Masaryk University
Faculty of Informatics

Normative Textual Representation of Mathematical Formulae

Master's thesis

Maroš Kucbel

Brno, 2013

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Maroš Kucbel

Advisor: doc. RNDr. Petr Sojka, Ph.D.

Acknowledgement

@todo Thanks!

Abstract

@todo Abstract

Keywords

MathML, StAX @todo more

Contents

Chapter 1

Introduction

1 page Estimated length is 30-40 pages of text, which will be supplemented with samples (possibly images) and maybe some source code snippets.

Chapter 2

Motivation

Chapter 3

Mathematical Markup Language - MathML

World Wide Web pages and their main publishing language, HTML, provide many ways to present desired information to the user. However, presenting mathematics is not so easy and straightforward. There aren't any special tags in the HTML specification for including mathematics. In most cases mathematical equations are presented in the form of an image. On one hand this approach renders the equation the same way in every web browser, on the other hand there is no way to copy the equation for further use, not to mention editing the equation.

MathML was specifically created to circumvent this obstacle. It was therefore designed to be used in web pages alongside HTML. It follows that MathML is an application of XML with special set of tags used to capture the content, structure and even presentation details of mathematical equations. We will take closer look at how this is achieved in the following sections. In April 1998 MathML became the recommendation of the W3C working group for including mathematics into web pages.

3.1 Structure

MathML as an application of XML consists of a tree of nodes. Each node is either empty, has textual value or has a list of descendant nodes. To provide additional information, a node can have an arbitrary amount of attributes, key – value pairs. Each MathML tree has to have a root node named math that belongs to the MathML namespace. Also it is important not to forget to include XML declaration and MathML doctype declaration.

MathML comes two distinct sets of tags. For the visual form of equations there are Presentation MathML tags, Content MathML tags focus on the semantics and meaning of formulas. Presentation tags are being used primarily by web browsers to display mathematical expressions to the users, Content tags are important for further processing of expressions by specialized mathematical programs.

3.1.1 Presentation MathML

The main focus of the Presentation MathML is displaying the equation. For this purpose there are around thirty elements, all starting with the prefix "m". Elements are divided into two classes, first of which is called tokens. It consist of elements that represent individual content and do not contain other nested elements. These include:

- <mi>x</mi> - identifiers,
- <mn>2</mn> - numbers,
- <mo>+</mo> - operators, fences, separators,
- <mtext>free text</mtext> - text.

The content of the token can of course be expressed in more than one character (<mo>sin</mo>) or with an XML and HTML character entities, for example `&#62;` and `&gt;` have the same meaning as `>`, greater than. It is completely up to the user which notation he will choose.

Even though nesting other elements in tokens is not allowed, there are exceptions. For example HTML5 allows almost any HTML inline tags inside the mtext element. So <mtext><b>free</b> text</mtext> would be rendered with the bold word free. The other class of elments is layout schemata. This collection of elements is further divided into following groups:

- General Layout
  - <mrow> - general horizontal grouping,
  - <mfrac> - fractions and binomial numbers,
  - <msqrt>, <mroot> - radicals
- Script and Limit - superscripts, subscripts,
- Tabular Math - tables and matrices,
- Elementary Math - notation for lower grades mathematics.

We can thought of the layout schemata as a form of expression constructors, that specify the way in which sub-expression are constructed into larger expressions, starting with tokens and ending with the math element. Therefore elements belonging to layout schemata class do not contain any characters, only other nested elements (layout schemata or tokens).
@todo image

MathML also provides over fifty attributes for fine tuning of expressions, like setting colors, dimensions, alignments and many others.

3.1.2 Content MathML

Mathematical notation, rigorous as it is, is still not standardized around the world and there are many different ways of writing mathematical expressions based on cultural customs. Even simple multiplication operation can be written as x*y, xy, x times y, x x y. In many situations there's no need to actually render the expression, only the underlying meaning is important. For this reason Content MathML provides a framework and a markup language that capture the semantics of mathematical expressions.

The structure of Content MathML tree is based on a very simple principle – applying an operator to sub-expressions. For example the quotient x/y can be thought of as applying the division operator to two arguments x and y. It follows that the cornerstone of Content MathML is the function application element <apply>. The first child of the <apply> element signifies the operator, which can be an <apply> element again, and the rest of the <apply> element direct descendants are arguments to which the operator is applied. Token elements <ci> and <cn> are available to represent numbers and variables respectively.

Content MathML provides two ways of defining the operator. The first one is a set of more than 100 elements, each corresponding to some mathematical operator. For example for the addition operator there is <plus>, for logical xor <xor> and so on. Then there is the second way, the element <csymbol>. This element contains a textual representation of the operator, that is bound to a definition in a content dictionary referenced by either the attribute cd or definitionURL. External content dictionaries are important for communication between agents and there exist several public repositories of mathematical definitions. Most notable is the OpenMath Society repository.

In MathML 3, a subset of Content MathML elements is defined: Strict Content MathML. This uses only minimal, but sufficient, amount of elements, most importantly allows only the usage of <csymbol> element for defining operators. Strict Content MathML is designed to be compatible with OpenMath standard.

3.2 Creating MathML Markup

Creating MathML documents is very simple. All that is needed is a word processor available on every operating system. However, this approach is prone to errors, be it the wrong letter case or unclosed tags. Since MathML is an XML language, the usage of some sophisticated word processor that supports tags highlighting, code completion and XML validation will greatly improve the efficiency of creating MathML documents. But the whole process is still very time consuming and impractical. As is the case with most XML documents, even writing simple mathematical expressions takes up a lot of space and time. Fortunately there are many dedicated MathML editors, that provide some degree of abstraction from the actual MathML markup.

Multiple software products designated for working with mathematical expressions provide an option to output the results of calculations in MathML format. These include popular web service Wolfram Alpha, Mathematica, Maple or Matlab. MathML can also be used as a data exchange format or input format for aforementioned programs.

Another way of creating MathML documents is a conversion from different formats. Among scientist, mathematicians particularly, TEX is the format of choice. It then comes as no surprise that there are many sources of mathematical texts written in TEX. However, not every publication comes with the source files. Often only print-ready PDF files are released. Also many academic writings, especially older ones, are available only in printed form. These have to be scanned using an Optical Character Recognition (OCR) software.

3.2.1 LATEXML

The lack of a suitable tool for converting LATEX to XML prompted the participants of the Digital Library of Mathematical Functions project to develop their own solution. Main goals of LATEXML design contain the aspiration to faithfully emulate TEX behavior, the ease of extensibility and the preservation of both semantic and presentation aspects of original documents. To this end LATEXML provides two main commands, `latexml` and `latexmlpost`. `latexml` converts the initial TEX document to XML based on a set of LATEXML-bindings files, that define the mapping of TEX macros to XML.`latexmlpost` then processes resulting XML output by converting mathematics and graphics, cross-referencing and applying an `XSLT` stylesheet.

Both commands come with a multitude of parameters that allow users to customize the whole process, such as loading user-defined bindings or choosing the output format. Based on the requested output format, mathematics is converted to graphics (png images for HTML) or Presentation MathML in case of XHTML or HTML5.

LATEXML is freely available online as an installation package for Linux systems as well as Windows and MacOS. An extensive and detailed manual is available online or as a PDF document.

3.2.2 Tralics

Tralics is a freeware software designed to translate LaTeXsources into XML documents, that can be further converted into either PDF or HTML. The generated XML is conforming to the local ad-hoc DTD (a simplification of the TEI) with mathematical formulas conforming to the Presentation MathML 2.0 recommendations.

Similarly to LaTeXML, Tralics provides many ways for customization of resulting documents, among them the possibility to change element and attribute names in the XML file. Besides Presentation MathML, mathematical formulae can be translated to LaTeX-like elements as well.

Tralics is readily available online in the form of source files or binaries for Linux, Windows and MacOS operating systems. An extensive documentation regarding customization and usage is also available online.

3.2.3 MaxTract

MaxTract is a tool that through spacial analysis of symbols and fonts in PDF document reverse-engineers its source files in the form of LaTeX or XHTML + MathML documents. The conversion process requires valid PDF files to work correctly as it needs the information about symbols, font encoding and width of objects contained in the PDF file. PDF documents created via LaTeX fulfill these requirements, therefore MaxTract is able to process most of the scientific and mathematical material.

3.2.4 INFTY Reader

INFTY is an Optical Character Recognition (OCR) software especially created for mathematical documents. INFTY reads scanned pages (images) and yields their character recognition results in multiple formats, including LaTeX and MathML. It does so in four steps: layout analysis, character recognition, structure analysis of mathematical expressions, and manual error correction (optional).

The most important phase, character recognition, is responsible for distinguishing mathematical expressions and running a character recognition engine originally developed for mathematical symbols together with non-specialized commercial OCR engine.

INFTY Reader is available free of charge for limited amount of time, then it is necessary to purchase a license key.

3.3 Possible Ambiguities

1-2 pages

3.4 Canonicalization

1-2 pages

3.5 Status Quo of Current Tools

2 pages total

3.5.1 MathPlayer

Chapter 4

Analysis and Solution Proposal

The goal of this thesis is the development of a conversion tool, that will be able to process XML document or documents and transform each occurrence of MathML markup into the appropriate textual representation. For example $5 \times \alpha = x + 3$ should be transformed into "five times alpha equals x plus three". In case of a visual (not semantic) change of the operator the tool should produce the same result. In our example $5 \times \alpha = x + 3$, $5 \cdot \alpha = x + 3$ and $5\alpha = x + 3$ all have the same meaning, so all should be transformed into identical strings.

Since the tool will be used on a corpus composed of thousands of mathematical documents, the computation speed and spacial requirements must be within reasonable limits.

4.1 Input Processing

The batch processing of a substantial amount of documents in a reasonably low time is one of the most important requirements of our conversion tool. Therefore the tool must be able to utilize the ability of modern processors to run applications in multiple threads at the same time.

4.1.1 Multithreading

Performing conversion one document at a time, especially on a batch of a substantial amount of files, is utterly impractical. The application would run for hours or even days or weeks. Therefore it is necessary to run the conversions in parallel, in multiple threads. The number of disposable threads depends on the hardware equipment of the user's computer. It would be desirable to allow the user to set the number of threads available in the thread pool of the application.

Another important factor of speeding up the conversion process represents the sharing of settings and resources among the threads. It lowers the time that each thread spends waiting for external resources, especially those residing on the hard-drive. This can be achieved by using an in memory cache.

It is also essential to ensure that all threads work with the same settings. This can be accomplished by providing a single point of entry to the settings instance. A globally visible class (object) that implements the Singleton design pattern is the best solution for accessing conversion settings across the whole application.

4.1.2 Document Object Model

Document Object Model is one of the most wide-spread methods of representing and working with XML and (X)HTML documents. It retains the tree structure of the XML document and is composed of nodes. Node represent elements in the XML structure and form a hierarchical structure.

Each node, except the top one called root, has a parent. Nodes can have arbitrary amount of children or siblings (nodes with the same parent). This makes traversing the document structure very easy and logical.

A major drawback of using DOM lies in creating the node tree for the whole document. The input document may be very extensive, but contains only a few MathML expressions (pieces of MathML markup). We are only interested in these pieces and therefore a lot of information provided by DOM is redundant and will be discarded. Also the memory and time requirements needed for creating DOM in computing memory increase with the size of input documents.

4.1.3 XSL

Extensible Stylesheet Language (XSL) is a family of languages used to transform and render XML documents. It was created by the W3C consortium and consists of three languages:

- XSL Transformation,
- XSL Formatting Objects,
- XML Path Language.

XSL Transformation (XSLT) XSLT is a language designed for transformation of XML documents into other XML documents, but can also be used to create different output formats, such as HTML, plain text XSL Formatting

Objects. During this process the input documents stays unchanged, rather a new document is created.

XSLT employs special stylesheets to transform document. XSLT stylesheets consist of a set of templates. Each template defines a rule that describes how to process a node that matches the template selector (most commonly an XPath expression).

The transformation process follows a simple algorithm: load stylesheets and create *source tree* from the input document; starting at the root node apply the best matching template; continue till there are nodes to process. This method of processing makes XSLT a declarative language.

XSL Formatting Objects (XSL-FO) XSL-FO document describes what the resulting pages look like and where the various components go, but does not specify the layout. XSL-FO documents are often created by transforming ordinary XML documents using XSL Transformation. An application called FO processor then takes these documents and converts them into some readable form, most often a PDF or PS file. Several FO processor can be employed to achieve different results.

XML Path Language (XPath) XPath is a query language capable of selecting nodes in an XML documents. It also comes with a multitude of functions, such as string operations, date functions or operations with numbers.

XPath expression consists of three components:

- an axis - navigation within the tree (parent, child, sibling, . . .);
- a node test - match the node name to a pattern;
- zero or more predicates - restricting a node set based on conditions.

A disadvantage of XSL Transformations is the high time complexity. However modern XSLT processor employ optimization techniques and different tree representations which makes the transformation process perform better than general-purpose DOM implementations.

4.1.4 Streaming XML

Streaming approach of processing XML documents is event-based and sequential. It follows that there is no tree representation of the document created in the memory. In fact the memory requirements of streaming processing are minimal and often considered negligible. The streaming parser, after reporting an event, usually discards almost all of the information. However it keeps some of the data, for example a list of unclosed nodes in order to catch possible errors.

There is more than a dozen available events that can be reported by the parser, such as:

- document start and end,
- element start and end,
- element text content,
- comments.

The parsing is unidirectional, the parser makes only one run through the document. Therefore every event is reported just once and there is no possibility of traversing the document structure. Hence the streaming processing comes into play when a single pass through the document is sufficient.

Based on the method of reporting events, streaming parsers can be divided into two groups:

- push principle,
- pull principle.

Push Principle When utilizing the push method of streaming XML documents all events are pushed (reported) to the application as soon as they are encountered in the document. The application needs to implement event handlers that will react to individual events and process them accordingly.

The industry standard for parsing based on push model is Simple API for XML (SAX) and has many implementations in multiple programming languages.

Pull Principle Unlike push method, with pull method the application requests the next event when it is ready to process it. The structure of the code that uses pull method resembles structure of XML documents, making it more readable and understandable than applications that use push methods.

There are various APIs that use pull method, among them XML Pull Parser (XPP)[1] and Streaming API for XML (StAX).

[1] `http://www.extreme.indiana.edu/xgws/xsoap/xpp/mxp1/index.html`

4.1.5 XML Data Binding

XML data binding refers to a process of mapping XML documents to objects in computer memory (deserialization). Objects in the application represent the types defined in the XML document (e.g. by XML Schema) and can be generated by various tools from XML Schema, minimizing the required for creating such objects (classes).

The process of mapping XML types onto application objects is called *unmarshalling*. The reverse process, serialization of objects, is called *marshalling*.

The obvious advantage of this method lies in the fact, that on application level we work only with objects and don't have to concern ourselves with XML specific issues. It makes the resulting code easily readable and comprehensive.

Unfortunately the disadvantages of XML data binding are numerous. The time and spacial requirements for marshalling and unmarshalling grow substantially with the size of input document. Also the round-trip (unmarshalling the document and then marshalling it back) may not preserve the sibling order of elements, physical structure of documents (CDATA sections), comments and entity references or DTDs.

Since MathML consists of hundreds of elements and each one has to be mapped to a class when using XML data binding method, writing the bindings by hand is out of a question. There is however an XML Schema of MathML available that could be used to generate said bindings. Unfortunately attempts of code generation from this schema failed. Moreover if MathML markup was embedded inside another XML markup, such as XHTML, the bindings for MathML only would not be sufficient.

4.1.6 Proposed Solution

As we can see every method comes with many advantages as well as disadvantages making it the right choice in various situations. We will try to select one method, or a combination of two methods, best suited for our use case and requirements.

From the developer's point of view, the best solution would be to use XML data binding. However the above outlined disadvantages make it impossible to actually implement this method.

The next best approach is using the DOM with its easy to traverse structure. But there remains the issue of memory and time requirements needed to create the model. Going by these requirements the streaming method wins as it is the fastest and most memory-efficient method.

The optimal solution would be to pass through the document using the streaming method. But streaming method does not provide any means to look ahead or behind on elements that might be needed to process the current element. On the other hand the DOM method is well suited for accessing arbitrary elements in the document tree. Combining these two methods seems to be the right way to go.

4.2 Recursive Approach

4.3 Division of Mathematical Operations

Chapter 5

Converter Implementation

5.1 Java and XML

3-4 pages Options of working with XML in Java, selecting the most suitable and why

5.1.1 StAX

1-2 pages More details about StAX

5.1.2 Custom XML Tree Representation

1-2 pages Creating custom tree with similar structure as DOM but less additional info and therefore lower space and time requirements

5.2 Presentation MathML

4-5 pages Specifics of impementing converter for presentation MathML

5.3 Content MathML

4-5 pages Specifics of impementing converter for content MathML, possibly longer

5.3.1 Element `apply`

Chapter 6

Results

6.1 Gensim

6.2 MREC Corpus

Chapter 7

Conclusion

1 page