Semantic Search for Quantity Expressions
Bachelor Thesis PROPOSAL / FIRST DRAFT

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

March 4, 2015

Abstract

In this proposal we describe how to introduce units to MathWebSearch. The aim of the project is build an extensible 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.

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 semantically 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.

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 it can deal with wildcards such as

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

$x + \sqrt{x}$. 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 in Section 3 with describing some of the approaches we will use and conclude in Section 4 with a list of work packages and a schedule.

2 An Overview of the existing MathWebSearch system

2.1 The Components of MathWebSearch

As mentioned above, MathWebSearch is a search engine for forumlae and key phrases. 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 query to search for, the LaTeXml daemon [Gin] is used to transform the query into 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.

There are several implementations of frontends and crawlers as well as extensions of MathWebSearch. One particular implementation is capable 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.

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

A mathematical theory in MMT is a set of typed symbols. Via the Curry–Howard isomorphism theorms are simply constants inside a theory. If a theorems type is non-empty, is is proveable and the definition is a representation of a proof.

As a simple example, the theory of semigroups can be written down as following:

Semigroup		
G	:	type
$\circ$	:	$G \to G \to G$
assoc	:	$\text{ded}\,(\forall x \forall y \forall z (x \circ y) \circ z = x \circ (y \circ z))$

In the example above, we first define the a type G (the base set of the Semigoup). Next we define a (binary) operation $\circ$ and then the law of associativity for the operation $\circ$ on G.

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 source 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 as a graph, called a theory graph. In Figure 1 we can see a simple example of a theory graph.

Every theory and every constant in MMT has a (globally unique) URI, called the MMT URI. It can be constructed via a triple (G, M, S) where G is a document URI, M is a module name within this document and S is the name of a constant [RK13]. We then seperate these components via a ? to get a URI of the form $G?M?S$. Because every theorem in MMT is only a symbol declaration it can also be represented via a URI. As explained above, if two theories are related via a view or import, theorems can be translated along this relation.

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 URI of this induced theorem can then be given using the view. This URI contains enough information for MMT to generate the theorem in explicit form [IKP14].

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 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.

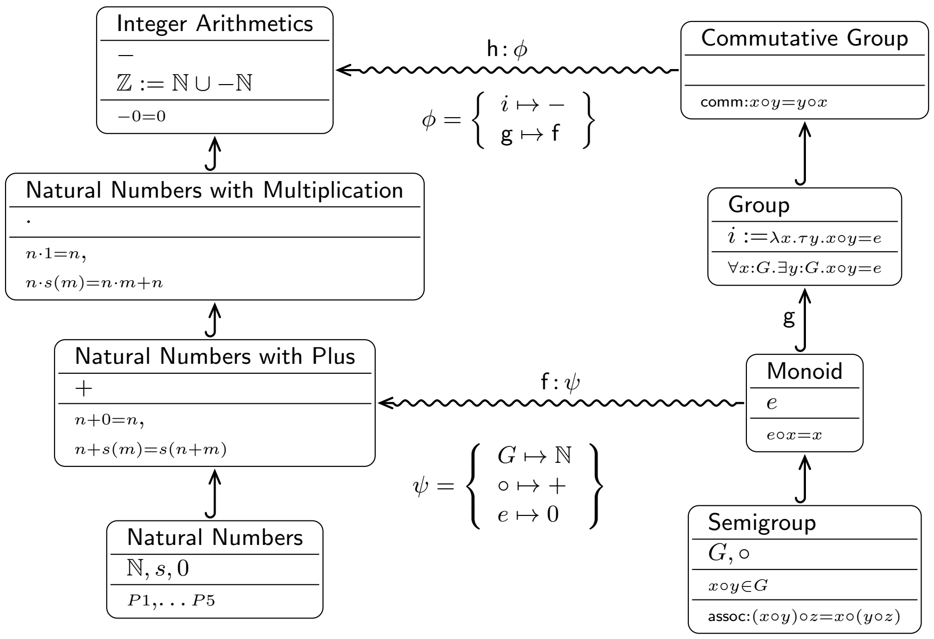
Integer Arithmetics
$-$
$\mathbb{Z} := \mathbb{N} \cup -\mathbb{N}$
$-0=0$
h : ϕ
Commutative Group
comm:$x \circ y = y \circ x$
$\phi = \left\{ \begin{array}{l} i \mapsto - \\ \mathsf{g} \mapsto \mathsf{f} \end{array} \right\}$
Natural Numbers with Multiplication
$\cdot$
$n \cdot 1 = n$,
$n \cdot s(m) = n \cdot m + n$
Group
$i := \lambda x. \tau y. x \circ y = e$
$\forall x:G. \exists y:G. x \circ y = e$
g
Natural Numbers with Plus
$+$
$n+0=n$,
$n+s(m)=s(n+m)$
f : ψ
Monoid
e
$e \circ x = x$
$\psi = \left\{ \begin{array}{l} G \mapsto \mathbb{N} \\ \circ \mapsto + \\ e \mapsto 0 \end{array} \right\}$
Natural Numbers
$\mathbb{N}, s, 0$
$P1, \ldots P5$
Semigroup
$G, \circ$
$x \circ y \in G$
assoc:$(x \circ y) \circ z = x \circ (y \circ z)$

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

3 A Semantic Search For Quantity Expressions Based On MathWebSearch

Most of the components will inherit from the existing MathWebSearch system. Some of the work packages, like the theory graph, will be easy to complete whereas others, like the unit system, need more work.

A view between theories of units is enough to translate between them. In practice however, because we want to use units in several components of the system, it is insufficient to just look at the theory graph views. We will need to develop a unit system that can efficiently handle units as well as translations.

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. When translating between 2 composite units, this can easily cause rounding errors. In particular, when an author gives an approximation of a quantity expression in a document, they might round differently depending on the units used. Thuse we might want to 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 and handled in the front end.

Apart from translating, we will also need to enter the units in the frontend and pass them to the API. The representation we choose for units when translating should also be flexible enough so that they can be entered easily. 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.

4 Work Plan

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 end users interface to the system.

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.
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 :

- **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

References

[Gin] Deyan Ginev. The LaTeXML 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.