Chapter 6
Mashups using Mathematical Knowledge
Why Formulae are Different

Christoph Lange and Michael Kohlhase

Abstract Mashups offer new functionality by combining, aggregating and transforming resources and services available on the Web. This chapter deals with *mathematical* mashups and focuses on those that process mathematical *knowledge* rather than, e.g., huge amounts of numeric *data*, as it is the structure of knowledge that distinguishes mathematics from other application domains.

The resources that mathematical mashups process primarily include *formulae* and the services they offer involve *computation*. In knowledge-rich mashups, the formulae are not hard-coded into the implementation but represented as explicit data structures, and often also presented to the user. These structures are different from the (re)presentations mashups usually process. To allow for automated processing, formulae need to be represented neither as plain text nor as images, but in a *symbolic* way. The representation of choice is, for compatibility and scalability reasons discussed in this chapter, usually not JSON or RDF, but semantic XML *markup*. Besides tables or graphs, mathematical mashups may also require formulae to be presented to the end user. The highest degree of interaction with formulae is offered by MathML – in those browsers that fully support it.

After introducing typical education and engineering use cases that benefit from mathematical mashups, this chapter reviews the conceptual and technical foundations for representing and presenting mathematical formulae, discussing MathML as well as alternatives. We continue with a review of mathematical web services and collections of mathematical knowledge that provide suitable building blocks for mathematical mashups. We then present the Planetarysystem, a math-enabled social semantic web portal that provides an environment for executable papers, and the SAlly framework that mashes up user interfaces of software applications with mathematical web services. Both environments mash up assistive services by hooking them into document structures, which have been annotated with terms from a mathematical background ontology. We conclude with an outlook towards contributing collections of mathematical knowledge to the Web of Data, and outline how such linked open datasets can drive further mathematical mashups in the near

Christoph Lange
School of Computer Science, University of Birmingham, UK, and Computer Science, Jacobs University Bremen, Germany, e-mail: c.lange@cs.bham.ac.uk

Michael Kohlhase
Computer Science, Jacobs University Bremen, Germany, e-mail: m.kohlhase@jacobs-university.de

future.

6.1 Mathematical Knowledge and its Management

Mathematics is a ubiquitous foundation of science, technology, and engineering. These fields share mathematics as a common foundation and consequently use the same rigorous style of argumentation and the same symbolic formula language. The process of understanding results is similar, too. For example, a software engineer can hardly understand a piece of software from its source code and the brief embedded documentation alone – i.e. the counterpart to rigorous mathematical notation –, but will usually have to consult external manuals – compare mathematical textbooks – and records of developers' communication about the code, such as discussions in mailing lists and bug trackers. The latter resemble transcribed dialogs about mathematical proofs (think "software features") and their refutations (think "bug reports"), as IMRE LAKATOS studied them [99].

While the work presented in this chapter is mainly motivated from a mathematical perspective, we point out connections to other STEM fields (science, technology, engineering, mathematics) wherever appropriate.

This chapter focuses on mashups that deal with mathematical *knowledge*. These are not the only mashups in the mathematical domain, but we argue that they are the most interesting ones, because the structure of mathematical knowledge – most prominently the structure of *formulae* – distinguishes them from other mashups, which we call "data mashups" for the purpose of distinction. The Developer Apps showcase of Data.gov, the open data site of the US government, shows a selection of typical data mashups [5]: DataMasher [4], for example, accesses data sets that contain various kinds of statistical figures per US state, such as population, crime rate, government spending figures, or unemployment; it allows to combine two such data sets, to choose an arithmetic operator (e.g. sum or division) and computes the result of applying this operator to the respective data points for each state (cf. figure 6.1). The result is displayed on a map (with different color shades per state) or as a table. Now, we are interested in mashups that handle potentially arbitrary mathematical functions, and that know the types of their data well enough to tell that it does not make sense to subtract the total tax revenue per capita from the percentage of population covered by health insurance.

Where do knowledge-rich mathematical mashups help? – Recently, an "industrialization" of mathematical research has been observed, exhibiting patterns such as big teams of authors, instant communication, more fluid collaboration, de-centralized modes of publication and knowledge authentication, and the usage of big computer systems [54, 60]; ANDREA ASPERTI et al. similarly argued that "mathematics is destined to assimilate some practices of software development" [45]. Nowadays, computers assist with all steps of "doing mathematics": numeric

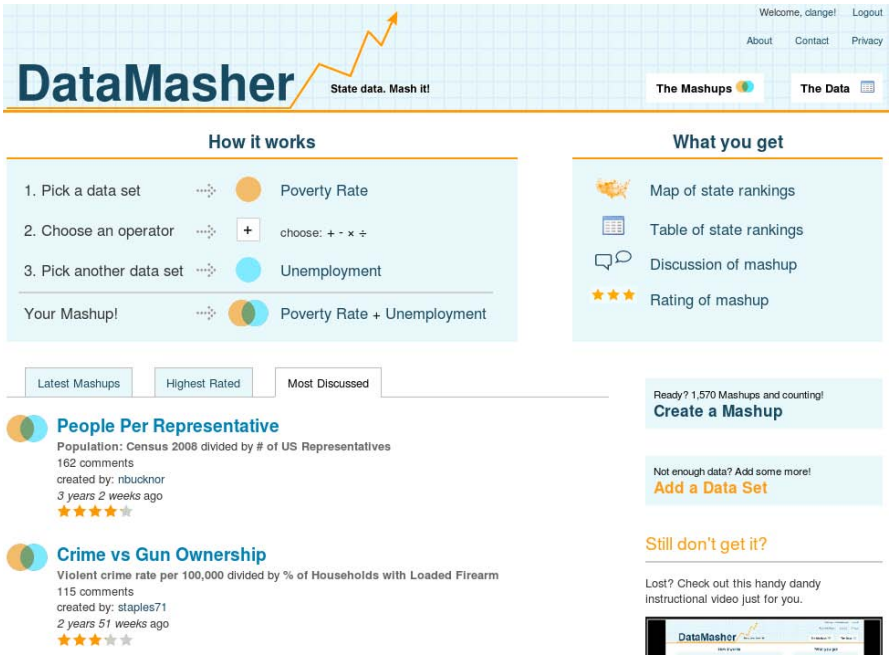

Fig. 6.1: DataMasher [4], a data mashup

as well as symbolic computation (the original purpose of *computers*!), verification and even generation of proofs [125] (these are subsumed under symbolic computation in the following), gaining intuition by experiment and generating counterexamples [81, p. 154], high-quality publishing (e.g. using the LaTeX typesetting system), and education (cf. e.g. [38, 114, 110, 64]). The proceedings of the previous conferences on intelligent computer mathematics (CICM) provide some further overview [1, 49, 58, 48, 67, 87].

Those aspects of computer mathematics that are not immediately concerned with numeric and symbolic computation are commonly referred to as *mathematical knowledge management* (MKM). The interdisciplinary MKM community consists of computer scientists, computer-savvy mathematicians, and digital library researchers, whose objective is "to develop new and better ways of managing mathematical knowledge using sophisticated software tools" [72][1], or, more specifically, "to serve (i) mathematicians, scientists, and engineers who produce and use mathematical knowledge; (ii) educators and students who teach and learn mathematics; (iii) publishers who offer mathematical textbooks and disseminate new mathematical results; and (iv) librarians and mathematicians who catalog and organize math-

[1] This notion of the term "knowledge management" is wider than that of its traditional definition as "a range of practices used [...] to identify, create, represent, distribute and enable adoption of insights and experiences. Such insights and experiences comprise knowledge, either embodied in individuals or embedded in organizational processes or practice." [140]

ematical knowledge" [72][2]. Knowledge-rich mathematical mashups are needed in such application settings, but MKM research also provides the know-how to build them: As soon as a mathematical mashup performs computations beyond standard library functions, or accesses structured data beyond arrays of numbers, it needs an explicit representation of knowledge in order to know where to get its data (e.g. numbers) from, what these data mean, and how to publish the results to its end users.

The rest of this chapter is structured as follows: Section 6.2 introduces the specific conditions of mathematical knowledge on the Web with a brief historical excursion. Section 6.3 provides an overview of existing mathematical mashups – or, rather, as very few of them exist to date, of *possible use cases* for mathematical mashups and of existing technology that helps building them. Section 6.4 introduces the Planetary system and the SAlly framework, two approaches to mashing up web or office documents with mathematical services. Section 8.5 concludes the chapter with an outlook towards a mathematical Web of Data.

6.2 Mathematical Knowledge on the Web

A lot of mathematical knowledge has been created and published on the Web[3], both by practitioners doing mathematical research, education, or applications, and in research projects that investigated the applicability of web technology to mathematics. This section reviews the state of the art, focusing on systems and projects that could use mashup technology. We first review traditional web 1.0 applications, as they are still widely in use. Web 2.0 applications, enabling better communication and collaboration, are becoming more and more commonplace also among mathematicians, whereas semantic web technology is barely on the verge of achieving a breakthrough in the mathematical domain.

6.2.1 Web 1.0 – Digital Libraries for Humans and Tools

Virtually all mathematicians nowadays use digital libraries. For example, Zentralblatt MATH [33] and MathSciNet [41] are the largest services that provide reviews and abstracts of mathematical publications. The knowledge base is searchable online, by full text and by metadata, such as author, title, and the Mathematics Subject Classification (MSC [34]; see also section 6.3.2.4).

Many of the libraries used by computer-based mathematics tools, such as computer algebra systems (CAS), proof assistants, and program verification systems, have also been published on the Web – while still being edited and maintained off

[2] numbering added by the authors

[3] To reduce eye strain, we only capitalize this term, as well as the terms "Web 2.0" and "Semantic Web", when they denote the Web as a whole, but not when they are in an adjective position, as in "semantic web services".

the Web. Consider, for example, the Journal of Formalized Mathematics, publishing machine-verified proofs from the Mizar Mathematical Library (MML [18]).

Finally, there are educational or general-purpose reference works, such as the Digital Library of Mathematical Functions (DLMF) and Wolfram MathWorld. The DLMF is a centrally edited reference of special functions [6]. MathWorld collects about 13,000 entries on mathematical topics, including downloadable files ("notebooks") for the Mathematica CAS [13].

Easy Access, but Poor Collaboration and Retrieval: Besides facilitating *access* to mathematical knowledge, these sites (i) offer limited internal interaction and do not facilitate collaboration other than sharing links with collaborators, and (ii) the means of automatically retrieving, using, and adaptively presenting knowledge, e.g. in mashups, are restricted. Web 2.0 technology addresses problem (i), and semantic web technology addresses problem (ii). The following subsections review to what extent these developments have been adopted for mathematical applications.

6.2.2 How Working Mathematicians have Embraced the Web 2.0

Working mathematicians, both researchers and instructors, are increasingly using the Web 2.0 for collaboratively developing new ideas, but also as a new publication channel for established knowledge. Research blogs, wiki encyclopedias, and open educational repositories are typical representatives.

Blogging about *established* mathematical knowledge follows the traditional practice of publishing short reviews and abstracts of previously published material (cf. section 6.2.1). Researchers have also found blogs useful to gather early feedback about preliminary findings. Successful collaborations among mathematicians have started in blogs and converged into conventional articles [50]. In the Polymath initiative, blogs are the exclusive communication medium for proving theorems in a massive collaborative effort [23, 51]. Compared to research blogs, the MathOverflow forum [16], where users can post their problems and solutions to others' problems, offers more instant help with smaller problems. Its reputation mechanism simulates the traditional scientific publication and peer review process in an agile way.

For ideas emerging from a blog discussion, or for creating permanent, short, interlinked descriptions of topics, wikis have been found more appropriate. The nLab wiki [20], a companion to the n-Category Café blog [19], is a prominent example. Where MathOverflow focuses on concrete problems and solutions, the Tricki [27] is a wiki repository of general mathematical techniques – reminiscent of a web 2.0 remake of GEORGE PÓLYA's classic "How to Solve It" [124].

Wikis that collect *existing* mathematical knowledge, for educational and general purposes, are more widely known. The PlanetMath encyclopedia [122] counts more than 8,000 entries at the time of this writing. The general-purpose Wikipedia with over 21 million articles in over 280 languages also covers mathematics [141]. Targeting a general audience, it omits most formal proofs but embeds the pure mathematical knowledge into a wider context, including, e.g., the history of mathematics,

biographies of mathematicians, and information about application areas. The lack of proofs is partly compensated by linking to the technically similar ProofWiki [25] with over 2,500 proofs, or to PlanetMath.

Similarly, open educational resources (OER) about mathematics can be found on general-purpose as well as mathematics-specific sites, some of them driven by wikis. For example, the Connexions [62] open courseware repository, having the more traditional ownership model of a content management system. It promotes the contribution of small, reusable course modules – more than 20,000, about 4,000 from mathematics and statistics, and about 7,000 from science and technology – to its *content commons*, so that the original author, but also others can flexibly combine them into collections, such as the notes for a particular course. i2geo [10, 106] is an example of a wiki dedicated to educational interactive geometry constructions.

Finally, established mathematical knowledge bases are starting to employ web 2.0 front ends to simplify and crowd-source maintenance – for example the recently developed prototypical wiki frontend for the Mizar Mathematical Library (MML) [39, 135] mentioned in section 6.2.1.

Little Reuse, Lack of Services: Web 2.0 sites facilitate collaboration but still require a massive investment of manpower for compiling a knowledge collection. Machine-supported intelligent knowledge reuse, e.g. from other knowledge collections on the Web, does not take place. Different knowledge bases are technically separated from each other by using document formats that are merely suitable for knowledge presentation but not for *re*presentation, such as XHTML with LaTeX formulae. The only way of referring to other knowledge bases is by untyped hyperlinks. The proof techniques collected in the Tricki cannot be automatically applied to a problem developed in a research blog, as neither of them is sufficiently formalized. Conversely, the Polymath community does not have any automated verification tools at hand.[4]

Intelligent information retrieval, a prerequisite for finding knowledge to reuse and to apply, is restricted to regular text search even though crucial parts of mathematical knowledge are only conveyed in formulae.

Finally, the integration of mathematical web 2.0 sites with automated reasoning and computation services is scarce. Interactive computation is available in mathematical eLearning systems, such as ActiveMath [38] or MathDox [110] – where document authors have sufficiently formalized the underlying mathematics in separate editing tools before publishing –, but less so in general-purpose digital libraries and collaboration environments. Mashups, which have otherwise been a driving force of Web 2.0 development, scarcely exist for mathematical tasks, as detailed in section 6.3.

[4] Other than the above-mentioned wiki frontend to the Mizar Mathematical Library and a similar one for the library of the Coq theorem prover, none of which are the *primary* entry points to these libraries yet, we are not aware of mathematical web 2.0 sites that integrate formal verification.

6.2.3 Adoption of the Semantic Web in MKM

In the early 2000s, when XML was increasingly used for mathematics, particularly for formulae (cf. section 6.3.2), the first building blocks of the Semantic Web vision approached standardization. This sparked interest in the emerging MKM community (cf. section 6.1), whose members hoped that semantic web technology would help to address their challenges. This seemed technically feasible, particularly as both communities made use of XML as a serialization format and URIs for identifying things [108]. The two main lines of applying semantic web technology to MKM focused on *digital libraries* – improving information retrieval and giving readers access to automated reasoning and computation services –, and *web services* – providing self-describing interfaces to automated reasoning and computation on the Web, so that they could solve problems sent to them by humans or other agents. We refer the reader to [102] for a survey of these (largely frustrated) early attempts of using Semantic Web techniques for MKM. Important systems of this time are the "Hypertextual Electronic Library of Mathematics" HELM [8, 46], the MathServe distribution architecture for automated reasoning [148], and the MONET project, which pioneered an architecture for mathematical semantic web services [117, 57]. The problems encountered by these systems can be attributed to (i) semantic web technologies were adopted before they were mature (which benefitted the Semantic Web, but not MKM), and (ii) a fundamental mismatch between RDF-based technologies and the requirements for dealing with mathematical formulae.

The combination of the Web 2.0 and the Semantic Web, sometimes called "Web 3.0", started to emerge in 2006 (cf. [42]) and is now conquering mainstream applications via incremental enhancement of successful Web 2.0 applications with semantic technology, as can be seen from general-purpose social semantic web engines such as Semantic MediaWiki [26] or Drupal 7 [65]. In 2007, STEFANO ZACCHIROLI suggested jumping on this train as a way of retrying the application of semantic web technology to mathematics after the initial failure: Web 2.0 technology would allow for interactively editing mathematical content in the web browser, and Social Web initiatives such as PlanetMath had already proven that there is "a community of people interested in collaboratively authoring rigorous mathematics on the web" [144].[5] Moreover, both HELM and MONET would now benefit from the wide support for SPARQL, the standard query language for RDF.

We are now witnessing a resurgence of mathematical applications on the Web 3.0. The emerging HTML5 [83] includes MathML without requiring the strict XML conformance that authors and UI widget toolkits often fail to achieve; soon, more browsers can be expected to support MathML. PlanetMath is being ported to the Drupal-based Planetary engine (cf. section 6.4.2) and is currently being deployed for beta testing [123, 95]. The Mathematics Subject Classification scheme (MSC), widely used in paper-based and digital libraries, has recently been officially published as a linked open dataset (cf. section 6.3.3.5), which enables easier access for

[5] Similarly, BAEZ suggests that the release of a TEX formula editor plugin for the popular WordPress blog engine was a major incentive for mathematicians to start blogging [50].

web services, as well as an easier construction of links between mathematics and related fields. We are even seeing first commercial systems like "truenumbers" [131] a system for supporting the representation, management and copy/pasting of engineering values as semantically enhanced data.

6.3 Mathematical Knowledge Mashups

Few mathematical knowledge mashups or mashup-based mathematical knowledge management systems exist to date. The ProgrammableWeb [24] mashup directory lists 3 mashups and 3 APIs tagged with "math", out of more than 6,000 mashups and more than 5,000 APIs overall – an observation that suggests that the mashups we classified as "data mashups" in section 6.1 are not commonly considered proper *mathematical* mashups. This can, however, be expected to change soon: In 2010, Wolfram, who had already released the Wolfram Alpha "computational knowledge engine" (cf. section 6.3.3.1), released an online development environment for building custom "widgets" as front-ends to Wolfram Alpha [30]. These widgets are mini-applications with a custom user interface. They perform simple computations backed by Wolfram Alpha and can be embedded into web pages. So far, users have developed several thousand widgets. However, these widgets only give a foretaste of the potential of mathematical mashups: (i) Most of them are rather trivial front-ends to Wolfram Alpha, merely offering input forms for information that one would otherwise include in a query to Wolfram Alpha, such as an amount of money, a source and a target currency, (ii) they are limited to acting as front-ends to Wolfram Alpha, and (iii) they can only be created within Wolfram's development environment.

Thus, as very few actual mathematical mashups exist to date, this section focuses on *possible* use cases for them, and on existing technology that helps building them.

6.3.1 Characterization and Purpose

The general **purpose** of mathematical mashups (including data mashups) can be subclassified into providing a user-friendly frontend to *computation* tasks (such as computing the derivative of a given function and plotting its graph), or retrieving and aggregating mathematics-related *information* (such as finding existing theorems applicable to a given formula). Both can occur in combination; Wolfram Alpha, for example, first retrieves functions related to the user's query and then evaluates them for some common values. Mathematical mashups can be further classified by their input and output interfaces. Possibilities for providing **input** include *entering* a *mathematical expression* (visually such as $\frac{4\pi r^3}{3}$ or in a linear text form such as `(4*PI*r^3)/3`) or a *sequence of data points* (e.g. to apply a statistical function to). Instead of manually entering such information, one may also *select* existing information – e.g. an existing expression in a document, or existing data points in a

table or chart. Mashups may provide their **output** as a *mathematical expression* (e.g. an expression representing the derivative of the input expression, or a plain number representing the result of a numeric computation) or as a *chart or graph* (e.g. for visualizing large amounts of numbers, or functions, or geometric shapes having a given property). If the input was provided by selecting existing information, a mashup may also provide its output by *rewriting or redrawing*, e.g., the selected expression or graph.

6.3.2 Technical Background

Data mashups that compute a straightforward result from some given numbers do not require any mathematics-*specific* mashup technology. This section focuses on handling *mathematical formulae*, a feature unique to mathematical web applications.

The symbolic language of mathematical formulae is non-trivial for its extensibility; particularly in mathematical research it is common practice to define new concepts and introduce new notation for them. Mathematics is not the only domain that employs its own symbolic notation: Formulae are also used in chemistry to express the structure of atoms forming a compound. An interesting aspect of musical notation is its multimodal combination with text in vocals. The notation of art music has been largely standardized until the 20$^{\text{th}}$ century[6] and then diversified. However, we argue that mathematics is distinct in that new notation can be introduced within the language of mathematics itself: Besides symbolic language, mathematics employs a conventionalized natural language (also called "mathematical vernacular"). In this language, the notation of a symbol is usually introduced with its first declaration, typical phrases being "We will denote by Z the set ...", "The notation aRb means that ...", etc. [132].

On the user interface, formulae are often entered and rendered in textbook style; the data formats for this are called *presentation markup* (section 6.3.2.1). For exchanging formulae with tools that perform computation or reasoning, they must not be encoded by their layout but by their meaning; this is called *content markup* (section 6.3.2.2). MathML, the most common language for both presentation and content markup of mathematical formulae, also allows for interacting with mathematical formulae in the browser – an important feature of the user interface of a mathematical knowledge mashup (section 6.3.2.3). Finally, mathematical mashups may draw on mathematical *data*, such as number series or mathematical background knowledge such as definitions or theorems. Section 6.3.2.4 explains how to publish such data on the Web.

[6] But see DONALD BYRD's documentation of "extremes of conventional music notation" [56].

Listing 6.1: The formula $a_1 + \frac{1}{2}$ in Presentation MathML (namespace declarations omitted)

```
<math>                                          <!-- Presentation MathML -->
  <msub>                                              <!-- subscript -->
    <mi>a</mi>                                      <!-- identifier -->
    <mn>1</mn>                                          <!-- number -->
  </msub>
  <mo>+</mo>                                          <!-- operator -->
  <mfrac>                                             <!-- fraction -->
    <mn>1</mn>
    <mn>2</mn>
  </mfrac>
</math>
```

6.3.2.1 Mathematical Formulae on the User Interface: Presentation Markup

Possibilities to present mathematical formulae in a human-readable textbook style include plain text, images, and MathML.

Plain text rendering is sufficient for simple mathematical formulae. The following Unicode blocks are of particular relevance [134]: Mathematical Operators, Miscellaneous Mathematical Symbols-A, Miscellaneous Mathematical Symbols-B, Supplemental Mathematical Operators, Letterlike Symbols, Miscellaneous Technical, Arrows, Miscellaneous Symbols and Arrows, and Mathematical Alphanumeric Symbols.

More complex mathematical formulae cannot be rendered within a line of text as they have a two-dimensional layout; consider operators embellished with sub-/superscripts, expressions in sub-/superscripts, fractions, or matrices. On the Web, they have traditionally been published as images (generated, e.g., with the LaTeX typesetting system). Images are, however, limited w.r.t. rendering quality, accessibility, reuse (e.g. copy/paste), and interaction possibilities.

Therefore, the preferred method for rendering complex mathematical formulae is the XML-based MathML (Mathematical Markup Language [47]) format. It was originally conceived for embedding mathematical objects into (X)HTML and is now part of HTML5. Besides a presentation-oriented sublanguage (unofficially called "Presentation MathML"), it features a content-oriented one (Content MathML). Listing 6.1 shows a formula in Presentation MathML.

As of 2012, Presentation MathML is natively supported by Mozilla's Gecko browser rendering engine used by Firefox. Safari supports it from version 5.1; Chrome, another browser using the WebKit engine, supports it from version 24. Opera has limited MathML support in that it treats MathML like generic XML but applies a custom built-in CSS stylesheet to it. Internet Explorer supports MathML via the MathPlayer plugin, which offers additional accessibility support by speech

Listing 6.2: The formula $a_1 + \frac{1}{2}$ in non-strict Content MathML (namespace declarations omitted)

```
<math>                          <!-- non-strict Content MathML -->
  <apply>       <!-- application of an operator to arguments -->
    <plus/>           <!-- short name for a common operator -->
    <!-- mixed presentation and content markup -->
    <ci>                                        <!-- identifier -->
      <msub><mi>a</mi><mn>1</mn></msub></ci>
    <!-- built-in constructor for a common type -->
    <cn type="rational">1<sep/>2</cn>  <!-- rational number -->
  </apply>
</math>
```

output and magnification [17].[7] Independently, any browser with CSS and JavaScript support can display MathML via MathJax [111], which replaces MathML subtrees in the DOM with CSS layouts and special fonts at display time.

6.3.2.2 Mathematical Formulae in Computation: Content Markup

In applications that go beyond high school mathematics, presentation markup leaves too many ambiguities to be useful as an input format for numeric or symbolic computation. This problem is addressed by content markup, which represents formulae (then usually called "mathematical objects") by their functional/operator structure, similar to an abstract syntax tree.

As an example for the ambiguity of presentation markup, consider three possible meanings of the[8] formula $O(n^2+1)$:

Landau symbol (a.k.a. big-O notation): the set of all functions that asymptotically grow at most as fast as n^2, where the $+1$ is actually superfluous

Function application: the application of *some* function named O – which needs not be the Landau set constructor function – to n^2+1

Invisible times: O (e.g. some variable) multiplied with n^2+1, where the multiplication operator is invisible[9]

Important building blocks of mathematical objects as supported by Content MathML [47, chapter 4] are numbers, variables, symbols (operators, functions, sets, constants), and applications of mathematical objects to other mathematical objects.

---

[7] The information on browser support has been taken from Wikipedia [142], "When can I use…" [37], and a review by TIMOTHY VISMOR [136].

[8] If we do not have the information that this is a *single* formula, then additional readings appear.

[9] Even in Presentation MathML, it is, however, best practice to mark up the distinction between multiplication and function application explicit using the special Unicode characters *FUNCTION APPLICATION* (U+2061) and *INVISIBLE TIMES* (U+2062). Both characters occupy no space on the screen and are thus invisible.

Listing 6.3: The formula $a_1 + \frac{1}{2}$ in strict Content MathML (namespace declarations omitted)

```
<math>                              <!-- strict Content MathML -->
  <apply>
     <!-- a symbol referenced by CD and name; the actual URI of
      this symbol is http://www.openmath.org/cd/arith1#plus -->
    <csymbol cd="arith1">plus</csymbol>
    <semantics>
      <ci>a1</ci>
      <!-- annotation ("parallel markup") -->
      <annotation-xml
       encoding="application/mathml-presentation+xml">
        <msub><mi>a</mi><mn>1</mn></msub>
      </annotation-xml>
    </semantics>
    <apply>           <!-- constructor for rational numbers -->
      <csymbol cd="nums1">rational</csymbol>
      <cn type="integer">1</cn>
      <cn type="integer">2</cn>
    </apply>
  </apply>
</math>
```

Content MathML comes with a default supply of symbols that cover high school and introductory university education. The non-strict sublanguage of Content MathML offers pragmatic shorthands for referring to these symbols (listing 6.2), whereas the strict sublanguage references all symbols by URI (listing 6.3). Their semantics is described in external vocabularies called *Content Dictionaries* (CDs); authors can create and use additional CDs as needed. MathML delegates the task of writing CDs to other languages, such as OpenMath [55].

OpenMath has originally been invented to facilitate data exchange between computer algebra systems (CAS). It comprises a sublanguage for mathematical objects and a language for CDs. This chapter does not cover OpenMath's object language, for three reasons: (i) HTML5 only supports MathML, but not arbitrary other XML namespaces (such as OpenMath's), (ii) OpenMath is isomorphic to strict Content MathML, and (iii) XSLT stylesheets for a bidirectional conversion exist [22]. OpenMath CDs usually do not fully specify the semantics of mathematical operators; instead, developers of CAS and other systems for numeric and symbolic computation are supposed to use the CDs as specification manuals when implementing *phrasebooks*, which translate OpenMath objects into the native languages of such systems.

Several CAS support a subset of the CDs built into Content MathML. Note, however, that (i) developers of mathematical software do not always use the OpenMath terminology of "CDs" and "phrasebooks" (instead, they may simply refer to the ability to import and export OpenMath or Content MathML), and that (ii) few systems support OpenMath or Content MathML as a part of their core functionality. In

Listing 6.4: The formula $a_1 + \frac{1}{2}$ in Popcorn (namespace declarations omitted)

```
a1{                                                    /* Popcorn */
  mathmlkeys.alternate-representation
  -> `application/mathml-presentation+xml<msub>
       <mi>a</mi><mn>1</mn></msub>`
} + 1 // 2
```

many cases, plugins particularly for OpenMath have been developed in academic research projects and are no longer maintained now. Systems offering varying degrees of OpenMath or Content MathML support include (in alphabetical order) GAP [66], Mathematica [35], MuPAD (now known as the Symbolic Toolbox of Matlab) [84], and Yacas [32].

Content MathML is encoded as XML; OpenMath additionally specifies a binary encoding. No JSON encoding has been developed so far,but with Popcorn [127, 85] there is a de facto standard text encoding for OpenMath objects (listing 6.4). RDF encodings for Content MathML have been suggested, but none has been implemented so far [102, section 4.3.2] as they are considered space-inefficient.

6.3.2.3 Prerequisites for Interacting with Formulae in the Browser

There are two important prerequisites for interacting with mathematical formulae in the browser: (i) their presentation markup should carry semantic annotations, which the client side of a mashup can use to interact with a server-side web service, and (ii) the browser must be able to redisplay them, completely or partly, according to the action the user performed.

Parallel markup is the most comprehensive solution satisfying requirement (i): MathML allows to combine the presentation and content markup of a (sub)-formula in one expression tree and identify corresponding sub-formulae by cross-references (see figure 6.2). This correspondence supports interactions with computational services where the user must identify sub-expressions of formulae. For instance, if we want to export a subexpression for evaluation in a CAS, then we could make use of the parallel markup in figure 6.2, there the light gray range is the user's selection, with the start and end node in bold face. As the CAS only accepts well-formed content expressions, we first look up their closest common ancestor that points to content markup via the *xref* attribute – here: *E.2*. Now, we can pass the target content markup tree to the CAS.

HELM [8, 46] was an early system that satisfied the interaction requirements listed under (ii) above. In HELM, one could perform actions on MathML formulae, e.g. simplifying a selected (sub)expression using an automated reasoning backend attached to the library. In modern MathML-aware browsers, the *maction* element [47, chapter 3.7.1] serves as a generic container for one or more Presentation MathML expressions, with which the user can interact. The @*actiontype* attribute

```
<semantics>
 <!-- a+ b 2 c -->
 <mrow xref="#E">
  <mi xref="#E.1">a</mi>
  <mo xref="#E.0">+</mo>
  <mrow xref="#E.2">
   <msup xref="#E.2.1">
    <mi xref="#E.2.1.1">b</mi>
    <mn xref="#E.2.1.2">2</mn>
   </msup>
   <mo xref="#E.2.0">&#x2062;
    <!-- INVISIBLE TIMES -->
   </mo>
   <mi xref="#E.2.2">c</mi>
  </mrow>
  <mo xref="#E.0">+</mo>
  <mi xref="#E.3">d</mi>
 </mrow>
```

```
<annotation-xml>
 <apply id="E">
  <csymbol cd="arith1" id="E.0">
   plus</csymbol>
  <ci id="E.1">a</ci>
  <apply id="E.2">
   <csymbol cd="arith1" id="E.2.0">
    times</csymbol>
   <apply id="E.2.1">
    <csymbol cd="arith1" id=
     "E.2.1.0">power</csymbol>
    <ci id="E.2.1.1">b</ci>
    <cn type="integer"
     id="E.2.1.2">2</cn>
   </apply>
   <ci id="E.2.2">c</ci>
  </apply>
  <ci id="E.3">d</ci>
 </apply>
</annotation-xml>
</semantics>
```

Fig. 6.2: Parallel markup with Presentation MathML elements (left column) pointing to Content MathML elements (right column).

allows for defining the type of interaction. The MathML recommendation suggests some (cycling through multiple alternative children, displaying a tooltip, requesting user input that can replace the current expression, etc.), but generally the interpretation *@actiontype* is up to applications. MathPlayer [17] supports those suggested by the MathML recommendation. No other browser has built-in support for specific action types, but the Gecko rendering engine allows for choosing the expression to be displayed from multiple alternative child expressions by changing the value of the *@selection* integer attribute (which is possible from JavaScript via the DOM).

6.3.2.4 Publishing Mathematical Data for Mashups

Mathematical knowledge is more than just formulae. Formulae occur in mathematical statements such as definitions, axioms, theorems, proofs, and examples. Like entries of CDs (cf. section 6.3.2.2), definitions and axioms may fix the semantics of new mathematical symbols; theorems assert additional properties of symbols, proofs validate theorems, and examples demonstrate the usage of symbols, definitions, axioms or theorems in practical settings, e.g. for symbolically simplifying an equation that describes the behavior of a technical system, or for numerically approximating a solution of such an equation. Publishing such knowledge in a suit-

able way for mashups requires implementing it in a data format that mashups can process, and making the data accessible on the Web. This section briefly reviews existing possibilities, whereas section 6.3.3.5 points to concrete datasets published on the Web.

The traditional format of choice for exchanging mathematical knowledge beyond formulae has been XML, RDF encodings have appeared more recently, whereas JSON is practically unknown. XML is the primary way of encoding OpenMath CDs, which have been introduced in section 6.3.2.2. OMDoc is an XML language that extends CDs by textbook-style statements and a notion of modular theories [21, 93]. Both the OpenMath CD language and OMDoc have unofficial RDF encodings [102, section 4.3.2].

Publishing knowledge as linked data [80] allows mashups to access it with little effort: They can download information about any resource (e.g. a mathematical theorem) by dereferencing its identifier (i.e. treating its URI as a URL), and these information records usually provide links to further relevant resources (e.g. examples in which the theorem is applied). Most commonly, such information is provided as RDF, but actually the client (here: the mashup) indicates its preferred format in the HTTP request header. The dominance of XML for mathematical formulae and the existence of both XML and RDF encodings for knowledge beyond formulae suggest a dual XML and RDF publishing approach; one way to achieve this is to maintain the knowledge primarily in XML and translate it to RDF (cf. [102, section 5.1]). A major complication to publishing mathematical knowledge that declares custom symbols is the OpenMath scheme of identifying symbols with URIs of the form *base-URI/cd#symbol* (consider, e.g., `http://www.openmath.org/cd/arith1#plus` used in listing 6.3), which Content MathML also relies on. The need to use "hash" URIs limits the possibility of publishing and may impair performance in the case of CDs that declare many symbols; see [102, section 5.2] for a detailed discussion of this and other related problems.

Finally note that the openness of the RDF data model allows for combining purely mathematical knowledge with related non-mathematical knowledge, such as knowledge about real-world application scenarios. Or, conversely, seen from the point of such application scenarios, linked data mechanisms allow for enriching existing datasets with mathematical semantics, as we have previously shown for governmental statistics datasets, which we enriched with pointers to OpenMath definitions of mathematical operators used to derive values from original data [137].

6.3.3 Tools

Tools useful for mathematical mashups can roughly be classified into: services performing computation, including verification (section 6.3.3.1), services for publishing mathematical content in web-compliant formats (section 6.3.3.2), user interface

components for input (section 6.3.3.3) and output (section 6.3.3.4), and useful data published for reuse (section 6.3.3.5).

We mainly focus on tools supporting content markup, as this is the most versatile format for communicating with computational tools. Beyond the tools reviewed here, mathematical mashups may employ further components, such as user models for recording knowledge items read or exercises mastered by the user (e.g. in the ActiveMath eLearning system [38, 113]), course generators that arrange learning objects into a sequence (e.g. PAIGOS [133], used in ActiveMath), or facilities that enable communication between users or between users and instructors. However, besides the possibility to *parameterize* them with a mathematical domain ontology (cf. PAIGOS [133] or the discussion facility of the SWiM wiki [100, chapter 6.6]), such tools work independently from a particular application domain.

6.3.3.1 Services for Computation

Tools that perform numeric and symbolic computation, including formal verification, have existed as standalone command-line or desktop applications for a long time. Few of them have a web service interface, and hardly any such tool is freely accessible on the Web.

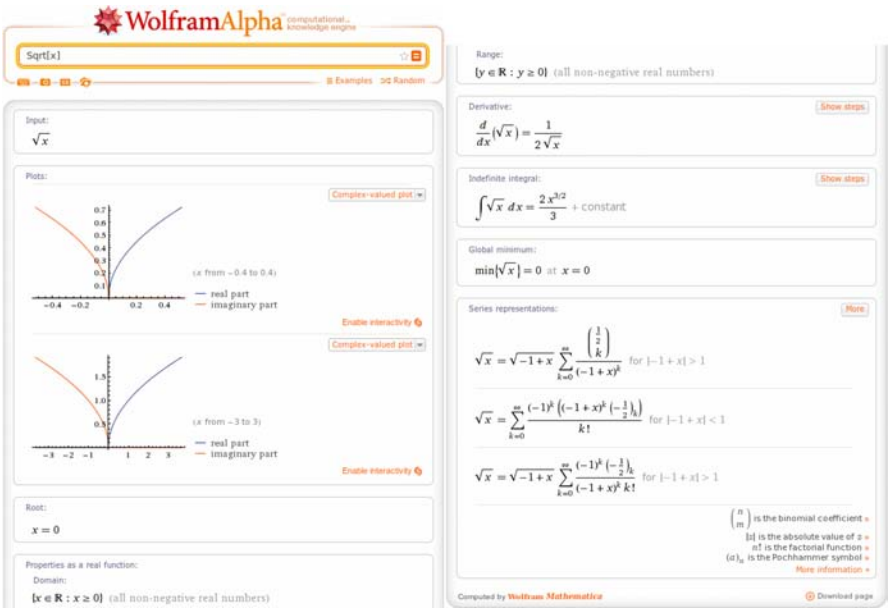

Fig. 6.3: Wolfram Alpha's results for `Sqrt[x]`

Wolfram Alpha [29], based on the webMathematica web frontend to the Mathematica CAS (which itself has a natural language input interface), offers publicly accessible computation facilities. Wolfram Alpha is primarily designed for interactive use via its web interface: users enter a mathematical expression (or a natural language phrase), and Wolfram Alpha tries to return everything that it can find out about this expression and that it deems relevant, such as a factorization, its roots, or a plot. Additionally, there is an API for non-interactive use, which allows for requesting output in different formats, but the Mathematica syntax is the only content markup supported; for details and possible workarounds see [69]. Using the API requires, depending on the feature, free registration or buying a license.

HIROSHI NAKANO et al. have developed a non-public mashup that integrates formulae resulting from symbolic computation, as well as plots of functions, into web pages [119]. On the server side, they obtain Presentation MathML formulae or gnuplot data from the Maxima CAS and wrap it into JSON-P.

Web service interfaces for exchanging content markup with computational software exist, e.g. as front-ends for CAS that support OpenMath (cf. section 6.3.2.2), but have not generally been made available for public use, e.g. by mashups. For developing a mathematical mashup, one would have to install one's own instance of any such system.

The MONET project pioneered an OpenMath-aware semantic web service architecture [117, 57]; some MONET services are still used internally in the MathDox eLearning system [110, 64] in a mashup-like architecture. The SCIEnce project (Symbolic Computation Infrastructure for Europe [126]), a more recent driving force of research on symbolic computation web services, developed SCSCP (Symbolic Computation Software Composability Protocol [79]), a lightweight XML remote procedure call protocol, which transfers OpenMath objects and whose communication semantics heavily relies on a custom OpenMath vocabulary. SCSCP front-ends have been developed for a number of CAS [126]. We are not aware of JavaScript clients libraries for MONET or SCSCP, which would facilitate the integration of such services into mashups.

6.3.3.2 Services for Publishing

Basic possibilities for publishing mathematical formulae on Web pages have been mentioned in section 6.3.2.1. This section focuses on Presentation MathML, the most advanced solution for this, and how to obtain it from content markup, as, e.g., returned by a computational service. Conceived as a function ρ: content $\rightarrow$ presentation, this translation is usually achieved by recursing over the abstract syntax tree represented by the content markup and applying different rules for different mathematical symbols, so that, e.g. $\rho(\text{plus}(a,b))$ would result in $\rho(a)+\rho(b)$. An individual per-symbol or per-symbol-pattern rule is also called a *notation definition* for this symbol.

Traditionally, the content$\rightarrow$presentation translation has been studied on the XML level and implemented as XSLT stylesheets. The probably largest collections of

notations natively defined in XSLT translates Content MathML to Presentation MathML and accompanies the MathML 3 specification as non-normative "example XSLT code" [138], and a related collection that defines 143 notations for the symbols of the official OpenMath 2 CDs [120].

To see why these are cautiously declared "examples", realize that mathematical notation depends on multiple dimensions of context. Consider, for example, the French/Russian notation of the binomial coefficient C_n^k vs. the German/English notation $\binom{n}{k}$, and see [114, 118] for systematic surveys.

XSLT as a low-level XML→XML translation language has been found inadequate for modeling notation definitions with contextual information. The rendering component of the ActiveMath eLearning system and the JOMDoc library developed in our research group offer high-level XML languages for defining context-sensitive notations. ActiveMath generates low-level XSLT stylesheets from the XML notation definitions [107], whereas JOMDoc implements a rendering algorithm internally [89, 96]. JOMDoc is particularly notable for producing cross-linked parallel markup of the presentation markup annotated with the original content markup.

Formula rendering engines are typically not ready for use in mashups; one would have to install them on the server and wrap them into a web service interface. Both ActiveMath [38] and TNTBase [146] – a document database that integrates JOMDoc, albeit without the presentation context features – provide such HTTP interfaces (cf. [147] for TNTBase). One can store content markup documents in the database and have them published as presentation markup, or POST an arbitrary content markup expression to the server and have it rendered to presentation markup on the fly (ActiveMath only).

As an alternative, Mozilla's Gecko browser rendering engine and Internet Explorer are capable of processing XSLT on the client, albeit with different APIs and limited to version 1.0. The above-mentioned MathML and OpenMath XSLT stylesheets are implemented in XSLT 1.0. The Sentido formula editor mentioned in the following section also comes bundled with client-side XSLT stylesheets for rendering a preview of the formula being edited.

6.3.3.3 User Interface Input Components

Input components for mathematical formulae can be subdivided as follows: linear text input vs. visual composition (also referred to as direct manipulation) of a formula, presentation markup vs. content markup generation, and client-side vs. server-side production of the resulting markup. Separately, this section addresses interaction with existing formulae.

A large number of visual formula editors, offering visual selection of mathematical symbols as well as visual cursor navigation, supports Presentation MathML [15]. Fewer visual editors exist for content markup. Visual editors for OpenMath objects – with a restricted set of supported CDs – have been developed by WIRIS [143, 109] and for the MathDox eLearning system [110]. The Connexions MathML

Table 6.1: Classification of formula input tools

Name	Input	Output	Site of execution	Ref.
WIRIS OpenMath Input editor	fully visual[a]	OpenMath	client (Java applet)	[143, 109]
MathDox formula editor	fully visual	OpenMath	client (HTML canvas)	[110]
Connexions MathML editor	fully visual	Content MathML	client (Presentation MathML)	[3]
Popcorn	linear	OpenMath	server (no interface provided)	[127, 85]
QMath	linear	OpenMath	client	[77]
Sentido	linear + visual[b]	OpenMath	client (Presentation MathML)	[78]
LaTeXML Daemon	linear (LaTeX)	Presentation MathML	server (HTTP POST)	[74, 76]

[a] visual symbol selection plus visual navigation
[b] just visual symbol selection

editor supports the built-in symbol vocabulary of Content MathML 2, including embedded Presentation MathML [3].

As a means of facilitating the implementation of content markup input interfaces, particularly when the supply of symbols/CDs is unlimited, one-dimensional ("linear") text input syntaxes have been developed. The Popcorn text encoding for OpenMath objects [127, 85] has been introduced in section 6.3.2.2. For a restricted set of symbols from the official OpenMath/MathML CDs, Popcorn supports intuitive infix notations such as $a+b$ instead of `arith1.plus(a,b)`. QMath [77] has originally been created as an extensible linear input syntax for OpenMath objects, with built-in support for the official CDs but the possibility to support arbitrary other CDs as well. In addition to QMath's own syntax, the QMath processor supports alternative syntaxes resembling the syntaxes of various CAS (Maxima, Yacas, Mathematica, Maple). The Sentido formula editor [78] combines linear input using QMath with a tool palette for visual selection of symbols.

The preferred linear input format for presentation-oriented formulae is LaTeX, as most users in academic mathematics and science are familiar with it. There is a large number of converters from LaTeX to Presentation MathML [14]. LaTeXML [116] in particular has a high output quality, as it reimplements the original TeX parser. The LaTeXML daemon [76] wraps LaTeXML into an HTTP POST interface publicly accessible at [74]. Content representations can be extracted from restricted subsets of LaTeX, e.g. by the above-mentioned QMath and LaTeXML tools. For unrestricted LaTeX this is a hard problem, due to the inherent ambiguity of mathematical notation. In a medium to long term perspective, linguistic techniques taking into account contextual information can be expected to solve this problem [75].

It is not always necessary that the author inputs a formula from scratch. Formulae may already exist in a mashup – embedded into a document that the mashup enriches, or resulting from an earlier step of computation. For sufficiently annotated formulae, as specified in section 6.3.2.3, the JOBAD toolkit developed in our research group (see section 6.4.1 for details) offers functions that determine the content markup associated to the range of presentation markup selected by the user. With cross-linked parallel markup this even works for subterms, as shown in figure 6.2.

6.3.3.4 User Interface Output Components

The output of a mathematical mashup can consist of structured text, formulae, or graphics. Structured text output, e.g. tabulating numerical results of a computation, does not work differently from non-mathematical mashups and is therefore not detailed here. Formula output has been covered in section 6.3.3.2.

Plots are an important and well-supported kind of mathematical and statistical graphics. For example, Flot [7] is a JavaScript library for producing graphical plots of arbitrary datasets, using the HTML canvas element. The mashup by NAKANO et al., mentioned in section 6.3.3.1, employs Flot for plotting graphs of functions. However, implementations that plot a sequence of data points do not generally live up to the complexity of mathematical functions, as they may have poles or other discontinuities. JSXGraph [12] is a library that does; in addition to function plotting, it supports interactive geometry, charting, and data visualization. Both Flot and JSXGraph allow the user to interact with their drawings; JSXGraph even supports multi-touch.

6.3.3.5 Data Published for Reuse

Huge amounts of mathematical data have been published on the Web, but only a small fraction in a machine-friendly form. This section first explains why many collections of mathematical data are not suitable for reuse in mashups, and then points out the few ones that are.

Most of the large existing collections of mathematical knowledge either target human end-users, or, when they are machine-comprehensible, their representation is specialized to a single system. Of the collections mentioned in section 6.2, Zentralblatt MATH [33], MathSciNet [41], DLMF [6], MathWorld [28], PlanetMath [122], Wikipedia [141] and the other wikis completely target human end-users. Some of them have machine-friendly APIs (cf. MathSciNet's "MR Lookup" [40], or the web service API of the MediaWiki engine that drives Wikipedia [112]), but these APIs are not uniform and thus not interoperable. Similarly, existing libraries of formalized mathematics, such as the MML [18], have been designed for automated processing but are only understood by one system. Some of the systems allow to produce representations of the libraries in a custom XML format. Since there are (to the best of our

knowledge) no publicly hosted versions, these libraries cannot be considered readily available for mashup purposes, but establishing a hosting service would be rather simple. While we have suggested OMDoc (cf. section 6.3.2.4) as a common interchange format among such systems [93, chapter 25.2], it has not yet been adopted on a large scale. Another knowledge collection that aims at machine-comprehensibility via XML but largely uses its own XML vocabulary is Connexions [62]. It uses Content MathML for the formulae, but the course documents (with explicit markup of definitions, rules, examples and exercises) and modules (including links and metadata) are written in a custom XML language [2], even though – at least for some aspects of the knowledge represented – more common vocabularies would exist (cf. the review in [102, section 4.1.5]). There are of course other educational repositories, but to the best of the authors' knowledge these do not expose the mathematical knowledge in a reusable way, but just the educational/pedagogical metadata; for example, i2geo (cf. section 6.2.2) exposes a LOM record (Learning Object Metadata) for each resource [9], which follows the LOM standard [86] with i2geo-specific extensions [82].

Linked data (cf. section 6.3.2.4) is an approach to publishing data in a uniform way for reusability. Major linked open datasets contain information that could be relevant in a mathematical context, but it not made sufficiently explicit. For example, DBpedia [70], the linked open dataset derived from Wikipedia, is not capable of making any mathematically relevant information explicit beyond a rough categorization by topic, as the Wikipedia sources only contain presentation-oriented LaTeX formulae. Statistical datasets as, e.g., published by governments, usually contain a lot of numbers (statistical data points), but hardly any information of how they have been derived (e.g. by direct observation, or by mathematical computation from other data points; see [137] for a problem statement).

As of 2012, linked open datasets with relevant mathematical knowledge are in their infancy. We have published the OpenMath CDs (cf. section 6.3.2.2), containing peer-reviewed semiformal descriptions of 260 mathematical symbols – those built into non-strict Content MathML and some more – as linked open data in 2011 [101]. In 2012, we published the Mathematics Subject Classification (MSC) as a linked open dataset [36, 103]. While these datasets provide two fundamental aspects of mathematical knowledge on the Web in a machine-comprehensible way – descriptions of the most commonly used mathematical operations, and a classification of all subjects of mathematical research and application –, further work needs to be done towards interlinking them among each other and with other existing linked datasets (such as those mentioned above), and towards annotating legacy collections of mathematical knowledge (such as those mentioned initially in this section) with pointers to these datasets. We refer to [102, section 6.1] for a detailed agenda.

6.3.4 Issues and Open Points

The foundational representation issues for enabling mashups with mathematical formulae and mathematical knowledge have been solved, and the web standards have been largely been established. There are some minor issues in the integration of the technologies into hybrid document markup systems like HTML5, but they are mostly being addressed in the ongoing standardization process.

The main remaining issue is the prevalence of presentation markup in mathematical practice, and our difficulties in semantics extraction, i.e. transforming it into content markup that can be used as a basis for mashup technologies. The semantics extraction problem is essentially equivalent to the information extraction problems from natural language, but in the case of mathematical/technical documents it is aggravated by the fact that mathematics (i) uses custom notation definitions that are particular to the respective community of practice, (ii) cannot be covered fully by standard concept and operator dictionaries (such as those of OpenMath/MathML), but introduces (and discharges) concepts, symbols, and notations dynamically (using the paradigmatic definition/theorem/proof forms), (iii) and tries to avoid duplication and redundancy of content, which allows to use information extraction algorithms with lower precision and recall.

Given that automated semantic annotation of legacy document collections using information extraction techniques is still in its infancy, we are missing large linked open datasets like DBpedia that could kick-start the explosion of mashup services we are seeing in other areas. First technologies that allow authors to input content-marked up formulae – and more generally semantically annotated mathematical documents – have been developed, but engender additional effort on the part of the authors, which would only be justifiable, if more services and mashup system were available. Thus the lack of semantics extraction techniques raises a prisoner's dilemma for the individual author and a chicken-and-egg problem for MKM; see [91] for additional discussion.

6.3.5 Evaluations

Evaluation criteria in the MKM domain often differ from those commonly applied to mashups. This section first explains the difference and then mentions a few known examples of evaluations that are relevant to the development of mathematical knowledge mashups.

MKM research has so far been biased towards formalized representations and applications in symbolic computation – less so towards web applications with a human user interface [59]. For MKM tools with a formal background, such as proof assistants or computer algebra systems, it is common to formally verify the correctness of their operation, as far as possible, on paper or using machine support. There is hardly any track record of systematic performance or usability evaluations, which would be useful for mashups. Performance evaluation has been restricted to the

evaluation of systems that do symbolic or numeric computation; see, e.g., the world championship for automated theorem proving [129, 130] or the ISSAC conference series (International Symposium on Symbolic and Algebraic Computation [11]). Performance evaluation of actual *MKM* technologies may not have been necessary so far, as large-scale deployment of advanced MKM technology is still in its infancy (cf. [97]).

Among the tools from which mathematical knowledge mashups can be built, formula editors have most frequently been evaluated for usability [43, 109, 90, 98]. Evaluation methods included performance testing – for example how many key presses or clicks users need for creating a given formula – and questionnaires assessing subjective usability. WILLIAM H. BILLINGSLEY has evaluated the usability of a graphical notation for mathematical objects and proofs, consisting of composable tiles, for students submitting solutions to exercises [52]. Test subjects were asked to prove a given set of theorems; BILLINGSLEY analyzed their solution attempts and classified the mostly unstructured feedback they had given by features of the software and aspects of the mathematical domain studied. Furthermore, he evaluated the notation against a Cognitive Dimensions of notations[10] questionnaire.

The pedagogical support of the ActiveMath integrated eLearning environment has been evaluated for usability [115]. However, that publication focuses on the qualitative results, not on the method of evaluation, and the mashup-like integration of different services in ActiveMath was not in the focus of that evaluation. In previous work, we have evaluated the usability of a wiki that integrates browsing, editing, publication, and discussion services for collaboratively maintaining OpenMath Content Dictionaries [100, chapter 10] in three steps: analyzing user-generated content, surveying the target community about the perceived utility of the wiki and their satisfaction with it, and conducting supervised experiments to assess learnability and effectiveness.

Where usability evaluations are commonly conducted via experiments with test users, mathematical applications have also been subject to *conceptual* analyses; cf., e.g., a structured comparison of the behaviors of visual formula editors [121], or an added-value analysis of a formula search engine [92].

6.4 Applications with Math Mashup Techniques

This section presents mathematical knowledge mashups that have been developed in our research group. As we have argued above, any mathematical mashup system has to combine semantic annotations with formula support – where we can interpret parallel markup as semantic annotations of formula presentations. As this is a rather big investment, there are few systems that are capable of managing mathematical content in a way that enables mashups. These include computer algebra systems with document-oriented interfaces (such as the notebooks of Mathematica, a commercial

[10] "an approach to analysing the usability of information artefacts" [53], which has also been applied to mashups [71, 104]

system with an open API but a proprietary content representation format [13]) and the technology stack based on our own open, web standards compliant OMDoc format, including our own tools, presented in the following subsections, but also the ActiveMath eLearning system and its authoring tools [105].

Rather than solving individual specific problems using mashup technology, we aim at designing general, open, and extensible *architectures*. While our reference implementations do provide services for specific problems, such as looking up the definition of a symbol that occurs in a formula, we also envision that third parties extend and customize them with their own services.

JOBAD (section 6.4.1) is an architecture for integrating assistive services into semantically annotated HTML5 documents. Planetary (section 6.4.2) integrates JOBAD plus further frontend and backend components into a web-based content management system. The SAlly framework generalizes this approach to a mashup enabler for arbitrary document-oriented interfaces, including desktop applications.

6.4.1 JOBAD, a Toolkit for Integrating Assistive Services into Interactive Documents

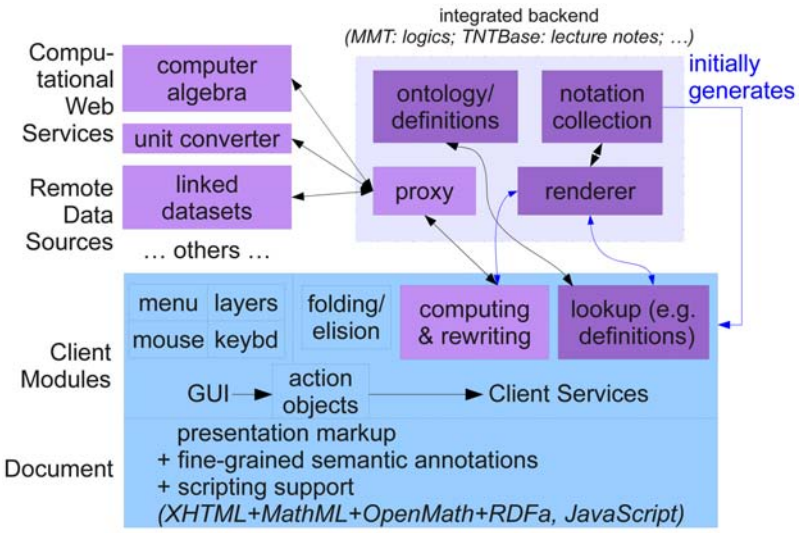

Fig. 6.4: The JOBAD architecture (concrete implementation systems/languages in *italics*)

JOBAD [88, 73] is an architecture for integrating interactive services into documents – services that assist readers in adapting the document's appearance to their

preferences or in looking up additional information and displaying it right in place, i.e. without forcing them to switch their attention away from the document.

The JOBAD architecture assumes documents that have been sufficiently prepared for interaction; section 6.3.2.3 states the particular requirements for the markup of mathematical formulae.[11] These annotations serve as hooks for assistive services, whose client side runs within the document. The level of interaction afforded by JOBAD depends on the service plugins loaded and on the depth of annotations in the documents; the set of functions applicable to a mathematical expression selected by the user – either by clicking on it or selecting a range as shown in figure 6.2 – depends on the structure of the selection and its annotations. For example, if a unit conversion service is available, it is only applicable to expressions that consist of a unit symbol applied to a quantity (cf. [61] for further background). If a concept definition lookup service is available, it is only applicable to symbols in a mathematical object, or to technical terms in a text, either of which is a concrete occurrence of that concept. Our JOBAD implementation makes the functions registered by its service plugins accessible in a context menu, at each time only showing those that claim to be applicable to the current selection. We have designed and implemented

1. client services that rely exclusively on annotations given in the rendered document – mostly for customizing its appearance, such as folding away subexpressions of long formulae, or document sections (as shown in figure 6.6) – and thus also work offline,
2. client services that retrieve additional information from background ontologies on the primary server backend that has also generated the document – such as definition lookup from a CD collection on the server, or in-place unit conversion using conversion rules given for unit symbols in such CDs –, and
3. client services that retrieve information from arbitrary external sources, with the primary server backend serving as a proxy due to the "same origin policy" [145, part 2]. This is the case for information lookup from Wolfram Alpha [69]; similarly, definition lookup could be extended to arbitrary external content dictionaries.

The client's communication with a web source may be as simple as downloading some data, e.g. the rendering of the definition of a symbol, from a URL, but it may also involve POSTing an expression to a web service and receiving a rewritten expression, as in the case of unit conversion. Information retrieved from the Web may be displayed in a tooltip-like popup – as for definition lookup and Wolfram Alpha lookup – or result in a part of the document, usually the selected mathematical expression, being rewritten – as for unit conversion.

[11] The original abbreviation means "JavaScript API for OMDoc-based Interactive Documents", but JOBAD-enabled documents do not have to be generated from an OMDoc source.

6.4.2 The Planetary eMath 3.0 System

Planetary is a comprehensive framework for semantic publishing and knowledge management, which has been instantiated prototypically in a variety of settings to validate the framework and to support communities. The portals realized with Planetary range from eLearning systems over scientific archives to theorem prover interfaces. All share a common basic architecture (see figure 6.5), which integrates previously developed components and interfaces into a central content management system (CMS; here: Drupal) that mediates all user interaction:

1. the TNTBase document database (cf. section 6.3.3.2) for storage and rendering,
2. the LaTeXML daemon (cf. section 6.3.3.3) for transforming TeX/LaTeX document fragments – not just formulae – to HTML5, including MathML formulae, SVG diagrams, and RDFa metadata,
3. sTeX [128, 94], a semantic variant of LaTeX that can be transformed to the OMDoc XML language (cf. section 6.3.3.5) and further to semantically annotated HTML (also covered by the JOMDoc library introduced in section 6.3.3.2),
4. and the JOBAD toolkit (see section 6.4.1) for embedding semantic services into web documents.

The Planetary system is mashup-based at the heart: The CMS supplies management and interaction at the "container level", i.e., without ever looking into the documents it manages, which is why we actually interpret the abbreviation "CMS" as *container* management system here. All other services are based on *structured document content* and background ontologies (providing, e.g., symbol definitions), which are provisioned by the TNTBase document database and the RDF triple store in Planetary. These subsystems, which are accessed via RESTful HTTP interfaces and may therefore be installed on different hosts, also perform semantic services, which are integrated (mashed up) into the mathematical documents via the JOBAD toolkit.

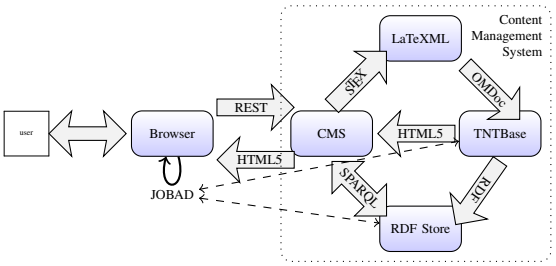

Fig. 6.5: Architecture of the Planetary System

Our different instances of the Planetary system have documents with different levels of annotation, and thus afford different levels of interaction, as explained in section 6.4.1: from simple folding and localized commenting services in a front-end system for the arXiv.org preprint library (see figure 6.6) to an in-place type reconstruction and elision of arguments and brackets for the fully formal LATIN atlas of logical theories [63].

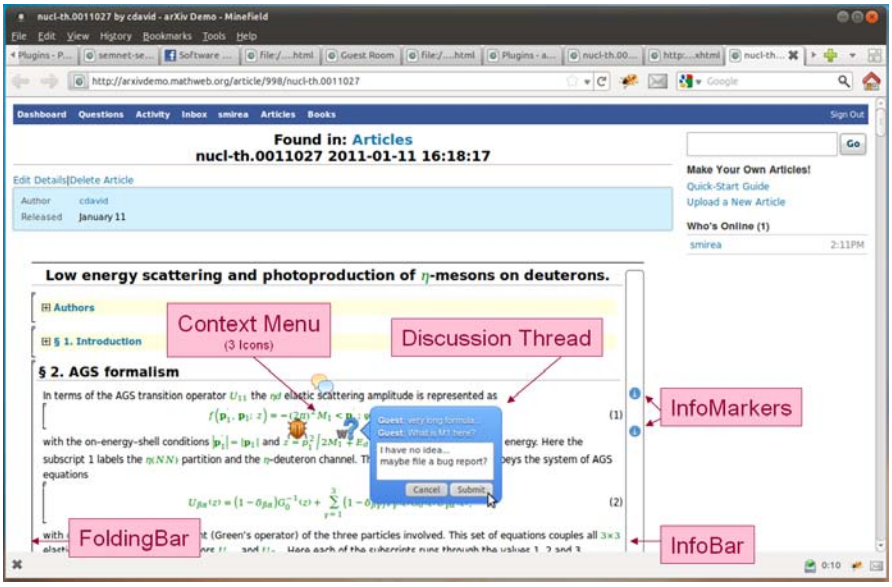

Fig. 6.6: Mashing up Semantic Services into arXiv.org Documents in Planetary

6.4.3 Semantic Allies: Mashing up User Interfaces

The basic mechanism behind the Planetary system introduced in the previous section is the mashup of semantic mathematical services in a web-based system. Planetary acts as a *document player* that presents documents to the user for interaction, where annotations in the documents provide hooks for JOBAD services. From this general perspective, we observe that document players for mathematical documents are ubiquitous not only on the Web, but also on the *desktop* and *mobile devices*. Furthermore, almost all documents with mathematical content are supported by some of these document players. Examples include spreadsheets, word processors, symbolic and numerical software, and even slide presenters. Users (readers as well as authors of documents) have usually invested heavily into becoming efficient in interacting with their preferred document players, and are therefore unlikely to leave them. Therefore, such document players are interesting candidates for integrating mathematical services beyond the built-in ones via mashup technology.

The SAlly framework [68] provides a mashup enabler for such situations. It builds on the observation that a service feels embedded into an application if it occupies a screen-area that is part of the area originally claimed by the application itself. This perception is amplified, if a service and its request refer to the local semantic objects. In particular, the service does not need to be implemented as application-specific invasive technology, and can be provided as a mashup service by a **semantic ally**, which in the SAlly framework has three components (see figure 6.7):

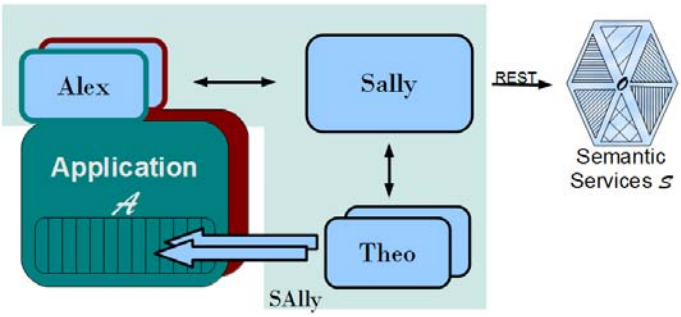

Fig. 6.7: SAlly as a Mashup Enabler for Semantic Allies

- a platform-independent semantic interaction manager "SAlly" (as a semantic ally), that has access to semantic services, and that
- partners with a set of invasive, thin, application-specific API Alex, that essentially only manages user interface events in the application, and that
- has access to a set of application-independent screen area managers Theo that can render the available services.

We have implemented the SAlly framework with Alexes for OO Calc, MS Excel, and a Theo based on XULRunner [31] – the layout and communication engine behind Mozilla Firefox and Thunderbird. Figure 6.8 shows the result of mashing up the JOBAD service for looking up definitions from a Planetary backend into an MS Excel spreadsheet using the SAlly framework.

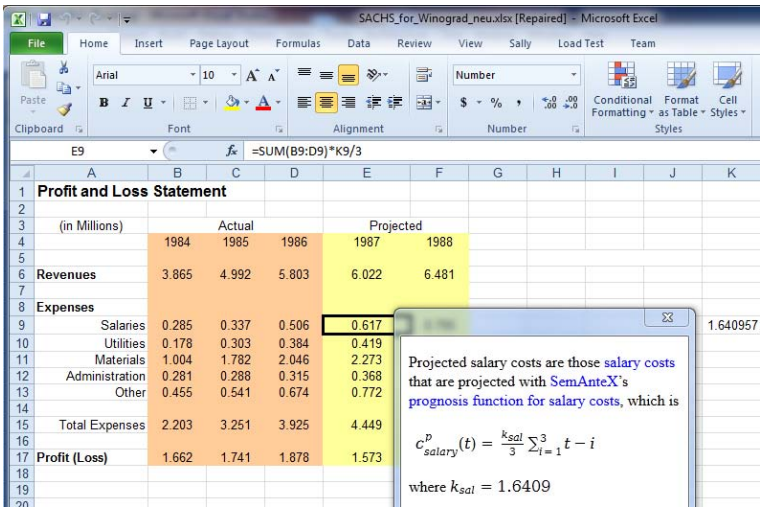

Fig. 6.8: Definition Lookup via SAlly in a Spreadsheet

6.5 Conclusion, Lessons Learned, and Future Prospects

As of summer 2012, proper mathematical *knowledge* mashups hardly exist (in contrast to data mashups that perform simple numeric computations), but, as this chapter proves, the technology to build them is there and waiting to be applied. In fact it is is already being applied in mashup-like settings, as our Planetary system demonstrates. Compared to other mashup application domains, the technology for mathematical knowledge mashups just took a few more years to consolidate. This is mainly due to the inherent complexity of mathematical formulae, which are a characteristic feature of mathematical applications and distinguish them from other applications.

But the added complexity of the application domain, which has delayed the provision of mashup services, will in our opinion also lead to added usefulness of the mashup systems when they eventually arrive. Math is generally considered hard – partly because of the embedded special-purpose sublanguage of formulae, and we claim that readers can reasonably expect more help in interacting with formulae. This is just what mathematical knowledge mashup services can offer. Indeed, mathematical formulae are often more semantic than mere language, and therefore afford more semantic services that can be mashed up.

Acknowledgements. The authors would like to thank Paul Libbrecht for his thorough review and constructive feedback.

References

1. Conference on intelligent computer mathematics (CICM). URL http://cicm-conference.org
2. Connexions – XML languages. URL http://cnx.org/help/authoring/xml
3. Connexions MathML editor. URL http://cnx.org/matheditor
4. Datamasher. URL http://www.datamasher.org
5. Developer apps showcase – data.gov. URL http://www.data.gov/developers/showcase
6. Digital Library of Mathematical Functions. URL http://dlmf.nist.gov
7. flot – attractive JavaScript plotting for jQuery. URL http://flot.googlecode.com
8. HELM. URL http://helm.cs.unibo.it
9. i2geo – i2g metadata. URL http://i2geo.net/xwiki/bin/view/About/I2GMetadata
10. i2geo – interoperable interactive geometry for europe. URL http://i2geo.net
11. ISSAC – international symposium on symbolic and algebraic computation. URL http://www.issac-conference.org/
12. JSXGraph – dynamic mathematics with JavaScript. URL http://jsxgraph.org
13. Mathematica. URL http://www.wolfram.com/products/mathematica/
14. MathML software – converters. URL http://www.w3.org/Math/Software/mathml_software_cat_converters.html
15. MathML software – editors. URL http://www.w3.org/Math/Software/mathml_software_cat_editors.html
16. MathOverflow. URL http://mathoverflow.net
17. MathPlayer. URL http://www.dessci.com/en/products/mathplayer

18. Mizar mathematical library. URL http://www.mizar.org/library
19. The n-Category Café. URL http://golem.ph.utexas.edu/category/
20. nLab. URL http://ncatlab.org/
21. OMDoc. URL http://omdoc.org
22. OpenMath software and tools. URL http://www.openmath.org/software/
23. The polymath blog. URL http://polymathprojects.org/
24. ProgrammableWeb. URL http://www.programmableweb.com
25. ProofWiki. URL http://www.proofwiki.org
26. Semantic MediaWiki. URL http://semantic-mediawiki.org
27. Tricki. URL http://www.tricki.org
28. Wolfram MathWorld. URL http://mathworld.wolfram.com
29. Wolfram—Alpha. URL http://www.wolframalpha.com
30. Wolfram—Alpha widgets. URL http://developer.wolframalpha.com/widgets/
31. Xulrunner runtime environment. URL https://developer.mozilla.org/en/XULRunner
32. The Yacas computer algebra system. URL http://yacas.sourceforge.net/
33. Zentralblatt MATH. URL http://www.zentralblatt-math.org
34. Mathematics Subject Classification MSC2010 (2010). URL http://msc2010.org
35. Working with MathML – Wolfram Mathematica 8 documentation (2011). URL http://reference.wolfram.com/mathematica/XML/tutorial/MathML.html
36. Mathematics subject classification (MSC) SKOS (2012). URL http://msc2010.org/resources/MSC/2010/info/
37. When can I use MathML? (2012). URL http://caniuse.com/mathml
38. ACTIVEMATH. URL http://www.activemath.org
39. Alama, J., Brink, K., Mamane, L., Urban, J.: Large formal wikis: Issues and solutions. In: Davenport et al. [67], pp. 133–148
40. American Mathematical Society: URL http://www.ams.org/mathscinet/help/mr_lookup_help.html
41. American Mathematical Society: MathSciNet Mathematical Reviews on the Net. URL http://www.ams.org/mathscinet/
42. Ankolekar, A., Krötzsch, M., Tran, T., Vrandečić, D.: The two cultures: Mashing up Web 2.0 and the Semantic Web. Web Semantics **6**(1), 70–75 (2008)
43. Anthony, L., Yang, J., Koedinger, K.R.: Evaluation of multimodal input for entering mathematical equations on the computer. In: G. van der Veer, C. Gale (eds.) CHI '05 extended abstracts on Human factors in computing systems, pp. 1184–1187. ACM (2005). DOI 10.1145/1056808.1056872
44. Asperti, A., Bancerek, G., Trybulec, A. (eds.): Mathematical Knowledge Management, MKM'04, no. 3119 in LNAI. Springer Verlag (2004)
45. Asperti, A., Geuvers, H., Natarajan, R.: Social processes, program verification and all that. Mathematical Structures in Computer Science **19**(5), 877–896 (2009)
46. Asperti, A., Padovani, L., Sacerdoti Coen, C., Guidi, F., Schena, I.: Mathematical knowledge management in HELM. Annals of Mathematics and Artificial Intelligence, Special Issue on Mathematical Knowledge Management, Kluwer Academic Publishers **38**(1–3), 27–46 (2003)
47. Ausbrooks, R., Buswell, S., Carlisle, D., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Ion, P., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Miner, R., Sargent, M., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical Markup Language (MathML) version 3.0. W3C Recommendation, World Wide Web Consortium (W3C) (2010). URL http://www.w3.org/TR/MathML3
48. Autexier, S., Calmet, J., Delahaye, D., Ion, P.D.F., Rideau, L., Rioboo, R., Sexton, A.P. (eds.): Intelligent Computer Mathematics, no. 6167 in LNAI. Springer Verlag (2010)
49. Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.): Intelligent Computer Mathematics, no. 5144 in LNAI. Springer Verlag (2008)

50. Baez, J.: Math blogs. Notices of the AMS p. 333 (2010). URL http://www.ams.org/notices/201003/rtx100300333p.pdf
51. Barany, M.J.: '[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: P. Ayers, F. Ortega (eds.) Proceedings of the 6th International Symposium on Wikis and Open Collaboration (WikiSym), ACM Press (2010). URL http://www.wikisym.org/ws2010/Proceedings/
52. Billingsley, W.H.: The intelligent book: technologies for intelligent and adaptive textbooks, focussing on discrete mathematics. Ph.D. thesis, University of Cambridge (2008). URL http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-719.pdf
53. Blackwell, A., Green, T.: Cognitive dimensions of notations resource site. URL http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/
54. Borwein, J., Stanway, T.: Knowledge and community in mathematics. The Mathematical Intelligencer **27**(2), 7–16 (2005)
55. Buswell, S., Caprotti, O., Carlisle, D.P., Dewar, M.C., Gaëtano, M., Kohlhase, M.: The Open Math standard, version 2.0. Tech. rep., The OpenMath Society (2004). URL http://www.openmath.org/standard/om20
56. Byrd, D.: Extremes of conventional music notation (2006). URL http://www.informatics.indiana.edu/donbyrd/CMNExtremes.htm
57. Caprotti, O., Dewar, M., Turi, D.: Mathematical service matching using description logic and OWL. In: Asperti et al. [44], pp. 73–87
58. Carette, J., Dixon, L., Sacerdoti Coen, C., Watt, S.M. (eds.): MKM/Calculemus Proceedings, no. 5625 in LNAI. Springer Verlag (2009)
59. Carette, J., Farmer, W.: A review of mathematical knowledge management. In: Carette et al. [58], pp. 233–246
60. Cartier, P.: Can we make mathematics universal as well as fully reliable? In: Autexier et al. [48]. Invited Talk
61. Cîrlănaru, M., Ginev, D., Lange, C.: Authoring and publishing of units and quantities in semantic documents. In: R. García Castro, D. Fensel, G. Antoniou (eds.) The Semantic Web: ESWC 2011 Workshops, no. 7117 in Lecture Notes in Computer Science, pp. 202–216. Springer Verlag, Heidelberg (2011). URL http://kwarc.info/clange/pubs/eswc2011-units.pdf
62. Connexions. URL http://cnx.org
63. Codescu, M., Horozal, F., Kohlhase, M., Mossakowski, T., Rabe, F.: Project abstract: Logic atlas and integrator (latin). In: Davenport et al. [67], pp. 289–291
64. Cohen, A.M., Cuypers, H., Verrijzer, R.: Mathematical context in interactive documents. Mathematics in Computer Science **3**(3), 331–347 (2010)
65. Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and Consume Linked Data with Drupal! In: A. Bernstein, D.R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, K. Thirunarayan (eds.) The Semantic Web – ISWC 2009, no. 5823 in LNCS, pp. 763–778. Springer Verlag (2009)
66. Costantini, M., Konovalov, A., Nicosia, M., Solomon, A.: GAP package OpenMath (2011). URL http://www.gap-system.org/Packages/openmath.html
67. Davenport, J., Farmer, W., Rabe, F., Urban, J. (eds.): Intelligent Computer Mathematics, no. 6824 in LNAI. Springer Verlag (2011)
68. David, C., Jucovschi, C., Kohlhase, A., Kohlhase, M.: Semantic Alliance: A framework for semantic allies. In: Jeuring et al. [87], pp. 49–64. URL http://kwarc.info/kohlhase/submit/mkm12-SAlly.pdf
69. David, C., Lange, C., Rabe, F.: Interactive documents as interfaces to computer algebra systems: JOBAD and Wolfram—Alpha. In: D. Delahaye, R. Rioboo (eds.) CALCULEMUS (Emerging Trends), pp. 13–30. Centre d'Étude et de Recherche en Informatique du CNAM (Cédric) (2010). URL https://svn.omdoc.org/repos/jomdoc/doc/pubs/calculemus10/jobad-cas.pdf
70. DBpedia. URL http://dbpedia.org

71. Ennals, R., Gay, D.: User-friendly functional programming for web mashups. In: ICFP '07 Proceedings of the 12th ACM SIGPLAN international conference on Functional programming, pp. 223–234. ACM, New York (2007). DOI 10.1145/1291151.1291187
72. Farmer, W.M.: MKM: A new interdisciplinary field of research. Bulletin of the ACM Special Interest Group on Symbolic and Automated Mathematics (SIGSAM) **38**(2), 47–52 (2004)
73. Giceva, J., Lange, C., Rabe, F.: Integrating web services into active mathematical documents. In: Carette et al. [58], pp. 279–293. URL `https://svn.omdoc.org/repos/jomdoc/doc/pubs/mkm09/jobad/jobad-server.pdf`
74. Ginev, D.: The LaTeXml daemon: Editable math for the collaborative web. URL `http://latexml.mathweb.org`
75. Ginev, D.: The Structure of Mathematical Expressions. Master's thesis, Jacobs University Bremen, Bremen, Germany (2011). URL `http://kwarc.info/people/dginev/publications/DeyanGinev_MScThesis.pdf`
76. Ginev, D., Stamerjohanns, H., Kohlhase, M.: The LaTeXML daemon: Editable math on the collaborative web. In: Davenport et al. [67], pp. 292–294. URL `https://svn.kwarc.info/repos/arXMLiv/doc/cicm-systems11/paper.pdf`
77. González Palomo, A.: QMath: A human-oriented language and batch formatter for OMDoc. In: OMDoc – An open markup format for mathematical documents [Version 1.2] [93], chap. 26.2. URL `http://omdoc.org/pubs/omdoc1.2.pdf`
78. González Palomo, A.: Sentido: an authoring environment for OMDoc. In: OMDoc – An open markup format for mathematical documents [Version 1.2] [93], chap. 26.3. URL `http://omdoc.org/pubs/omdoc1.2.pdf`
79. Hammond, K., Horn, P., Konovalov, A., Linton, S., Roozemond, D., Zain, A.A., Trinder, P.: Easy composition of symbolic computation software: A new lingua franca for symbolic computation. In: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation (ISSAC), pp. 339–346. ACM Press (2010)
80. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space, 1 edn. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, San Rafael, CA (2011). URL `http://linkeddatabook.com`
81. Heintz, B.: Die Innenwelt der Mathematik. Zur Kultur und Praxis einer beweisenden Disziplin. Springer Verlag, Wien (2000)
82. Hendriks, M., Libbrecht, P., Creus-Mir, A., Dietrich, M.: Metadata specification. Deliverable D2.4, Intergeo (2008). URL `http://i2geo.net/files/deliverables/D2.4-Metadata-Spec.pdf`
83. Hickson, I.: HTML5. W3C Working Draft, World Wide Web Consortium (W3C) (2012). URL `http://www.w3.org/TR/2011/WD-html5-20120329/`
84. Horn, P.: MuPAD OpenMath package. URL `http://mupad.symcomp.org/`
85. Horn, P., Roozemond, D.: OpenMath in SCIEnce: SCSCP and POPCORN. In: Carette et al. [58], pp. 474–479
86. IEEE Learning Technology Standards Committee: Standard for Learning Object Metadata. Tech. Rep. 1484.12.1, IEEE (2002)
87. Jeuring, J., Campbell, J.A., Carette, J., Dos Reis, G., Sojka, P., Wenzel, M., Sorge, V. (eds.): Intelligent Computer Mathematics, no. 7362 in LNAI. Springer Verlag (2012)
88. JOBAD framework – JavaScript API for OMDoc-based active documents. URL `http://jobad.omdoc.org`
89. JOMDoc project — Java library for OMDoc documents. URL `http://jomdoc.omdoc.org`
90. Kawata, T., Kataoka, M., Kai, H., Tamura, Y.: A MathML content markup editor on the xfy. In: E. Smirnova, S.M. Watt (eds.) Applications for Computer Algebra (2008)
91. Kohlhase, A., Kohlhase, M.: CPoint: Dissolving the author's dilemma. In: Asperti et al. [44], pp. 175–189. URL `http://kwarc.info/kohlhase/papers/mkm04.pdf`
92. Kohlhase, A., Kohlhase, M.: *Re*examining the MKM Value Proposition: From Math Web Search to Math Web *Re*Search. In: M. Kauers, M. Kerber, R. Miner, W. Windsteiger (eds.) Towards Mechanized Mathematical Assistants. MKM/Calculemus, no. 4573 in LNAI, pp. 266–279. Springer Verlag (2007). URL `http://mathweb.org/projects/mws/pubs/mkm07.pdf`

93. Kohlhase, M.: OMDoc – An open markup format for mathematical documents [Version 1.2]. No. 4180 in LNAI. Springer Verlag (2006). URL http://omdoc.org/pubs/omdoc1.2.pdf
94. Kohlhase, M.: Using LaTeX as a semantic markup format. Mathematics in Computer Science **2**(2), 279–304 (2008). URL https://svn.kwarc.info/repos/stex/doc/mcs08/stex.pdf
95. Kohlhase, M., Corneli, J., David, C., Ginev, D., Jucovschi, C., Kohlhase, A., Lange, C., Matican, B., Mirea, S., Zholudev, V.: The planetary system: Web 3.0 & active documents for stem. Procedia Computer Science **4**, 598–607 (2011). DOI 10.1016/j.procs.2011.04.063. URL https://svn.mathweb.org/repos/planetary/doc/epc11/paper.pdf. Finalist at the Executable Paper Grand Challenge
96. Kohlhase, M., Müller, C., Rabe, F.: Notations for living mathematical documents. In: Autexier et al. [49], pp. 504–519. URL http://omdoc.org/pubs/mkm08-notations.pdf
97. Kohlhase, M., Rabe, F., Zholudev, V.: Towards MKM in the Large: Modular Representation and Scalable Software Architecture. In: S. Autexier, J. Calmet, D. Delahaye, P. Ion, L. Rideau, R. Rioboo, A. Sexton (eds.) Intelligent Computer Mathematics, *Lecture Notes in Computer Science*, vol. 6167, pp. 370–384. Springer (2010)
98. Kovalchuk, A., Levitsky, V., Samolyuk, I., Yanchuk, V.: The formulator MathML editor project: User-friendly authoring of content markup documents. In: Autexier et al. [48], pp. 385–397
99. Lakatos, I.: Proofs and Refutations. Cambridge University Press (1976)
100. Lange, C.: Enabling Collaboration on Semiformal Mathematical Knowledge by Semantic Web Integration. No. 11 in Studies on the Semantic Web. AKA Verlag and IOS Press, Heidelberg and Amsterdam (2011)
101. Lange, C.: Krextor – an extensible framework for contributing content math to the web of data. In: Davenport et al. [67], pp. 304–306. URL http://kwarc.info/clange/pubs/krextor-system.pdf
102. Lange, C.: Ontologies and languages for representing mathematical knowledge on the semantic web. Semantic Web Journal (2012). URL http://www.semantic-web-journal.net/content/ontologies-and-languages-representing-mathematical-knowledge-semantic-web
103. Lange, C., Ion, P., Dimou, A., Bratsas, C., Sperber, W., Kohlhase, M., Antoniou, I.: Bringing mathematics to the web of data: the case of the mathematics subject classification. In: E. Simperl, P. Cimiano, A. Polleres, O. Corcho, V. Presutti (eds.) The Semantic Web, no. 7295 in Lecture Notes in Computer Science, pp. 763–777. Springer (2012). DOI 10.1007/978-3-642-30284-8_58. URL http://kwarc.info/clange/pubs/eswc2012-msc-skos.pdf
104. Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: J. Quemada, G. León, Y.S. Maarek, W. Nejdl (eds.) Proceedings of the 17th WWW conference, pp. 581–590. ACM Press (2009)
105. Libbrecht, P.: What you check is what you get: Authoring with jEditOQMath. In: 10th International Conference on Advanced Learning Technologies (ICALT), pp. 682–686. IEEE (2010)
106. Libbrecht, P., Kortenkamp, U., Mercat, C.: I2Geo: a web-library of interactive geometry. In: P. Sojka (ed.) Towards Digital Mathematics Library, pp. 95–106. Masaryk University Press, Brno (2009)
107. Manzoor, S., Libbrecht, P., Ullrich, C., Melis, E.: Authoring Presentation for OpenMath. In: M. Kohlhase (ed.) Mathematical Knowledge Management, MKM'05, no. 3863 in LNAI, pp. 33–48. Springer Verlag (2006)
108. Marchiori, M.: The mathematical semantic web. In: A. Asperti, B. Buchberger, J.H. Davenport (eds.) Mathematical Knowledge Management, MKM'03, no. 2594 in LNCS, pp. 216–223. Springer Verlag (2003). Keynote

109. Marquès, D., Eixarch, R., Casanellas, G., Martínez, B.: WIRIS OM tools: a semantic formula editor. In: P. Libbrecht (ed.) Mathematical User Interfaces Workshop 2006 (2006). URL http://www.activemath.org/~paul/MathUI06
110. MathDox – interactive mathematics. URL http://www.mathdox.org
111. MathJax: Beautiful math in all browsers. URL http://mathjax.com
112. McCabe, D., Garrett, A.: MediaWiki – api. URL http://www.mediawiki.org/w/index.php?title=API:Main_page&oldid=574291
113. Melis, E., Andrés, E., Büdenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M., Ullrich, C.: ActiveMath: A generic and adaptive web-base learning environment. International Journal of Artificial Intelligence in Education **12**(4), 385–407 (2001)
114. Melis, E., Goguadze, G., Libbrecht, P., Ullrich, C.: Culturally adapted mathematics education with ActiveMath. AI & Society **24**(3), 251–265 (2009)
115. Melis, E., Weber, M., Andrès, E.: Lessons for (pedagogic) usability of eLearning systems. In: A. Rossett (ed.) Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, pp. 281–284. AACE (2003)
116. Miller, B.: LaTeXML: A LaTeX to XML converter. URL http://dlmf.nist.gov/LaTeXML/
117. MONET – Mathematics on the net. URL http://monet.nag.co.uk
118. Müller, C.: Adaptation of Mathematical Documents. Ph.D. thesis, Jacobs University Bremen (2010). URL http://kwarc.info/cmueller/papers/thesis.pdf
119. Nakano, H., Nagai, T., Yunpeng, J., Wannous, M., Kita, T.: Mashup approach for embedding algebraic manipulations, formulas and graphs in web pages. In: Learning Environments and Ecosystems in Engineering Education, pp. 691–694. IEEE (2011)
120. OpenMath Society: OM 2 Presentation MathML XSLT test release. URL http://www.openmath.org/standard/omxsl/
121. Padovani, L., Solmi, R.: An investigation on the dynamics of direct-manipulation editors for mathematics. In: Asperti et al. [44], pp. 302–316
122. PlanetMath.org – math for the people, by the people. URL http://planetmath.org
123. PlanetMath Redux.org – math for the people, by the people. URL http://planetmath.mathweb.org
124. Pólya, G.: How to Solve it. Princeton University Press (1973)
125. Robinson, A., Voronkov, A. (eds.): Handbook of Automated Reasoning, vol. I and II. Elsevier Science and MIT Press (2001)
126. The SCIEnce project – Symbolic Computation Infrastructure for Europe. URL http://www.symcomp.org/
127. SCIEnce EU Project: The Popcorn OpenMath Representation (2009). URL http://java.symcomp.org/FormalPopcorn.html
128. Semantic Markup for LaTeX. URL http://trac.kwarc.info/sTeX/. Project Homepage
129. Sutcliffe, G., Suttner, C.: The CADE ATP system competition. URL http://www.cs.miami.edu/~tptp/CASC/
130. Sutcliffe, G., Suttner, C.: The state of casc. AI Communications **19**(1), 35–48 (2006)
131. truenumbers. URL http://www.truenum.com
132. Trzeciak, J.: Writing Mathematical Papers in English. Gdańskie Wydawnictwo Oświatowe (1995)
133. Ullrich, C.: Pedagogically Founded Courseware Generation for Web-Based Learning. LNCS. Springer Verlag (2008). URL http://www.springerlink.com/content/k604618p5351/
134. Unicode, Inc.: Unicode (2009). URL http://www.unicode.org/versions/Unicode5.2.0/
135. Urban, J., Alama, J., Rudnicki, P., Geuvers, H.: A wiki for Mizar: Motivation, considerations, and initial prototype. In: Autexier et al. [48], pp. 455–469
136. Vismor, T.: Viewing mathematics on the internet (2012). URL https://vismor.com/documents/site_implementation/viewing_mathematics/

137. Vrandečić, D., Lange, C., Hausenblas, M., Bao, J., Ding, L.: Semantics of governmental statistics data. In: Proceedings of WebSci'10: Extending the Frontiers of Society On-Line. Web Science Trust (2010). URL http://journal.webscience.org/400/
138. W3C Math Working Group: Example XSLT code for transforming XML languages for the web. URL http://web-xslt.googlecode.com
139. Wikipedia, the free encyclopedia. URL http://www.wikipedia.org
140. Knowledge management (2009). URL http://en.wikipedia.org/w/index.php?title=Knowledge_management&oldid=329227520
141. Portal: Mathematics. URL http://en.wikipedia.org/w/index.php?title=Portal:Mathematics&oldid=329137789
142. MathML (Software support/Web browsers) (2012). URL http://en.wikipedia.org/w/index.php?title=MathML&oldid=482267822#Web_browsers
143. WIRIS Editor – a tool for graphical edition of mathematical formulas. URL http://www.wiris.com/content/view/20/
144. Zacchiroli, S.: User interaction widgets for interactive theorem proving. Ph.D. thesis, Università di Bologna (2007)
145. Zalewski, M.: Browser security handbook. Tech. rep., Google (2010). URL http://browsersec.googlecode.com
146. Zholudev, V., Kohlhase, M., Rabe, F.: A [insert xml format] database for [insert cool application]. In: Proceedings of XML Prague 2010, pp. 317–339 (2010). URL http://kwarc.info/vzholudev/pubs/XMLPrague.pdf
147. Zholudev, V., et al.: TNTBase – restful api (2010). URL http://tntbase.org/wiki/restful
148. Zimmer, J.: MathServe – a framework for semantic reasoning services. Ph.D. thesis, Universität des Saarlandes (2008)