# Semantic Search for Quantity Expressions
# Bachelor Thesis PROPOSAL / FIRST DRAFT* EdN:1

Tom Wiesing
Supervisor: Michael Kohlhase
Co-supervisor: Tobias Preusser
Jacobs University, Bremen, Germany

May 3, 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. [2][3] EdN:2 EdN:3

## Contents



*EdNote: Remove draft status
[2]EdNote: Physics search
[3]EdNote: Re-write abstract

# 1 Introduction

* we want to write an extensible unit system. * explain in a summary what to do

## 2 The structure of Mathmatics: Theories, Views and Imports

Before we start looking at Quantity Expressions and how to build a search engine for them, we want to give an introduction to meta-mathmatical structure. We will use this knowledge later to build a better search engine. For this we first need to take a look at the concept of theories.

### 2.1 Modeling Mathematics with the help of theories

Theories, in this sense, are simply a set of symbols. Each of the symbols optionally can have a type and a definition. Within each theory, we can then use these symbols to write down terms (or expressions) within this theory. Types and definitions of these symbols are terms themselves[1]. As a simple example of this, let us consider the theory of semigroups:

| Semigroup | | |
|---|---|---|
| $G$ | : | type |
| $\circ$ | : | $G \to G \to G$ |
| assoc | : | $\mathrm{ded}\,(\forall x \in G.\forall y \in G.\forall z \in G.(x \circ y) \circ z = x \circ (y \circ z))$ |

In this theory, we define 3 symbols: $G$, $\circ$ and assoc. In the first line we define a type $G$. Next we define a function $\circ$ that takes 2 arguments of type $G$ and returns another term of type $G$. In the last line, we make the statement that associativity holds[4]. EdN:4

### 2.2 Extending theories using imports

Sometimes we want to extend theories without having to define everything again. For example, we want to say that a Monoid is a semi-group along with an identity element. In the semi-group example above, we have also used terms from other theories to define $G$ as a type.

We can model this concept by using imports. An import from one theory into another makes symbols of the imported theory available in the target theory. We can thus define a Monoid as follows[5]: EdN:5

| Monoid | | |
|---|---|---|
| import Semigroup | | |
| $e$ | : | $G$ |
| id | : | $\mathrm{ded}\,(\forall x \in G.x \circ e = e \circ x = x)$ |

### 2.3 Views as mappings between theories

However imports are not the only way theories can be related. If we have 2 theories, we sometimes want to have a map between them. In addition to the theory of monoids above, we could for example declare the following theory of non-negative integers:

[1] They are not terms over the same theory however.
[4] EdNote: possibly explain / mention Curry–Howard isomorphism
[5] EdNote: Inline import syntax?

| Non-negative integers | | |
|---|---|---|
| $\mathbb{Z}_0^+$ | : | type |
| $0$ | : | $\mathbb{Z}_0^+$ |
| $+$ | : | $\mathbb{Z}_0^+ \to \mathbb{Z}_0^+ \to \mathbb{Z}_0^+$ |
| assoc | : | ded $\left(\forall x \in \mathbb{Z}_0^+ . \forall y \in \mathbb{Z}_0^+ . \forall z \in \mathbb{Z}_0^+ . (x \circ y) \circ z = x \circ (y \circ z)\right)$ |
| id | : | ded $\left(\forall x \in \mathbb{Z}_0^+ . x + 0 = 0 + x = x\right)$ |

A map from the theory of monoids to the theory of positive integers should map all symbols from the theory of monoids to symbols from the theory of positive integers. Furthermore, such a map should be truth preserving, i. e. if I write down a true statement as a term over the theory of monoids and translate this term, it should still be true in the theory of positive integers. Such a mapping is called a *View* from the theory of monoids to the theory of Positive integers. Such a view $\phi$ could look as follows:

$$\phi = \left\{ \begin{array}{l} G \mapsto \mathbb{Z}_0^+ \\ e \mapsto 0 \\ \circ \mapsto + \\ \mathsf{assoc} \mapsto \mathsf{assoc} \\ \mathsf{id} \mapsto \mathsf{id} \end{array} \right\}$$

If we take a closer look at this view, we notice that we also have to map the imported symbols. This is needed so that we can translate any term or statement in theory of monoids to a term or statment into the theory of non-negative integers.

### 2.4 Building theory graphs

[6] We have seen in the examples above that we can model mathematics with the help of theories, views and imports. To make this structure even more obvious, we can represent it in a graph, a so-called theory graph. We consider the theories as vertices of such a graph and the views and imports as edges. An example can be found in figure 1. EdN:6

### 2.5 Using MMT to write down terms and theories

MMT is a **M**odule system for **M**athematical **T**heories [RK13]. With the help of MMT we can represent theories, views and imports in *.mmt* files. It is easy to write these files and anyone without programming knowledge can easily extend existing ones. The objects defined in these files can then be used via an API to write down terms and transform these using definitions and views.

This furthermore allows us to easily model an extensible system for units for use within a search engine. We will come back to this later in section 4.1.

[6] EdNote: Remove section heading or extend this section.

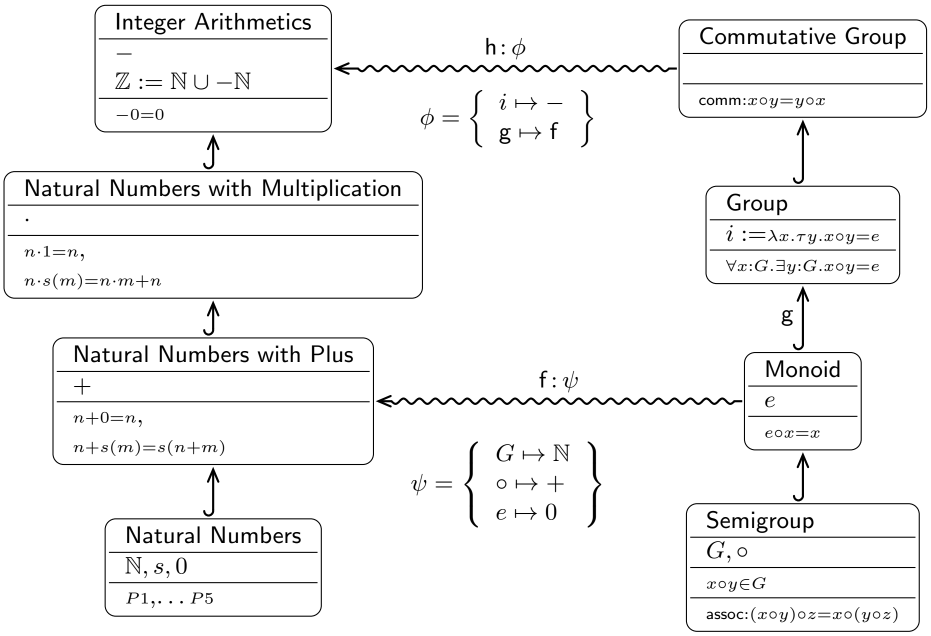


Figure 1: A simple theory graph. Imports are represented as solid edges and views as squiged edges.

# 3 Modeling Quantity Expressions

The first step in developing a good search engine for Quantity Expressions is to take a closer look at Quantity Expression.

## 3.1 Compositional behaviour of Quantity Expressions

For this purpose let us take a look at:

$$x = 25\ \frac{\mathrm{m}}{\mathrm{s}}$$

We notice that $x$ consists of 2 parts, a scalar (25) and a scalar-free $\frac{\mathrm{m}}{\mathrm{s}}$. Furthermore, the unit consists of two primitive units $m$ and $s$. Since they are divided by one another, we can conclude that $x$ describes a velocity.

While $m$ and $s$ are certainly primitive units (they can not be decomposed further), it is not easy to define a unit as a composition of simple units. Consider the following example:

$$y = \frac{\mathrm{L}}{100\ \mathrm{km}}$$

$y$ is certainly a unit and also a quantity expression. It consists of 2 sub-expressions, L and 100 km. The first one is a primitive unit and the second one a multiplication of a number and the unit km. It is thus reasonable to define the following 4 types of quantity expressions:

1. A primitive unit, such as m (meter). This is the most obvious one.
2. A number, such as 100. This can mostly be used to compose existing quantity expressions.

3. The multiplication $\cdot$ which takes 2 existing quantity expression and generates a new one, for example $\cdot(100, \mathrm{m}) = 100$ m
4. The division \which again takes 2 quantity expressions and generates a new one, for example $\backslash(\mathrm{m}, \mathrm{s}) = \frac{\mathrm{m}}{\mathrm{s}}$

This allows us to easily generate the quantity expressions $x$ and $y$ from primitive units m, s, L and km.

## 3.2 Dimensions of Quantity Expressions

Now let us briefly examine the dimensions of Quantity Expressions. The dimension of a quantity expression is the type of quantity it expresses. For example 5 m describes some length. According to the International System of Units[7] there are seven basic dimensions: EdN:7

- length
- mass
- time
- electric current
- temperature
- luminous intensity
- amount of substance.

In addition to this we add the unit dimension which simply describes a number.

Similar to the compositional behaviour of quantity expressions, dimensions can be multiplied and divided. Unlike quantity expressions however they can not be multiplied with numbers. Furthermore, when multiplying to quantity expressions of dimensions $a$ and $b$ their resulting dimension is $a \cdot b$, the multiplication of the dimensions. The same goes for division. In this regard the dimension of a quantity expression behaves like a type.

## 3.3 Mathematical Theory of Quantity Expressions

## 3.4 Transforming Quantity Expressions from one form into another

[7] EdNote: Quote something properly here

# 4 Making Quantity Expressions searchable

Having investigated the structure of quantity expression in the previous section, we can now define a meta-mathmatical theory of quantity expressions as our next step towards a search engine.

## 4.1 A meta-mathmatical formalisation of quantity expressions

## 4.2 Normalisation of Quantity Expressions

* describe normalisation algorithm, 2 steps

## 4.3 Serialising Quantity Expressions to XML

* W3C RFC, find it again

# 5 Caveats of the current implementation

## 5.1 Type Equalities

## 5.2 Unification Queries

# 6 Future work

## 6.1 Extension of the theory graph of units

## 6.2 Integration with MathWebSearch

## 7 Conclusion

## References

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