Semantic Search for Quantity Expressions
Guided Research Proposal

Tom Wiesing
Supervisor: Michael Kohlhase
Jacobs University, Bremen, Germany

March 2, 2015

[1] EdN:1

Abstract

In this proposal we describe how to introduce units to MathWebSearch. The aim of the project is build a complete semantics-aware system that searches a corpus of documents for quantity expressions. The project will be based on the existing MathWebSearch system and related technologies. [2] EdN:2

Contents

1 Introduction

In this paper we want to give an approach to Semantic Search for Quantity Expressions. A quantity expression is a scalar together with a physical unit, for example $25\frac{\mathrm{m}}{\mathrm{s}}$ where 25 is the scalar and $\frac{\mathrm{m}}{\mathrm{s}}$ the unit. This quantity expression is equivalent to $90\frac{\mathrm{km}}{\mathrm{h}}$ (with equivalent in this sense meaning it expresses the same quantity) although the units are different. In a semantic search for quantity expressions we want to be able to search for a certain quantity expression and find any equivalent ones as well.

[1] EdNote: Find a Math Co-supervisor
[2] EdNote: Spin

MathWebSearch (MWS for short) is an existing semantics-aware system to search LaTeX documents [1] for mathematical formulae [HKP14]. As it is semantics-aware it not only searches for formulae in a simple-minded "text search" way but also includes simple transformation rules, such as $a + b = b + a$. Additionally it can deal with wildcards such as $x + \sqrt{x}$. For this query MWS would deliver forumlae as above where x has been substituted with any sub-formula.

So far the transformation system has been used by MWS exclusively for mathmatical forumlae. In this paper we propose an extension for quantity expressions which should prove useful for physicists. The end user will search, for example, 100 °C and also get results which show 212°F or 373.15K.

This proposal is organised as follows: In section 2 we shortly describe the existing MathWebSearch system. We continue in section 3 with a list of work packages and a schedule. Then we conclude in section 4 with describing our approach to building the system components.

2 An Overview of the existing MathWebSearch system

2.1 The Components of MathWebSearch

As mentioned above, MathWebSearch is a search engine for forumlae. MWS consists of 3 main components as well as a frontend [2] [KP].

The backend consists of 3 main components,

1. a crawler,
2. a core system and
3. a public rest API.

The crawler, as its name suggests, crawls corpera for forumlae. For each corpus MWS uses, a seperate crawler has to be implemented. The crawled formulae are passed to the core system and indexed in an MWS Harvest. The core system is also responsible for parsing queries and sending results back to the REST API. This is done by searching the harvest only. In order to make a semantic search for quantity expressions we will adapt this crawler to find and harvest quantity expression instead.

Because MWS is semantics-aware, the harvest can not only contain the exact formulars that are found in the original corpus but also all versions that are quivalent to it. These are generated with the help of MMT and theory graphs, see sections 2.2 and 2.3

The frontend for MathWebSearch, which is not part of MWS itself but running client-side in a web browser, is written in HTML5, CSS and JavaScript. It accesses the REST backend and depends on MathML support to render Mathematics. When the client enters a LaTeX formular to search for, the LaTeXml daemon [Gin] is used to transform the query in content MathML. Next, the client renders the MathML (to show the formular the user is searching for) and then sends the query off to the MWS API. Upon receiving results, the client renders them and links to the original documents.

[1] Technically, MWS itself can only search XHTML documents. However with the help of LaTeXml [Mil] it also handles LaTeX documents.

[2] The frontend is not part of MWS directly but rather built on top of the REST API, more on this later.

There are several implementations of frontends and crawlers as well as extensions of MathWebSearch. One particular implementation is capabale of crawling the arXMLiv corpus, which contains approximatly 750.000 documents. A list of demos can be found at [Mat].

2.2 Mathematical Theory Modeling with MMT

MMT is a **M**odule system for **M**athematical **T**heories [RK13]. In MMT we can represent mathematical theories, views and imports.

A mathematical theory in MMT is a set of typed symbols and statements. As a simple example, the theory of semigroups can be written down as following:

Semigroup		
G		
$\circ$	:	$G \to G \to G$
assoc	:	$(x \circ y) \circ z = x \circ (y \circ z)$

In the example above [3], we first define the a symbol G (the set of the Semigoup), as well as a (binary) operation $\circ$ and then the law of associativity for the operation $\circ$. EdN:3

Theories in MMT can be related in two ways, with imports and views. If theory A imports theory B then theory A contains all symbols and statements from theory B. In particular, every satisfiable statement that can be made in theory A is also a satisfiable statement in theory B.

As an example of an import, the theory of a *commutative groups* imports the theory of *groups*. The *commutative group* theory additionally has the axiom of commutativity. We can consider an import as a type of inheritance relation.

In contrast to this a view is a type of interpretation relation. For example, there is a view from the theory of *Monoids* to the theory of *Natural numbers*. Again, a satisfiable statement in the viewed theory can be translated into a satisfiable statement in the target theory.

Thus a theorem in one theory can be translated along views as well as along imports. For example, given a theorem in the theory of groups, we automatically get a theorem in the theory of commutative groups.

2.3 Using Theory Graphs

The theories and their relations can be represented with the help of a graph. This graph is then called theory graph. In figure 1 we can see a simple example of a theory graph.

Every theory in MMT has a (globally unique) URL, called the MMT URL [4]. Furthermore, every theorem in MMT can also be represented via a URI. As explained above, if 2 theories are related via a view or import, theorems can be translated along this relation. EdN:4

Sometimes it is useful not to write down a theorem explicitly, but only give an existing theorem and translate this along a view or import. The MMT URL of this induced theorem can then be given using the view[5]. This URL contains enough information for MMT to generate the theorem in explicit form [IKP14]. EdN:5

In an MWS Harvest we exploit this property of induced theorems. We have several represented theorems, the forumlae in the corpus, and many more induced theorems, other

[3] EdNote: Check with miancu if the example is correct this way
[4] EdNote: Example URL here
[5] EdNote: Example URI here

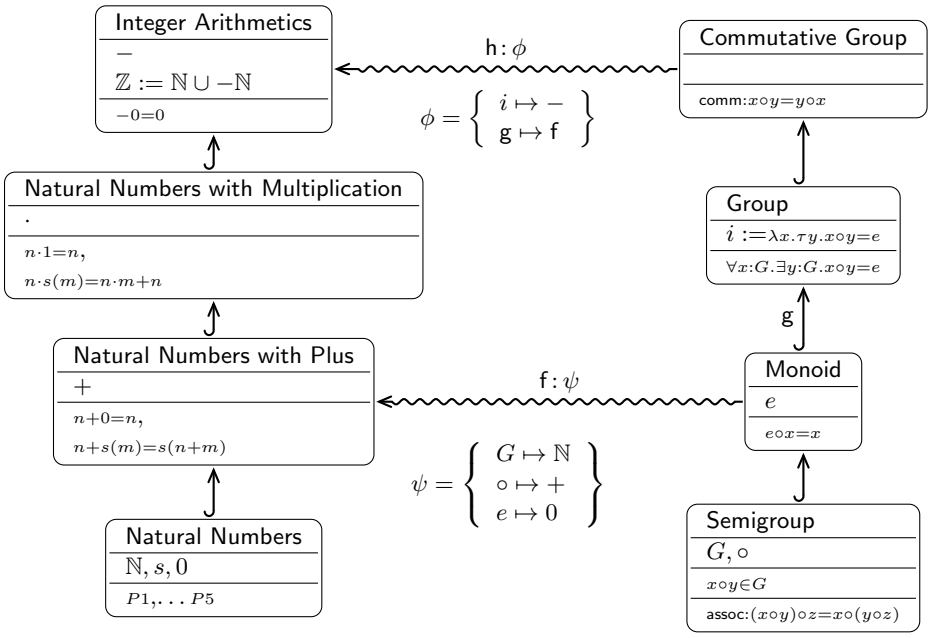

Figure 1: A simple theory graph. Imports are represented as solid edges and views as dashed edges.

equivalent representations of these formulae. With the help of MMT, the MMT Harvest is built by representing all theses theories (forumlae) explictly.

Applying this principle to quantity expressions, we can consider different units as different theories and different quantity expressions different theorems belonging to their respective unit theories. The unit conversions can then be represented via views. The information on how to translate from one unit to another will be contained within the view. If we know all formulae in a corpus we can then generate an MWS Harvest that contains all the representations. [6] EdN:6

3 Schedule and Work Packages

[7] [8] EdN:7

Because we want to build a complete system based on MathWebSearch that can perform a semantic search of quanity expressions on a specific corpus we need 4 major components: (1) *A crawler* that crawls documents for quantity expressions, (2) *a core system* that transforms these quantity expressions and harvests them as described above as well as provides search functionality, (3) *a rest API* that exposes this search functionality and (4) *a frontend* that servers as the endusers[9] interface to the system. EdN:8 EdN:9

This results in several work packages:

1. **A corpus** that is needed for a demo to work.

2. **A formula spotter** that crawls the corpus and finds formulae.

[6] EdNote: Should we move this paragraph to the next section?
[7] EdNote: Think of a better section title?
[8] EdNote: Subsections?
[9] EdNote: Spelling?

3. **A unit theories graph** as described above that can be used to transform units.
4. **A unit system** that can be used by the API and the frontend.
5. **A harvester** which uses the theory graph to perform an MWS Harvest of the corpus
6. **An API** that understands the unit system
7. **A frontend** that works together with the API.

We will start by taking on work package 3 and building a small unit theories graph. We can then develop a unit system (work package 4). Once that is done we will be able to build a harvester (work package 5).

Although we need work packages 1 and 2 to properly test these components, work package 1 will be postponed for now. Work package 2 will be taken on by Stiv Sherko in a seperate effort [She14]. Work package 1 will be postponed as well.

After the harvester is completed we can proceed with writing an API (work package 6) and the frontend (work package 7). Finally we can return to work packages 1-3 and expand the corpus and unit theories.

This results in the following timeline [10]: EdN:10

- **Week 1** - Work Package 3 - Building a small unit theories graph
- **Week 2** - Work Package 4 - Developing a unit system
- **Weeks 3 & 4** - Work Package 5 - Building a harvester
- **Weeks 4 & 5** - Work Package 6 & 7 - Building an API and a frontend
- **Weeks 6 & 7** - Work Packages 1, 2 & 3 - Finishing up the unit theories graph and integrating a corpus

4 A Semantic Search For Quantity Expressions Based On MathWebSearch

[11] EdN:11

[12] EdN:12

We will build the frontend using HTML, CSS and JavaScript. Because it serves as the main interface to the system

Furthermore the system will be flexible with respect to

- *the quanity expressions it understands,* Adding new units should be a simple process so that the system can be easily extended to also capture rare units and
- *the corpus* which should be exchangable easily.

The crawler will be based on existing MWS crawlers. As it is very difficult to find and mark up all units in a corpus automatically, we will restrict ourself to an artificial corpus

containing only lorem ipsum [13]documents with randomly inserted units. This does not solve the problem of writing a crawler, however it will enable us to build a proof-of-concept system first. EdN:13

The core system will almost completly be inherited from existing implementations. As it does not search formulae, it has to be made aware of unit translations. This will be done with the help of theory graphs. It is not yet clear how exactly these will look, a speculation can be found below in [14]. EdN:14

The rest-based API will have to take the user input from the client and pass the entire quantity expression on to the core system. Once it receives a result, it will have to translate these back into human-readable form and then pass these on to the frontend.

As with the existing system, the frontend will be a web page that incorperates 3 main elements, (1) a search input for a value (2) a search input for a unit and (3) a result page that displays results and links to the found documents.

[15] EdN:15

A bigger challenge with the approach will be to find a standardised representation for quantity expressions, mainly for units[3] [16]. EdN:16

Because we want to translate between these representations however, it is insufficent to just have such a representation. We additionally need to find a standardised way of translating between units need to exist as well. Because MWS uses Theory Graphs and MMT to translate between equivalent formulae, we will have to write down all units in the form of a theory graph. Translations between units will be represented as views.

To translate between units we could use simple translation formulae such has $x\mathrm{K} = x{+}273.15°C$. However because we want to support composite units (such as meters per second $\frac{\mathrm{m}}{\mathrm{s}}$) as well, this can cause problems. In machine representation [17], every quantity expression has to have a finite precision representation. When translating between 2 composite units, this can easily cause rounding errors. In particular, when an author gives an approximation of a quantity in a document, they might round differently depending on the units used. EdN:17

Hence it will not be sufficient to use exact translations. Thus we should search for a range of values instead. This has recently been implemented as an extension to MathWebSearch [Ham]. It remains to be seen how exactly these ranges should be used.

Furthermore the units will have to be entered in some fashion [18] in the frontend. While it is trivial to design an interface where a single unit can be entered, it is non-trivial when we want to recognise composite units as well. Furthermore units with (si-)prefixes (such as kilo or mega) can either be recognised seperatly or as part of the unit (multipling directly into the value of the quantity expression). There are several input formats that can be used: AsciiMath, LaTeX and MathML, to name only a few. Independent of the input format, the end result (delivered to the backend of the search engine) will have to be represented in one EdN:18

[10] EdNote: Replace week numbers with actual dates
[11] EdNote: More sub
[12] EdNote: Is the section title correct?
[13] EdNote: What is my contribution, the exact details are not important.
[14] EdNote: Fix reference
[15] EdNote: Next section continues here
[3] Here, standardised means that all quantity expressions are represented in a standardised, machine-readable fashion, not equivalent quantity expressions being representeted in exactly one way.
[16] EdNote: MathML note OM CDs
[17] EdNote: Seperate problem
[18] EdNote: The same?

way.

References

[Gin] Deyan Ginev. TheLaTeXml daemon: Editable math for the collaborative web.

[Ham] Radu Hambasan. Ranges in MathWebSearch. private communication.

[HKP14] Radu Hambasan, Michael Kohlhase, and Corneliu Prodescu. MathWebSearch at NTCIR-11. In Noriko Kando and Hideo Joho andKazuaki Kishida, editors, *NTCIR 11 Conference*, pages 114–119, Tokyo, Japan, 2014. NII, Tokyo.

[IKP14] Mihnea Iancu, Michael Kohlhase, and Corneliu-Claudiu Prodescu. Representing, archiving, and searching the space of mathematical knowledge. In Hoon Hong and Chee Yap, editors, *Mathematical Software - ICMS 2014 - 4th International Congress*, volume 8592 of *Lecture Notes in Computer Science*, pages 26–30. Springer, 2014.

[KP] Michael Kohlhase and Corneliu Prodescu. MathWebSearch manual. Web manual, Jacobs University.

[Mat] Math WebSearch a semantic search engine. web page at http://search.mathweb.org. seen September 2008.

[Mil] Bruce Miller. LaTeXML: A LaTeX to XML converter.

[RK13] Florian Rabe and Michael Kohlhase. A scalable module system. *Information & Computation*, 0(230):1–54, 2013.

[She14] Stiv Sherko. Extracting quantity and unit semantics from technical documents. Bachelor thesis proposal, Computer Science, Jacobs University, Bremen, 2014.