Authoring, Publishing and Interacting with Units and Quantities in Technical Documents

Mihai Cîrlănaru

Computer Science
Jacobs University Bremen
Campus Ring 1
28759 Bremen
Germany

Type: *Guided Research Thesis*
Date: *May 10, 2011*
Supervisor: *Prof. Dr. Michael Kohlhase*

Abstract

Despite their ubiquitous presence, units and quantities lack a comprehensive semantic representation in scientific and technical documents. Moreover, when it comes to authoring and interacting with units, there is very little support by current technology and also there is no formal, standardized way to do so. Problems arise from the different flavors (country specific unit standards) and formats (abbreviations, special cases of occurrences) units possess, making it hard to fully perceive their meaning. Concrete authoring and publishing examples in the context of a large collection of legacy scientific documents (the arXMLiv corpus) detail the approach to formalize units at the authoring and representational ends. This confers a basis for applications and services such as *unit preference settings*, *unknown unit lookup*, *unit and quantity semantic search*, *unit and quantity manipulation.* One service is demonstrated in further detail to exemplify the potential and practical advantages of semantically publishing units w.r.t. user experience. The thesis concludes with an outlook to the further potential of my approach towards a better manipulation and interpretation of units and quantities in scientific publications and technical documents.

Contents

Appendices

1 Introduction and Motivation

Units and quantities[1] are present *everywhere*.
Formally, a **unit** is *any determinate quantity, dimension, or magnitude adopted as a basis or standard of measurement for other quantities of the same kind and in terms of which their magnitude is calculated or expressed* [Oxf], but from the top-most level of perception, it simply provides information on a wide range of quantifiable aspects. This leads to a very high occurrence rate in one's daily life. For example, most of the general-use, common devices, like watches, phones, cars and so on, provide information (feedback) through the use of units and quantities. Descriptions, specifications, routines also take advantage of the ability to quantify information and also to provide a certain perception level through units. Concrete examples comprise of things as unrelated as cooking recipes, medical prescriptions, scientific papers and many other, thus emphasizing the great extent units and quantities possess.

Units and quantities are a very *important* aspect of our daily lives.
At the perception level, aside from quantifying properties and relations between objects, units bring the meaning of scale. One of their most important uses was and still is in the field of science: they have allowed scientists throughout the past couple of centuries to better transmit and exchange knowledge between them.

Units and quantities come in very different forms and flavors (i.e. metric, imperial). Be it just for the sake of simplicity (easier calculations, easier perception of their scales) or just for the sake of legacy (used for centuries), these flavors, although standardized, pose a lot of problems in real life. Consider losing a $125 million satellite [Mar] because of the differences between metric and imperial unit systems or running out of fuel in mid-flight with an aircraft whose fuel sensors were faulty configured in displaying the units [Air]; and these are just a few of the most notable errors caused by units. Fields like medicine, commerce, civil engineering have also been marked by such types of errors and pitfalls [Usm]. Semantic publishing solves most such problems by disambiguating the unit and quantity occurrences, which, further on, will enable a wide range of applications and services to interact with them (see sections 4.6 and 5).

The high density of units and quantities in technical documents and day-to-day published content enables an extensive analysis of different context occurrences. This confers a broader view on their hidden semantics and also helps to understand their different aspects, thus avoiding the common pitfalls and hazardous interpretations mentioned above [Usm]. Furthermore,

[1] The *units and quantities* wording was chosen in order to emphasize the semantic dependence between the *unit* and its *quantity* (i.e. amount, magnitude). An in-depth analysis of the two concepts is proposed in section 2 of [Col09].

providing semantics to units and their quantities for the publishing industry, either by supplying semantic authoring tools or by semantically enriching their occurrences in legacy documents, has high impact benefits. It will enable transparent exchange of scientific knowledge between different academic communities, typical of technical papers with high occurrences of units and quantities, and also enhance the reader's experience, via novel interactive services with day-to-day published material, e.g. cooking recipes or technical manuals.

In the following section the state of the art (section 2) for *units and quantities* is introduced in order to have a basis for the *semantic publishing pipeline* core components (section 4.2), the central point of my thesis. Immediate strategies (section 2.2) for extending the benefits of semantic units to legacy documents (section 4.5) are outlined and conclude with a summary of our outlook of future work (section 6).

2 State of the Art

This section concentrates on the main technologies in focus for the research project, namely the content markup languages OpenMath and *Content* MathML in the context of publishing technical documents. Also previous research on units and related services is described in section 2.1. The relevant prior work involving units and quantities in the context of semantic publishing is separated in three major components, *representation* (cf. section 2.1), *authoring* (cf. section 2.2) and *interaction* (cf. section 2.3) which are individually reviewed in the context of existing technologies and tools.

2.1 Representation

The semantic publishing aspect of units in scientific documents has not yet accumulated a sizable body of prior work. Previous research has been mainly concerned with the standardization of unit and quantity representation which is far from complete (not covering every unit occurrence possibility) or sufficiently machine comprehensible. There is a number of units-related semantic web ontologies: The authors of the Measurement Units Ontology [BP09] review a number of ways of representing units in RDF. The SWEET ontology (Semantic Web Earth and Environmental Terminology [Swe; RP05]) is particularly remarkable for linking units to the fields of science where they occur. A general weakness of RDF/OWL unit ontologies is, however, that the computation of derived units (and thus unit conversion) cannot be described in a straightforward way (and is rarely done). The following sections detail how the representation of (semantic) units is enabled.

2.1.1 OpenMath and *Content* MathML

In the past couple of years the web was enriched with an important number of XML-based, content-oriented markup languages for mathematics and natural sciences. The focus of the investigations discussed by my thesis is on OpenMath [Bus+04] and *Content* MathML [Aus+10], which are standards for the representation and communication of mathematical objects [KR09]. Both markup languages are concerned with the encoding of object's semantics rather than with its visual representation, to enhance the ease of knowledge exchange and processing by software systems and also human beings.

OpenMath provides an extensible framework with well-defined extension mechanisms through Content Dictionaries (CDs) [Bus+04]. Their main purpose is to provide semantics for the symbols used by OpenMath objects. This modular aspect allows an easy expansion of the unit knowledge base [SD08] to further disambiguate units in all the possible common occurrences they might have. OpenMath's minimal core language enables a high degree of portability and extensibility which can only be an advantage for the study of units and quantities.

For OpenMath, a representation of units and quantities has been proposed (cf. [DN03]), and several Content Dictionaries (CDs) covering common units have been provided. The in-depth analysis of the prospective representations of units and their dimensions that [DN03] proposes (taking into account the pros and cons of each approach) allows for a broader view to the multitude of semantic publishing possibilities. The two most significant sets of OpenMath unit CDs have been developed by James Davenport and Jonathan Stratford [SD08] and Joseph Collins [Col09], respectively. The former are remarkable for their explicit representation of conversion rules (see also section 2.3). The latter ones provide a standards-compliant implementation of SI[2] quantities and units, providing strong insight on the concepts of *quantity* and *unit* and on the prospects of capturing more of their semantics in the representation. The representation of units in OpenMath is further discussed in section 3.1.

Content MathML is concerned with the representation of object semantics through an extensive collection of construction elements whose intuitive format covers all of school and engineering mathematics (the "K-14" fragment) [KR09]. *Presentation* MathML is the other sublanguage of MathML and is concerned with the layout schemata for describing the two-dimensional notation of mathematical formulae and will not be treated in this thesis (for more details see section *Presentation Markup* of [Aus+10]). Units and dimensions were also considered in MathML which provides multiple

[2] The International System of Units [Sib]

encoding possibilities [HD03], one of which also integrates semantics in unit expressions (see section 3.2).

Structure-wise, both OpenMath and MathML provide a valuable basis for machine processing of content and are ideal markup languages for the purpose of interacting with units and quantities. The extensiveness of MathML, provided by its close to 100 intuitive (XML) structure elements for school and engineering mathematics [KR09] and multiple representation possibilities, and the modularity and extensibility of OpenMath, given by Content Dictionaries, would allow the development of great applications and services (some of which are discussed in sections 4.6 and 5) that will ease the understanding of units' and quantities' semantics.

2.1.2 OMDoc

OMDoc (Open Mathematical Documents) semantic markup format for mathematical documents [Koh06] integrates the two mentioned content markup languages in section 2.1.1 (at object level representation) thus providing an extensible framework for the semantic representation of units and quantities. One extension of OMDoc which goes into this direction is PhysML (Physics Markup Language) [HKS06] which provides a *Theory of SI units* in a nutshell. This semantic markup format provides a basis for future work in the field of semantically published scientific papers.

2.1.3 Logical Framework LF module

The module system for the logical framework LF [RS09] allows the definition of unit signatures (i.e. declarations and constant definitions) and unit signature morphisms (i.e. conversion factors/relations from one unit to another). The advantages of an LF unit module are given by its robustness, convenient use and extensibility that enable an elegant formalization of units. With the use of the Twelf system [PS] such a module can be easily transformed into OMDoc and thus enable all the advantages presented in the previous sections.

2.2 Authoring

In "pre-semantic" environments, such as LaTeX, there are first approximations of content-oriented macros that represent units. A prominent example is the LaTeX package *SIunits* [Hel] which covers the full range of base and derived units in the SI system, as well as SI prefixes, a range of widely accepted units external to SI and a couple of generic mechanisms for creating custom author-specified unit constructs. The package enables a large set of abbreviating commands, which are internally built up from the compositional application of atomic building blocks. In this sense, the authoring process via *SIunits* is *nearly semantic* on the interface level, but *entirely presentational* on the output side.

Still, all major semantic authoring systems (e.g. the semantic LaTeX extensions sTeX [Koh08], SALT [Gro+07], the Ontology Add-in for Microsoft Office Word [Fin+10], or the semantic content management system PAUX [PAU]) have so far neglected the specific use case of units. This can be partially explained by the lack of a widely agreed standard representation, as well as different primary development foci – mathematics for sTeX, rhetorical structures for SALT, life sciences terminology for the Microsoft Office Word ontology add-in, and educational texts from areas unrelated to physics, such as law, for PAUX. Notably, sTeX could, in principle, support units already, as its wide coverage of the conceptual model of OpenMath and its generic mechanism for defining new symbols and concepts could easily be utilized for the specification of the relevant unit and quantity symbols. Section 4.4 presents how this has been done without disrupting existing LaTeX authoring practices. While LaTeX is commonly used in mathematics, science, and engineering, the proposed authoring solution is unlikely to appeal to life scientists, where Microsoft Office Word is more widely used; however, unit support for word processors is left to future work.

2.3 Interaction

Applications taking advantage of the semantic publishing of units and their quantities using OpenMath have also been experimented with by various authors, albeit the lack of authoring support. The unit conversion service [Str08; SD08] by Jonathan Stratford, which users can easily extend by uploading new Content Dictionaries (CDs) with new units and conversion rules, provides a good example of the power of semantically annotated units. Besides the implementation of such a service, Stratford's research also identifies the difficulties of unit conversion and the limitations of OpenMath's current state with regard to unit representation.

Stratford's conversion service is interactive in that users can enter quantities into a web form and upload definitions of new units. Previous research [GLR09] has made the conversion service interactively accessible from web documents that contain MathML formulas with OpenMath annotations, as created by the *semantic publishing pipeline* explained in section 4.2. This interaction with units in publications has, however, remained a proof of concept so far, as *producing* suitably annotated documents required manual authoring of quantity expressions in OpenMath XML markup – a barrier that my research tries to overcome with the work presented in this report. Although another (web) interactive service, Wolfram|Alpha [Wola], provides unit conversion capabilities through its API [Url] and widgets[3] [Wolb], the interpretation of the input and presentation of the output are purely linguistic (i.e. non-semantic) and thus it is not considered for my research.

3 Units in Technical Documents

This section represents a preliminary stage of the research project by providing a more in-depth look at the representation (cf. section 2.1) of units and quantities in (semantic) markup languages an analysis of *unit and quantity* (common) occurrences. The detailed state of the art of *units' and quantities'* representation in OpenMath and in *Content* MathML (with concrete examples) is described in the first two subsections, while the last subsections describe the different occurrences of units and quantities in technical documents (selected from the arXiv e-Print archive [Arx]) and the generic idea of semantic units in the context of semantic publishing.

3.1 Units and Quantities in OpenMath

OpenMath encompasses units through Content Dictionaries [Col09]. Although generically described in section 2.1.1, it is important to emphasize OpenMath's extensibility through the creation of new content dictionaries that can add new symbols (referred to as OMS in OpenMath objects) with their respective semantics via Formal Mathematical Properties (known as FMPs) or simply through the extension of the already existent unit CDs.

The real problem of OpenMath is the way units and quantities are interlinked. For example, the OpenMath representation of a quantity (i.e. a number) would be:

```
    <OMI> 100 </OMI>
```

while for units, take the unit "gramme" as example, the following representation exists:

```
    <OMS name="gramme" cd="units_metric1" />
```

The representation of *100 grammes* ($\boxed{100\,\mathrm{g}}$) would look as follows:

```
  <OMA>
      <OMS name="times" cd="arith1" />
```

[3] Mini-apps built on top of Wolfram|Alpha queries [Wolc].

```
        <OMI> 100 </OMI>
        <OMS name="gramme" cd="units_metric1" />
    </OMA>
```

Listing 1: OpenMath representation of $\boxed{100\,\mathrm{g}}$

which is one out of multiple representations of ***quantity*** *unit* [DN03], thus making it difficult to develop standardized interaction services. Moreover, all such representations make abuse of the `times` operator which is presented as a best practice for unit semantic annotation [SD08]. A different semantically-appropriate representation that replaces `times` with a quantity constructor ($quantityFN : real \times unit \rightarrow quantity$ – see also appendix A) is presented in section 4.4.

3.2 Units and Quantities in *Content* MathML

The characteristics of units in *Content* MathML are extensively described in [HD03], but for the scope of the research project only the significant ones have been listed below:

1. **Simple Units**: *"In expressing a quantity with units [...] it is recommended that the unit be the last child of an* ***apply*** *element which has the* ***times*** *element its first child"* in Section 3 of [HD03].This implies the following type of representation for a simple quantity with unit (e.g. $\boxed{1.7\,\mathrm{m}}$):

```
      <apply>
          <times/>
          <cn> 1.7 </cn>
          <csymbol> m </csymbol>
      </apply>
```

Listing 2: *Content* MathML representation of $\boxed{1.7\,\mathrm{m}}$

2. **Compound Units**: *"It would also be best if compound units are kept separate as a nested apply at the end of a product"* in Section 3 of [HD03]. This implies the following possible representation of a quantity with simple, compound units (e.g. $\boxed{350\,\mathrm{m/s}}$):

```
      <apply>
          <times/>
          <cn> 350 </cn>
          <apply>
              <divide/>
              <csymbol> m </csymbol>
              <csymbol> sec </csymbol>
          </apply>
```

```
        </apply>
```

Listing 3: *Content* MathML representation of $\boxed{350\,\mathrm{m/s}}$

   Here the compound unit is represented by the division of two simple units, each enclosed in its own **csymbol** element. The use of simple units in the **csymbol** element is actually enforced throughout Section 5 of [HD03]. Semantically speaking, there exist simple units (e.g. Watt) that have a compound representation (e.g. Watt $=$ Nm : *Newton* × *meter*) and thus two identical expressions can have different representations.

3. **Semantic Units** (Expressions): *"In general, it is quite difficult to pick a unit from an expression. [...] a method for facilitating the transmission of unit information by making use of the* ***semantics*** *element. [...] Each* ***semantics*** *element would contain the attribute* ***definitionURL="http://.../units/"***. *The first child element would be the unit encoded in either* Content *or* Presentation MathML.*"*. These types of expressions are detailed starting from Section 6 in [HD03].

   The introduction of the `definitionURL` attribute (for `semantics` elements) which identifies the definition of a unit expression (e.g. `<semantics definitionURL='http://.../units/'>`) enables an easier spotting of units in *Content* MathML and also facilitates the disambiguation procedure. Note that the `csymbol` element for units also has a `definitionURL` which uniquely identifies its definition (e.g. `<csymbol definitionURL='http://.../units/meter>m</csymbol>`).

3.3 Common Occurrence of Units and Quantities

For the purpose of identifying the different ways units appear in the "wild", seven papers [CB10; SGHPD10; LJ10; Stu+10; EFK10; Haz+10; Fuk+10] from the fields of Physics, Astronomy and Mathematics were randomly chosen from the arXiv e-Print archive [Arx] and all occurrences of quantities and units were highlighted. Although the selected pool of technical documents is quite narrow, each of the selected papers contained a sufficient number of occurrences in a extensive variety of contexts[4]:

- ***Inner-paragraph occurrences*** which consist of embedded unit-quantity or unit-formula blocks within sentences:

[4] Only selected occurrences have been provided as examples to highlight their classification in the given types.

- **simple occurrence**: simple or prefixed units together with quantities containing:
    - abbreviations of units:
      a broad temperature range ($4K - 300K$) - [Haz+10], p. 1
      with frequencies around $1kHz$ - [SGHPD10], p. 1
      use a $500M\Omega$ *thick film resistor* - [Haz+10], p. 2
    - full names of units:
      over a hundred gigabytes *of street network data* - [LJ10], p. 2
      with a few meters *gap in between* - [LJ10], p. 2
      costs only 25 seconds *to generate* - [LJ10], p. 6
- **complex occurrence**: compound units together with quantities (including composed quantities) or mathematical expressions:

  absorption column $N_H >= 5 \times 10^{23} cm^{-2}$ - [SGHPD10], p. 3
  cables are cut into $8 \times 1.85m$ *shorter lenghts* - [CB10], p. 6

- ***Formula / Mathematical expression occurrences*** which consist of independent formula / mathematical expression blocks linked with units: Such examples were not encountered in the seven selected papers mentioned at the beginning of this section but one expected occurrence would be

$$F_{q_1 q_2} = \frac{1}{4\pi\epsilon_0} \frac{q_1 q_2}{d^2} N$$

  where N represents the unit for force in Physics ($Newtons$) and not a variable (q_1, q_2, d) or constant ($\frac{1}{4\pi\epsilon_0}$).

- ***Graph axis occurrence***: occurrence of units as axis labels exemplified in Figure 1. Although it is quite common to have the units as axis labels for plots, it is practically impossible to interact with the data in such plots given that their representation format does not allow its reiteration. Nevertheless, one could make LaTeXML [Mil] generate SVG+RDFa from figures that are accessible to the LaTeX compiler but this would be treated as potential future work and is not considered for the in-depth approach in section 4.

4 Research Approach and Results

4.1 Analysis of Units and Measurable Quantities

Although section 3 mentions a wide range of types of occurrences for units and quantities, when it comes to the practical aspect of interacting with them, from the *user* perspective, the focus should be shifted to their underlying (semantic) representation in the OpenMath and *Content* MathML markup languages.

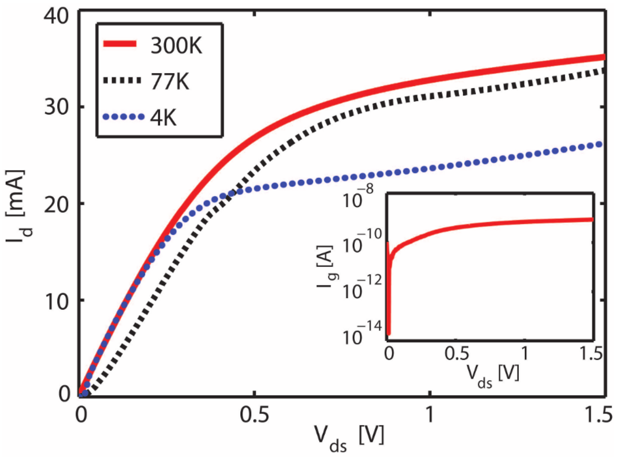


Figure 1: Output characteristics of Fujitsu FHX35X HEMT Integrated Capacitance Bridge [Haz+10]

In OpenMath the existence of content dictionaries for units [Col09] allows the capture of their semantics, while in *Content* MathML the formal representation of units is extensively defined in [HD03]. A detailed description of their syntactic representation has been already mentioned in sections 3.1 and 3.2, however, from the *author* point of view, writing a technical document in OpenMath or *Content* MathML, including all the underlying semantics, is quite tedious and undesirable.

The highlighted problems are solved by a connected three-tier flow commonly called the *Semantic Publishing Pipeline*, consisting of the following processes: *authoring*, *publishing* (i.e. semantic representation) and *interaction*. This represents the core focus of my research and is detailed in the following sections.

4.2 Semantic Publishing Pipeline

Semantic Publishing, as a process, consists of three core components: *authoring*, *publishing* and *interaction*. Usually, these processes imply three different groups of contributors – authors, publishers and readers. Incorporating the full publishing life cycle into a single system, striving for integration and collaboration between the different participants, brings great benefits. For my thesis, I took into consideration the benefits of the social web for well-established[5] and accepted and focused on the novel developments in the field of publishing. To this extent, I developed my work in the context of the

[5] For mathematics, including the mathematical foundations of science and engineering, see, e.g., the PlanetMath free encyclopedia [Pla] and the Polymath wiki/blog-based collaboration effort [Bar10].

Planetary eMath3.0 system (see [Koh+11] for an introduction). Notably, the Planetary framework implements the architecture introduced in [Dav+10] for publishing its documents as XHTML+RDFa+MathML, enabling interactive semantic services.

4.3 Semantic Units – Idea Outline

In order to understand how a semantic representation of units and quantities will integrate with the publishing flow of the chosen framework, one first needs to pinpoint what they comprise and how they are *represented*.

A computational *semantic entity* is an object with explicit *structure*, representable in a machine-understandable form, and denoting a corresponding real-world entity. The denotation is usually encoded via a machine-readable ontology. This definition is directly applicable to semantic units and quantities, which are exactly the machine-readable representations of their physical counterparts.

For the *representation* OpenMath is chosen (cf. section 2.1.1), since it enables extensibility through the creation of new Content Dictionaries (CDs) that can add new symbols or simply through the extension of the existing unit ontologies/CDs.

As an example, consider the semantic representation of the physical ***quantity*** $\boxed{100\,\mathrm{km/h}}$ (further examples can be consulted in section 3.1); one possibility to represent it in OpenMath is[6]:

```
<OMA>
    <OMS cd="arith1" name="times"/>
    <OMI>100</OMI>
    <OMA>
       <OMS cd="arith1" name="divide"/>
       <OMA>
         <OMS cd="units_ops1" name="prefix"/>
         <OMS cd="units_siprefix1" name="kilo"/>
         <OMS cd="units_metric1" name="metre"/>
       </OMA>
       <OMS cd="units_time1" name="hour"/>
    </OMA>
</OMA>
```

Listing 4: OpenMath representation of $\boxed{100\,\mathrm{km/h}}$

[6] This is one out of several ways of representing units (cf. [DN03]). For a detailed description of the XML schema see section 3.1.2 of [Bus+04]

Although this representation is not semantically accurate because of the inappropriate use of the `times` operator (i.e. a quantity being represented as *real* × *unit*), it is widely used in practice [Str08; SD08]. A new approach (briefly explained in section 3.1) is detailed in the following section.

4.4 Semantic Authoring of Units

The available methods and technologies for authoring have been revised and it was noticed that a semantic authoring support for units does not formally exist at present. Consequently, I set out to make the first steps towards extending one of the more prepared software solutions, namely sTEX, with a special authoring module for units, by building on the existing pre-semantic toolbox of the *SIunits* LATEX package with the help of two other contributors (see Acknowledgements). sTEX [Koh08] is essentially a collection of LATEX packages that offer semantic macros. sTEX can be translated into XML markup using LATEXML [Mil] bindings[7], thus enabling easier subsequent processing – including semantic web publishing (cf. [Dav+10]). The units extension follows a similar approach[8].

As described in section 2.2, *SIunits* provides an sTEX-like content authoring interface. Considering the example mentioned in the previous sections, we are interested in authoring $\boxed{100\,\mathrm{km/h}}$ in order to create the content representation shown in Listing 4. There are many ways to author the representation in LATEX, e.g. via `$\textrm{100\,km/h}$`. The *SIunits* package makes the process less ad-hoc by focusing on the content and factoring out the presentational quirks, in the form of package options. Hence, one would instead write the more semantic `\unit{100}{\kilo\metre\per\hour}`. Moreover, using the sTEX unit module I defined, it enables a more appropriate semantic (markup) representation of a quantity-unit pair by eliminating the inadequate `times` operator (cf. section 3.1) with a generic quantity constructor of the form: $quantityFN : real \times unit \rightarrow quantity$. Also, individual unit constructors can be defined (e.g. $unitFN : real \rightarrow quantity$) to further simplify the authoring process, e.g. `\unit{100}{\gramme}` would be authored as `\gramme{100}` (a representation example can be seen in table 2).

It is interesting to observe that a completely different motivation than mine, namely to provide a convenient and centralized interface to control the *presentation* of the unit entities on a document level through the LATEX

[7] Files containing mapping rules from TEX markup to XML markup for macros, primitives and constructors; the LATEXML counterpart of a LATEX package.

[8] The SIunits bindings and sTeX extension will be released in the respective bundles (the ARXMLIV binding library and the sTEX package on CTAN) with the authors' strong commitment to free software licenses compatible with the originals.

SIunits package, leads to the essentially same result which is desired – a *semantics-oriented* authoring interface.

In my effort to leverage this functionality, I first created a LaTeXML binding for the *SIunits* package. It helped to pinpoint the semantic mapping between the interface and the OpenMath representation and provided a non-invasive semantic enrichment for LaTeX documents based on the package. Next, this gained understanding was used in building a native sTeX module for units, roughly based on the *SIunits* interface. Table 1 shows a small snippet comparing the different stages. One easily notices the abbreviating power of the sTeX approach, which hides the verbose and overly complex binding declaration under its hood, exposing the author to a controlled LaTeX vocabulary and facilitating reuse (part of the sTeX module for units, `units_cd.tex`, can be consulted in appendix A).

Language	Definition	Semantics

LaTeX	`\newcommand{\kilo}{\ensuremath{\mathrm{k}}}`
	`\newcommand{\metre}{\ensuremath{\mathrm{m}}}`
		✗

LaTeXML	`DefConstructor('\kilo{}','`
	`<ltx:XMApp>`
	`    <ltx:XMTok meaning="prefix" cd="units_ops1"/>`
	`    <ltx:XMTok meaning="kilo" cd="units_siprefix1">`
	`       k`
	`    </ltx:XMTok>`
	`    #1`
	`</ltx:XMApp>');`
	`DefConstructor('\metre','`
	`<ltx:XMTok meaning="metre" cd="units_metric1">`
	`    m`
	`</ltx:XMTok>');`
		✓

sTeX	`\symdef[name=kilo,cd=units_siprefix1]{kiloPX}{\mathrm{k}}`
	`\symdef[name=metre,cd=units_metric1]{metre}{\mathrm{m}}`
	`\symdef[name=prefix,cd=units_siprefix1]{prefixFN}{}`
	`\symdef{kilo}[1]{\mixfixii{}{\kiloPX}{\prefixFN}{#1}{}}`
		✓

Table 1: Definitions for \kilo\metre, typeset as ‘km’

4.5 Enabling Semantic Units in Legacy Corpora

The arXMLiv corpus is the ideal environment for semantic units and quantities since it contains a collection of more than 600,000 scientific publications that could take great advantage of semantically enriched content. It is based on Cornell University's arXiv e-Print archive [Arx] originally typeset in LaTeX, converted to XML in order to achieve easy machine-readability, partial semantics recovery and clear separation of document modalities such as natural language and mathematical expressions [Sta+10]. Currently, the project has achieved a successful conversion rate of nearly 70% to a semantically enriched XHTML+MathML representation, natively understandable by modern web browsers [Koh+08].

A proof-of-concept check, performed via the arXMLiv build system (see [Sta+10]) revealed roughly 150 arXiv articles using the *SIunits* package, with an outlook for close to tripling the number when considering sibling packages such as *units* and *SIunitx*. This gives the work on creating a semantic binding for *SIunits* an even stronger benefit, as it can directly and non-invasively enrich legacy publications, putting them one step further on the path to semantic publishing. An additional, mid-term benefit is the opportunity to build a linguistic *Gold Standard* for units; we created both legacy (to presentational MathML) and semantic (to OpenMath) bindings in order to provide a raw, presentational output and its annotated, semantic counterpart. Having both as a basis, unit spotters can then be developed using methods of Computational Linguistics and Machine Learning, further enriching the arXMLiv corpus. This represents an important future research topic in the field of unit identification and is definitely worth investigating for the greater purpose of creating a closed formal unit knowledge base containing not only the authoring, publishing and interaction parts, but also the linguistics component.

The mentioned enhancements not only enable the interactive services of semantic publishing on legacy corpora, but also provide a tempting outlook to the development of an ecosystem of linguistic analysis modules, which can draw on the captured semantics of units and quantities, as originally envisioned by the LaMaPUn project [Gin+09].

4.6 Unit (System) Preference Service

Given the provisions for authoring support, we move to the added-value benefits one could reap from interacting with a published document. This section details a relevant use case and explains the prerequisites that are already available, detailed in the previous sections of the thesis.

A concrete scenario for a prospective service that would take advantage of semantically published papers, based on the ideas from section 4.3, can be evolved on top of common published material like *cooking recipes*. These provide a good use case thanks to the high density of units and quantities they contain. Moreover, the physical quantities are restricted to a small subset (mass/volume related units) including special types of *units* [21c] which are not formally defined and might prove to be misleading:

$$\begin{aligned}
1\ \textbf{teaspoon (tsp)} &\approx 5\ \textit{millilitres (mL)} \\
1\ \textbf{tablespoon (tbsp)} &\approx 15\ \textit{millilitres (mL)} \\
1\ \textbf{cup} &\approx 250\ \textit{millilitres (mL)}
\end{aligned}$$

The idea of the *unit (system) preference* service is to allow the user/reader to choose a preferred system of units (e.g. imperial, metric) or simply preferred types of units (e.g. “minutes” instead of “hours”, “kilogrammes” instead of “grammes”) for the representation of physical quantities and then seamlessly adapt the document to these preferences. This can only be achieved at the end of the semantic publishing pipeline, since the process requires the technologies described in sections 2.1 and 2.2 for the *representation* and *authoring* parts. Once these prerequisites have been met, one can embed interactive scripts into the published document (here: XHTML with OpenMath-annotated MathML formulae), which invoke a web service for any computation. In my implementation, the JOBAD (Javascript API for OMDoc-based Active Documents) framework [GLR09] provides for client-server communication and manipulation of the document. Figure 2 visualizes the work flow for a cooking recipe example (Chocolate Chip Cookies [Crc]) and the following section describes the implementation details.

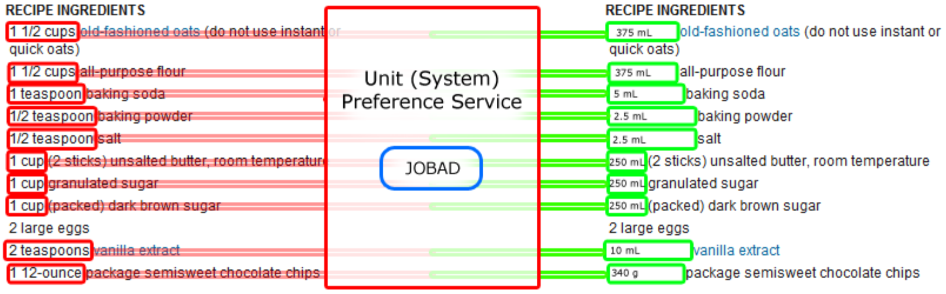

RECIPE INGREDIENTS
1 1/2 cups old-fashioned oats (do not use instant or quick oats)
1 1/2 cups all-purpose flour
1 teaspoon baking soda
1/2 teaspoon baking powder
1/2 teaspoon salt
1 cup (2 sticks) unsalted butter, room temperature
1 cup granulated sugar
1 cup (packed) dark brown sugar
2 large eggs
2 teaspoons vanilla extract
1 12-ounce package semisweet chocolate chips
Unit (System) Preference Service
JOBAD
RECIPE INGREDIENTS
375 mL old-fashioned oats (do not use instant or quick oats)
375 mL all-purpose flour
5 mL baking soda
2.5 mL baking powder
2.5 mL salt
250 mL (2 sticks) unsalted butter, room temperature
250 mL granulated sugar
250 mL (packed) dark brown sugar
2 large eggs
10 mL vanilla extract
340 g package semisweet chocolate chips

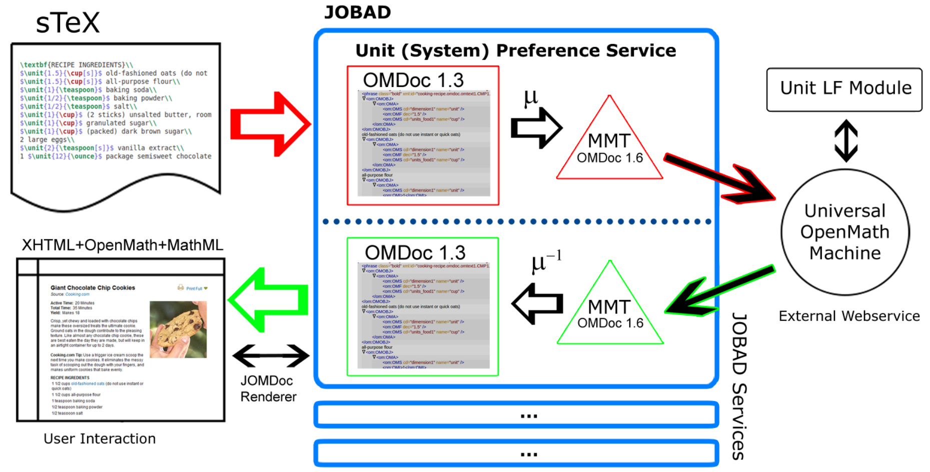
Figure 2: Work flow for the Chocolate Chip Cookies recipe

4.6.1 Implementation

The implementation of the unit (preference) service for the cooking recipe use case is separated in three major parts: authoring (including unit definitions), OpenMath-compatible web service (for computing unit relations) and JOBAD service (for interaction). The relation between them can be easily observed in the publishing and interaction flow from Figure 3.

From the *authoring* point of view, the given example has been written semantically (see `cooking-recipe.tex` in appendix B) using sTEX (cf. section 4.4) in order to prove the *authoring* concept and emphasize the ease of integration of arbitrary documents containing units within the publishing pipeline. While sTEX required the semantics for unit symbols (cf. `units_cd.tex` in appendix A), the relation between units (formal theory) was encapsulated in a LF (logical framework) module (cf. section 2.1.3 and appendix C).

Before reaching the external web service end, the sTEX authored cooking



```
\textbf{RECIPE INGREDIENTS}\\
$\unit{1.5}{\cup[s]}$ old-fashioned oats (do not
$\unit{1.5}{\cup[s]}$ all-purpose flour\\
$\unit{1}{\teaspoon}$ baking soda\\
$\unit{1/2}{\teaspoon}$ baking powder\\
$\unit{1/2}{\teaspoon}$ salt\\
$\unit{1}{\cup}$ (2 sticks) unsalted butter, room
$\unit{1}{\cup}$ granulated sugar\\
$\unit{1}{\cup}$ (packed) dark brown sugar\\
2 large eggs\\
$\unit{2}{\teaspoon[s]}$ vanilla extract\\
1 $\unit{12}{\ounce}$ package semisweet chocolate
```

Figure 3: Publishing and interaction flow for the cooking recipe use case

recipe has to pass through several preparatory stages. The first one occurs at the author's end, by compiling the sTEX file into OMDoc. Since the LF formal theory (definition) of units is authored in the context of MMT (a Module system for Mathematical Theories) [RS09], which requires a rather new (conceptually different) version of OMDoc, v1.6, and the compiled OMDoc cooking recipe example is based on v1.3, there was the need of a mapping (μ) from OMDoc v1.3 to OMDoc v1.6 (MMT) (consult table 2 for the structural/conceptual differences) which was implemented at the JOBAD service side. The inverse mapping (μ^{-1}) was also required to accommodate the restriction of the JOMDoc-based [Jom] renderer which does not yet support OMDoc v1.6. It is important to mention that the required mappings(μ and μ^{-1}) are a temporary work-around until support for OMDoc v1.6 will be integrated for sTEX and the renderer. Once this will be achieved, the flow will be significantly simplified by the removal of these stages.

The two representations exemplified in table 2 are a direct result of the detailed unit authoring possibilities presented in section 4.4. The OMDoc v1.3 shows the representation of a quantity constructor, $quantityFN : real \times unit \to quantity$ (i.e. `\unit{1.5}{\cup}`), while the OMDoc v1.6 shows the representation of a unit-specific constructor, $unitFN : real \to quantity$ (i.e. `\cup{1.5}`). Each representation possesses its pros and cons especially at the authoring end (e.g. v1.6 entails shorter authoring expressions), but for the purpose of my research the former (quantity constructor) was in focus.

The web service is based on the Universal OpenMath Machine [Zam11] which reasons and computes with OpenMath (OM) objects in the context of MMT. Thus, the v1.6 mapped example is sent to the web service which

Language	Definition

OMDoc v1.3	`<om:OMA>`
	`      <om:OMS cd="dimension1" name="unit"/>`
	`      <om:OMF dec="1.5"/>`
	`      <om:OMS cd="units_food1" name="cup"/>`
	`</om:OMA>`

OMDoc v1.6	`<om:OMA>`
	`      <om:OMS base="http://.../units.elf"`
	`        module="Cooking"`
	`        name="cup"/>`
	`      <om:OMF dec="1.5"/>`
	`</om:OMA>`

Table 2: The differences between OMDoc v1.3 and OMDoc v1.6 (MMT) relevant for the mapping (μ) and inverse mapping (μ^{-1}) scripts

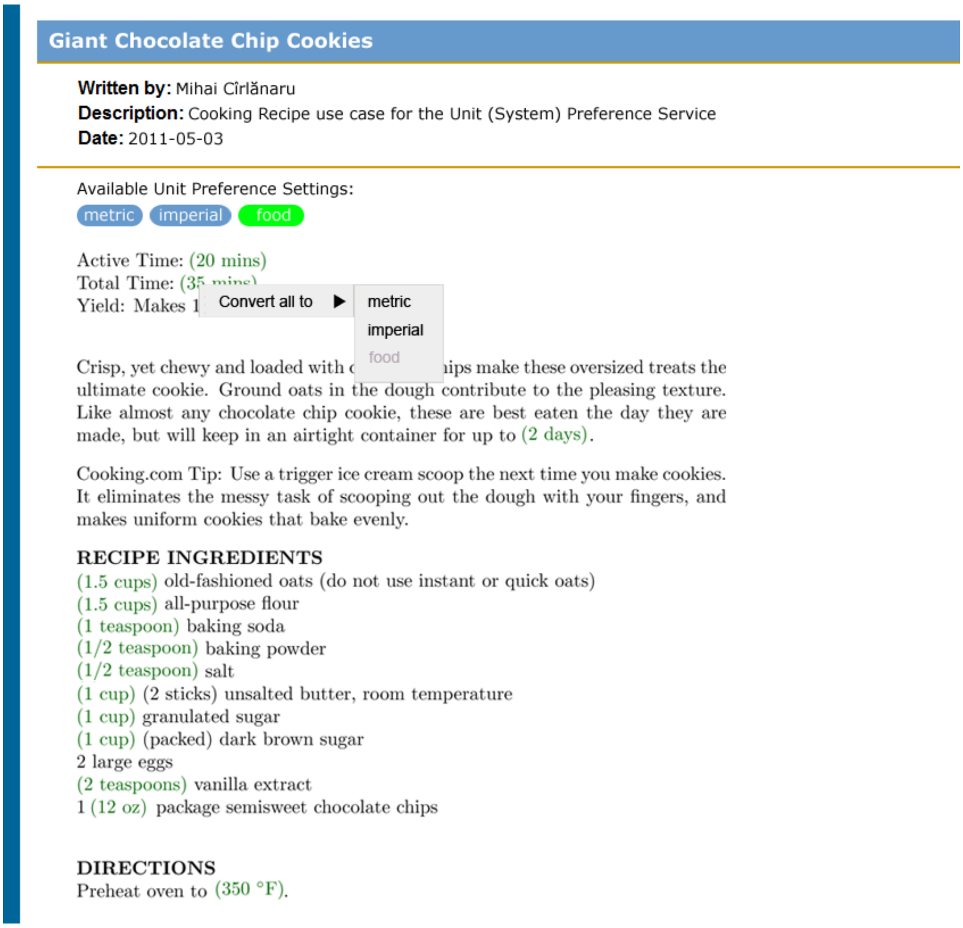
consults the LF unit theory for the morphism's definitions/operations from one unit to another. The resulting OM object is sent back to JOBAD which has to apply the inverse mapping (μ^{-1}) to OMDoc v1.3 for the renderer end. The JOMDoc library [Jom], which implements the rendering algorithm described in [Koh+09; KMR08], provides a basis for the TNTBase-exposed [ZL10] rendering service. Thus, human-readable XHTML combined with Presentational MathML and OpenMath objects is provided (rendered) to the end user, enabling the desired interaction features through JOBAD (see section 4.6.2 for more details).

It can be observed that the detailed flow follows very closely the semantic publishing pipeline by extending each of its components. The *authoring* side is achieved through sTeX and LF, the *publishing* side is done through OMDoc (v1.3) and the *interaction* is enabled by JOBAD and the external (Universal OpenMath Machine) web service thus achieving the desired goal of providing an general-purpose use case for the topic of the thesis.

4.6.2 User Interaction

This section explains how interaction is achieved on the presented cooking recipe example as sketched in figure 2.

The user is provided with the *Available Units Preference Settings* at the top of the interface and with a context menu showing the available options once he right-clicks on a unit (see figure 4). The design choice was to present the user only with the generic commonly-used unit systems (e.g. metric, imperial) and not specific units as options for preference especially due to the numerous existing types of units (some of which might not even apply



Giant Chocolate Chip Cookies

Written by: Mihai Cîrlănaru
Description: Cooking Recipe use case for the Unit (System) Preference Service
Date: 2011-05-03

Available Unit Preference Settings:
metric imperial food

Active Time: (20 mins)
Total Time: (35 mins)
Yield: Makes 1

Crisp, yet chewy and loaded with c hips make these oversized treats the ultimate cookie. Ground oats in the dough contribute to the pleasing texture. Like almost any chocolate chip cookie, these are best eaten the day they are made, but will keep in an airtight container for up to (2 days).

Cooking.com Tip: Use a trigger ice cream scoop the next time you make cookies. It eliminates the messy task of scooping out the dough with your fingers, and makes uniform cookies that bake evenly.

RECIPE INGREDIENTS
(1.5 cups) old-fashioned oats (do not use instant or quick oats)
(1.5 cups) all-purpose flour
(1 teaspoon) baking soda
(1/2 teaspoon) baking powder
(1/2 teaspoon) salt
(1 cup) (2 sticks) unsalted butter, room temperature
(1 cup) granulated sugar
(1 cup) (packed) dark brown sugar
2 large eggs
(2 teaspoons) vanilla extract
1 (12 oz) package semisweet chocolate chips

DIRECTIONS
Preheat oven to (350 °F).

Figure 4: Screenshot of the Unit (System) Preference service for the cooking recipe

to the subset of unit occurrences in the document). Once the user selects a unit preference, either from the top menu or from the context menu, each semantically represented unit from the document passes the process flow from figure 3 and its presentation is replaced with its converted value and new unit (again the choice was made for the most commonly used units, e.g. *tablespoons* would be converted to *milli litres* for the metric preference).

Although the use case covers a small subset of units and unit systems[9], the detailed authoring and publishing technologies enable the extension of semantic unit modules and thus offer a lot of opportunities for future work.

5 Future Technologies and Services

The detailed unit (system) preference service (cf. section 4.6), which enhances the user experience through the interaction with semantically published units, is a convincing proof of concept for the topic of the thesis. Nevertheless, having a standardized semantic representation of units given by formalized authoring tools can furthermore enable a great extent of potential technologies and services that can totally change the perception of units:

- **Mapping Natural Sciences Concepts to their respective Units:** defining Content Dictionaries that would describe the connection of units to general natural sciences concepts like *force* (measured in Newtons: $N = \frac{kgm}{s^2}$ or any variant of the ratio) or *energy* (measured in Joules: $J = Nm = \frac{kgm^2}{s^2} = ...$) and plenty of other examples. The interconnection of concepts in sciences: $Energy = Force \times displacement$ can further enable scientific formula "spell checking", at the authoring end, which might prove to be of great value to physicists, astronomers and many others.

- **Mathematical Formula Validation:** at the authoring end of the publishing pipeline unit checking within a mathematical expression can easily allow the verification of dimensionality of the concept the expression defines. This can be further extended to a *scientific formula validation* service that can identify a wrongly written formula by matching the units of the right hand side of the equality to the units of the left hand side.

- **Unknown Unit Lookup:** In theoretical scientific papers authors usually use abbreviations for concepts (e.g. N for *Newtons* – the unit for *force*) without mentioning anything about units/dimensions,

[9] Special unit "systems" like *food* were virtually defined and might not have an equivalent in practice. The service and detailed authoring (cf. section 4.4) tools give the author the freedom to create his own unit system.

which might turn out to be difficult for the readers who would be interested to know, for example, the order of measurement (magnitude) for the unknown physical quantities and also a (small) description of the respective concept (e.g. Pa is the unit for *pressure*). Defining a generic way in which semantics can be added to such unknown symbols will enable showing/hiding units for expressions/formulas, enhancing the interaction end of the publishing pipeline.

- **Unit and Quantity Semantic Search:** a library-level service that would allow searching for units by their type, name and magnitude and return the relevant results independently of the measuring standard of the occurrences in the paper (e.g. imperial or metric) and also independent of their form (N or $\frac{kgm}{s^2}$).[10]
- **Quantity and Unit's Magnitude Manipulation:** a document interaction service that is able to transform for example $100N \rightarrow 0.1kN$ or $0.1 \times 10^3 N$ or $0.1 \times 10^3 \frac{kgm^2}{s^2}$. This can be useful when it comes to simplifying representations and adapting them consistently to a certain type of magnitude (for example *all occurrences of force expressions should have their unit represented in kN*).

As detailed at the beginning of this thesis, having a standard, uniform understanding of units and quantities can prevent hazards and even eliminate entire (unit) compatibility check processes in industry. The presented list of prospective enabling technologies and services shows only a few of the numerous opportunities of interacting with units and quantities in semantically published documents and serves as strong motivation for future research in this direction.

6 Conclusions

Units and quantities are sufficiently wide-spread and important to not be disregarded from the context of semantic documents. Unfortunately, by now, there have been only isolated approaches (see section 2) to exploit the semantic power of units. Also considering the wide range of existing unit types and representations (some of which can be consulted in section 3), it makes it almost impossible to identify and semantically enrich all of them, especially when we are talking about occurrence contexts as unrelated as cooking recipes, medical prescriptions, technical documents or scientific papers.

Through the analysis of the semantic publishing process for units, my research

[10] In contrast, state-of-the-art scientific publication search services, such as Springer's LaTeX search [Spr], do not support the semantics of units.

emphasized the importance of three major components: *representation* (cf. section 2.1), *authoring* (cf. section 2.2) and *interaction* (cf. section 2.3), detailing technologies that can improve each of them.

From the *representation* point of view, the contribution of the thesis relates to the analysis of the different available mark-up languages (cf. section 2.1.1) and the semantically wrong, obsolete, best-practice that the current state of the art presents.

Regarding the *authoring* component, the contribution of the *SIunits* LaTeXML binding for the larger context of legacy corpora (cf. section 4.5) together with the sTEXunit module (cf. section 4.4) and the LF unit theory module (cf. section 2.1.3) provides a strong basis for authoring semantic units and quantities in a minimized user-friendly way.

Moreover, *interaction* was achieved by providing a preference-based unit conversion service (cf. section 4.6.1) throughout a complete document applicable not only to cooking recipes and also a series of further potential services and applications on top of semantically published units, my thesis contributes means of better manipulation and interpretation of *units and quantities* to the Semantic Publishing Industry and to legacy corpora.

Although not treated in the thesis, the (automated) identification of unit and quantity occurrences through linguistic analysis is an important (last) research component of a closed formal unit knowledge base that would fully take advantage of the mentioned (cf. section 5) future technologies and services that could change the perception of units in published material and would prevent hazards and costly errors in real-life use applications.

Acknowledgements

I would like to thank Christoph Lange and Deyan Ginev for their extensive support and guidance throughout the guided research. Furthermore, special thanks to Prof.Dr. Michael Kohlhase for his valuable suggestions and evaluations and to Anton Antonov for contributing on the LaTeXML bindings for the *SIunits* LaTeX package. I would also like to acknowledge that this final version of the thesis is partially based on a joint publication [CGL11] for which Christoph Lange contributed parts of the state-of-the-art review on unit ontologies (section 2.1), OpenMath unit CDs (section 2.1.1), authoring (section 2.2) and interaction (section 2.3) and Deyan Ginev contributed parts of the publishing pipeline (section 4.2), semantic authoring of units (section 4.4) and background on legacy corpora (section 4.5). A big thank you for their contribution and also to the anonymous reviewers for the constructive feedback received for the respective paper.

References

[21c] *Code of Federal Regulations – Food and Drugs*. Apr. 1, 2004. URL: http://edocket.access.gpo.gov/cfr_2004/aprqtr/21cfr101.9.htm.

[Air] *Aviation Safety – Air Canada Accident Report*. URL: http://aviation-safety.net/database/record.php?id=19830723-0 (visited on 10/25/2010).

[Arx] *arxiv.org e-Print archive*. URL: http://www.arxiv.org.

[Aus+10] R. Ausbrooks et al. *Mathematical Markup Language (MathML) Version 3.0*. W3C Recommendation. World Wide Web Consortium (W3C), 2010. URL: http://www.w3.org/TR/MathML3.

[Aut+08] S. Autexier et al., eds. *Intelligent Computer Mathematics*. 9th International Conference, AISC, 15th Symposium, Calculemus, 7th International Conference MKM (Birmingham, UK, July 28–Aug. 1, 2008). LNAI 5144. Springer Verlag, 2008.

[Bar10] M. J. Barany. "'[B]ut this is blog maths and we're free to make up conventions as we go along': Polymath1 and the modalities of 'massively collaborative mathematics'". In: *Proceedings of the 6th International Symposium on Wikis and Open Collaboration (WikiSym)* (Gdansk, Poland, July 7–9, 2010). Ed. by P. Ayers and F. Ortega. ACM Press. 2010. URL: http://www.wikisym.org/ws2010/Proceedings/.

[BP09] D. Berrueta and L. Polo. *Measurement Units Ontology*. Nov. 9, 2009. URL: http://forge.morfeo-project.org/wiki_en/index.php?title=Measurement_Units_Ontology&oldid=12301 (visited on 04/16/2011).

[Bus+04] S. Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: http://www.openmath.org/standard/om20.

[CB10] R. T. Cahill and D. Brotherton. *Experimental Investigation of the Fresnel Drag Effect in RF Coaxial Cables*. 2010. eprint: 1009.5772v1.

[CGL11] M. Cîrlănaru, D. Ginev, and C. Lange. "Authoring and Publishing of Units and Quantities in Semantic Documents". In: *1st Workshop on Semantic Publication (SePublica)* (Hersonissos, Crete, Greece, May 30, 2011). Ed. by A. García Castro et al. accepted. 2011. URL: http://kwarc.info/clange/pubs/units-sepublica.pdf.

[Col09] J. B. Collins. "OpenMath Content Dictionaries for SI Quantities and Units." In: *Calculemus/MKM*. Ed. by J. Carette et al. Vol. 5625. Lecture Notes in Computer Science. Springer, Sept. 2, 2009, pp. 247–262. ISBN: 978-3-642-02613-3. URL: http://dblp.uni-trier.de/db/conf/mkm/mkm2009.html#Collins09.

[Crc] *Cooking.com – "Giant Chocolate Chip Cookies"*. URL: http://www.cooking.com/recipes-and-more/recipes/Giant-Chocolate-Chip-Cookies-recipe-5112.aspx (visited on 03/05/2011).

[Dav+10] C. David et al. "Publishing Math Lecture Notes as Linked Data". In: *The Semantic Web: Research and Applications (Part II)*. 7th Extended Semantic Web Conference (ESWC) (Hersonissos, Crete, Greece, May 30–June 3, 2010). Ed. by L. Aroyo et al. Lecture Notes in Computer Science 6089. Springer Verlag, 2010, pp. 370–375. arXiv: 1004.3390v1.

[DN03] J. H. Davenport and W. A. Naylor. *Units and Dimensions in OpenMath*. 2003. URL: http://www.openmath.org/documents/Units.pdf.

[EFK10] N. Evans, J. French, and K.-Y. Kim. *Holography of a Composite Inflaton*. 2010. eprint: 1009.5678v1.

[Fin+10] J. L. Fink et al. "Word add-in for ontology recognition: semantic enrichment of scientific literature". In: *BMC Bioinformatics* 11.103 (Feb. 2010).

[Fuk+10] K. Fukumura et al. *Modeling High-Velocity QSO Absorbers with Photoionized MHD Disk-Winds*. 2010. eprint: 1009.5644v1.

[Gin+09] D. Ginev et al. "An Architecture for Linguistic and Semantic Analysis on the arXMLiv Corpus". In: *Applications of Semantic Technologies (AST) Workshop at Informatik 2009*. 2009. URL: http://www.kwarc.info/projects/lamapun/pubs/AST09_LaMaPUn+appendix.pdf.

[GLR09] J. Giceva, C. Lange, and F. Rabe. "Integrating Web Services into Active Mathematical Documents". In: *MKM/Calculemus Proceedings*. Ed. by J. Carette et al. LNAI 5625. Springer Verlag, July 2009, pp. 279–293. ISBN: 9783642026133. URL: https://svn.omdoc.org/repos/jomdoc/doc/pubs/mkm09/jobad/jobad-server.pdf.

[Gro+07] T. Groza et al. "SALT – Semantically Annotated LaTeX for Scientific Publications". In: *The Semantic Web: Research and Applications*. 4th European Semantic Web Conference (ESWC) (Innsbruck, Austria, June 3–7, 2007). Ed. by E. Franconi, M. Kifer, and W. May. Lecture Notes in Computer Science 4519. Springer Verlag, 2007, pp. 518–532. ISBN: 978-3-540-72666-1.

[Haz+10] A. Hazeghi et al. *An integrated capacitance bridge for high-resolution, wide temperature range quantum capacitance measurements*. 2010. eprint: 1009.5407v1.

[HD03] D. W. Harder and S. Devitt. *Units in MathML*. World Wide Web Consortium. Nov. 2003. URL: http://www.w3.org/Math/Documents/Notes/units.xml.

[Hel] M. Heldoorn. *The SIunits package: Consistent application of SI units*. Self-documenting LaTeX package. URL: http://mirror.ctan.org/macros/latex/contrib/SIunits/SIunits.pdf (visited on 03/13/2011).

[HKS06] E. Hilf, M. Kohlhase, and H. Stamerjohanns. "Capturing the Content of Physics: Systems, Observables, and Experiments". In: *Mathematical Knowledge Management (MKM)*. Ed. by J. Borwein and W. M. Farmer. LNAI 4108. Springer Verlag, 2006, pp. 165–178. URL: http://kwarc.info/kohlhase/papers/mkm06physml.pdf.

[Jom] *JOMDoc Project — Java Library for OMDoc documents*. URL: http://jomdoc.omdoc.org.

[KMR08] M. Kohlhase, C. Müller, and F. Rabe. "Notations for Living Mathematical Documents". In: *Intelligent Computer Mathematics*. 9th International Conference, AISC, 15th Symposium, Calculemus, 7th International Conference MKM (Birmingham, UK, July 28–Aug. 1, 2008). Ed. by S. Autexier et al. LNAI 5144. Springer Verlag, 2008, pp. 504–519. URL: http://omdoc.org/pubs/mkm08-notations.pdf.

[Koh06] M. 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] M. Kohlhase. "Using LaTeX 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.

[Koh+08] M. Kohlhase et al. "MathWebSearch 0.4, A Semantic Search Engine for Mathematics". manuscript. 2008. URL: http://mathweb.org/projects/mws/pubs/mkm08.pdf.

[Koh+09] M. Kohlhase et al. *Notations for Active Mathematical Documents*. KWARC Report 2009-1. Jacobs University Bremen, Feb. 2009. URL: http://kwarc.info/publications/papers/KLMMR_NfAD.pdf.

[Koh+11] M. Kohlhase et al. "The Planetary System: Web 3.0 & Active Documents for STEM". In: accepted for publication at ICCS 2011 (Finalist at the Executable Papers Challenge). 2011. URL: https://svn.mathweb.org/repos/planetary/doc/epc11/paper.pdf.

[KR09] M. Kohlhase and F. Rabe. "Semantics of OpenMath and MathML3". In: *22nd OpenMath Workshop*. Ed. by J. H. Davenport. July 2009. URL: http://kwarc.info/kohlhase/papers/om09-semantics.pdf.

[LJ10] X. Liu and B. Jiang. *Defining and Generating Axial Lines from Street Center Lines for better Understanding of Urban Morphologies*. 2010. eprint: 1009.5249v1.

[Mar] *CNN – "NASAs metric confusion caused Mars orbiter loss"*. Sept. 1999. URL: http://articles.cnn.com/1999-09-30/tech/9909_30_mars.metric_1_mars-orbiter-climate-orbiter-spacecraft-team?_s=PM:TECH (visited on 10/29/2010).

[Mil] B. Miller. *LaTeXML: A LaTeX to XML Converter*. URL: http://dlmf.nist.gov/LaTeXML/.

[Oxf] *Oxford English Dictionary. "unit" definition*. URL: http://dictionary.oed.com/entrance.dtl (visited on 10/29/2010).

[PAU] PAUX Technologies GmbH, ed. *PAUX Technologies*. URL: http://paux.de (visited on 10/10/2010).

[Pla] *PlanetMath.org – Math for the people, by the people*. URL: http://planetmath.org (visited on 01/06/2011).

[PS] F. Pfenning and C. Schürmann. "System Description: Twelf — A Meta-Logical Framework for Deductive Systems". In: *Proceedings of the 16th Conference on Automated Deduction*. Pp. 202–206.

[RP05] R. G. Raskin and M. J. Pan. "Knowledge representation in the semantic web for Earth environmental terminology (SWEET)". In: *Computers & Geosciences* 31 (2005), pp. 1119–1125. DOI: 10.1016/j.cageo.2004.12.004.

[RS09] F. Rabe and C. Schürmann. "A Practical Module System for LF". In: *Proceedings of the Workshop on Logical Frameworks: Meta-Theory and Practice (LFMTP)*. Ed. by J. Cheney and A. Felty. ACM Press, 2009, pp. 40–48.

[SD08] J. Stratford and J. H. Davenport. "Unit Knowledge Management". In: *Intelligent Computer Mathematics*. 9th International Conference, AISC, 15th Symposium, Calculemus, 7th International Conference MKM (Birmingham, UK, July 28–Aug. 1, 2008). Ed. by S. Autexier et al. LNAI 5144. Springer Verlag, 2008, pp. 382–397.

[SGHPD10] J. D. Sanabria-Gomez, J. L. Hernandez-Pastora, and F. Dubeibe. *Innermost stable circular orbits around magnetized rotating massive stars*. 2010. eprint: 1009.0320v1.

[Sib] *The International System of Units (SI) 8. Edition*. Bureau International des Poids et Mesures, 2006. URL: http://www.bipm.org/utils/common/pdf/si_brochure_8_en.pdf.

[Spr] Springer, ed. *LaTeX Search*. URL: http://www.latexsearch.com (visited on 04/16/2011).

[Sta+10] H. Stamerjohanns et al. “Transforming large collections of scientific publications to XML”. In: *Mathematics in Computer Science* 3.3 (2010): *Special Issue on Authoring, Digitalization and Management of Mathematical Knowledge*. Ed. by S. Autexier, P. Sojka, and M. Suzuki, pp. 299–307. URL: http://kwarc.info/kohlhase/papers/mcs10.pdf.

[Str08] J. Stratford. *Creating an extensible Unit Converter using OpenMath as the Representation of the Semantics of the Units*. Tech. rep. 2008-02. University of Bath, June 2008. URL: http://www.cs.bath.ac.uk/pubdb/download.php?resID=290.

[Stu+10] M. Stumpf et al. *Resolving the L/T transition binary SDSS J2052-1609 AB * (Research Note)*. 2010. eprint: 1009.5672v1.

[Swe] *Semantic Web for Earth and Environmental Terminology (SWEET)*. NASA. URL: http://sweet.jpl.nasa.gov/ (visited on 08/22/2010).

[Url] *Wolfram—Alpha API*. URL: http://www.wolframalpha.com/developers.html (visited on 05/05/2011).

[Usm] *US Metric Association “Unit Mixups” article*. URL: http://lamar.colostate.edu/~hillger/unit-mixups.html (visited on 10/25/2010).

[Wola] *Wolfram—Alpha*. URL: http://www.wolframalpha.com (visited on 05/05/2011).

[Wolb] *Wolfram—Alpha Units and Measures Widgets*. URL: http://developer.wolframalpha.com/widgets/gallery/category/?cat=units (visited on 05/05/2011).

[Wolc] *Wolfram—Alpha Widgets*. URL: http://developer.wolframalpha.com/widgets/ (visited on 05/05/2011).

[Zam11] V. Zamdzhiev. *Universal OpenMath Machine*. B. Sc. Thesis. 2011.

[ZL10] V. Zholudev and C. Lange. “TNTBase – a Versioned XML Database”. submitted. 2010. URL: http://kwarc.info/vzholudev/pubs/xsym2010.pdf.

Appendices

A Semantic Units (Modules) Definition in sTEX

Find below the semantic definition of the units relevant to the cooking recipe example (cf. section 2.3) defined in sTEX (as exemplified in Table 1 from section 4.4):

```
%% Unit Symbols relevant for the Cooking Recipe Example
%% ---------------------------------------------------
%% Guided Research Thesis - Spring 2011
%% KWARC Group - author Mihai Cirlanaru

%% 1. Generic unit and quantity functions
\begin{module}[id=dimension1]
\symdef{unit}[2]{\mixfixii[p=500]{}{#1}{\quantityFN}{#2}{}}
\symdef[name=quantity]{quantityFN}{\ }
\end{module}

%% 2. Time units
\begin{module}[id=units_time1]
\symdef[name=second]{second}{\mathrm{s}}
\symvariant{second}{s}{\mathrm{s}}
\symdef[name=minute]{minute}{\mathrm{min}}
\symvariant{minute}{s}{\mathrm{mins}}
\symdef[name=hour]{hour}{\mathrm{h}}
\symvariant{hour}{s}{\mathrm{hours}}
\symdef[name=day]{day}{\mathrm{day}}
\symvariant{day}{s}{\mathrm{days}}
\end{module}

%% 3. Cooking units
\begin{module}[id=units_food1]
\symdef[name=tablespoon]{tablespoon}{\mathrm{tablespoon}}
\symvariant{tablespoon}{tbsp}{\mathrm{tbsp}}
\symvariant{tablespoon}{s}{\mathrm{tablespoons}}

\symdef[name=teaspoon]{teaspoon}{\mathrm{teaspoon}}
\symvariant{teaspoon}{tsp}{\mathrm{tsp}}
\symvariant{teaspoon}{s}{\mathrm{teaspoons}}

\symdef[name=cup]{cup}{\mathrm{cup}}
\symvariant{cup}{s}{\mathrm{cups}}
\end{module}

%% 4. Temperature units
\begin{module}[id=units_temp1]
\symdef[name=degree_Fahrenheit]{Fahrenheit}{\mathrm{^{\circ}F}}
\symdef[name=degree_Celsius]{Celsius}{\mathrm{^{\circ}C}}
\symdef[name=degree_Kelvin]{Kelvin}{\mathrm{K}}
\end{module}
```

```
%% 5. Miscellaneous units present in example
\begin{module}[id=units_imperial1]
\symdef[name=ounce]{ounce}{\mathrm{oz}}
\symdef[name=inch]{inch}{\mathrm{inch}}
\end{module}
```

B Cooking Recipe Authored in sTeX

Find below the use-case example of a cooking recipe for Chocolate Chip Cookies [Crc] authored using sTeX as detailed in section 4.4:

```
%% Guided Research Thesis - Spring 2011
%% KWARC Group - author Mihai Cirlanaru
%% ------------------------------------
%% Cooking Recipe semantic authoring of units

\documentclass{omdoc}
\usepackage{stex,dcm,modules}
\usepackage{amssymb,lststex,lstomdoc}

%% Import Unit sTeX modules
\importmodule[units_cd]{dimension1}
\importmodule[units_cd]{units_time1}
\importmodule[units_cd]{units_food1}
\importmodule[units_cd]{units_temp1}
\importmodule[units_cd]{units_imperial1}

\begin{document}
\setlength{\parindent}{0pt}
\setlength{\parskip}{2ex}

\textbf{Giant Chocolate Chip Cookies}\\
\tiny Source: Cooking.com

\normalsize

Active Time:  $\unit{20}{\minute[s]}$\\
Total Time:  $\unit{35}{\minute[s]}$\\
Yield:  Makes 18\\

Crisp, yet chewy and loaded with chocolate chips make these oversized treats
the ultimate cookie. Ground oats in the dough contribute to the pleasing
texture. Like almost any chocolate chip cookie, these are best eaten the day
they are made, but will keep in an airtight container for up to $\unit{2}{\day[s]}$.

Cooking.com Tip: Use a trigger ice cream scoop the next time you make cookies.
It eliminates the messy task of scooping out the dough with your fingers, and
makes uniform cookies that bake evenly.

\textbf{RECIPE INGREDIENTS}\\
$\unit{1.5}{\cup[s]}$ old-fashioned oats (do not use instant or quick oats)\\
$\unit{1.5}{\cup[s]}$ all-purpose flour\\
$\unit{1}{\teaspoon}$ baking soda\\
$\unit{1/2}{\teaspoon}$ baking powder\\
$\unit{1/2}{\teaspoon}$ salt\\
$\unit{1}{\cup}$ (2 sticks) unsalted butter, room temperature\\
$\unit{1}{\cup}$ granulated sugar\\
$\unit{1}{\cup}$ (packed) dark brown sugar\\
2 large eggs\\
$\unit{2}{\teaspoon[s]}$ vanilla extract\\
1 $\unit{12}{\ounce}$ package semisweet chocolate chips\\

\textbf{DIRECTIONS}\\
Preheat oven to $\unit{350}{\Fahrenheit}$.

Butter 3 large ($\unit{16}{\inch}$ by $\unit{14}{\inch}$) nonstick baking sheets.

Place oats and $\unit{1/2}{\cup}$ flour in processor; blend until oats are almost
ground, about 10 seconds. Transfer oat mixture to medium bowl. Mix in remaining
$\unit{1}{\cup}$ flour, baking soda, baking powder and salt.

Using electric mixer on medium-high speed, beat butter in large bowl until light,
about 1 minute. Add both sugars and beat until well blended, about
$\unit{1}{\minute}$. Add eggs and vanilla; beat until well blended, about
$\unit{1}{\minute}$. On low speed, beat in dry ingredients just until blended
(do not over beat). Stir in chocolate chips.

Drop batter by $\unit{1/4}{\cup}$fuls onto prepared sheets, spacing evenly apart
and forming 6 mounds per sheet. Bake cookies $\unit{8}{\minute[s]}$. Reverse
cookie sheets and continue baking until golden brown, about $\unit{8}{\minute[s]}$
longer. Cool cookies on sheets for $\unit{5}{\minute[s]}$. Using metal spatula,
transfer cookies to racks and cool completely. (Can be made $\unit{2}{\day[s]}$
ahead. Store in airtight container at room temperature.)

\textbf{OPTIONS:}
Mix $\unit{1}{\cup}$ coarsely chopped toasted pecans, walnuts or macadamia nuts
into dough.
Replace semisweet chocolate chips with milk chocolate or white baking chips, or
use a mix of all three.


\tiny{Recipe created exclusively for Cooking.com by Sarah Taverner.}

\end{document}
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
```

C Unit Morphism Definitions in LF

The unit definition formalization has been authored in LF (cf. section 2.1.3) in the following format (excerpt from `units.elf` file) by having a theory (i.e. `%sig` definition) for each unit system:

```
%sig SI = {
   %include Quantities.

   length : dimension.
   time   : dimension.
   mass   : dimension.
   volume : dimension = length *d* length *d* length.

   %abbrev milli : unit D -> unit D = [u] [r] (r / 1000) ’ u.           %prefix 200 milli.
   %abbrev centi : unit D -> unit D = [u] [r] (r / 100) ’ u.            %prefix 200 centi.
   %abbrev deci  : unit D -> unit D = [u] [r] (r / 10) ’ u.             %prefix 200 deci.
   %abbrev deca  : unit D -> unit D = [u] [r] (r * 10) ’ u.             %prefix 200 deca.
   %abbrev hecto : unit D -> unit D = [u] [r] (r * 100) ’ u.            %prefix 200 hecto.
   %abbrev kilo  : unit D -> unit D = [u] [r] (r * 1000) ’ u.           %prefix 200 kilo.

   second       : quantity time.
   %abbrev seconds    : unit time = multiples-of second.
 }.

 %sig metric = {
   %include SI.
   meter  : quantity length.
   gramm  : quantity mass.
   litre  : quantity volume.

   %abbrev meters     : unit length = multiples-of meter.
   %abbrev gramms     : unit mass = multiples-of gramm.
   %abbrev litres     : unit volume = multiples-of litre.

   tons  : unit mass = multiples-of (1000 ’ kilo gramms).
}.

%sig imperial = {
   %include SI.

   fluid_ounce  : quantity volume.

   %abbrev fl_oz : unit volume = multiples-of fluid_ounce.
   gallon        : unit volume = multiples-of (160 ’ fl_oz).

   %* The units that are in common for both imperial and food domains *%

   %include SI.
   inch         : quantity length.
   ounce        : quantity mass.

   %abbrev inches     : unit length = multiples-of inch.
   %abbrev oz         : unit mass = multiples-of ounce.

   feet         : unit length = multiples-of (12 ’ inches).
   yard         : unit length = multiples-of (36 ’ inches).
   mile         : unit length = multiples-of (19080 ’ inches).
}.

%sig food = {
   %include SI.
   %* Special Units for Cooking Recipes *%
   tablespoon : quantity volume.
   teaspoon   : quantity volume.
   cup        : quantity volume.

   %abbrev tablespoons : unit volume = multiples-of tablespoon. %postfix 200 tablespoons.
   %abbrev tbsp        : unit volume = multiples-of tablespoon.

   %abbrev teaspoons   : unit volume = multiples-of teaspoon. %postfix 200 teaspoons.
   %abbrev tsp         : unit volume = multiples-of teaspoon.

   %abbrev cups        : unit volume = multiples-of cup. %postfix 200 cups.

      %* The units that are in common for both imperial and food domains *%

   inch         : quantity length.
   ounce        : quantity mass.

   %abbrev inches     : unit length = multiples-of inch.
   %abbrev oz         : unit mass = multiples-of ounce.

   feet         : unit length = multiples-of (12 ’ inches).
   yard         : unit length = multiples-of (36 ’ inches).
   mile         : unit length = multiples-of (19080 ’ inches).
}.

%view convertFoodMetric : food -> metric = {
%*  tbsp := [x: real] (x * 15) ’ litres.  *%
   tablespoon :=  15 ’ milli litres.
   teaspoon := 5 ’ milli litres.
   cup := 250 ’ milli litres.

   inch := (127/5) ’ milli meters.
   ounce := 28 ’ gramms.
}.
```