/8.pdf`.
15. Wikipedia. Wikipedia:copyrights. `http://en.wikipedia.org/wiki/Wikipedia:Copyrights`, 2011.
16. Guizhen Yang and Michael Kifer. *FLORA-2: User's Manual*. Department of Computer Science, State University of New York at Stony Brook, July 2001.
17. Yingbo Zhang. *Fundamentals of Algebra (2)*. Beijing Normal University, 2010.