Interactive Documents as Interfaces to Computer Algebra Systems: JOBAD and Wolfram|Alpha*

Catalin David, Christoph Lange, Florian Rabe

Computer Science, Jacobs University Bremen,
{c.david,ch.lange,f.rabe}@jacobs-university.de

Abstract. Interactivity and customization are common trends guiding the design of services on the web. Not only can users adapt content to their preferences, they can also dynamically aggregate content from various sources on interactive pages in their browser that thus turn into powerful command centers (e. g. iGoogle). Our JOBAD architecture embeds mathematical services into XHTML+MathML documents. JOBAD is a modular JavaScript framework for interactive services such as term folding or definition lookup.
We have now enhanced it with a client for computer algebra. It lets the user select mathematical expressions and ask a CAS to compute, graph, or rewrite them. We have done first steps towards an integration with the Wolfram|Alpha web service API, which gives access to Mathematica as well as a large mathematical knowledge base. We are currently working on a generalization towards arbitrary CAS backends and thus promoting documents as interfaces to computer algebra systems.

Keywords: Interactive Documents, Computer Algebra Systems, Web Services, MathML, OpenMath

1 Introduction

Writing and reading mathematics is difficult. Due to its abstract and succinct notation and the practice of omitting information that can be inferred from the context, readers — especially if not experts on the subject matter — often lack knowledge about certain characteristics and properties of the objects under consideration. Moreover, due to the hierarchic nature of mathematical theories, many of the terms defined and used in a mathematical document actually depend on other terms defined elsewhere.

From another point of view, a current development of the Internet is to provide ever more possibilities for the users to receive data from different sources, thus overwhelming the user with useless information. We try to cope with this by setting up intelligent contextual filters that act as interfaces to the raw data.

* We would like to thank Wolfram Research for providing us with a pioneer grant for accessing the Wolfram|Alpha web service API, and Jan Willem Knopper from the MathDox team for support in using their translation web service.

The point where the above two perspectives meet is the JOBAD architecture [13, 9]. Its goal is to facilitate the integration of diverse web services into mathematical documents — inspired by the Web 2.0 technology of *mashups* [23, 2]. 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 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 conversion.

JOBAD handles enriched mathematical documents presented on the web, as a combination of XHTML, RDFa, MathML [28] (for both display and semantics via Presentation and Content MathML) and OpenMath [3] (as an alternative language to Content MathML in which the semantic structure of the formulae is annotated) that are (inter)active and, therefore, customizable. For example, JOBAD provides services for the user that change the display and the content of the active document by retrieving additional data from different web services.

We have used JOBAD to enrich the General Computer Science lecture notes at Jacobs University which are written using sTEX(semantically enhanced TEX), uploaded to an installation of TNTBase, a versioned XML database [31, 27], which translates them to OMDoc, and finally to XHTML+MathML+RDFa [7]. The architecture is not constrained to that and can also be used for other purposes. For example, the LATIN project (Logic ATlas and INtegrator) [17], 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); see [6] for the details of the JOBAD integration. Another example in this area can be given from history: in the beginnings of the 20^{th} century, a group of (mainly) French mathematicians wrote 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.

The JOBAD architecture is modular and easily extensible, which gives other developers the ability to develop customized service modules for different tasks. This is where Computer Algebra Systems (CASs) come into the picture. In this work we present a new JOBAD service that can interact with CASs. Our generic service is instantiated to connect to Wolfram|Alpha using the Wolfram|Alpha web service API. Thus JOBAD can provide a lot of background information for a term or an equation, e. g., we can use it to simplify the selected term, plot it, or compute solutions to equations.

2 Related Work

Similarly to JOBAD, the ActiveMath project [1] deals with *aggregated 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 [5] 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, in part, to interact with CASs like Mathematica, Maxima and GAP via OpenMath phrasebooks (cf. [3]), so this can be considered as related work regarding our project. 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 [10] 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 refering 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.

3 Computer Algebra Services for JOBAD

The research in the field of CAS has received mass recognition in May 2009 when Wolfram|Alpha [30], a computational knowledge engine, was launched. 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. Mathematica [20], whose 7^{th} version was released in the first quarter of 2009, is a well-known CAS, which provides many possibilities in interacting with mathematical formulae. From a mathematical point of view, the way Wolfram|Alpha works is that it always tries to return everything that it knows about a certain formula (factorization, roots, plot) or already knows about a certain formula (as one can see in Figure 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 computable 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, 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 CASs that deal with OpenMath content and which can be reached via the SCSCP protocol [12].

The work envisioned with this project has two main parts that, in turn, regard the improvement and extension of the already existing JOBAD architecture [9]. 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 select an annotated MathML fragment (formula) and redirect it 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, identifying equations, terms etc. Still, this would only be the first use 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, should work with any CAS, the only constraint being 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 desired operation to be performed by the CAS on the user interface. The difference between Wolfram|Alpha and other CASs is that Wolfram|Alpha will automatically return plots, derivatives, related formulas, while these, if supported, have to be explicitly asked for in the other CASs. This functionality needs to be embedded in the respective service GUI elements and will have to be adjusted per CAS. In Figure 2 you can see an updated diagram of the entire JOBAD architecture, with the components in red being the ones to be added (also, the proxy will be redesigned).

Another improvement of the architecture comprises modifications to the user interface that, right after the document has been 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.

4 Wolfram|Alpha in JOBAD

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.

Fig. 1. Wolfram|Alpha results for "subset"

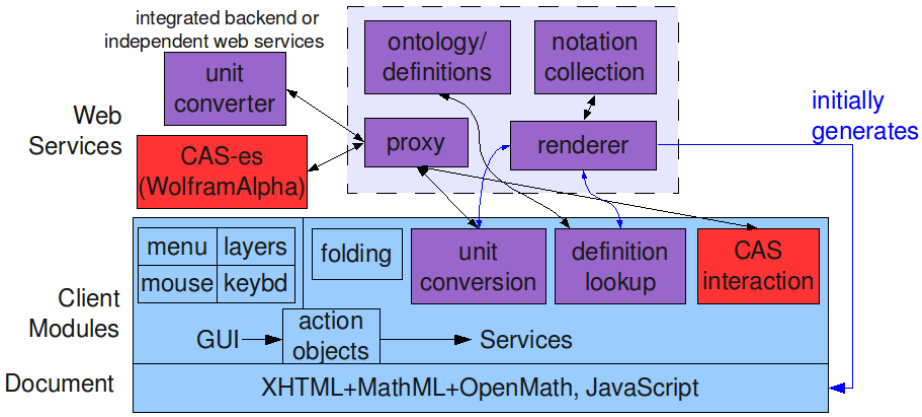

Fig. 2. JOBAD architecture

4.1 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 desired content from the webpage via XPath or other means of accessing XML fragments 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 means that retrieving the loaded page through our proxy (which is needed to circumvent security restrictions; see below) 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)

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 allowing for specifying the type of desired information more exactly, and providing additional functionality. As the above solution seems infeasible for an automated client, we have applied for and obtained a grant for research purposes on the Wolfram|Alpha architecture, which consists of a Wolfram|Alpha API key. Regarding the extra functionality provided by the API, the output of the Wolfram|Alpha webservice

can be filtered according to the developer's wish in order to better accommodate the needs of the user. The filtering regards the various possible representations of the result, providing options for visual representations (e. g. images, HTML, PDF) or textual representation (e. g. plain text, Mathematica syntax, an XML encoding of the Mathematica syntax called ExpressionML [8] etc.). Once there is one possible way to query the Wolfram|Alpha engine, the only problem that still remains is establishing a common language for the interaction between our document and Wolfram|Alpha.

As Wolfram|Alpha does not support querying mathematical content via semantically enriched MathML (Content MathML) or OpenMath, one is required to query the engine via other mechanisms, this being the first important step towards querying the knowledge base. The common syntax for querying seems to be the Mathematica language which is familiar to the Wolfram|Alpha engine, as it relies on a Mathematica backend, therefore requiring a translation from the existing Content MathML and OpenMath standard served by the server (in our case TNTBase) to Mathematica. We are aware of existing transformations between OpenMath and Mathematica, e. g. on the server as a part of the MathDox infrastructure [18, 5], or on the client as a part of the Sentido formula editor [11], and have decided to use the MathDox infrastructure for translation.

Also, taking into consideration the limitations of JavaScript, we will need to set up a proxy in order to access Wolfram|Alpha, as JavaScript code is not allowed to provide data unless it comes from the same domain ("Same Origin Policy") [21]. The purpose of the proxy will be to only prepare a request for forwarding to the Wolfram|Alpha server and, when the data is available, to provide it back to the JavaScript client, given that the key that we received from Wolfram|Alpha may not be exposed and will be stored in the proxy. Also, depending on the nature of the data, the proxy might alter the structure of the retrieved document in order to save post-processing on the client side. 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, there are two possible ways of presenting the information retrieved from Wolfram|Alpha on the client: embedding it as parallel presentation and content markup (i. e. Presentation MathML annotated with Content MathML or OpenMath) into the original document, thus effectively rewriting formulae in the document, or simply displaying it for the user's information.

The communication flow required for rewriting expressions is vizualized in Figure 3. 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 expression and make an HTTP 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

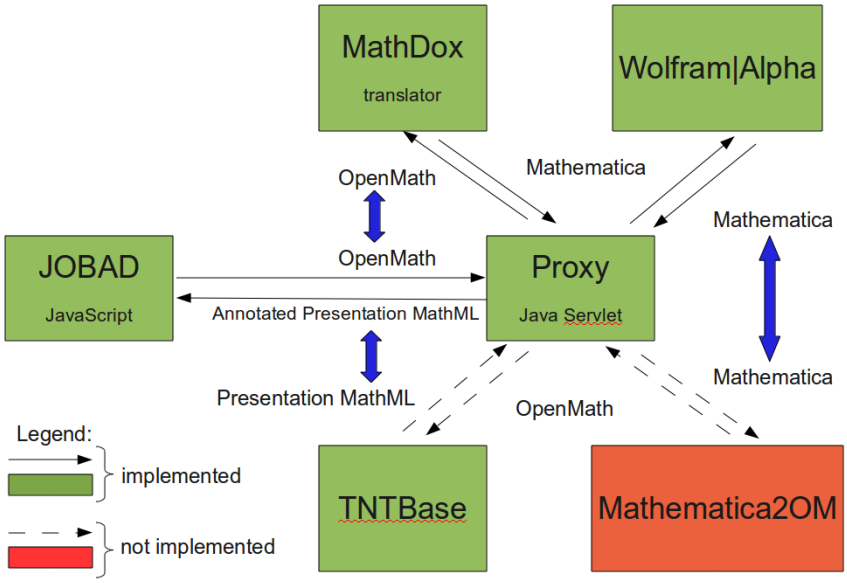

Fig. 3. Proxy architecture for content-math oriented tasks

request to a webservice running on the MathDox [19] 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 (NLP) 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 processing 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 [26] is a Mathematica package that is able to transform Mathematica code (version 4.2, latest version is 7) to OMDoc (an early draft of 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 adition to that, one would have to transform (render) the OMDoc content to Presentation MathML with OpenMath annotations, a step to be executed by TNTBase and its embedded renderer from the JOMDoc library [14], a Java API for OMDoc documents (and illustrated in the picture).
- *Mathematica web service*: As pointed out here in the Mathematical online tutorial[1], 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 [29], 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

[1] `http://reference.wolfram.com/mathematica/XML/tutorial/MathML.html`

not simultaneously (a large number of users might mean interference in the result)
- *Sentido formula editor*: Developed by Alberto González Palomo as part of the Sentido [25] 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, bypassing the other functionality.

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

4.2 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 before in the definition lookup service, we decided to use the same jQuery UI [15] widget that allows the developer to populate a dialog window with content, in this case the results provided by Wolfram|Alpha, translated from their XML representation to XHTML. 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 4 for definition lookup, 5 for the module loading utility and 8 for the Wolfram|Alpha lookup). An alternative that is appropriate for some types of queries and results is rewriting mathematical expressions in place, which JOBAD so far does for the results of unit conversions [9].

4.3 Preserving Document Settings

The last part of this project regarded the extension of the interface with another service that allows 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 5), 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

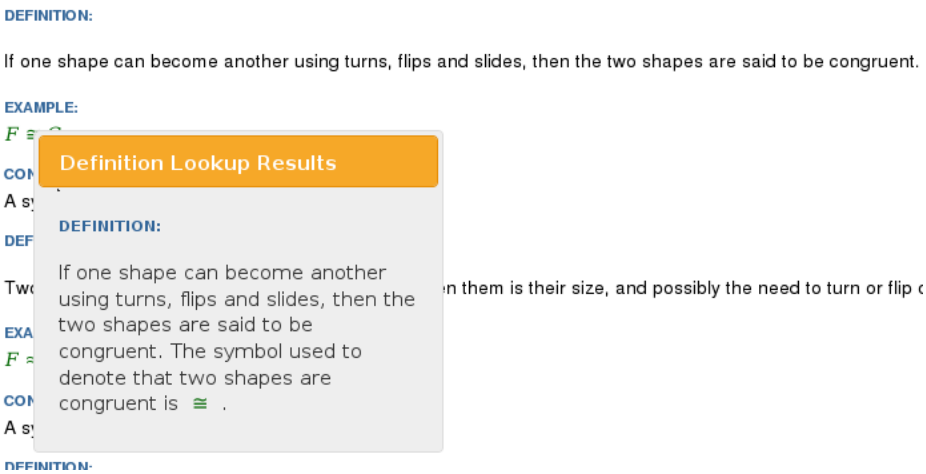

Fig. 4. Definition Lookup Example

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. This can be further expanded in a server-side saved user profile that can be used in other projects as the adaptive document browser panta rhei [24, 22].

5 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 5). Once both the *wolframalpha* and the 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 7 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 6. If a user were to access the Wolfram|Alpha website and search for the Mathematica representation of the OpenMath fragment, in this case *Sqrt[x]*, the result page would look like Figure 9. After the request is processed, the Wolfram|Alpha content is retrieved on the client side and the user will experience something resembling Figure 8.

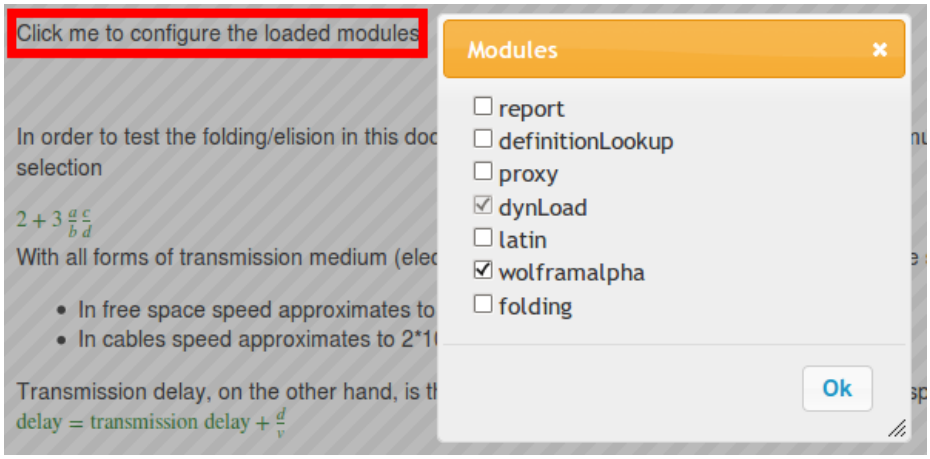

Fig. 5. The user loads the *wolframalpha* service

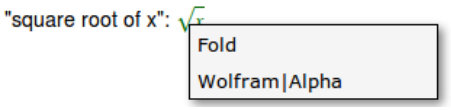

Fig. 6. The user performs a right on the $\sqrt{x}$ symbol

```
<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>
```

Fig. 7. The Presentation MathML and the associated OpenMath content annotation representations of $\sqrt{x}$

Fig. 8. Part of the Wolfram|Alpha results embedded into the original document (The question marks in the MathML formulae result from Wolfram|Alpha using a wrong character encoding; we are working on a workaround for that.)

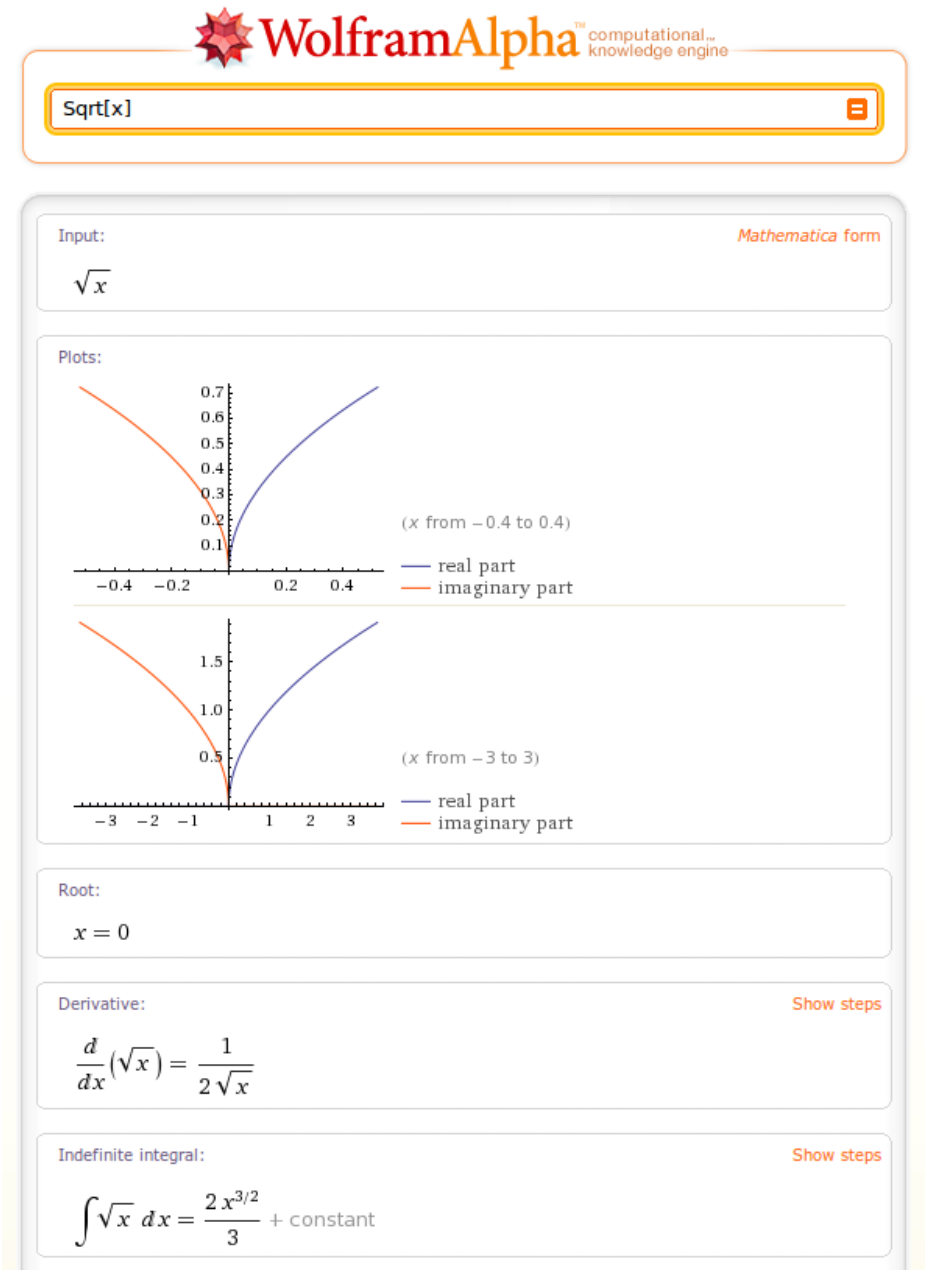

Fig. 9. Part of the Wolfram|Alpha website search results for *Sqrt[x]*

5.1 Extending the Wolfram|Alpha interface to other CASs

This section of the project represents more of a further research topic. While Wolfram|Alpha is very competent, the amount of information it provides might not be adequate for other services or facilities or might not be detailed enough, thus requiring a more involved or specialized CAS. Some things also to keep in mind are the efficiency of the CAS, as this will have a great impact on the users, and the licensing of the CAS (the Wolfram|Alpha API is not freely accessible and requires licensing). Therefore, instead of using Wolfram|Alpha for simple or very specific computations, one can use a locally installed CAS or just another CAS that handles things better and/or faster than the webservice provided for Mathematica. Still, as presented before, Wolfram|Alpha is only a specialized application of JOBAD's "CAS" service, as this feature could be implemented for any existing webservice, given that there is a common interaction language between the MathML/OpenMath formulae in the documents and the CAS. However, given that Wolfram|Alpha can accomplish much more than a simple CAS, as it also relies on an index for words and that its output is a bit different from the one of a CAS (which is punctual), a transition from Wolfram|Alpha to a simple CAS might be a bit more involved and will require some further investigations both in terms of CASs and in terms of integration with the JOBAD architecture.

6 Conclusion and Future Work

We have presented the design and first implementation steps for an integration of CAS services into the JOBAD architecture for interactive mathematical documents. Specifically, we provide a service that lets users interact with the Wolfram|Alpha web service API. 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.

We will evaluate the CAS module for JOBAD using the existing integration of JOBAD with the first-year undergraduate computer science course taught at Jacobs University, whose slides and lecture notes are written using the LaTeX package sTeX. sTeX documents are automatically converted into OMDoc [16] documents and from there into JOBAD-enabled XHTML+presentation/content-MathML documents [7]. Thus, students will be able to interact with Wolfram|Alpha-generated content.

In future work, we will extend the architecture to other CASs. This will also act as a simple integration platform between CASs as results received from one system can be sent to another one. Similarly, the current design can be extended to theorem provers where, instead of querying the service for the simplification of an expression, we query for a proof of a theorem. This could be the first step to a document-centric integration of theorem provers both among each other and with CASs.

References

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

[2] 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.

[3] Stephen Buswell, Olga Caprotti, David P. Carlisle, et al. *The Open Math Standard, Version 2.0*. Tech. rep. The Open Math Society, 2004. URL: http://www.openmath.org/standard/om20.

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

[5] 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, June 2008, pp. 5177–5182. URL: http://go.editlib.org/p/29092.

[6] Catalin David, Michael Kohlhase, Christoph Lange, et al. "JOBAD/MMT – Interactive Mathematics". In: *AI Mashup Challenge 2010, ESWC*. Ed. by Adrian Giurca, Brigitte Endres-Niggemeyer, Christoph Lange, et al. June 2010. URL: http://sites.google.com/a/fh-hannover.de/aimashup/home/jobad.

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

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

[9] 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, July 2009, pp. 279–293. URL: https://svn.omdoc.org/repos/jomdoc/doc/pubs/mkm09/jobad/jobad-server.pdf.

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

[11] Alberto González Palomo. "Sentido: an Authoring Environment for OMDoc". In: OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. Chap. 26.3. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

[12] 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, July 2009, pp. 474–479.

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

[14] *JOMDoc Project — Java Library for OMDoc documents*. URL: http://jomdoc.omdoc.org (visited on 10/22/2009).

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

[16] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.pdf.

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

[18] *MathDox – OpenMath Translation Servlet*. URL: http://mathdox.org/phrasebook/ (visited on 03/16/2010).

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

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

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

[22] Christine Müller and Michael Kohlhase. "panta rhei". In: *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) conference proceedings*. Ed. by Alexander Hinneburg. Martin-Luther-University Halle-Wittenberg, 2007, pp. 318–323.

[23] 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).

[24] *Panta: The PHP Frontend of panta rhei*. URL: http://trac.kwarc.info/panta (visited on 01/2010).

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

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

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

[28] *W3C Math Home*. URL: http://www.w3.org/Math/ (visited on 01/2009).

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

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

[31] 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.