Smart Pasting for ActiveMath Authoring

Paul Libbrecht
paul@activemath.org
Competence Center for E-Learning
DFKI GmbH, Saarbrücken, Germany

Eric Andrès
eandres@mx.uni-saarland.de
CC Visu, Saarland University
Saarbrücken, Germany

Yecheng Gu
yecheng@activemath.org
Competence Center for E-Learning
DFKI GmbH, Saarbrücken, Germany

Abstract

Authoring mathematical formulæ for ActiveMath is a challenge because it requires them to be semantic, along the OpenMath standard and (extensible) set of symbols. The authoring environment of ActiveMath, jEditOQMath eases this by providing a readable linear syntax for formulæ input based on QMath.
This paper approaches the support to authors to encode formulæ by bringing together several types of conversion methods into one *smart paste* approach. Such sources of formulæ as Wikipedia, and Planet Math are considered and start to work.

Keywords: copy-and-paste, conversion, OpenMath

Introduction

One of the central, most difficult, and yet most interoperability-promising parts of authoring content for the ActiveMath learning environment is to write mathematical formulæ. These have to be encoded according to the model of the OpenMath semantic language. The current practice is to use QMath, a processor supporting a linear syntax as readable as that of TeX. Encoding is done by direct typing which can be a difficult task that has a steep learning curve.

jEditOQMath, the editor for ActiveMath content, has been enriched with a "smart paste" facility to paste mathematical formulæ encoded in alternative formats. Smart paste transforms these external formats to OpenMath and, from there, to QMath. This enables authors to reuse formulæ from many external places by pasting them into jEditOQMath. There are many possible formats for formulæ around and there are many possible converters; the smart paste facility tries to bundle all of them.

The approach we describe thus far bases on a web-service which is accessed to by smart paste:

- MathML-presentation to OpenMath using the configurable Wiris OpenMath Tools currently based on the default notations of ActiveMath
- MathML-presentation to Content-MathML based on WebEQ
- the TeX of wikipedia to MathML-presentation running the blahtex processor

We expect to make use of these in the conversion of large content collections initiated in the Math-Bridge project.

We report on successful copy and paste from the Web and how feedback is provided to support the user. Furthermore, we describe other uses of the smart paste paradigm, e.g. to support beginners in the insertion of complex objects such as the a flash animation.

Typical Scenario

A simple scenario that we wish to achieve is the conversion from a page found on the Web to a source document for the ActiveMath learning environment. For example, Wikipedia (`http://www.wikipedia.org/`) or MathWorld (`http://www.mathworld.com/`) both are web-sites with a rich wealth of mathematical content. In many conditions it is possible to copy and paste content from these pages for re-use, e.g., in educational activities.

This paper follows a common dialogue we, as educators of the usage of jEditOQMath, had with future ActiveMath authors asking "So you are supporting TeX?": So far, this question could only be answered negatively since *supporting TeX* meant encoding many ill-formed formulæ.

Outline

The paper starts with a very brief description of the ActiveMath learning environment and its current authoring infrastructure. A definition of *smart pasting* is then proposed followed by its concretization in jEditOQMath. One of the fundamental applications, smart-paste for learning, is then proposed. We conclude after a coverage of existing smart paste approaches we have encountered.

1 ActiveMath Authoring in Short

ActiveMath [5] is a Web-based learning environment for mathematics. ActiveMath uses the OMDoc [2] semantic format for its mathematical documents, which it can present in several formats, notably HTML and PDF. It leverages the semantic information in order to enrich formulæ with features that support

readability and transferability (Drag&Drop) and to use some of the formulæ for exercise evaluation, among others.

ActiveMath content consists of sets of OMDoc files called *content collections* which share a practice of authoring. One of the most common collection types is based on the OQMath format described in [3]: this format is mostly made of a readable OMDoc encoding with mathematical formulæ written in compact linear form and compiled with the QMath processor [6].

This processor uses *notation contexts* which define new notations using Unicode characters. Very often a single notation context is attached to a collection and it includes the standard notation of other contexts.[1]

Many ActiveMath content collections are edited using jEditOQMath, the classical java-based editor jEdit extended to support the authoring experience with such features as the content-storage loading process; see [3].

2 The Smart Paste Paradigm

We define *smart paste* as the action of transferring the content of the clipboard to an editing zone using a transformation appropriately chosen for the editing context at hand. Smart paste may involve interactions with the user in order to guide the program in choosing the best alternative. That guidance should, as much as possible, be leveraged to minimize interaction requests in subsequent smart paste invocations since it is expected to be used repetitively.

2.1 Smart paste in jEditOQMath

jEditOQMath has been extended with a smart paste functionality. Earlier versions contained a start of conversion as a replacement of the normal paste, this was mostly used to convert dropped ActiveMath URLs to references. We have decided, however, that the smart paste should really only be used when requested, hence it has been reassigned to an alternate shortcut, similar to word processors' "special paste".

The smart paste of jEditOQMath tries to convert formulæ from:

- the TeXvc syntax of Wikipedia
- MathML-presentation
- MathML-content
- OpenMath

[1] An extract of a notation context can be seen at `http://www.activemath.org/~george/work/qmath/contexts/en/Mathematics/OpenMath/arith1.qmath`.

to the QMath format, using the notation of the file being edited, otherwise to OpenMath.

A few *content sniffers* are used in order to detect the format of the input which, thus far, is still in the plain text media type. The determined formats are then used to choose the conversion pipelines which are made of the following ingredients:

- MathML-presentation to MathML-content, in a fixed and comprehensive way, as provided by the WebEQ developer tools[2]
- Content MathML to OpenMath as provided by the stylesheet of David Carlisle [3]
- MathML-presentation to OpenMath as provided by the Wiris OpenMath tools [4], in a way that can be configured by the *notation domain*
- TeXvc to MathML-presentation through the usage of the blahtex command-line tool[4]
- OpenMath to QMath thanks to an XSLT stylesheet that is generated on the fly from the notation context of the current file

In the current implementation, all the services except the last are run in a central server which is called over an HTTP post from the jEditOQMath editor. This serves both the license needs as well as the debugging necessities, allowing us to obtain information about the formulæ attempted to be converted, successfully or not.

Once all the possible conversion pipelines are assembled and run, a list of the results is presented to the user who is requested to choose. The presentation generated by the pipeline selected on the last run is *pre-chosen*.

In case of an error, the pipeline is ignored, in the case that all pipelines fail, the user is invited to the log.

2.2 Example Smart Paste Usage

One of the first targets is to use the Wikipedia web-site whose mathematical content is ever growing. For example many formulæ on `http://de.wikipedia.org/wiki/Formelsammlung_Geometrie` can be easily copied if using Firefox since it puts the content of the alt-attribute of an image into the

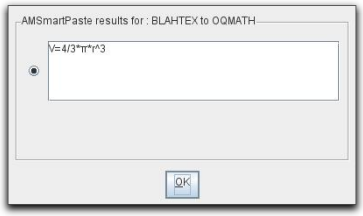

Figure 1: Pasting the formula for the volume of a ball from Wikipedia

[2]See `http://www.dessci.com/`.

[3]See `http://svn.openmath.org/OpenMath3/xsl/om2cmml.xsl` for the stylesheet Pragmatic-Content-MathML to OpenMath

[4]See `http://gva.noekeon.org/blahtexml/` for the current development place of the Blah-TeX converter from TeXvc-syntax to MathML presentation.

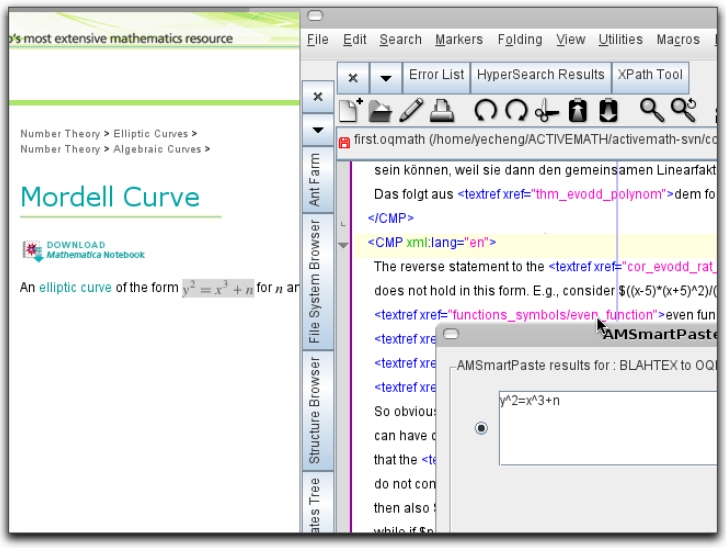

Figure 2: Transferring the Mordell curve from MathWorld

clipboard. The TeXvc processor kindly fills that attribute with the source.

- $A = \frac{g \cdot h}{2}$: with the converter chain
- $V = \frac{4}{3}\pi r^3$: with converter chain, as in figure 1
- $h^2 = q \cdot p$
- $O = r^2 \cdot \pi + \pi \cdot r \cdot s = r \cdot \pi \cdot (r + s)$

A similar gesture is done from MathWorld, for example the generic formula of Mordell Curves at `http://mathworld.wolfram.com/MordellCurve.html` presented in figure 2.

The case of JS-math powered websites is supported by this software's inbuilt ability to show the TeX-like source of each formula when one double clicks on it. The page `http://planetmath.org/encyclopedia/GoniometricFormulae.html` offers a wealth of formulæ to test where a few succeed.

2.3 Failures of Smart Pasting

Not all formulæ succeed on this page and our heuristics certainly need to be boosted to raise the success rate.

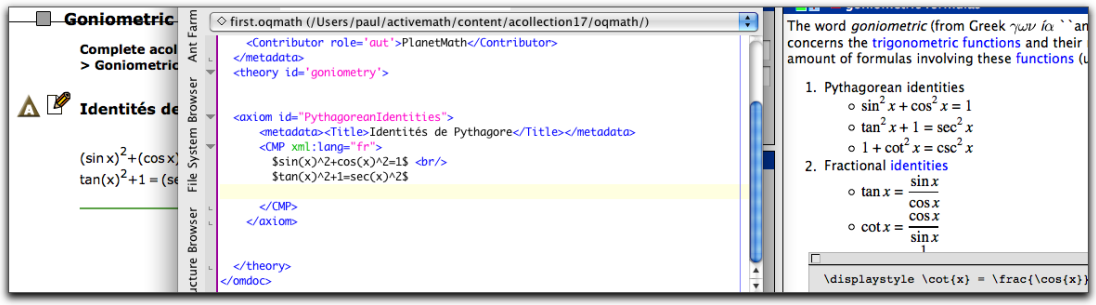

Figure 3: Transferring a formula of goniometry from PlanetMath (on the right) until ActiveMath (on the left) using the smart-paste in jEditOQMath (middle).

A first very common family of errors is the inclusion of the punctuation at the end of a formulæ as in figure on the right. Not surprisingly, this breaks all conversions from presentation to content since it corrupts the mathematical semantics of the formula. A safe way to remove such punctuation is wished and will probably be best placed within a process of MathML-processing.

$$\int_0^1 \sqrt{x}\,dx.$$

Figure 4: formula ending with a punctuation.

Another common cause of error is the usage of subscripts which should, generally, be interpreted as variable names but often fails to be converted to variable names. It is not yet clear at which stage such a conversion should be included, especially since some index notations are defined and actually converted, for example x_A which means the first coordinate of the point A. Maybe the right strategy is to fallback on *escaping* variable names in case the conversion to content fails. Note, however, that the notation context should be enriched to process the indexed variable name as such.

All these failures are somewhat visible to the author and adjusting things is possible, from the easy crafting till the elaborate adjustments.

2.4 Adjusting the Smart Paste Process

The first and basic dimension that an author can use is rely on the fact that the smart paste is separated from paste. A natural debug action is, thus, to first paste, inspect and arrange the text, then cut and smart paste. This is a manual process which is not particularly comfortable but works in many situations.

The author has the possibility to adapt his environment for the smart paste function to operate with more success and quality.

The smart paste result may be guilty of dropping some symbols which is generally due to missing symbols in the notations context. The author, who is responsible for enriching it, can do so relatively easily, adding a new notation for the OpenMath content he expects to receive.

Clearly, the OpenMath to QMath conversion could be more explicit, for example indicating that a symbol is missing. Unfortunately, there is no *warning* mechanism available yet in smart paste, which is where such information should be inserted. Moreover, such warnings could be detailed enough to actually add the notation if the author agrees.

Another adaptation to the OpenMath to QMath conversion process could be the change of order of notations so that the right QMath notation is used for a given OpenMath. QMath having been originally designed as a processor from short-syntax to OpenMath, this has never been an issue but surfaces now. For example, we could experience the appearance of the notation `a minus b` instead of the much more natural `a-b`. Thus far there has not been a *smart ordering* of the notations in the context, for example preferring shorter or binary operators, so that the author can change the notation context's order to indicate his preference.

A last dimension of adaptation is the rich domain of the Wiris MathML-presentation to OpenMath converter. As described in [4], this can be easily done using a desktop application called the *domain editor*. Thus far, however, it would require authors to install their own converter service and configure it which is rather a developer task. We see it possible, however, that one day the domain is pushed from the client to the server. This approach is particularly relevant in complement to the WebEQ conversion which is non-extensible and unable to read many localized notations, such as the simple French $\mathrm{pgcd}(p, q)$ (meaning $\gcd(p, q)$) or the russian notation for binomial coefficient C_a^b denoting C_b^a.

3 Smart Paste to Learn to Input

It should be noted that the smart-paste paradigm is considerably different than a batch conversion process; the small-grained steps of the conversion and the ability to fix are two major differences. A most important one is also that it produces readable input that looks like the user had typed it by hand.

This ability allows a user knowing, for example, the LaTeX syntax, to discover ways to write QMath expressions by simply inputting desired things first in LaTeX then seeing its result in QMath. We expect this to become one of the major usages of the smart paste function.

4 Smart Paste Beyond Mathematical Formulæ

The same paradigm could be used in many other situations where the construction of the OMDoc encoding may require thoughts.

One particular aspect which has little interest to be developed is that of multimedia embedding, such as Java or Flash applets. Their HTML pendant has always been the result of an authoring-tool output, one that authors barely look at. The smart-paste function of jEditOQMath supports HTML tags for applet

and flash embeddings, inserting the necessary `omlet` and `private` elements of OMDoc. But the tag itself is not enough and the author still have to adjust the paths and/or copy the relevant files.

A natural wish would be to extend the smart paste function to larger paragraphs. This is however, much more difficult. Using the plain-text content that browsers put in the clipboard, there's almost no way to recognize formulæ islands. We shall need to get the source content in a different media-type such as HTML or RTF.

5 Other Smart Paste Approaches

The closest comparison to the smart paste approach is the paste function included in Maple or Mathematica computer algebra systems. Both of these classical commercial desktop systems use the plain-text of the clipboard and detect when the text being pasted "looks like MathML" and try to process it. This is expected to be useful from external tools such as word-processors.

For both of these systems, the scope is different than jEditOQMath since, in most cases, the conversion from presentation stays presentation and is only attempted to be converted to content when pasted in particular places. That conversion is incomplete and seems to be not influenceable, just as the conversion from content (e.g. linear syntax) to presentation.

Other forms of *special paste* that we have encountered, for example in desktop word processors, seem to follow the approach of a dialog to ask the user what is to be considered but seem to be only asking about the choice of conversion pipelines by their name. jEditOQMath's smart paste, instead, lets the user choose from possible results which appears to us to be a helpful approach even though, it may be computationally expensive.

6 Conclusion

We have introduced jEditOQMath's smart paste functionality as a potential helpful function for authors to be able to input formulæ using the compact syntax of the QMath processor. We expect this function to be widely used in the Math-Bridge project which is starting with the conversion of a significant amount of content.

The implementation is based on multiple conversion pipelines and only the result is evaluated by the user which, we believe, is a useful approach that we have seen nowhere else. It is probable that the set of pipelines will be extended since, indeed, many converters can be found around but activating needs often a convoluted programming.

We hope smart-paste to be able to refine and mature with usage. We have identified a few enhancement directions in the text above, from the ease of configuration of the authors till an enhanced heuristics for content processing.

One direction which needs strong polishing is the conversion from OpenMath to QMath: from the software side, to report more efficiently to the user about incompleteness, as well as from the default notation context side which receive a new role, that of going from OpenMath to QMath.

Overall the polishes to the converters seems to appear mostly because of the appearance of the new easy activation of this feature. This should be compared to the computer-algebra-systems' approaches to only offer content-MathML to functions invokers, i.e. far from an easy copy-and-paste. The appearance of such a feature may wake-up awareness at the user-side of the *semanticalizability* of the many formulæ that they manipulate.

A new direction appears with the expected implementations of tools putting multiple content formats into the clipboard following the clipboard section of the MathML 3 specification currently in draft [1]. This may give a much finer control over the choice of involved formats and could open the door to the ability of selecting an expression in the simple view and obtaining the underlying mathematical representation; currently, all browsers we have met with all possibilities exploited invariably copy the plain-text conversion of the content which makes for example $1 - \tan^2 x$ become `1-tan2x` where no converter can act.

Acknowledgements

This publication is partly a result of the ActiveMath-EU project funded by the European Commission, as well as the Math-Bridge eContentPlus project. This paper represents the opinions of the authors only.

We thank George Goguadze for his first contribution to the XSLT generation to convert OpenMath to QMath. We thank Design Science Inc. for their collaboration in the integration of the rich WebEQ conversion facilities.

References

[1] D. Carlisle, P. Ion, and R. Miner. Mathematical markup language, version 3.0. Working Draft, 2008. `http://www.w3.org/TR/MathML3/`.

[2] Michael Kohlhase. *OMDoc – An Open Markup Format for Mathematical Documents*. Springer Verlag, 2006.

[3] P. Libbrecht and C. Groß. Experience report writing LeActiveMath Calculus. In Jon Borwein and William Farmer, editors, *Proceedings of Mathematical Knowledge Management 2006*, number 4108 in LNAI, pages 251–265. Springer Verlag, aug 2006. Available at `http://www.activemath.org/pubs/bi.php/Libbrecht-Gross-Experience-Report-Authoring-LeAM-MKM-2006`.

[4] Daniel Marquès, Ramon Eixarch, Glória Casanellas, Bruno Martínez, and Tim Smith. WIRIS OM Tools a Semantic Formula Editor. In *Proceedings of MathUI'06*, August 2006. Available from `http://www.activemath.org/~paul/MathUI06/`.

[5] E. Melis, G. Goguadze, M. Homik, P. Libbrecht, C. Ullrich, and S. Winterstein. Semantic-Aware Components and Services of ActiveMath. *British Journal of Educational Technology*, 37(3):405–423, may 2006.

[6] Alberto González Palomo. QMath: A human-oriented language and batch formatter for OMDoc. volume 4180/2006 of *LNCS*, chapter 26.2. Springer Verlag Heidelberg, 2006. See `http://www.mathweb.org/omdoc`.