JACOBS UNIVERSITY BREMEN

Interactive Documents and Computer Algebra Systems: JOBAD and Wolfram|Alpha[1]

by

Catalin David

Guided Research: Final Report
supervised by: Prof.Dr. Michael Kohlhase

in the
School of Engineering and Science
Computer Science

July 2010

[1]We would like to thank Wolfram Research for providing us with a pioneer grant for accessing the Wolfram|Alpha web service API.

JACOBS UNIVERSITY BREMEN

Executive Summary

School of Engineering and Science
Computer Science

Guided Research Thesis

by Catalin David

In today's world, interactivity and per-user satisfaction are some of the most common trends guiding design of services, being increasingly implemented, especially on the Internet. From simple design features that let the user arrange the workspace in a certain layout to systems that provide more complex services regarding the content that is provided, everything works on a certain set of rules, learns from the user behavior so that, in the end, the user will receive news, ads, notifications or any other provided service only about the area(s) of interest or only about some pre-selected topics (we see Google Ads and Facebook as perfect examples in this domain). The JOBAD architecture takes care of embedding such services in mathematical documents and, besides displaying them on the Internet, actually lets the user interact with the document. The services provided are structured in client-side applications (e.g.: term folding) and server-interactive applications (e.g.: definition lookup), but none of them treats the interaction with a Computer Algebra System (CAS) which would provide additional information regarding a mathematical formula or part of it. Wolfram|Alpha [46] is classified as an "answer engine" by its authors, relying, besides the Mathematica [27] backend (which provides the actual CAS), on a structured knowledge base through which the user can actually find the answers to their questions. The goal of the proposed project is to provide a service which would facilitate the communication between the user and any CAS (Wolfram|Alpha in particular).

Acknowledgements

I would like to thank Christoph Lange and Florian Rabe for the great guidance and direct support throughout the whole guided research. Furthermore special thanks for Prof.Dr.Michael Kohlhase and the whole KWARC group for helping me with their suggestions, and evaluations. I would also want to thank Klaus Sutner for making available to us the *NB2OMDoc* Mathematica package and last, but not least Wolfram Research for providing us with a pioneer grant for accessing the Wolfram|Alpha API.

Contents

1

Introduction

1.1 Active Mathematical Documents

Throughout the years, the standard for spreading knowledge, mathematical knowledge in particular, was printed documents (books, articles etc.). In the last years, there has been a shift in this standard, from printed documents to electronic documents, and, with the spread of Internet, web documents that would provide certain extra features besides the printed counterpart. With electronic documents, one can go "green" and not print them, but more interesting is the chapter of web documents which provides more content (e.g.: the actual meaning of a formula) and format (e.g.: one does not have to carry books around any more, a simple mobile phone will be great at that) related features which put the whole knowledge available online at your fingertips. But, with all that knowledge, the current tendency on the Internet market is to provide more possibilities for the users to receive more and more data from different sources, thus intoxicating the user with useless information. By setting up contextual and intelligent filters, the companies try to help the users feel less distracted by the huge amount of information that the Internet is generating every second so that the users actually get only the information that they want and that information fits their interests and needs.

Despite all the progress in knowledge and information propagation, mathematics is still regarded as one of the hard topics of publishing, not as much as in writing it, but reading and understanding it. Due to the abstract and minimalistic tendency people have towards expressing their knowledge, the readers usually find themselves in the position of ignoring certain characteristics and properties of the objects being talked about. Also, because of the hierarchy of knowledge used in math, many of the terms defined and used in the mathematical documents actually depend on other terms which, in turn depend on more documents, thus constraining the reader to having a solid background before reading a paper.

The point where the two above mentioned perspectives meet is the JOBAD architecture [16, 10], whose purpose is to facilitate the integration of diverse (web) services - via the Web 2.0 technology of *mashups* [33, 3] - for mathematical documents. The work we are presenting right now regards enriched interactive mathematical documents presented on the web. Our vision of an *interactive document* is a document that the user can not just read, but adapt according to his preferences and interests *while* reading it — not only by customizing the display of the rendered document in the browser, but also by changing the mathematical notations (which requires re-rendering) or retrieving additional information from services on the web. Consider a student reading lecture notes: whenever he is not familiar with a mathematical symbol occurring in some formula, JOBAD enables him to look up its definition without opening another document, but right in his current reading context. Or consider the problem of converting between physical units (e. g., imperial vs. SI). Instead of manually opening a unit converter website and copying numbers into its entry form, we have enabled an in-place unit conversion.

The documents are displayed as a combination of XHTML, RDFa (annotations made in OMDoc[20, 41], preserved using this format in XHTML) and MathML [44] that we intend to make (inter)active and, therefore, customizable, thus enriching the user experience. The JOBAD project's [14, 23, 10] purpose is to provide services for the user that will affect both the display and the content of the active document by retrieving additional data from different web services. The architecture of JOBAD is also interactive: some of the services it provides are only based on the data in the document itself (only client side), but, most of the services provided have also a server backend that provides additional data asynchronously. The server backends (such as MMT [35] and TNT-Base [48, 42]) will provide the actual document and for security considerations, a simple proxy to access the other web services that are needed by the JOBAD functionality.

Also, regarding the architecture, it is modular and, therefore, easily extensible, giving other developers the ability to develop customized modules for different tasks. The extension of services provided by the server interaction provides a whole new set of user content possibilities, as the server can combine the pre-existing knowledge with fresh knowledge that it can retrieve from different services. This is where Computer Algebra Systems (CAS) come into place, as they can provide a lot more background information for a term or an equation (e.g. plots, roots, simplifications etc.).

A Computer Algebra System is a software package which is used for symbolic mathematics, the manipulation of mathematical formulae by computers in a symbolic manner, as opposed to manipulating approximations of numerical quantities represented by those symbols. CAS are usually used to automate tedious and difficult algebraic manipulation tasks. Computer Algebra systems often include facilities for graphing equations and provide a programming language for the user to define his/her own procedures. Examples of popular systems include Maple, Mathematica, and MathCAD. Computer Algebra systems can be used to simplify rational functions, factor polynomials, find the solutions to a system of equation, and various other manipulations. In Calculus, they can be used to find the limit of, symbolically integrate, and differentiate arbitrary equations.

In this work we present a new JOBAD service that can interact with CASs. The generic service should provide interaction with any CAS available, provided that there is a common interaction language between them, but, for our experiment, we instantiated this service to connect to the Wolfram|Alpha using the Wolfram|Alpha web service API. Thus JOBAD can provide background information for a symbol, a term of the equation, e.g.: we can use it to simplify a certain term, plot it, solve equations.

1.2 Related work

Similar to JOBAD, the ActiveMath project [1] deals with *aggregate documents* which are retrieved from a knowledge base depending on the user's topics of interest (what the user wants to learn) and its prerequisites. It is presented as a platform for learning mathematics in school and university.

MathDox [7] is an XML based format for interactive mathematical documents which can be transformed to interactive mathematical web pages using the MathDox Player. MathDox uses OpenMath for semantic representation and was actually designed, among others, to interact with CAS-es like Mathematica, Maxima and GAP via OpenMath phrasebooks. For MathDox, in order to evaluate a certain mathematical formula in a CAS, several steps need to be taken by the MathDox player for transforming the underlying formats of the document (DocBook, OpenMath, MONET etc.) to HTML which embeds the result of the CAS query.

Interactive exercises have been developed by both ActiveMath [12] and MathDox which rely on a user's answer and a solution checker in order to return feedback to the user. In order to evaluate the user input, ActiveMath needs a CAS to check for correctness and relies on one of the following: Yacas, Wiris-CAS or Maxima. Still, the features provided by the JOBAD architecture are more inclined towards modularity and client-side services that neither presuppose a single backend nor a particularly powerful one, whereas the two aforementioned projects are less modular (we are referring here at the addition of new services by third party sources, which JOBAD can handle very well) and more of the required computation is done on the server instead of the client.

1.3 Technologies used

In this section, we give a brief introduction to the technologies which were used during the Guided Research experiment and which will be used later on this paper.

On the client side:

- **JavaScript [29]:** scripting language used on the client side in the development of dynamic websites. It constitutes, alongside with XHTML and MathML, the foundation for developing active mathematical documents. Although a proof of concept can be found at [34] that active documents can be created without JavaScript, this is only recent and is not very efficient and, most important, is not as versatile as JavaScript. Therefore, JavaScript is regarded as one of the foundations of this project.
- **jQuery [18]:** a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and AJAX interactions for rapid web development. It is widely used on the Internet and is regarded as a basic JavaScript tool. It is heavily used in the development of the JOBAD client side, as it simplifies the code and the tasks that need to be achieved.
- **XPath [47]:** the XML Path Language is a query language for selecting nodes from an XML document, enabling querying of XML documents (thus XHTML). XPath was defined by the W3C (World Wide Web Consortium) and has proven to be useful in places where the functionality provided by *jQuery* was not enough.

On the server side:

- **Java Servlet [39]:** A Java program that runs as part of an HTTP server and responds to requests from clients. This was chosen as a programming language for the proxy, as this can be easily integrated with the existing server backends (TNTBase and MMT).
- **Apache Tomcat [4]:** a Java servlet container for Java classes. It provides an HTTP server environment for Java code to run.
- **JOMDoc [17]:** a Java API for OMDoc documents which allows, among others, the transformation of OMDoc documents to XHTML

On the document level:

- **MathML** [43]**:** the Mathematical Markup Language is an XML application for describing mathematical notation and capturing both its structure and content. This, alongside with JavaScript and XHTML represents the foundation of the JOBAD architecture. Presentation MathML focuses on the display of mathematical expressions, while the Content MathML deals with the semantic meaning of mathematics. The two representations of mathematical formulae can be linked together via "parallel markup", a concept which puts the two XML trees side by side, the Content MathML being an annotation of the Presentation counterpart (as browsers can only render Presentation MathML).
- **OpenMath** [32]**:** a markup language for specifying the meaning of mathematical formulae that can be used to complement MathML. In addition to that, it can be encoded in both XML and binary formats.
- **OMDoc** [31]**:** or Open Mathematical Documents is a semantic markup format for mathematical documents. While MathML only covers mathematical formulae and the related OpenMath standard only supports formulae and "content dictionaries" containing definitions of the symbols used in formulae, OMDoc covers the whole range of written mathematics.

Communication Protocol:

- **AJAX** [2]**:** or "Asynchronous JavaScript And XML", provides the interface for communication between the server and client. It provides asynchronous communication between the client and the server, making the browser experience as interactive as desktop one, while providing a connection with the Internet.

2

The existing JOBAD infrastructure

The initial motivation of JOBAD was to provide a simple and interactive way to view and adapt (to users' preferences) the lecture notes provided in the General Computer Science course at Jacobs University Bremen by Prof. Dr. Michael Kohlhase. Once with progress in technology and understanding of the concepts, this view has evolved, revolving more towards the concept of e-learning and providing added-value services for the users. These added-value services would be oriented more towards the context and the semantics of mathematical documents (in this case, our lecture notes) such as bracket elision and the folding of mathematical terms in formulas. This was the basis of JOBAD and the first services are the two presented above and, in addition to that, the unit conversion service.

After getting a better grasp of the concepts once time has passed and the first real use cases appeared, we have gained more experience and realized that JOBAD needs a change in terms of modularity. Thus, I have redesigned the entire JOBAD codebase in the spring of 2010 to be more modular and to allow the addition and load of new JOBAD services by third parties in a non-obtrusive way. In Figure 2.3 one can see a detailed diagram of the JOBAD architecture and the interactions it has with the external sources. The JOBAD architecture is structured in three parts, the core JavaScript code that can be used as an interface for developing added value services (see Figure 2.1), a library registry and a service registry. Since the JOBAD framework is intended to be as modular as possible and it is not desired to load very much JavaScript code on the client side, as this might decrease performance and user satisfaction, we have required having a library registry which provides basic functions to be used by several services (e.g.: display the popup, fill the popup with this content) and a service registry which provides access to the (third party) services that can be loaded dynamically.

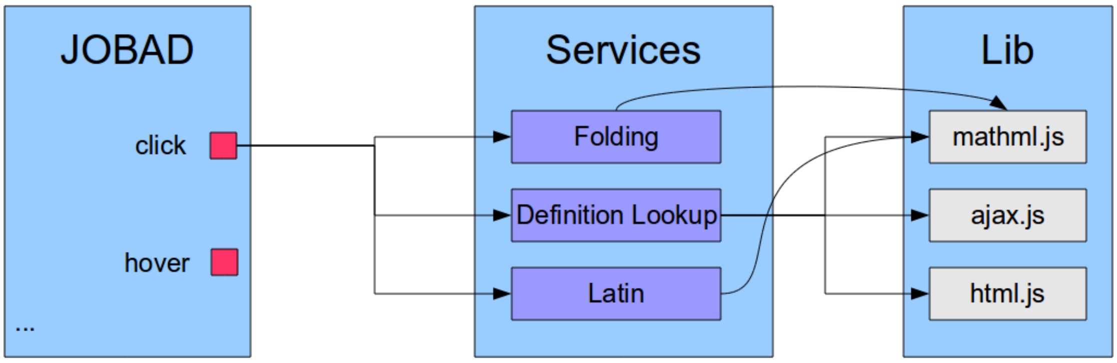


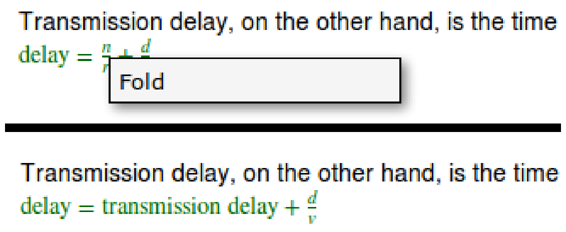
FIGURE 2.1: JOBAD internal architecture

2.1 The core of JOBAD

The core of JOBAD provides an interface between the services and the document being viewed. Thus, the JavaScript code in the core allows the loading and unloading of services, loads,computes and displays the context menu and triggers events for basic interactions (e.g.: click, hover) on the document level. The loading is done on the document level, requiring extra scripts from the backend which are then executed, while unloading requires just deleting this code from the memory.

The context menu is based on an open-source context menu which was slightly modified so that it can be dynamically populated, as the context menu is populated with regard to the current context in which the user has right-clicked. For example, when one would right-click on an element, the respective functions of each loaded service will be called which provide, in turn, a tree that represents the context menu entries that should be displayed (tree because the context menu can have multiple levels and a service may provide multiple interactions).

The last part which is taken care of by the core of JOBAD is providing the interface with both the services and the document. On a document level in modern browsers, JavaScript events are handled via a process called “event bubbling”. When the user clicks on an element, the event is registered on the innermost element on which the user clicked and then the event “bubbles” until the event reaches the document level, calling the click event handlers for all the parents of the clicked document until one is actually implemented. We are using this concept such that we are interacting only on a document level, when the event has bubbled up to the last level, thus realizing the interface with the document. Once such an event has been triggered, the registry of loaded services is accessed and the respective functions are called which handle the request further (in Figure 2.2, one can see the interaction provided by the folding service).


Transmission delay, on the other hand, is the time
delay = $\frac{n}{r} + \frac{d}{}$

Transmission delay, on the other hand, is the time
delay = transmission delay + $\frac{d}{v}$

FIGURE 2.2: The folding service interaction

2.2 The library registry: *Lib*

As previously mentioned, there is library registry which provides JavaScript code with common functions which can be used in certain services. The choice of a modular library (instead of a monolithical one) has been made mainly because of extensibility (one can easily add new JavaScript code0 and size (the responsiveness of web applications is important and loading large quantities of JavaScript code for only one function is slow and does not make sense). In the future, this will be readjusted once the library increases in size in order to support namespaces, as right now one can find namespace, redefining and code pollution issues, as each entry in the registry might define a function with the same name, while the JavaScript code only takes in consideration the last loaded one.

2.3 The *Services* registry

In Figure 2.3 you can see an updated diagram of the entire JOBAD architecture, with the components in red being the ones which were added, while the proxy communication with the external services has been redesigned.

Generically speaking, the (third party) services which were and will be developed follow a certain pattern in order to mimic the inheritance behavior: all the services developed interact with the document via the interface defined in the JOBAD core. The new services need to clone the interface and implement the appropriate methods (e.g.: the method to be called when the document is clicked). In this registry reside all the available services, including folding, bug reporting, definition lookup (example can be viewed in Figure 2.4) and others.

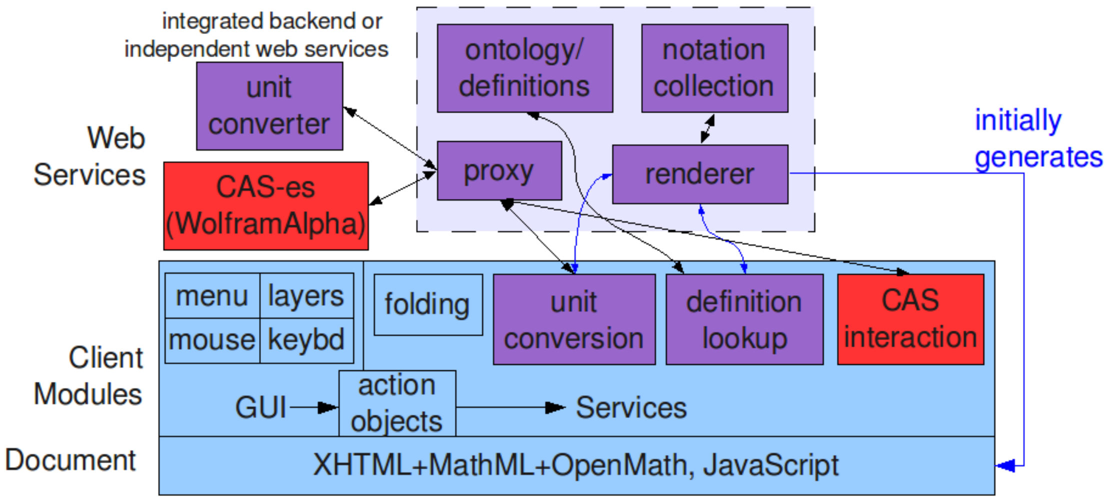
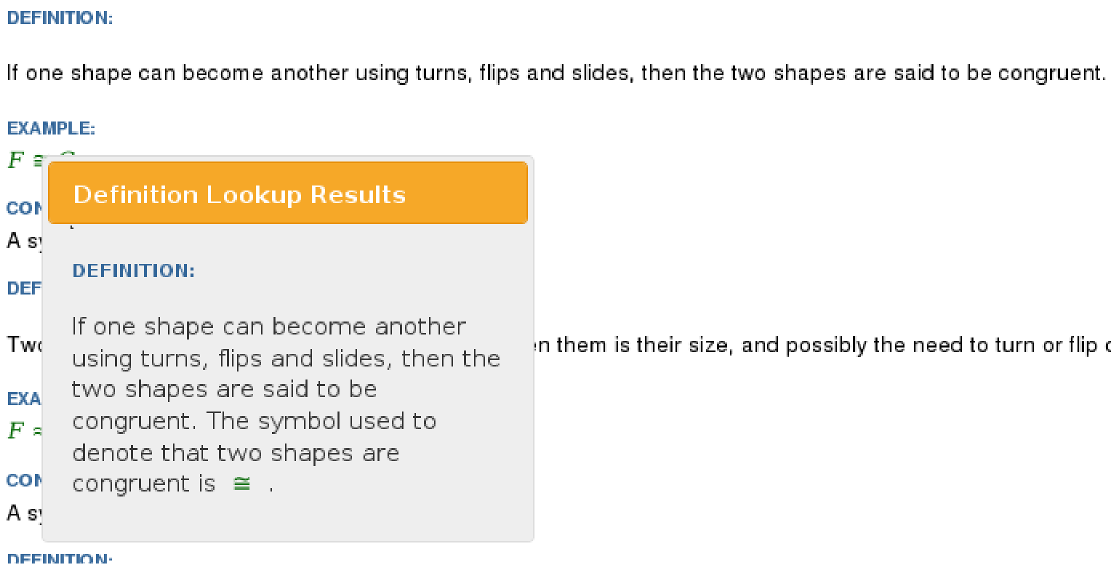


FIGURE 2.3: JOBAD architecture


DEFINITION:

If one shape can become another using turns, flips and slides, then the two shapes are said to be congruent.

EXAMPLE:

F ≅

CON

A s

DEF

Two

n them is their size, and possibly the need to turn or flip

EXA

F ≈

CON

A s

DEFINITION:

Definition Lookup Results

DEFINITION:

If one shape can become another using turns, flips and slides, then the two shapes are said to be congruent. The symbol used to denote that two shapes are congruent is ≅ .

FIGURE 2.4: Definition Lookup example

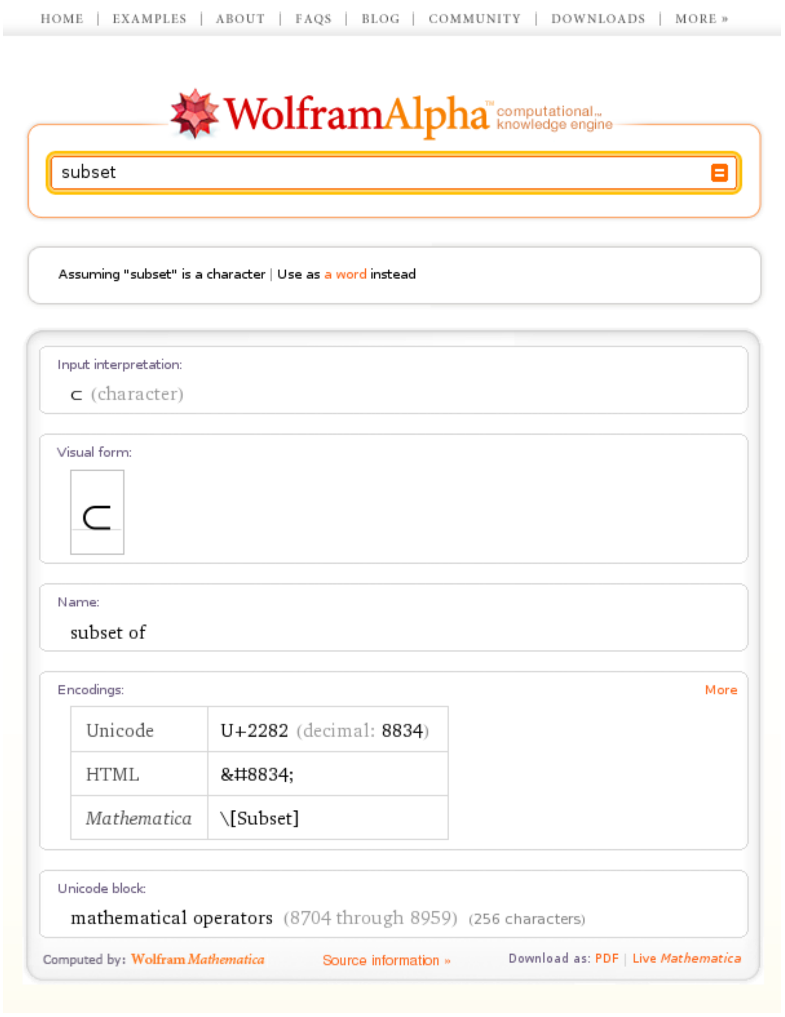
3

Computer Algebra Services for JOBAD

3.1 A simple introduction to Wolfram|Alpha

The research in the field of CAS has received mass recognition in May 2009 when Wolfram|Alpha [46], a computational knowledge engine, was launched. An example of a CAS is Mathematica [27], whose 7th version was released in the first quarter of 2009, a system that provides many possibilities in interacting with mathematical formulae. Wolfram|Alpha is based on two primary resources for the answers it provides (as it is classified as being an "answer engine"), the Mathematica backend and the knowledge base. From a mathematical point of view, the way this works is that Wolfram|Alpha always tries to return everything that it can compute (factorization, roots, plot) or already knows about a certain formula (as one can see in Figure 3.1).

An integration of the services provided by Wolfram|Alpha with the JOBAD architecture makes sense, as this would facilitate the users' immediate access to more information regarding the formulae that the user explores in a document, thus providing, besides the already existing information services (e. g.: definition lookup), another way of acquiring background information regarding the topic, thus making it easier for users to understand complex mathematical formulae. This data is instantly computed and is available for access via the Wolfram|Alpha website or via a webservice API specially designed for developers. Still, Wolfram|Alpha is only one example of a CAS (one can see Maxima [28] as a possible candidate) and the JOBAD architecture should not be confined to only using this one. Another example of a similar system with which JOBAD could interact are those CAS-es that deal with OpenMath content and which can be reached via the SCSCP protocol [13].


HOME | EXAMPLES | ABOUT | FAQS | BLOG | COMMUNITY | DOWNLOADS | MORE »
WolframAlpha computational... knowledge engine
subset

Assuming "subset" is a character | Use as a word instead

Input interpretation:
⊂ (character)

Visual form:
⊂

Name:
subset of

Encodings:

Unicode	U+2282 (decimal: 8834)
HTML	&#8834;
Mathematica	\[Subset]

Unicode block:
mathematical operators (8704 through 8959) (256 characters)

Computed by: Wolfram *Mathematica* Source information » Download as: PDF | Live *Mathematica*

FIGURE 3.1: Sample result provided by Wolfram|Alpha when looking up “subset”

3.2 Architecture extension

The work envisioned with this project has two main parts that, in turn, regard the improvement and extension of the already existing JOBAD architecture [10]. The idea is to develop another module for the already existent JOBAD architecture that will allow the user to interact with more web-driven mathematics, via the Wolfram|Alpha computational knowledge engine. The extension also regards a generalized method of a “Send To” menu that will allow the user to redirect an annotated MathML fragment (formula) to some other sources of information, in this case, a CAS, in particular, Wolfram|Alpha. Wolfram|Alpha was chosen as an initiator for the “Send To” method, as this seems the most useful, rational and complex choice for a user who wants to look up mathematical content on the web, as Wolfram|Alpha is also capable of plotting different functions, identify equations, terms etc. Still, this would be only a test case for the menu, as this can be further expanded and further destinations for the “Send To” menu can be provided. This new extension, in theory, can be extended to work with any CAS, the only constraints being that the CAS and the mathematical content in the document should have a common language — such as OpenMath —, or that there should be a one-to-one mapping between these languages and that there is a way to specify the wanted query from the CAS in the interface. The difference between Wolfram|Alpha and other CAS-es is that Wolfram|Alpha will automatically return plots, derivatives, related formulas, while these, if supported, have to be explicitly asked for in the other CAS-es. This functionality needs to be embedded in the respective service GUI elements and will have to be adjusted per CAS.

Another improvement of the architecture comprises modifications to the user interface that, right after the document has loaded and JOBAD will come in to place, it will display a notification on the page that will allow the user to select the necessary and wanted services available for the respective document. Then, the data will be stored for later access so that each and every time the user will display the document again, the offered services are seamlessly loaded and the document is prepared for interaction according to the user's preferences.

The steps required for the extension of JOBAD are related to two important tasks, first, querying the CAS system (in this case the Wolfram|Alpha engine) and as for the improvement of the user interaction, retrieving the results from the system and displaying them for the user.

3.3 Querying Wolfram|Alpha

In the initial phase of the project, there were two ideas around which the retrieving of data revolved. First of all, there was the brute force way, query the Wolfram|Alpha website, wait for it to load (as it contains a lot of JavaScript and AJAX requests) and retrieve the content on the webpage via XPath or something similar and display it in the associated dialog box. Still, there were some issues regarding this:

- The Wolfram|Alpha website only provides instant results image-wise. The images are generated on-the-fly and then deleted shortly after the AJAX request has been completed; therefore, there was no way to return the images which are rather important in the case of functions.
- The results were displayed via JavaScript and AJAX which would mean that retrieving the loaded page through the Java proxy would be hard, if not impossible.
- One can not set the content of the retrieved results which is, by default images, even for mathematical formulae.

- Further content cannot be retrieved or filtered (we are interested in retrieving the meaning of the formulas, not just the graphical representation)

3.3.1 Wolfram|Alpha API

Due to the above mentioned reasons, we had to come up with a better solution to this issue, have applied for and received a research pioneer grant consisting of a key to use the Wolfram|Alpha webservice which exposes more functionality. The Wolfram|Alpha service provides a web-based API for clients to integrate the computational and presentation capabilities of Wolfram|Alpha into their own applications or web sites. The Wolfram|Alpha webservice allows one to query the database as if one were to query the actual website, but also providing additional functionality.

For example the output of the Wolfram|Alpha webservice can be filtered according to the developer's wish:

- Visual Representations:
  - *Image* – unlike the actual Wolfram|Alpha website, these images are stored for some time and can be later accessed (also via the cache mechanism provided by Wolfram|Alpha)
  - *HTML* – the content is retrieved in HTML + CSS + JavaScript format, just as if it were retrieved from the Wolfram|Alpha website
  - *PDF* – one can choose to retrieve links to PDF files that store the information
- Textual Representations:
  - *Plain text*: this is the text format that you see in the "Copyable plaintext" popup that appears when you click on results on the Wolfram|Alpha site
  - *Mathematica Input*: this is the format that is used in Mathematica (the computational engine behind Wolfram|Alpha) to compute and generate some of the results for graphs, plots etc. (e.g.: *D[Sqrt[x], x]* in order to compute the derivative of $\sqrt{X}$)
  - *Mathematica Output*: this is the format of the output, in the Mathematica language, which can be later on fed to Mathematica or a Mathematica parser (e.g.: considering the same example as above, as the derivative of $\sqrt{X}$ is $\frac{1}{2\sqrt{X}}$, the Mathematica output would be *1/(2 Sqrt[x])*)
  - *MathML*: this is an additional format of textual representation that one can query for and returns a Presentation MathML version of an equation, which can be later on rendered in the browser.

- *ExpressionML [9]*: is a format that provides a way to represent Mathematica expressions in XML.
- *XML*: a generic XML representation of the results. More details on this format are forthcoming.

Although the options are rather numerous, they regard the presentation part of formulae, rather than the content of the formulae, which would be more interesting for computers, as this can provide some knowledge.

3.3.2 The Proxy

As previously stated, we have received a grant consisting of an application ID which allows us to query the Wolfram|Alpha webservice for free, as long as it remains in research purposes. Still, due to the Same Origin Policy [30] which is implemented by most of the browsers (and Firefox in particular, as the Gecko engine – which powers Firefox – is the only engine capable of rendering MathML without additional plugins), one can only retrieve content via GET / POST methods from websites which are running on the same domain and port number as the page being served. This is a very important security measure, as it does not allow a client (browser) to retrieve content from unknown hosts. Still, this has the disadvantage that *mash-ups* have to be created on the server side and then displayed or transfered to the user (at [25] one can find an article which helps in designing a mashup). Therefore, as for all the existing web *mash-ups*, a proxy had to be created which would handle the communication with the external sources: CAS in general, Wolfram|Alpha and the dependent services in particular.

The purpose of the proxy is to interact with the "outside world" of the application, retrieve necessary content, fit it together nicely and then return it to the user. Given the choice of different formats in which Wolfram|Alpha can return the result, this proxy divides the task in two subtasks: a content oriented task and a general information retrieval task.

Regarding the content oriented task of the proxy, the workflow can be visualized in Figure 3.2 and will be also explained in the next sentences. The JOBAD architecture was thought to be used, at first, in the General Computer Science lecture notes at Jacobs University which are written using sTEX(semantically enriched TEX) [21, 22] and are transformed and hosted in TNTBase [48, 42]. Still, the system is not constrained to that and can be used in much more useful purposes. For example, the LATIN project (Logic ATlas and INtegrator) [24], which aims to use a "logics as theories/translations as morphisms" approach to achieve the interoperability of both system behavior and represented knowledge (the Logic Integrator), and to obtain a comprehensive and interconnected network of formalizations of logics of computational logic systems (the Logic Atlas). Another example in this area can be given from history: in the beginnings of the 20th century, a group of (mainly) French mathematicians has started writing the basis of set theory and published under the common pseudonym: Nicolas Bourbaki. But, for this collection of books, there is no digitized version which would allow the users to explore (e.g.: the basis of set theory) properly.

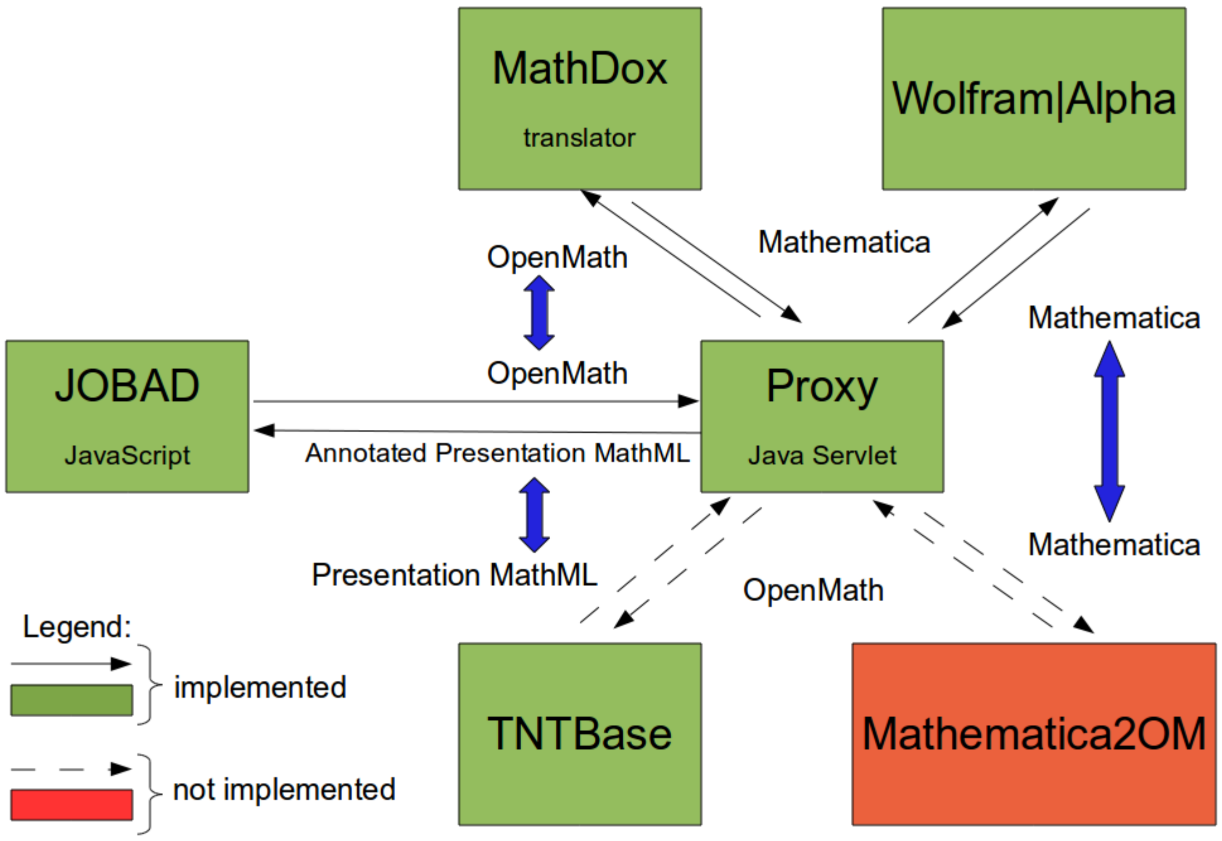


FIGURE 3.2: Proxy architecture - context oriented

The test case for the integration of Wolfram|Alpha services into JOBAD was the previously mentioned lecture notes which are usually displayed in MathML (Presentation) and also have content annotations in OpenMath. As the annotations are made up to a per symbol level, it is easy for the Wolfram|Alpha service of the JOBAD architecture to find the associated OpenMath representation of the selected text and make a POST request to the proxy which runs on the same domain and port (due to the “Same Origin Policy”). The proxy will then determine if the content is OpenMath and, in this case, will send a request to a webservice running on the MathDox [26] website which will translate the OpenMath content to a Mathematica expression. As Wolfram|Alpha is based on Mathematica (the plots, expansions etc. are computed via Mathematica), the Mathematica language is easier to understand by the engine and the computed results are more relevant to the search, as no Natural Language Processing techniques need to be employed to transform the input (e.g.: on a basic level, an input as "Sqrt[x]" might produce more relevant results than "square root of x"; for this simple test case, the results are identical, but for more complex queries, NLP tagging might not work) . So, the converted OpenMath expression is then passed to Wolfram|Alpha for evaluation in two steps: the first request is for Mathematica output (a representation of the formula in Mathematica language) and is directed towards the content and meaning of the formula, while the second request is sent in order to retrieve pictures and a Presentation MathML representation of the results.

Given that the first query was successful (which can be easily verified in the result of Wolfram|Alpha query), the system should proceed in transforming the retrieved Mathematica content to a displayable form (Presentation MathML), while still preserving the associated content annotation. For this, the following possibilities have been investigated:

- *NB2OMDoc*: Developed by Klaus Sutner NB2OMDoc [40] is a Mathematica package that is able to transform Mathematica code (version 4.2, latest version is 7) to OMDoc (version 1.2). The disadvantages of this system would be that it requires Mathematica to be installed on the proxy computer and that it is designed for an old format of both Mathematica and OMDoc. In addition to that, one would have to transform (render) the OMDoc content to MathML (Presentation and Content MathML), step which would be provided by TNTBase and JOMDoc, a Java API for OMDoc documents (and illustrated in the picture).

- *Mathematica web service*: As pointed out here [`http://reference.wolfram.com/mathematica/XML/tutorial/MathML.html`], Mathematica is capable of exporting its formulas to both Content and Presentation MathML. So, one can design a web service that would start Mathematica, input a formula, convert it to MathML and then retrieve the result. This is not feasible, as the Mathematica files (with extension *nb*) have a proprietary format and extracting content from that file is not easy. Also, another drawback is that one would have to start Mathematica each time (as we are not aware of a Mathematica daemon) which, even on a new computer, takes more than 10 seconds which makes a webservice not user friendly. An example in this area is WITM [45], Web Interface to Mathematica which provides a Mathematica interaction inside the browser. Still, the main constraint is that WITM (and similar attempts) is intended to allow a small number of licenced users access to Mathematica kernels remotely, but not simultaneous (a large number of users might mean interference in the result)

- *Sentido formula editor*: Developed by Alberto Gonzalez Palomo as part of the Sentido [36] editor, browser and environment for OMDoc, it is a JavaScript extension that allows the translation between different mathematical formulae representation formats. This would mean that all the translation between the different formats (Mathematica to OMDoc) should be done on the client side. The drawbacks of using this method is that the entire library is necessary for this and there seems to be no interface to just transform between the different formats, without enabling the other features.

Since each of the methods presented above has its own (major) drawbacks and would require more time to integrate than the one allotted for the Guided Research, we consider the integration of a Mathematica to OpenMath/OMDoc/MathML translator as future work.

3.4 Displaying the results

Once the data is retrieved in a displayable format (images, Presentation MathML, rendered MathML from Mathematica), it needs to be displayed. Following the design pattern used in the definition lookup service, we decided to use the same jQuery UI [19] widget that allows the developer to populate a dialog with the necessary data, in this case an adjusted XML representation of the results provided by Wolfram|Alpha. The expansion is made in place, where the user clicked and allows the user to move the dialog around (examples can be viewed in Figures 2.4 for definition lookup, 3.3 for the module loading utility and 3.6 for the Wolfram|Alpha lookup).

3.5 Preserving Document Settings

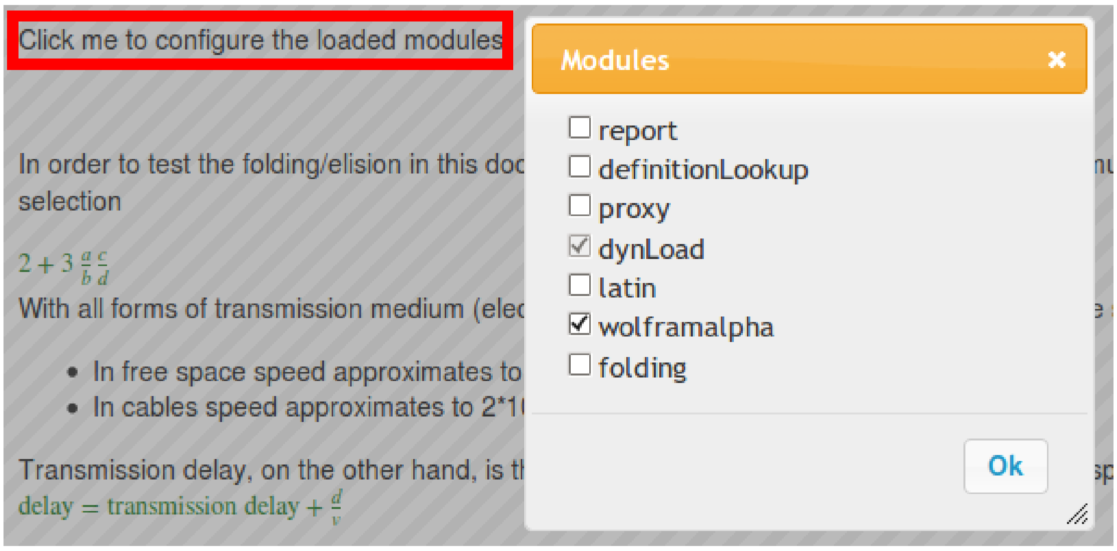
The last part of this project regarded the extension of the interface with another service that would allow the dynamic loading of other services, thus providing even more freedom of configuration on the user side, leading towards more personalized active mathematical documents.

This extension first adds a text at the top of the document (“Click me to configure the loaded modules”) and uses the same jQuery UI dialog (as one can see in Figure

3.3), only that this time, the dialog is made modal: everything else except the dialog is grayed out and it does not allow access to the underlying document until either the form is confirmed (via the *Ok* button) and the necessary modules are loaded or the dialog is closed via the **x** button. In addition to that, we imagine students that might access a document or documents on the same domain for multiple times and having to load the same modules over and over might become irritating and annoying. Therefore, in addition to loading the necessary services, this module also stores the loaded services in a cookie for further usage and each time a page is loaded, the cookie is retrieved and the modules that have been loaded at the last access of the web page are loaded again. The list of available services is not static, but rather dynamic and each time the top of the page is clicked, a request is sent to the server, asking for the available services.

3.6 Example test case

In the following section we present an example workflow for a test document. The user arrives at the test document and no services (besides the service loading system) are loaded, resulting in no obvious functionality. Once the user clicks the top of the page text which allows the loading of additional modules, a request is sent to the server asking for the available services, the dialog is populated and pops up and allows the user to check the *wolframalpha* checkbox (see Figure 3.3). Once both the *wolframalpha* and the


Click me to configure the loaded modules

In order to test the folding/elision in this doc
selection

$2 + 3\frac{a}{b}\frac{c}{d}$

With all forms of transmission medium (elec

- In free space speed approximates to
- In cables speed approximates to 2*1

Transmission delay, on the other hand, is t

delay = transmission delay + $\frac{d}{v}$

Modules ×

- [ ] report
- [ ] definitionLookup
- [ ] proxy
- [x] dynLoad
- [ ] latin
- [x] wolframalpha
- [ ] folding

Ok

FIGURE 3.3: The user loads the *wolframalpha* service

additional *folding* services are loaded, the user proceeds to the document and after each right click is presented with a contextual menu, dynamically created for that document element. Assuming the user would right click on a mathematical fragment which is $\sqrt{x}$, with the associated XHTML fragment presented in Figure 3.5 which contains both

Presentation MathML and annotations in OpenMath format, he would then receive a visual confirmation of his action via a context menu, as one can see in Figure 3.4.

If a user were to access the Wolfram|Alpha website and search for the Mathematica

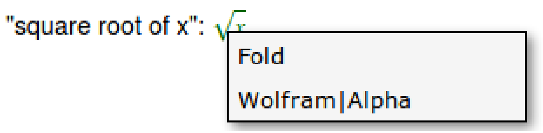
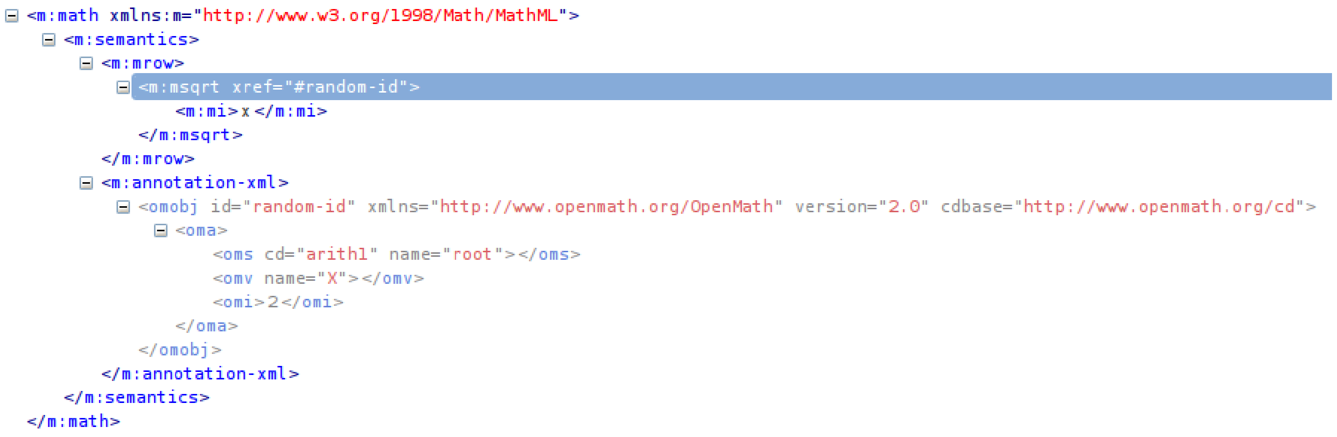


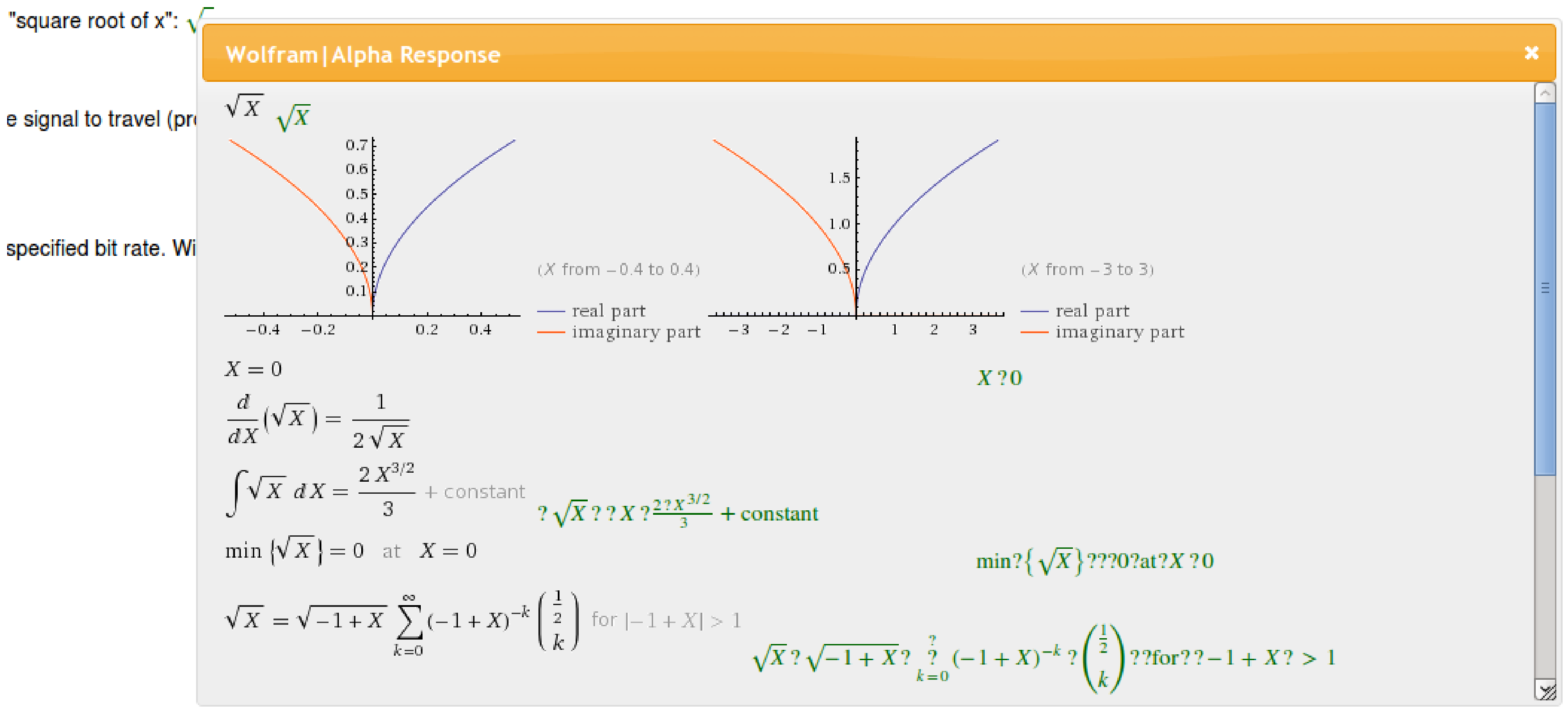
FIGURE 3.4: The user performs a right on the $\sqrt{x}$ symbol



```
<m:math xmlns:m="http://www.w3.org/1998/Math/MathML">
  <m:semantics>
    <m:mrow>
      <m:msqrt xref="#random-id">
        <m:mi>x</m:mi>
      </m:msqrt>
    </m:mrow>
    <m:annotation-xml>
      <omobj id="random-id" xmlns="http://www.openmath.org/OpenMath" version="2.0" cdbase="http://www.openmath.org/cd">
        <oma>
          <oms cd="arith1" name="root"></oms>
          <omv name="X"></omv>
          <omi>2</omi>
        </oma>
      </omobj>
    </m:annotation-xml>
  </m:semantics>
</m:math>
```

FIGURE 3.5: The Presentation MathML and the associated OpenMath content annotation representations of $\sqrt{x}$

representation of the OpenMath fragment, in this case *Sqrt[x]*, the result page would look like Figure 3.7. After the request is processed, the Wolfram|Alpha content is retrieved on the client side and the user will experience something resembling Figure 3.6.



"square root of x": $\sqrt{}$

e signal to travel (pr

specified bit rate. Wi

Wolfram|Alpha Response

$\sqrt{X}$ $\sqrt{X}$

(X from −0.4 to 0.4)

real part
imaginary part

(X from −3 to 3)

real part
imaginary part

$X = 0$ $X ?0$

$\frac{d}{dX}(\sqrt{X}) = \frac{1}{2\sqrt{X}}$

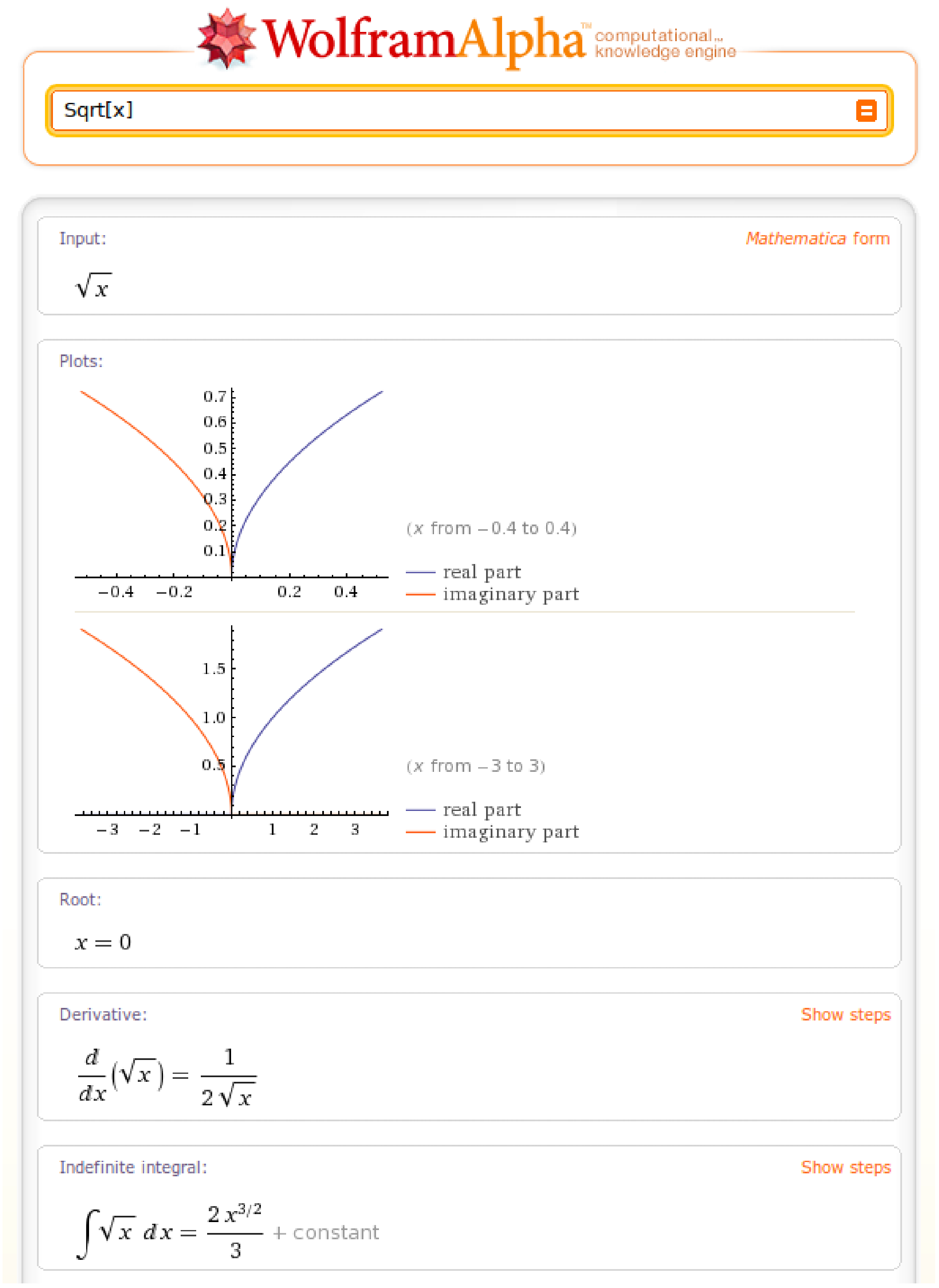
$\int \sqrt{X}\, dX = \frac{2X^{3/2}}{3}$ + constant ? $\sqrt{X}$? ? X ? $\frac{2?X^{3/2}}{3}$ + constant

min $\{\sqrt{X}\} = 0$ at $X = 0$ min?$\{\sqrt{X}\}$???0?at?X ?0

$\sqrt{X} = \sqrt{-1+X}\sum_{k=0}^{\infty}(-1+X)^{-k}\binom{\frac{1}{2}}{k}$ for $|-1+X| > 1$

$\sqrt{X}$? $\sqrt{-1+X}$? $\overset{?}{\underset{k=0}{?}}$ $(-1+X)^{-k}$? $\binom{\frac{1}{2}}{k}$??for??−1 + X? > 1

FIGURE 3.6: Part of the Wolfram|Alpha results embedded into the original document



Sqrt[x]

Input:

$\sqrt{x}$

Plots:

(x from -0.4 to 0.4)

(x from -3 to 3)

Root:

$x = 0$

Derivative:

$$\frac{d}{dx}\left(\sqrt{x}\right) = \frac{1}{2\sqrt{x}}$$

Indefinite integral:

$$\int \sqrt{x}\, dx = \frac{2x^{3/2}}{3} + \text{constant}$$

FIGURE 3.7: Part of the Wolfram|Alpha website search results for *Sqrt[x]*

4

Future Work and Conclusion

4.1 Implementation Improvements

The main feature needed for improving and extending the service is to allow the interaction with other CAS-es. We can see examples of such CAS interaction in the ActiveMath and MathDox projects which already provide some use case scenarios. Still, these use case scenarios are constrained on already generated content. One of the future applications of JOBAD is to be integrated into annotated XHTML + MathML generated documents. We see the ongoing effort of the `arXMLiv` project as relevant in this direction [38, 37, 5] which tries to transform the entire collection of articles stored at `http://arxiv.org` into a content based form. We consider this as being the perfect example of "math in the wild", a use case scenario for which JOBAD would fit well.

The high-level view of this project is that with minimum effort for a developer, any CAS can be integrated in this service and, from a user point of view, easy, direct and fast interaction with the CAS is available. From a developer point of view, the extension would allow the integration of a CAS by providing an URL of the webservice on which the CAS is running, the services it provides (the functions which can be called; e.g.: *Plot(formula)*), a function that computes the requested URL (if needed), a parser function for results and a placeholder for the results. From the user point of view, the interaction with the CAS must be intuitive, the GUI elements should not be obtrusive in any way and should only appear when is actually looking for further content or seems confused. The available services should be dynamically created and listed on-the-fly and provide background or additional information which would otherwise be hard to obtain for the user.

Even for this current (and particular) implementation of the CAS integration services, there are several possible extensions that regard, in part either the proxy or the JavaScript service. The most ardent issue at this point is found on the proxy and regards the translation from Mathematica to Presentation MathML with annotations, since having content annotations enables the computer to actually understand the mathematical meaning behind. Although there are some possible solutions to this problem, the integration of them with the current architecture seems to be tedious and, even if successful, each of these possible solutions has some major drawbacks. Investigations on this issue are still made and we are confident a solution will be found soon (the ExpressionML language in which Wolfram|Alpha can output seems more easily "translatable" to OpenMath via XSLT).

Another issue that needs to be resolved is the filtering of the content retrieved. Although Wolfram|Alpha will retrieve everything that it knows about the query, it will also retrieve the same content in different formats (images and MathML), content which needs to be filtered and the best possible format should be chosen (which should be Presentation MathML with content annotations). This should be done because Wolfram|Alpha returns graphs as images, as well as rendered formulas which are of no use since we have either the Presentation MathML or an annotated version of Presentation MathML. This can be done either on the proxy (by removing from the response) or directly in the client (by hiding the unnecessary images). As mentioned before. additions to the JavaScript code should be made in order to be more permissive with CAS-specific GUI elements.

4.2 Future Services

4.2.1 Requirements for a generalized Wolfram|Alpha service

In the end, this service can be generalized in order to work with any CAS or an interface to CAS, such as the SCSCP implementation mentioned earlier. For this purpose, the CAS and the web document should have a common language through which they can communicate, such as OpenMath, which is now used to represent annotations to the Presentation MathML and of which the SCSCP implementation is aware. If there is no possibility for a direct language connection between the document and the CAS, one must employ a *two-way* translator, once for document language to CAS language, as the one designed by the MathDox team for converting OpenMath to Mathematica, and the reverse, which we are currently lacking.

4.2.2 Other Services

One more pending issue is to be able to query for both natural language and mathematics at the same time. While querying for natural language and mathematics is possible right now (but independently), modifications need to be made to both the proxy and the JavaScript code in order to get the selected text, filter out the irrelevant XHTML fragments (the MathML code), while on the proxy the content needs to be separated and part-by-part, if needed, converted to Mathematica syntax.

As before, the services that will be developed can be split in two categories: client side services and services which require a server backend. On the client side, in the near future we have envisioned services that perform highlighting of the matching brackets (when hovering over a bracket, the matching bracket would be highlighted). This service is useful for the case in which one looks at a very complicated formula and needs help in deepening the understanding of it. Another similar client side service looks at variable highlighting. Imagine a complicated formula resulted from the interaction with a CAS (the derivative of a formula) with many occurrences of similarly notated variables (x_1, x_2 etc.). Even for a proficient mathematical reader, things might get complicated and this is where this service would come into place: when hovering over a certain mathematical variable, it would highlight all the occurrences of that variable in the formula. This can be even extended to the document level and using more advanced Natural Language Processing tools, one can actually retrieve and highlight the definition of a certain symbol (e.g.: "Let x be a natural number") [11]. For the future, we envision a service that can embed, besides the current CAS lookup, also definition lookup of math symbols and related information linked by RDFa annotations [8].

4.3 Conclusion

The concept of interactive content, alongside with per-user adaptation of the served information, by providing additional knowledge from external sources seems to be a more and more popular idea in today's world. This idea is challenged and employed more and more often by the creation of web-mashups, a technology of Web 2.0 which will soon become the standard, resulting in interactive documents on the Internet (e.g.: combining housing offers with geographic data provided by Google Maps).

The main purpose of this project is to extend this version of the JOBAD framework via a service that will enrich the user experience and exploit an unimplemented feature, by combining the vast amount of information provided by CAS with the existing content and putting it all together at the fingertips of the user. Wolfram|Alpha has provided a perfect testcase for this scenario, with its extensive knowledge base which can be of great help for a novice user exploring advanced content. We have made the first steps towards providing in-place solutions for exploring mathematical documents and we report on progress towards embedding the computing power of any full-featured CAS at the touch of a button, in a web browser. Its final version will permit users to send symbols or entire annotated mathematical formulae for evaluation to a CAS, the result of which will be displayed contextually.

Bibliography

[1] ActiveMath. URL: http://www.activemath.org (visited on 06/05/2010).

[2] *AJAX - descriptive article*. URL: http://www.adaptivepath.com/ideas/essays/archives/000385.php (visited on 06/05/2010).

[3] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, et al. "The two cultures: Mashing up Web 2.0 and the Semantic Web". In: *Web Semantics* 6.1 (2008), pp. 70–75.

[4] *Apache Tomcat website*. URL: http://tomcat.apache.org/ (visited on 06/05/2010).

[5] *arXMLiv Build System*. http://arxmliv.kwarc.info. URL: http://arxmliv.kwarc.info.

[6] Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, et al., eds. *MKM/Calculemus 2009 Proceedings*. LNAI 5625. Springer Verlag, 2009.

[7] Hans Cuypers, Arjeh M. Cohen, Jan Willem Knopper, et al. "MathDox, a system for interactive Mathematics". In: *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008*. Vienna, Austria: AACE, 2008, pp. 5177–5182. URL: http://go.editlib.org/p/29092.

[8] Catalin David, Michael Kohlhase, Christoph Lange, et al. "Publishing Math Lecture Notes as Linked Data". In: *ESWC*. Ed. by Lora Aroyo, Grigoris Antoniou, and Eero Hyvönen. Vol. II. Lecture Notes in Computer Science 6089. Springer, 2010, pp. 370–375. arXiv: 1004.3390.

[9] *ExpressionML specification*. URL: http://reference.wolfram.com/mathematica/ref/format/ExpressionML.html (visited on 06/05/2010).

[10] Jana Giceva, Christoph Lange, and Florian Rabe. "Integrating Web Services into Active Mathematical Documents". In: *MKM/Calculemus 2009 Proceedings*. Ed. by Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, et al. LNAI 5625. Springer Verlag, 2009, pp. 279–293. URL: https://svn.omdoc.org/repos/jomdoc/doc/pubs/mkm09/jobad/jobad-server.pdf.

[11] Deyan Ginev, Constantin Jucovschi, Stefan Anca, et al. “An Architecture for Linguistic and Semantic Analysis on the arXMLiv Corpus”. In: *Applications of Semantic Technologies (AST) Workshop at Informatik 2009*. 2009. URL: http://www.kwarc.info/projects/lamapun/pubs/AST09_LaMaPUn+appendix.pdf.

[12] George Goguadze and Erica Melis. “Feedback in ActiveMath Exercises”. In: *International Conference on Mathematics Education (ICME)*. 2008.

[13] Peter Horn and Dan Roozemond. “OpenMath in SCIEnce: SCSCP and POPCORN”. In: *MKM/Calculemus 2009 Proceedings*. Ed. by Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, et al. LNAI 5625. Springer Verlag, 2009, pp. 474–479.

[14] *JOBAD Framework – JavaScript API for OMDoc-based active documents*. URL: http://jomdoc.omdoc.org/wiki/JOBAD (visited on 08/05/2010).

[15] *JOBAD Framework – JavaScript API for OMDoc-based active documents*. URL: http://jomdoc.omdoc.org/wiki/JOBAD.

[16] *JOBAD Framework – JavaScript API for OMDoc-based active documents*. http://jomdoc.omdoc.org/wiki/JOBAD. 2008. URL: http://jomdoc.omdoc.org/wiki/JOBAD.

[17] *JOMDoc Project — Java Library for OMDoc documents*. http://jomdoc.omdoc.org. seen Feb. 2010. URL: http://jomdoc.omdoc.org.

[18] *jQuery main website*. URL: http://jquery.com/ (visited on 06/05/2010).

[19] *jQuery UI website*. URL: http://jqueryui.com/ (visited on 06/05/2010).

[20] Michael Kohlhase. OMDoc*: An open markup format for mathematical documents (latest released version)*. Specification, http://www.omdoc.org/pubs/spec.pdf. URL: http://www.omdoc.org/pubs/spec.pdf.

[21] Michael Kohlhase. “Semantic Markup for TEX/LATEX”. In: *Mathematical User Interfaces*. Ed. by Paul Libbrecht. 2004. URL: http://www.activemath.org/~paul/MathUI04.

[22] Michael Kohlhase. “Using LATEX as a Semantic Markup Format”. In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: https://svn.kwarc.info/repos/stex/doc/mcs08/stex.pdf.

[23] Michael Kohlhase, Jana Giceva, Christoph Lange, et al. “JOBAD – Interactive Mathematical Documents”. In: *AI Mashup Challenge 2009, KI Conference*. Ed. by Brigitte Endres-Niggemeyer, Valentin Zacharias, and Pascal Hitzler. 2009. URL: https://svn.omdoc.org/repos/jomdoc/doc/pubs/ai-mashup09/jobad.pdf.

[24] *LATIN: Logic Atlas and Integrator*. https://trac.omdoc.org/latin/. Project Homepage. URL: https://trac.omdoc.org/latin/.

[25] *Mashup Styles, Part 1: Server-Side Mashups*. URL: http://java.sun.com/developer/technicalArticles/J2EE/mashup_1/ (visited on 06/05/2010).

[26] *MathDox webservice for OpenMath to Mathematica conversion*. URL: http://mathdox.org/phrasebook/mathematica/eval_openmath_native (visited on 06/05/2010).

[27] *Mathematica*. URL: http://www.wolfram.com/products/mathematica/index.html (visited on 06/05/2010).

[28] *Maxima - A GPL CAS based on DOE-MACSYMA*. web page at http://maxima.sourceforge.net. URL: http://maxima.sourceforge.net.

[29] *Mozilla Developer Center - JavaScript*. URL: https://developer.mozilla.org/en/JavaScript (visited on 06/05/2010).

[30] *Mozilla Developer Center documentation for Same Origin Policy*. URL: https://developer.mozilla.org/En/Same_origin_policy_for_JavaScript (visited on 06/05/2010).

[31] *OMDoc*. project page at http://omdoc.org. URL: http://omdoc.org.

[32] *OpenMath*. web page at http://wiki.openmath.org. 2009.

[33] Tim O'Reilly. *What is Web 2.0*. Sept. 2005. URL: http://oreilly.com/web2/archive/what-is-web-20.html (visited on 10/22/2009).

[34] *Proof of Concept: AJAX without JavaScript*. URL: http://jonathanscorner.com/ajax/ (visited on 04/26/2010).

[35] Florian Rabe. "The MMT Language and System". 2009. URL: https://svn.kwarc.info/repos/kwarc/rabe/Scala/doc/mmt.pdf (visited on 12/12/2009).

[36] *Sentido Formula Editor*. URL: http://www.matracas.org/sentido/index.html.en (visited on 06/05/2010).

[37] Heinrich Stamerjohanns and Michael Kohlhase. "Transforming the arχiv to XML". In: *Intelligent Computer Mathematics, 9th International Conference, AISC 2008 15th Symposium, Calculemus 2008 7th International Conference, MKM 2008 Birmingham, UK, July 28 - August 1, 2008, Proceedings*. Ed. by Serge Autexier, John Campbell, Julio Rubio, et al. LNAI 5144. Springer Verlag, 2008, pp. 574–582. URL: http://kwarc.info/kohlhase/papers/mkm08-arXMLiv.pdf.

[38] Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, et al. "Transforming large collections of scientific publications to XML". In: *Mathematics in Computer Science* (2010). in press. URL: http://kwarc.info/kohlhase/papers/mcs09.pdf.

[39] *Sun Java Servlet documentation*. URL: http://java.sun.com/products/servlet/ (visited on 06/05/2010).

[40] Klaus Sutner. "Converting MATHEMATICA Notebooks to OMDoc". In: OMDOC – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, 2006. Chap. 26.17. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[41] *The OMDoc Wiki*. http://www.mathweb.org/omdoc/wiki/. URL: http://www.mathweb.org/omdoc/wiki/.

[42] *TNTBase Demo*. Available at http://alpha.tntbase.mathweb.org:8080/tntbase/lectures/. 2010. URL: http://alpha.tntbase.mathweb.org:8080/tntbase/lectures/.

[43] *W3C Math Home*. URL: http://www.w3.org/Math/.

[44] *W3C Math Home*. Web site at http://www.w3.org/Math/. 2009. URL: http://www.w3.org/Math/.

[45] *Web Interface to Mathematica*. URL: http://witm.sourceforge.net/ (visited on 06/05/2010).

[46] *Wolfram|Alpha*. URL: http://www.wolframalpha.com (visited on 06/05/2010).

[47] *XPath reference*. URL: http://www.w3.org/TR/xpath/ (visited on 06/05/2010).

[48] Vyacheslav Zholudev and Michael Kohlhase. "TNTBase: a Versioned Storage for XML". In: *Proceedings of Balisage: The Markup Conference 2009*. Vol. 3. Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2009. DOI: 10.4242/BalisageVol3.Zholudev01. URL: http://www.balisage.net/Proceedings/vol3/html/Zholudev01/BalisageVol3-Zholudev01.html.