Transforming the arχiv to XML

Heinrich Stamerjohanns and Michael Kohlhase

Computer Science, Jacobs University Bremen
{h.stamerjohanns,m.kohlhase}@jacobs-university.de

Abstract. We describe an experiment of transforming large collections of LaTeX documents to more machine-understandable representations. Concretely, we are translating the collection of scientific publications of the Cornell e-Print Archive (arXiv) using the LaTeX to XML converter which is currently under development.
The main technical task of our arXMLiv project is to supply `LaTeXML` bindings for the (thousands of) LaTeX classes and packages used in the arXiv collection. For this we have developed a distributed build system that reiteratively runs `LaTeXML` over the arXiv collection and collects statistics about e.g. the most sorely missing `LaTeXML` bindings and clusters common error events. This creates valuable feedback to both the developers of the `LaTeXML` package and to binding implementers. We have now processed the complete arXiv collection of more than 400,000 documents from 1993 until 2006 (one run is a processor-year-size undertaking) and have continuously improved our success rate to more than 56% (i.e. over 56% of the documents that are LaTeX have been converted by `LaTeXML` without noticing an error and are available as XHTML+MathML documents).

1 Introduction

The last few years have seen the emergence of various XML-based, content-oriented markup languages for mathematics and natural sciences on the web, e.g. OpenMath, Content MathML, or our own OMDoc. The promise of these content-oriented approaches is that various tasks involved in "doing mathematics" (e.g. search, navigation, cross-referencing, quality control, user-adaptive presentation, proving, simulation) can be machine-supported, and thus the working mathematician can concentrate in doing what humans can still do infinitely better than machines.

On the other hand LaTeX is and has been the preferred document source format for thousands of scientists who publish results that include mathematical formulas. Millions of scientific articles have been written and published using this document format. Unfortunately the LaTeX language mixes content and presentation and also allows to create additional macro definitions. Therefore machines have great difficulties to parse and analyze LaTeX documents and to extract enough information to represent the written formulas in a XML representation.

In this paper, we will present an experiment of translating a large corpus of mathematical knowledge to a form that is more suitable for machine processing. The sheer size of the arXiv [ArX07] poses a totally new set of problems for MKM technologies, if we want to handle (and in the future manage) corpora of this size. In the next section we will review the translation technology we build on and then present the corpus-level build system which is the main contribution of this paper.

2 TeX/LaTeX to XML Conversion

The need for translating LaTeX documents into other formats has been long realized and there are various tools that attempt this at different levels of sophistication. We will disregard simple approaches like the venerable `latex2html` translator that cannot deal with user macro definitions, since these are essential for semantic preloading. The remaining ones fall into two categories that differ in the approach towards parsing the TeX/LaTeX documents.

Romeo Anghelache's Hermes [Ang07] and Eitan Gurari's TeX4HT systems use special TeX macros to seed the `dvi` file generated by TeX with semantic information. The `dvi` file is then parsed by a custom parser to recover the text and semantic traces which are then combined to form the output XML document. While Hermes attempts to recover as much of the mathematical formulae as Content-MathML, it has to revert to Presentation-MathML where it does not have semantic information. TeX4HT directly aims for Presentation-MathML.

The systems rely on the TeX parser for dealing with the intricacies of the TeX macro language (e.g. TeX allows to change the tokenization (via "catcodes") and the grammar at run-time). In contrast to this, Bruce Miller's `LaTeXML` [Mil07] system and the SGLR/Elan4 system [vdBS03] re-implement a parser for a large fragment of the TeX language. This has the distinct advantage that we can fully control the parsing process: We want to expand abbreviative macros and recursively work on the resulting token sequence, while we want to directly translate semantic macros[1], since they directly correspond to the content representations we want to obtain. The `LaTeXML` and SGLR/Elan4 systems allow us to do just this. In our conversion experiment we have chosen the `LaTeXML` system, whose LaTeX parser seems to have largest coverage.

The `LaTeXML` system consists of a TeX parser, an XML emitter, and a post-processor. To cope with LaTeX documents, the system needs to supply `LaTeXML` **bindings** (i.e. special directives for the XML emitter) for the semantic macros in LaTeX packages. Concretely, every LaTeX package and class must be accompanied by a `LaTeXML` binding file, a Perl file which contains `LaTeXML` constructor-, abbreviation-, and environment definitions, e.g.

```
DefConstructor("\Reals","<XMTok name='Reals'/>");
DefConstructor("\SmoothFunctionsOn{}",
          "<XMApp><XMTok name='SmoothFunctionsOn'/>#1</XMApp>");
DefMacro("\SmoothFunctionsOnReals","\SmoothFunctionsOn\Reals");
```

[1] See [Koh08] for a discussion of semantic and abbreviative macros.

`DefConstructor` is used for semantic macros, whereas `DefMacro` is used for abbreviative ones. The latter is used, since the `latexml` program does not read the package or class file and needs to be told, which sequence of tokens to recurse on. The `LaTeXML` distribution contains `LaTeXML` bindings for the most common base LaTeX packages.

For the XML conversion, the `latexml` program is run, say on a file `doc.tex`. `latexml` loads the `LaTeXML` bindings for the LaTeX packages used in `doc.tex` and generates a temporary LTXML document, which closely mimics the structure of the parse tree of the LaTeX source. The LTXML format provides XML counterparts of all core TeX/LaTeX concepts, serves as a target format for `LaTeXML`, and thus legitimizes the XML fragments in the `LaTeXML` bindings.

In the semantic post-processing phase, the LaTeX-near representation is transformed into the target format by the `latexmlpost` program. This program applies a pipeline of intelligent filters to its input. The `LaTeXML` program supplies various filters, e.g. for processing HTML tables, including graphics, or converting formulae to Presentation-MathML. Other filters like transformation to OpenMath and Content-MathML are currently under development. The filters can also consist of regular XML-to-XML transformation process, e.g. an XSLT style sheet. Eventually, post-processing will include semantic disambiguation information like types, part-of-speech analysis, etc. to alleviate the semantic markup density for authors.

3 The Build System

To test and give feedback to improve `LaTeXML`, and to extend our collection of valid XHTML+MathML documents which are being used for other projects such as our MathWebSearch [Mat08], we have chosen to use the articles that have been published in the arXiv. This large heterogenous collection of scientific articles is a perfect source for experiments with scientific documents that have been written in LaTeX.

The huge number of more than 400.000 documents (each one may include figures and its own style files and is located in its own subdirectory) in this collection made simple manual handling of conversion runs impossible. To handle the conversion process itself (invocation of ttlatexml and `latexmlpost`) `Makefiles` have been automatically created by scripts. But the usage of `make` has also some limitations: It does not easily allow to run distributed jobs on several hosts, a feature that is essential to be able to massively convert thousands of documents in one day. While distributed make utilities (such as `dmake`) or other grid tools may support distributed builds, all these tools are of limited use when only a restricted set of specific documents with certain error characteristics should be converted again, which would require complex and continuous rewriting of makefiles.

To overcome these limitations we have developed an arXMLiv build system which allows `make` jobs to be distributed among several hosts, extracts and analyzes the conversion process of each document and stores results in its own

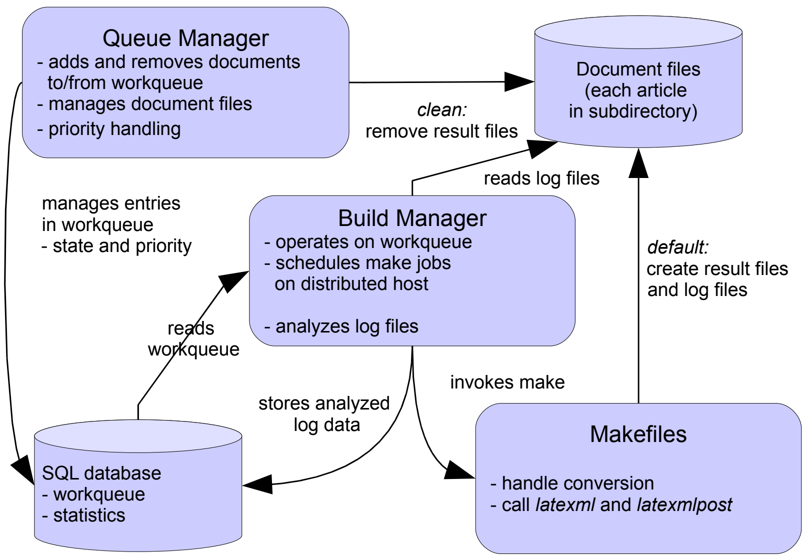

Queue Manager
- adds and removes documents to/from workqueue
- manages document files
- priority handling
Document files (each article in subdirectory)
clean: remove result files
reads log files
manages entries in workqueue
- state and priority
Build Manager
- operates on workqueue
- schedules make jobs on distributed host
- analyzes log files
default: create result files and log files
reads workqueue
invokes make
stores analyzed log data
Makefiles
- handle conversion
- call *latexml* and *latexmlpost*
SQL database
- workqueue
- statistics

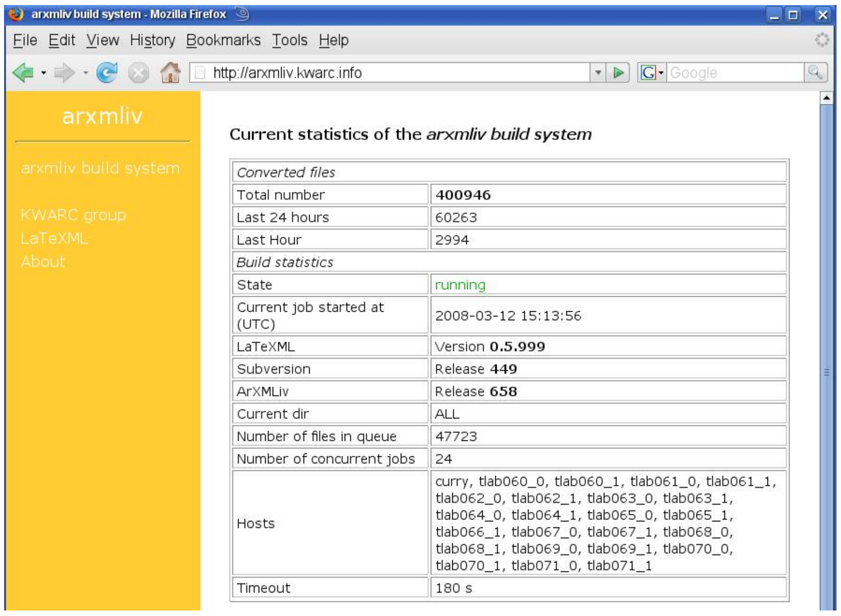
Fig. 1. Schematic overview of the arXMLiv build system

SQL database. The database allows to easily cluster documents which include a macro that is only partially supported or to gather statistics about the build process.

The arXMLiv build system consists of a file system which is shared among all hosts, a queue manager, a build manager, and a relational database, which stores a workqueue and results statistics about each single converted file. The file system contains all the documents ($\approx$ 150 Gigabytes), classified by topic and each one located in its own subdirectory. The file system is exported via NFS to all hosts which take part in the build process.

To schedule conversion jobs, we operate the queue manager via the command line. A command like `php workqueue.php default cond-mat` will add all documents inside the `cond-mat` subdirectory — the arXiv section for papers concerning *condensed matter* — to the current work queue, which is stored in the relational database.

The build manager is implemented in PHP, where SQL databases as well as process control functions can be easily used. It keeps an internal list of available hosts, reads the files to be converted next from the workqueue and distributes jobs to remote hosts. For each document that is to be converted the build manager forks off a new child process on the local machine. The child sets a timer to enable a limiting timeout of 180 s for the conversion process and then creates another child (the grandchild) which then calls the `make` on a host via remote `ssh` execution. The `make` process will then invoke `latexml` and `latexmlpost` to


arxmliv build system - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://arxmliv.kwarc.info
Google
arxmliv
arxmliv build system
KWARC group
LaTeXML
About

Current statistics of the *arxmliv build system*

Converted files	
Total number	**400946**
Last 24 hours	60263
Last Hour	2994
Build statistics	
State	running
Current job started at (UTC)	2008-03-12 15:13:56
LaTeXML	Version **0.5.999**
Subversion	Release **449**
ArXMLiv	Release **658**
Current dir	ALL
Number of files in queue	47723
Number of concurrent jobs	24
Hosts	curry, tlab060_0, tlab060_1, tlab061_0, tlab061_1, tlab062_0, tlab062_1, tlab063_0, tlab063_1, tlab064_0, tlab064_1, tlab065_0, tlab065_1, tlab066_1, tlab067_0, tlab067_1, tlab068_0, tlab068_1, tlab069_0, tlab069_1, tlab070_0, tlab070_1, tlab071_0, tlab071_1
Timeout	180 s

Fig. 2. The web interface shows the current state of the build system

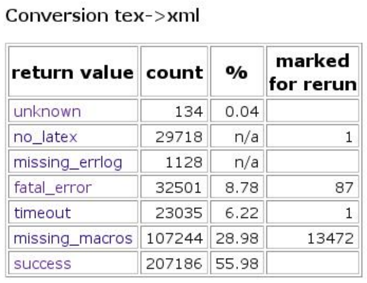
convert a TEX file to XML and XHTML. `LaTeXML` logs the inclusion of style files and also reports problems while converting from `latex` to `xml` to special log files.

After a timeout or the completion of the conversion process the build manager is notified via typical Unix signal handling. The build manager then parses and analyzes log files, extracts information about the result state of conversion and collects the names of missing macros that are not yet supported. If the conversion has failed, the error message given by `LaTeXML` is also extracted. For each processed document the analyzed result data is then stored into the database for later use. With the stored result data it is also possible to instruct the queue manager to rerun specific documents which use a certain macro or to rerun all documents which resulted in a fatal error in the conversion process. The queue manager will then take care of removing the appropriate files and reset the status and add these files to the workqueue again. This has been especially useful when changes to the binding files have been applied or when an improved version of `latexml` becomes available.

Developers are able to retrieve the results via a web-interface which is available at http://arxmliv.kwarc.info. The main page describes the number of documents, the number of converted files in the last 24 hours and the current state of the build system. Since the binding files as well as `LaTeXML` are being developed in distributed environments, subversion is being used as a version control system. The current release version of the binding files and `LaTeXML` are also shown.

Furthermore the active hosts that are currently being used for the conversion process is also displayed. For our experiment we have used 13 different hosts on 24 processors.

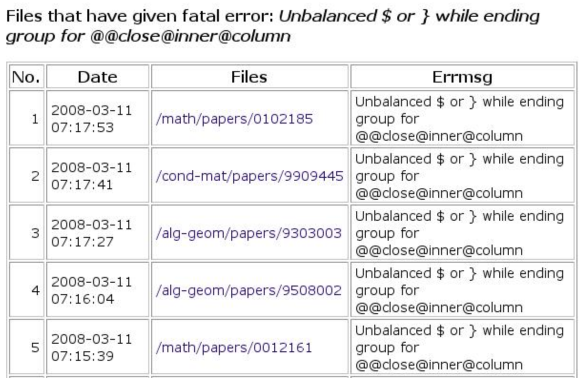
A further table gives detailed information about the results of the conversion process. The most important states are *success* where latexml has only issued some minor warnings, *missing macros*, where the conversion has been successfully completed, but some macro definitions could not be resolved. In this case the rendered layout may contain unexpected elements or not properly displayed elements. The status *fatal_error* is returned from the conversion process if there are too many unresolved macros or if some internal error condition during the LaTeXML conversion process has been triggered.



Conversion tex->xml

return value	count	%	marked for rerun
unknown	134	0.04	
no_latex	29718	n/a	1
missing_errlog	1128	n/a	
fatal_error	32501	8.78	87
timeout	23035	6.22	1
missing_macros	107244	28.98	13472
success	207186	55.98	

Fig. 3. The result status of converted documents



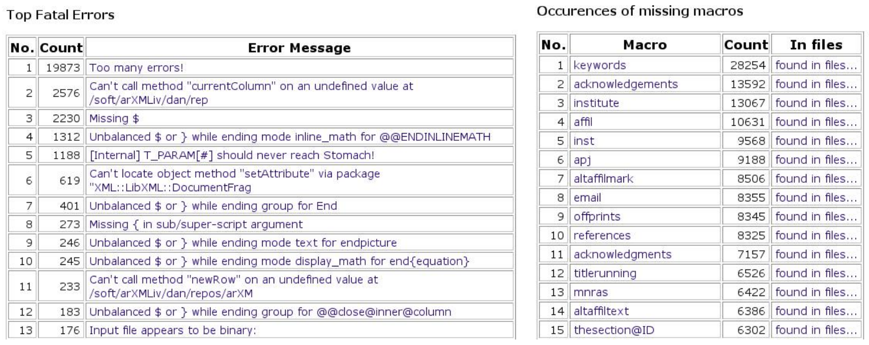
Files that have given fatal error: *Unbalanced $ or } while ending group for @@close@inner@column*

No.	Date	Files	Errmsg
1	2008-03-11 07:17:53	/math/papers/0102185	Unbalanced $ or } while ending group for @@close@inner@column
2	2008-03-11 07:17:41	/cond-mat/papers/9909445	Unbalanced $ or } while ending group for @@close@inner@column
3	2008-03-11 07:17:27	/alg-geom/papers/9303003	Unbalanced $ or } while ending group for @@close@inner@column
4	2008-03-11 07:16:04	/alg-geom/papers/9508002	Unbalanced $ or } while ending group for @@close@inner@column
5	2008-03-11 07:15:39	/math/papers/0012161	Unbalanced $ or } while ending group for @@close@inner@column

Fig. 4. Documents that could not be successfully converted

All these states are clickable and lead to a list of recently converted files with the specified status. The clickable file name leads to the source directory of the document where the document can be investigated in all its different representations, such as the TEX source, as an intermediate XML file that LaTeXML produces or as the XHTML+MathML form. Also the full log file containing detailed error messages can be easily be retrieved via the web browser.

The backend behind the web interface is also able to analyze the database content and create cumulated statistics. It applies some regular expressions to the error messages and clusters and cumulates these. By creating this information the backend is able to gather statistics such as a list of *Top Fatal Errors* and *Top Missing Macros* on-the-fly. Especially these two lists have proven to give valuable information not only the developer of the LaTeXML system but also to the implementers of binding files that are needed to support the conversion from LATEX to XML. With this information one can easily determine the most severe bugs in the still evolving conversion tool as well as determine the macros that are being used by many documents and that need further support.



Top Fatal Errors

No.	Count	Error Message
1	19873	Too many errors!
2	2576	Can't call method "currentColumn" on an undefined value at /soft/arXMLiv/dan/rep
3	2230	Missing $
4	1312	Unbalanced $ or } while ending mode inline_math for @@ENDINLINEMATH
5	1188	[Internal] T_PARAM[#] should never reach Stomach!
6	619	Can't locate object method "setAttribute" via package "XML::LibXML::DocumentFrag
7	401	Unbalanced $ or } while ending group for End
8	273	Missing { in sub/super-script argument
9	246	Unbalanced $ or } while ending mode text for endpicture
10	245	Unbalanced $ or } while ending mode display_math for end{equation}
11	233	Can't call method "newRow" on an undefined value at /soft/arXMLiv/dan/repos/arXM
12	183	Unbalanced $ or } while ending group for @@close@inner@column
13	176	Input file appears to be binary:

Occurences of missing macros

No.	Macro	Count	In files
1	keywords	28254	found in files...
2	acknowledgements	13592	found in files...
3	institute	13067	found in files...
4	affil	10631	found in files...
5	inst	9568	found in files...
6	apj	9188	found in files...
7	altaffilmark	8506	found in files...
8	email	8355	found in files...
9	offprints	8345	found in files...
10	references	8325	found in files...
11	acknowledgments	7157	found in files...
12	titlerunning	6526	found in files...
13	mnras	6422	found in files...
14	altaffiltext	6386	found in files...
15	thesection@ID	6302	found in files...

Fig. 5. Lists of Top fatal errors and of macros that are currently not supported

The arXiv articles use a total of more than 6000 different LaTeX packages. Some of these style files are well known ones which are widely used, while other are private enhancements which are used only once or very few times. While same macro names for different things are not a problem, since the binding files are created for each LaTeX package, there might be a problem that authors add private additions to well-known style files. We have chosen to ignore this problem since it is statistically insignificant.

With these statistics we have been able to focus on the most important macros and have been able to improve the success rate to now more than 58%. Although another 29% of the documents have also been successfully converted and are available as XHTML+MathML, we do not currently count them as full successes since support for some macros is still lacking and the layout that is rendered in a web browser might not be fully appropriate.

4 Conclusion and Outlook

By using the `LaTeXML` tool with our arXMLiv build system to support the conversion process of large document collections, we have been able to successfully convert more than half of the more than 400,000 scientific articles of the arXiv written in LaTeX to a semantically enriched XHTML+MathML representation. We have been able to expand our collection of scientific MathML documents which we need for further studies by more than 200,000 (real-world) documents. The build system has enabled us to cope with the conversion process of this huge collection of documents and helped us to improve our binding files that are needed to support various style files. The statistics the build system gathers have also been valuable contribution to the developer of `LaTeXML` since they clearly point to bugs and give hints where to enhance the software.

Although still under development, `LaTeXML` has shown to be a very promising tool to convert LaTeX documents to XML and hence XHTML+MathML. Many existing LaTeX documents can already be fully converted to an

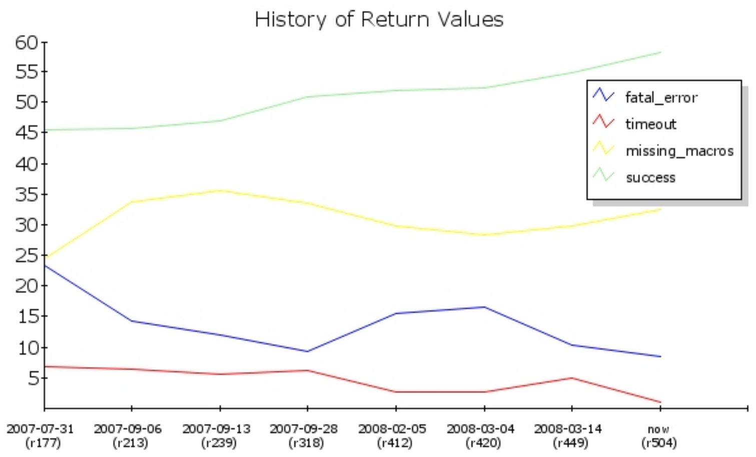


Fig. 6. The history of return values in our conversion experiment

XHTML+MathML representation which may then be nicely rendered inside a browser. The possibility to render and fully integrate these documents inside a browser will enable us to add features on top of existing articles and offer added-value services which we might not even think of yet.

Up to now, the work in the arXMLiv project has focused on driving up the coverage of the translation process and build a tool set that allows us to handle large corpora. With a conversion rate of over 80% we consider this phase as complete. We are currently working to acquire additional corpora, e.g. Zentralblatt Math[2] [ZBM07].

The next steps in the analysis of the arXiv corpus will be to improve the `LaTeXML` post-processing, and in particular the OpenMath/MathML generation. Note that most of the contents in LaTeX documents are presentational in nature, so that content markup generation must be heuristic or based on linguistic and semantic analysis. Rather than relying on a single tool like the `latexmlpost` processor for this task we plan to open up the build system and compute farm to competing analysis tools. These get access to our corpus and we collect the results in a analysis database, which will be open to external tools for higher-level analysis tasks or end-user MKM services like semantic search [Mat08]. For this, we will need to generalize many of the build system features that are currently hard-wired to the translation task and the `LaTeXML` system. We also plan to introduce facilities for *ground-truthing* (i.e. for establishing the intended semantics of parts of the corpus, so that linguistic analysis can be trained on this). For the arXiv corpus this will mean that we add feedback

[2] First tests show that due to the careful editorial structure of this collection and the limited set of macros that need to be supported, our system can reach nearly perfect translation rates.

features to the generated XHTML+MathML that allow authors to comment on the generation and thus the arXMLiv developers to correct their `LaTeXML` bindings.

To allow manual tests for the developers, the build system also includes an additional interface (available at http://tex2xml.kwarc.info) where LaTeX files can be manually uploaded and then converted. It allows to test the conversion without the need to install `LaTeXML` and also makes use of the many additional binding files that we have created to support additional style files and are not part of the standard `LaTeXML` distribution. This interface may also be used to convert private LaTeX files to XHTML+MathML, but because of limited resources only few users can concurrently use this system.

The build system itself is open source software and can be obtained from the authors upon request.

References

[Ang07] Romeo Anghelache. Hermes - a semantic xml+mathml+unicode e-publishing/self-archiving tool for latex authored scientific articles. web page at http://hermes.roua.org/, 2007.

[ArX07] `arXiv.org` e-Print archive, seen December2007. web page at http://www.arxiv.org.

[Koh08] Michael Kohlhase. sTeX: Using TeX/LaTeX as a semantic markup format. *Mathematics in Computer Science; Special Issue on"Management of Mathematical Knowledge"*, 2008. in press.

[Mat08] Math Web Search. web page at http://kwarc.info/projects/mws/, seen October 2008.

[Mil07] Bruce Miller. `LaTeXML`: A LaTeX to xml converter. Web Manual at http://dlmf.nist.gov/LaTeXML/, seen September2007.

[vdBS03] Mark van den Brand and Jürgen Stuber. Extracting mathematical semantics from latex documents. In *Proc. Intl. Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2003)*, number 2901 in LNCS, pages 160–173, Mumbai, India, 2003. Springer.

[ZBM07] Zentralblatt MATH, seen December2007. web page at http://www.zentralblatt-math.org.