

The Mathematical Markup Language

and Web technologies for mathematics

Luca Padovani

`lpadovan@cs.unibo.it`

Department of Computer Science
University of Bologna

Summary

- Brief introduction to XML, emphasis on encoding of mathematics
- MathML markup
- Transformations of MathML markup
- $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, and MathML
- Some practical issues
- Live demonstration

<http://www.w3.org/Math>

XML

Unicode

- XML documents are *made of* Unicode characters

character The smallest component of written language that has semantic value; refers to the abstract meaning and/or shape, rather than a specific shape

glyph, glyph image The actual, concrete image of a glyph representation having been rasterized or otherwise images onto some display device

a a a a a a

Unicode

character repertoire A set of distinct characters, with no specific internal representation in computers or data transfer sessions

character code A mapping that defines a one-to-one correspondence between characters in a character repertoire and a set of nonnegative, not necessarily consecutive integers

character encoding A method (algorithm) for representing characters in digital form by mapping sequences of code numbers of characters into sequences of octets

Unicode

character repertoire

$\{+, \times, 0, 1, \infty\}$

character code

$\{+ \mapsto 002B_{16}, 0 \mapsto 0030_{16}, 1 \mapsto 0031_{16}, \infty \mapsto 221E_{16}\}$

character encoding

$\{\{00_{16}, 2B_{16}\}, \{00_{16}, 30_{16}\}, \{00_{16}, 31_{16}\}, \{22_{16}, 1E_{16}\}\}$

Among the most popular encodings: **UTF-8**, **UTF-16**,
UCS-4

Unicode and Mathematics

- mathematical symbols and operators
- technical symbols commonly used in mathematical literature (arrows)
- definition of mathematical properties for characters (operator, digit, delimiter, ...)
- letter-like symbols ($a \mapsto 1D44E_{16}$, mathematical italic small A, typically used for variables)

XML

XML makes explicit what we take for granted...

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 28$$

XML

XML makes explicit what we take for granted...

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 28$$

$$\lim_{[i x]_i [o \rightarrow]_o [n 0]_n} \frac{[i \sin]_i [i x]_i}{[i x]_i} [o =]_o [n 28]_n$$

XML

XML makes explicit what we take for granted...

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 28$$

$$\lim_{[i x]_i [o \rightarrow]_o [n 0]_n} \frac{[i \sin]_i [i x]_i}{[i x]_i} [o =]_o [n 28]_n$$

$$\lim_{[e [i x]_i [o \rightarrow]_o [n 0]_n]_e} \frac{[e [i \sin]_i [i x]_i]_e}{[i x]_i} [o =]_o [n 28]_n [e]_m$$

... basically because “we” doesn’t comprise the machines (yet)

XML

- The classification in terms of labels is completely arbitrary
- There is no intended semantics in the labels themselves
- The level of structure is arbitrary
- XML specifies *how* to apply labels
- XML does **not** specify what labels mean
- XML does **not** specify how fine the structuring should be

The last two points are up to the specific *application domain* of concern

XML documents

```
<xml? version="1.0" encoding="UTF-8"?>  
<!DOCTYPE math "mathml2.dtd">  
<math xmlns="http://www.w3.org/1998/Math/MathML">  
  ...  
</math>
```

- XML document compliant to XML 1.0 specification
- (Unicode) characters are encoded using **UTF-8**
- the root “element” of the document is `math`
- the document should conform the “grammar” specified in the `mathml2.dtd` DTD
- the “default namespace” of the “elements” under `math` and `math` itself is...

Elements

Elements are a pair of matching labels with other things (possibly nothing) in between:

```
<workshop> . . . </workshop>
```

Empty elements are honored by an abbreviated syntax:

```
<workshop />
```

Elements may have *attributes*:

```
<workshop name="MathML and . . ."> . . .
```

Elements

```
<xml? version="1.0" encoding="UTF-8"?>
<!DOCTYPE workshop "workshop.dtd">
<workshop name="Web technologies">
  <lecture title="XML">
    <lecturer name="X" />
    <when date="Wed" from="8" to="12" />
    <when date="Thu" from="8" to="12" />
  </lecture>

  <lecture title="MathML">
    <lecturer name="Y" />
    <when date="Thu" from="14" to="16" />
    <when date="Fri" from="8" to="12" />
  </lecture>
</workshop>
```

Characters

- We can use characters outside attribute values. It is always a problem to decide what is data and what is an attribute
- A possible rule is: think of the plain document; the actual content you see is data; make structure explicit; refine the structure with attributes.

Example:

```
<lecturer name="X" />  
<lecturer>X</lecturer>
```

We can associate a sequence of character to a symbolic name and use the name instead (*entity reference*):

```
<lecturer>&lecturer_with_very_long_name;</lecturer>
```

Well-formedness and Validity

A well-formed XML document is characterized by:

- markup must be properly structured. `<a>` is **not** properly structure
- unique attributes. **Bad:** ``
- referenced entities must be declared

Validity is a property of documents w.r.t. a DTD. A document is valid if it respects the constraints on the attributes and the content model of elements as specified in the DTD.

```
<!ELEMENT a (b, (c|d))>
```

```
<!ATTLIST a id CDATA #IMPLIED>
```

```
<a><b /><d /></a>
```

Namespaces

Complexity arise when two XML applications must mix. For instance XHTML + MathML. The solution is to use namespaces:

- XML languages are identified by a URI
 - MathML:
`http://www.w3.org/1998/Math/MathML`
 - XHTML: `http://www.w3.org/1999/xhtml`
- They can be indicated in two ways
 - By using an `xmlns` attribute on an element
 - By adding a prefix to element names

Namespaces

Use the `xmlns` attribute on the outermost element of the embedded markup. This places the element on which the attribute is set, and its descendant in the indicated namespace.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
  ...
  <math
    xmlns="http://www.w3.org/1998/Math/MathML" >
    <mi>x</mi><mo>+</mo><mn>2</mn>
  </math>
  ...
</html>
```

Namespaces

To use prefixes, you must

- Associate a prefix and a namespace using an `xmlns:prefix` attribute on a containing element
- Use the prefix to identify elements (and attributes) that should be in the namespace

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:m="http://www.w3.org/1998/Math/MathML" >
  ...
  <m:math>
    <m:mi>x</m:mi><m:mo>+</m:mo><m:mn>2</m:mn>
  </m:math>
  ...
</html>
```

MathML

Overview of MathML

- The Mathematical Markup Language (MathML, version 1) was first published as a recommendation in April 1998, then revised and re-published in February 2001 (version 2)
- From the “Math Activity Statement” of the W3C Math Working Group:

Overview of MathML

- The Mathematical Markup Language (MathML, version 1) was first published as a recommendation in April 1998, then revised and re-published in February 2001 (version 2)
- From the “Math Activity Statement” of the W3C Math Working Group:
 - “Designed as an XML application, MathML provides two sets of tags, one for the *visual presentation* of mathematics and the other associated with *the meaning behind expressions*.”

Overview of MathML

- The Mathematical Markup Language (MathML, version 1) was first published as a recommendation in April 1998, then revised and re-published in February 2001 (version 2)
- From the “Math Activity Statement” of the W3C Math Working Group:
 - “. . . two sets of tags. . . .”
 - “MathML is not designed for people to enter by hand but *specialized tools provide the means for typing in and editing mathematical expressions.*”

XML + math = MathML

Goal: define an XML application to encode mathematical expressions for inclusion in Web pages, communication between applications, long- and short-term storage.

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 28$$

XML + math = MathML

Goal: define an XML application to encode mathematical expressions for inclusion in Web pages, communication between applications, long- and short-term storage.

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 28$$

notation the three letters 'l' 'i' and 'm' in roman, below them the expression made of the letter 'x' in italic, followed by a right arrow, then the digit 0. Next to this, a horizontal line with...

XML + math = MathML

Goal: define an XML application to encode mathematical expressions for inclusion in Web pages, communication between applications, long- and short-term storage.

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 28$$

notation the three letters 'l' 'i' and 'm' in roman, below them the expression made of the letter 'x' in italic, followed by a right arrow, then the digit 0. Next to this, a horizontal line with...

meaning consider the function sine of x over x as x approaches 0, take the limit of this function, the result is 28.

MathML

- About 30 MathML presentation elements which accept about 50 attributes

MathML

- About 30 MathML presentation elements which accept about 50 attributes
 - Most elements represent templates or patterns for laying out subexpressions. For example, there is an `mfrac` element for fractions, and an `msqrt` element for square roots
 - Attributes generally specify additional optional information about the element layout. For example, the `mfrac` element has an attribute called `linethickness`

MathML

- About 30 MathML presentation elements which accept about 50 attributes
- Using presentation markup, it's possible to precisely control how an expression will look when displayed

MathML

- About 30 MathML presentation elements which accept about 50 attributes
- Using presentation markup, it's possible to precisely control how an expression will look when displayed
- About 120 content elements, accepting about a dozen attributes

MathML

- About 30 MathML presentation elements which accept about 50 attributes
- Using presentation markup, it's possible to precisely control how an expression will look when displayed
- About 120 content elements, accepting about a dozen attributes
 - Most content elements represent either operators or mathematical data types. For example, there is a `divide` element for division, and an `emptyset` element to denote the empty set

MathML

- About 30 MathML presentation elements which accept about 50 attributes
- Using presentation markup, it's possible to precisely control how an expression will look when displayed
- About 120 content elements, accepting about a dozen attributes
- Content markup facilitates applications other than display, like computer algebra and speech synthesis

MathML Presentation Overview

- tokens (atomic entities)
- general layout schemata (fractions, square roots, . . .)
- scripts and limits
- tables and alignment
- “live” expressions

MathML Presentation Overview

- tokens (atomic entities)
- general layout schemata (fractions, square roots, . . .)
- scripts and limits
- tables and alignment
- “live” expressions

There is a fair amount of *semantics* even in presentation elements:

- refine formatting, higher quality
- “meaningful” presentation (conversions)

Presentation: tokens

- `mi` identifiers
- `mo` operators
- `mn` numbers
- `mtext` text

Presentation: tokens

- `mi` identifiers
- `mo` operators
- `mn` numbers
- `mtext` text

Examples:

<code><mi>x</mi></code>	x
<code><mo>+</mo></code>	+
<code><mn>28</mn></code>	28
<code><mtext>suppose that</mtext></code>	suppose that

Presentation: tokens

More examples:

`<mi>sin</mi>`

sin

`<mo>lim</mo>`

lim

`<mi>Π</mi>`

π

`<mn>two</mn>`

two

`<mi mathvariant="bold">x</mi>`

x

`<mi mathcolor="red">x</mi>`

x

Presentation: tokens

More examples:

<code><mi>sin</mi></code>	sin
<code><mo>lim</mo></code>	lim
<code><mi>&Pi;</mi></code>	π
<code><mn>two</mn></code>	two
<code><mi mathvariant="bold">x</mi></code>	x
<code><mi mathcolor="red">x</mi></code>	<i>x</i>

Remark: token elements are the only elements that can directly contain character data.

Presentation: grouping

The `mrow` element groups subexpressions and align their content on a horizontal line

```
<mrow>  
  <mrow>  
    <mi>x</mi>  
    <mo>+</mo>  
    <mn>1</mn>  
  </mrow>  
  <mo>=</mo>  
  <mn>0</mn>  
</mrow>
```

$$x + 1 = 0$$

Presentation: fractions

```
<mfrac> <mi>a</mi> <mi>b</mi> </mfrac>
```

$$\frac{a}{b}$$

```
<mfrac linethickness="0">...</mfrac>
```

$$\frac{a}{b}$$

```
<mfrac numalign="left">...</mfrac>
```

$$\frac{a}{x + 1}$$

```
<mfrac bevelled="true">...</mfrac>
```

$$a/b$$

Presentation: radicals

```
<msqrt> <mn>2</mn> </msqrt>
```

$$\sqrt{2}$$

```
<mroot>  
  <mn>2</mn>  
  <mrow>  
    <mi>n</mi>  
    <mo>+</mo>  
    <mn>1</mn>  
  </mrow>  
</mroot>
```

$${}^{n+1}\sqrt{2}$$

Presentation: scripts

```
<msub> <mi>x</mi> <mn>2</mn> </msub>
```

 x_2

```
<msup> <mi>x</mi> <mn>2</mn> </msup>
```

 x^2

```
<msubsup>
```

```
  <mi>x</mi> <mi>k</mi> <mn>2</mn>
```

```
</msubsup>
```

 x_k^2

Presentation: multiple scripts

```
<mmultiscripts>  
  <mi>x</mi>  
  <mi>i</mi>  
  <none/>  
  <mi>j</mi>  
  <mi>k</mi>  
  <mprescripts/>  
  <none/>  
  <mi>m</mi>  
</mmultiscripts>
```

$$m x_{ij}^k$$

Presentation: accents and limits

```
<mover>  
  <mi>x</mi>  
  <mo>&#x0307;</mo>  
</mover>
```

\dot{x}

Hint: U+0307 is the Unicode character “combining dot above”. In this context it means “derivative” (Newtonian notation).

Compare with

```
<mi>x&#x0307;</mi>
```

\dot{x}

Presentation: accents and limits

```
<munder>  
  <mo>lim</mo>  
  <mrow>  
    <mi>x</mi>  
    <mo>&#x2192;</mo>  
    <mn>0</mn>  
  </mrow>  
</munder>
```

$$\lim_{x \rightarrow 0}$$

Hint: U+2192 is the Unicode character “rightwards arrow.”

This is an example of *embellished operator*.

Presentation: matrices and tables

```
<table>  
  <tr>  
    <td> <mn>1</mn> </td>  
    <td> <mn>0</mn> </td>  
  </tr>  
  <tr>  
    <td> <mn>0</mn> </td>  
    <td> <mn>1</mn> </td>  
  </tr>  
</table>
```

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

Presentation: stretchy operators

```
<mrow>  
  <mo stretchy="true">( </mo>  
  <mfrac> <mn>1</mn> <mn>2</mn> </mfrac>  
  <mo stretchy="true">)</mo>  
</mrow>
```

$$\left(\frac{1}{2}\right)$$

```
<mover>  
  <mrow> <mi>x</mi> <mo>+</mo> <mn>1</mn> </mrow>  
  <mo stretchy="true">&#x0305;</mo>  
</mover>
```

$$\overline{x + 1}$$

Hint: U+0305 is the Unicode character “combining overline.”

Presentation: live expressions

The `maction` element binds actions to a sub-expression:

```
<mrow>
```

```
  <mrow>
```

```
    <mn>1</mn>
```

```
    <mo>+</mo>
```

```
    <mn>1</mn>
```

```
  </mrow>
```

```
  <mo>=</mo>
```

```
  <maction selection="1" actiontype="toggle">
```

```
    <mtext>?</mtext>
```

```
    <mn>2</mn>
```

```
  </maction>
```

```
</mrow>
```

1 + 1 = ?

1 + 1 = 2

Presentation: fine points

Sometimes we want, or it is better, to make explicit even what is normally invisible:

```
<mathrow>  
  <math>\sin</math>  
  <math>x</math>  
</mathrow>
```

$\sin x$

An `mathrow` with no operators: *suspicious*

Presentation: fine points

```
<mathrow>
  <math>\sin</math>
  <math>\#x2061</math>
  <math>x</math>
</mathrow>
```

$\sin x$

Hint: U+2061 is the Unicode character “function application”

Presentation: fine points

A formatting engine for MathML can exploit invisible operators to produce better output.

$\sin x$

$\sin(x)$

Presentation: fine points

A formatting engine for MathML can exploit invisible operators to produce better output.

$\sin x$

$\sin(x)$

Other invisible operators:

- U+2062 invisible times: xy
- U+2063 invisible separator: xij
- invisible plus: $1\frac{1}{2}$

Presentation: example

```
<math xmlns="http://www.w3.org/1998/Math/MathML" >
  <mrow>
    <mrow>
      <munder>
        <mo>lim</mo>
        <mrow>
          <mi>x</mi>
          <mo>&RightArrow;</mo>
          <mn>0</mn>
        </mrow>
      </munder>
    <mfrac>
      <mrow>
        <mi>sin</mi>
        <mo>&ApplyFunction;</mo>
        <mi>x</mi>
      </mrow>
      <mi>x</mi>
    </mfrac>
  </mrow>
  <mo>=</mo>
  <mn>28</mn>
</mrow>
</math>
```

MathML Content Overview

- Token elements (atomic entities)
- Arithmetic, Algebra, and Logic (functions, operators)
- Relations (equality)
- Calculus and Set Theory
- Object constructors (sets, lists)

MathML Content Overview

- Token elements (atomic entities)
- Arithmetic, Algebra, and Logic (functions, operators)
- Relations (equality)
- Calculus and Set Theory
- Object constructors (sets, lists)

`apply` is the main grouping element in MathML content markup. It is semantical, not presentational (`mathrow`).

Content: Tokens

- `ci` identifiers
- `cn` numbers
- `csymbol` extensions

Content: Tokens

- `ci` identifiers
- `cn` numbers
- `csymbol` extensions

Examples:

<code><ci>x</ci></code>	x
<code><cn>28</cn></code>	28
<code><csymbol>bang</csymbol></code>	!

Content: grouping

The `apply` element groups subexpressions and relates them with respect to a function or an operator:

```
<apply>
  <eq/>
  <apply>
    <plus/>
    <ci>x</ci>
    <cn>1</cn>
  </apply>
  <cn>0</cn>
</apply>
```

```
<mrow>
  <mrow>
    <mi>x</mi>
    <mo>+</mo>
    <mn>1</mn>
  </mrow>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
```

$$x + 1$$

Content: grouping

```
<apply>
  <eq/>
  <apply>
    <cos/>
    <ci>pi</ci>
  </apply>
  <apply>
    <minus/>
    <cn>1</cn>
  </apply>
</apply>
```

```
<mrow>
  <mrow>
    <mi>cos</mi>
    <mo>&ApplyFunction;</mo>
    <mi>&pi;</mi>
  </mrow>
  <mo>=</mo>
  <mrow>
    <mo>-</mo>
    <mn>1</mn>
  </mrow>
</mrow>
```

$$\cos \pi = -1$$

Content: extensions

- MathML content defines a *finite* set of functions and operators.
- MathML content \subset K-12
- MathML content is aimed at presentation

We can use the `csymbol` element to introduce new functions or operators:

```
<apply>
  <csymbol>par</csymbol>
  <ci><msub><mi>P</mi><mn>1</mn></msub></ci>
  <ci><msub><mi>P</mi><mn>2</mn></msub></ci>
</apply>
```

$$P_1 \mid P_2$$

Content: semantics

- How do we know that operators and functions mean what we mean?
- How do we refine the meaning of a pre-defined MathML symbol?
- How do we give meaning to an extension (`csymbol`)?
- How do we relate a piece of content markup with the corresponding presentation markup?

These problems are not solvable *within* MathML markup. MathML gives us structure and a limited amount of “intuitive” semantics.

Applications can exploit the markup (structure, annotations) to refine the “meaning.”

Content: `definitionURL`

Most MathML content elements accept a `definitionURL` whose purpose is to *point* to another document where the meaning of the element is defined “in some way.”

```
<apply>  
  <plus definitionURL="http://.../plus_1.xml/">  
    ...  
</apply>
```

```
<apply>  
  <plus definitionURL="http://.../plus_2.xml/">  
    ...  
</apply>
```

Approximately, intuitively, structurally the two `pluses` do the same thing (they add stuff)

Content: definitionURL

Still hard to think of two different pluses?

- integers vs. reals
- numbers vs. strings
- with overflow vs. without overflow
- unary encoding vs. binary encoding
- many others...

Content: definitionURL

Still hard to think of two different pluses?

- integers vs. reals
- numbers vs. strings
- with overflow vs. without overflow
- unary encoding vs. binary encoding
- many others...

Some applications:

- cut and paste
- computing
- searching

Content: definitionURL

The `definitionURL` attribute can also be used on the `csymbol` element:

```
<apply>  
  <csymbol definitionURL="...">...</csymbol>  
</apply>
```

This way MathML content has an extension mechanism that we can target to our application domain.

Examples:

- things not covered by MathML
- mathematics outside K-12
- non-mathematical formulas (logic, chemistry, process algebra, ...)

Content: `definitionURL`

MathML does **not** specify what we find at `definitionURL`:

- XML document (OpenMath content dictionary)
- text document
- another document
- nothing (the meaning is in the URL itself)

Content: definitionURL

MathML does **not** specify what we find at definitionURL:

- XML document (OpenMath content dictionary)
- text document
- another document
- nothing (the meaning is in the URL itself)

The standardization of this mechanism is *hard*.

Some companies are developing large dictionaries of functions and operators \Rightarrow feedback.

Content: semantics

Sometimes we need to relate presentation and content markup.

Example:

- presentation is ambiguous
- presentation is meaningless (re-structuring)

Content: semantics

Sometimes we need to relate presentation and content markup.

Example:

- presentation is ambiguous
- presentation is meaningless (re-structuring)

Or the other way round:

- we want to associate a non-standard rendering to a piece of content markup
- we want to associate presentation to an extension for which there is no default rendering

Content: semantics

```
<semantics>
  <mrow>
    <mi>x</mi>
    <mo>+</mo>
    <mn>1</mn>
  </mrow>
  <annotation-xml encoding="MathML-Content">
    <apply>
      <csymbol>succ</csymbol>
      <ci>x</ci>
    </apply>
  </annotation-xml>
</semantics>
```

Content: semantics

```
<semantics>
  <mrow id="i1">
    <mi id="i2">x</mi>
    <mo>+</mo>
    <mn>1</mn>
  </mrow>
  <annotation-xml encoding="MathML-Content">
    <apply xref="i1">
      <csymbol>succ</csymbol>
      <ci xref="i2">x</ci>
    </apply>
  </annotation-xml>
</semantics>
```

Content: semantics

The use of *mixed markup* can increase exponentially the size of the documents!

The two light possibilities are:

- top-level `semantics` element
- parallel markup in a different document

Content: semantics

The use of *mixed markup* can increase exponentially the size of the documents!

The two light possibilities are:

- top-level `semantics` element
- parallel markup in a different document

When presentation and content are not embedded in the *same* document:

- Pro: documents are light, if the parallel markup is not needed we don't need to parse it
- Cons: documents are not self-contained, it is harder to retrieve the information (XML pipelines usually have a single flow-path)

Aural rendering

MathML markup has support for aural rendering in both Content markup and presentation markup.

Two equivalent ways to render (aurally) “the sum of n elements x_1, \dots, x_n .”

$$x_1 + x_2 + \cdots + x_n$$

Aural rendering

MathML markup has support for aural rendering in both Content markup and presentation markup.

Two equivalent ways to render (aurally) “the sum of n elements x_1, \dots, x_n .”

$$x_1 + x_2 + \cdots + x_n$$

Or:

$$\sum_{i=1}^n x_i$$

Aural rendering

MathML markup has support for aural rendering in both Content markup and presentation markup.

Two equivalent ways to render (aurally) “the sum of n elements x_1, \dots, x_n .”

$$x_1 + x_2 + \cdots + x_n$$

Or:

$$\sum_{i=1}^n x_i$$

Probably a combination of content and presentation is the most flexible way in this context.

Transformations

Content \Rightarrow Presentation

It is easy to conceive a transformation process from MathML content to MathML presentation.

Example:

```
<apply>  
  <sin/>  
  <mi>x</mi>  
</apply>
```

Can be *automatically* transformed to:

```
<mrow>  
  <mi>sin</mi>  
  <mo>&ApplyFunction;</mo>  
  <mi>x</mi>  
</mrow>
```

Content \Rightarrow Presentation

MathML content can be seen as an “abstract” form of presentation.

Example: at the content level the notion of “division” is unique, whereas in presentation we have many different forms.

```
<apply>
  <divide/>
  <ci>a</ci>
  <ci>b</ci>
</apply>
```

$$\frac{a}{b}$$

```
<mfrac>
```

$$a/b$$

```
<mfrac bevelled="true">
```

$$a/b$$

```
<mo>/</mo>
```

XSLT

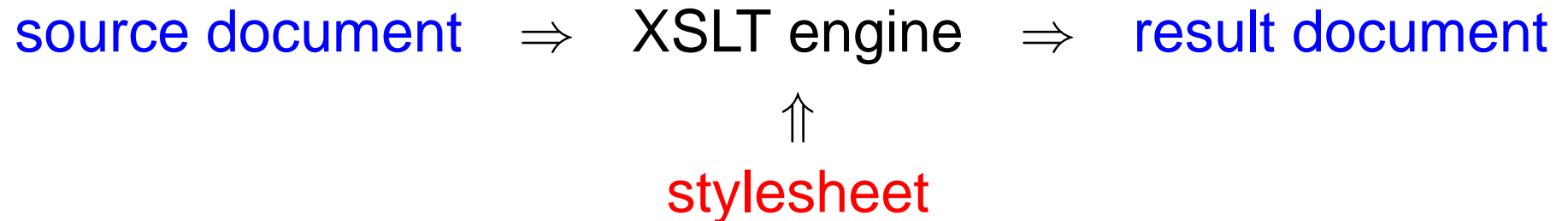
XSLT (eXtensible Stylesheet Language Transformation) can be used to this purpose:

- targeted to transformation of XML documents (tree transformation language)
- based on *pattern-matching*

XSLT

XSLT (eXtensible Stylesheet Language Transformation) can be used to this purpose:

- targeted to transformation of XML documents (tree transformation language)
- based on *pattern-matching*



XSLT and MathML

The following fragment of XSLT transform a `divide` application into presentation markup:

```
<xsl:template match="m:apply[*[1][self::m:divide]]">
  <m:mfrac>
    <xsl:apply-templates select="*[2]"/>
    <xsl:apply-templates select="*[3]"/>
  </m:mfrac>
</xsl:template>
```

```
<xsl:template match="m:apply[*[1][self::m:divide]]">
  <m:mrow>
    <xsl:apply-templates select="*[2]"/>
    <m:mo>/</m:mo>
    <xsl:apply-templates select="*[3]"/>
  </m:mrow>
</xsl:template>
```

Not shown: precedence of operators, etc.

XSLT and MathML

- it is possible to design a stylesheet that generates mixed markup (content + presentation) at any level of detail.
- it is possible to automatically generate cross-links between fragments of content and presentation markup.

<http://www.orcca.on.ca/MathML>

See also “Content-Faithful Transformations for MathML” at

<http://www.mathmlconference.org/2002/presentations.html>

MathML, T_EX, and L^AT_EX

Syntax

\LaTeX syntax:

- easy to type: “experience shows that untrained personnel can learn how to type \TeX without difficulty” in a short amount of time (Knuth)
- no Unicode
- loose rules (e.g. grouping)

Syntax

\LaTeX syntax:

- easy to type: “experience shows that untrained personnel can learn how to type \TeX without difficulty” in a short amount of time (Knuth)
- no Unicode
- loose rules (e.g. grouping)

MathML syntax:

- MathML is an XML application \Rightarrow syntax of XML documents
- Unicode characters
- stricter rules (proper grouping, nesting, invisible operators)

Semantics

$\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ are purely presentational. MathML is concerned with both semantics and presentation.

Semantics in presentation:

- structure is semantics: grouping reflects the operator precedence $[\text{mrow } x + [\text{mrow } x \times y]_{\text{mrow}}]_{\text{mrow}}$ (line-breaking)

Semantics

$\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ are purely presentational. MathML is concerned with both semantics and presentation.

Semantics in presentation:

- structure is semantics: grouping reflects the operator precedence $\text{[mrow } x + \text{[mrow } x \times y \text{]mrow]mrow}$ (line-breaking)
- Unicode classifies characters according to their semantics, rather than their shape. Compare: $\backslash\text{mathbf}\{A\}$ in $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and U+1D404 “mathematical bold capital A” in Unicode.

Semantics

$\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ are purely presentational. MathML is concerned with both semantics and presentation.

Semantics in presentation:

- structure is semantics: grouping reflects the operator precedence $\text{mrow}x + \text{mrow}x \times y\text{mrow}\text{mrow}$ (line-breaking)
- Unicode classifies characters according to their semantics, rather than their shape. Compare: $\backslash\text{mathbf}\{A\}$ in $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and U+1D404 “mathematical bold capital A” in Unicode.
- no shortcuts: invisible operators

Semantics

$\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ are purely presentational. MathML is concerned with both semantics and presentation.

Semantics in presentation:

- structure is semantics: grouping reflects the operator precedence $\left[\left[\left[\text{mrow } x + \left[\text{mrow } x \times y \right] \right] \right] \right]$ (line-breaking)
- Unicode classifies characters according to their semantics, rather than their shape. Compare: $\backslash\text{mathbf}\{A\}$ in $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and U+1D404 “mathematical bold capital A” in Unicode.
- no shortcuts: invisible operators
- **content markup**

Formatting Quality

- Quality of T_EX's formatting excellent, but its formatting rules are *fixed*.

Formatting Quality

- Quality of T_EX's formatting excellent, but its formatting rules are *fixed*.
 - authors know how to tweak it to do what they want
 - it is guaranteed that all T_EX implementations will render it identically

Formatting Quality

- Quality of T_EX's formatting excellent, but its formatting rules are *fixed*.
 - authors know how to tweak it to do what they want
 - it is guaranteed that all T_EX implementations will render it identically
- MathML formatting rules are much looser

Formatting Quality

- Quality of T_EX's formatting excellent, but its formatting rules are *fixed*.
 - authors know how to tweak it to do what they want
 - it is guaranteed that all T_EX implementations will render it identically
- MathML formatting rules are much looser
 - the fine points of the formatting process are up to the formatting engine
 - MathML does suggest certain conventions
 - tweaking a MathML document might not be a good idea (in fact, it may compromise decent rendering in some contexts)

Formatting Quality

However:

- provided that quality fonts are available (e.g. Computer Modern), and...

Formatting Quality

However:

- provided that quality fonts are available (e.g. Computer Modern), and...
- ... the formatting engine has been designed carefully

MathML can achieve the same formatting quality of $\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ (without cheating).

Formatting Quality

However:

- provided that quality fonts are available (e.g. Computer Modern), and...
- ... the formatting engine has been designed carefully

MathML can achieve the same formatting quality of $\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ (without cheating).

Some publishers (Wiley) are already adopting MathML for encoding mathematics in scientific articles, encyclopedias.

Macros and Transformations

- $\text{T}_{\text{E}}\text{X}$ is based on *macros* (named, possibly parameterized sequence of $\text{T}_{\text{E}}\text{X}$ commands that expands wherever the name occurs in the document)
- macros make $\text{T}_{\text{E}}\text{X}$ usable
- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ is “just” a large collection of macros

Macros and Transformations

- $\text{T}_{\text{E}}\text{X}$ is based on *macros* (named, possibly parameterized sequence of $\text{T}_{\text{E}}\text{X}$ commands that expands wherever the name occurs in the document)
- macros make $\text{T}_{\text{E}}\text{X}$ usable
- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ is “just” a large collection of macros
- in MathML there are no macros
- it is not obvious whether a macro mechanism would be useful (MathML is not hand-written)
- MathML content can be seen as a “macro” version of MathML presentation
- XSLT can be seen as a macro-expanding language

From T_EX, L^AT_EX to MathML (and back)

At the MathML implementations page:

`http://www.w3.org/Math/implementations.html`

there are several tools transforming T_EX and L^AT_EX to MathML. A few of them do the opposite.

From T_EX, L^AT_EX to MathML (and back)

At the MathML implementations page:

<http://www.w3.org/Math/implementations.html>

there are several tools transforming T_EX and L^AT_EX to MathML. A few of them do the opposite.

Applications:

- converting L^AT_EX documents for Web presentation
- using T_EX, L^AT_EX syntax for writing MathML
- rendering MathML using T_EX formatting engine

From T_EX, L^AT_EX to MathML (and back)

At the MathML implementations page:

<http://www.w3.org/Math/implementations.html>

there are several tools transforming T_EX and L^AT_EX to MathML. A few of them do the opposite.

Applications:

- converting L^AT_EX documents for Web presentation
- using T_EX, L^AT_EX syntax for writing MathML
- rendering MathML using T_EX formatting engine

Benefits:

- MathML is Web-friendly
- semantics (search, computation, . . .)
- long-time storage (easy processing)

MathML in practice

Support for MathML

- Windows...
 - IE 5.0 with the Techexplorer plug-in
 - IE 5.5 with either the MathPlayer or Techexplorer plug-ins
 - IE 6.0, optionally with MathPlayer or Techexplorer plug-ins
 - Netscape 6.1 with Techexplorer plug-in
 - Netscape 7.0 PR1
 - Amaya (Presentation MathML only)
 - Mozilla 0.9.9

Support for MathML

- Windows...
- Macintosh...
 - IE 5.0 with Techexplorer plug-in
 - Mozilla 0.9.9

Support for MathML

- Windows...
- Macintosh...
- Linux/Unix...
 - Netscape 6.1 with Techexplorer plug-in
 - Netscape 7.0 PR1
 - Mozilla 0.9.9
 - Amaya (Presentation MathML only)

MathML Stylesheet

There are different ways of embedding MathML markup embedded inside XHTML markup.

- platform: Windows, Macintosh, Linux
- browser: Internet Explorer, Mozilla.
- plug-in: Mathplayer, Techexplorer, ...

MathML Stylesheet

There are different ways of embedding MathML markup embedded inside XHTML markup.

- platform: Windows, Macintosh, Linux
- browser: Internet Explorer, Mozilla.
- plug-in: Mathplayer, Techexplorer, ...

Example:

- Mozilla/Netscape will only render your document if it is XML
- Internet Explorer will only render your document if it is HTML

MathML Stylesheet

By adding the MathML Universal Stylesheet in your document you can write XHTML+MathML document without caring about what platform the document will be rendered on.

- the stylesheet is referenced from the XHTML+MathML document using a Processing instruction:

```
<?xml-stylesheet type=text/xsl  
href=style/mathml.xsl?>
```

MathML Stylesheet

By adding the MathML Universal Stylesheet in your document you can write XHTML+MathML document without caring about what platform the document will be rendered on.

- the stylesheet is referenced from the XHTML+MathML document using a Processing instruction:

```
<?xml-stylesheet type=text/xsl  
href=style/mathml.xsl?>
```
- the stylesheet sits on the server with your document

MathML Stylesheet

By adding the MathML Universal Stylesheet in your document you can write XHTML+MathML document without caring about what platform the document will be rendered on.

- the stylesheet is referenced from the XHTML+MathML document using a Processing instruction:

```
<?xml-stylesheet type=text/xsl  
href=style/mathml.xsl?>
```
- the stylesheet sits on the server with your document
- the stylesheet runs in the client to transform your document for viewing (Mozilla/Netscape and I.E. have an embedded XSLT engine)

MathML Stylesheet

The MathML universal stylesheet can

- detect what browser it is running in and output either XML or HTML accordingly
- detect what add-ons are installed and output the necessary wrappings for MathML markup
- convert content to presentation markup

MathML Stylesheet

The MathML universal stylesheet can

- detect what browser it is running in and output either XML or HTML accordingly
- detect what add-ons are installed and output the necessary wrappings for MathML markup
- convert content to presentation markup

It is available at

<http://www.w3.org/Math/XSL/Overview-tech.html>

gtkmathview

Software component for rendering MathML presentation markup (licensed under GPL).

`http://helm.cs.unibo.it/mml-widget/`

- platform independent MathML rendering engine
- binding for GTK+ (the Gimp ToolKit, `http://www.gtk.org`)
- binding for PostScript

gtkmathview

Software component for rendering MathML presentation markup (licensed under GPL).

`http://helm.cs.unibo.it/mml-widget/`

- platform independent MathML rendering engine
- binding for GTK+ (the Gimp ToolKit, `http://www.gtk.org`)
- binding for PostScript

Design and development of applications using MathML markup, in which interaction with the document can be customized.

Exercises

Exercises I

Create a Web page with the following expressions encoded in MathML presentation markup:

$$1 + x + x^2 + x^3$$

$$\frac{1 + x + x^2 + x^3}{1 + x}$$

$$(x + 1)^2$$

$$(x + 1)^2 = x^2 + 2x + 1$$

$$\sin^2 x + \cos^2 x = 1$$

Exercises II

$$\int_0^1 \frac{\sin x}{x} dx$$

$$\text{even}(x) = \begin{cases} 1, & \text{if } x \bmod 2 = 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta_k \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} \alpha + 2\beta \\ \gamma + 2\delta_k \end{pmatrix}$$