Mathematics in XML MAIDEN 2.1

George Chavchanidze

Department of Theoretical Physics, A. Razmadze Institute of Mathematics, 1 Aleksidze Street, Ge 0193 Tbilisi, Georgia

Abstract. XML MAIDEN is science oriented, XML based markup language, designed to simplify authoring, interchange and delivery of short mathematical manuscripts (articles, letters, notes). XML MAIDEN aims to capture general structure of scientific articles (front matter, headers, sections, statements, paragraphs, references etc) and basic structure of mathematical formulae in the way suitable for further formatting with Cascading Style Sheets (2.1 or later). The present document describes mathematical part of XML MAIDEN 2.1 markup. It briefly explains role of basic elements used to encode mathematical formulae in XML MAIDEN 2.1

Keywords: Mathematical Markup Language, XML, CSS

22 June 2006 (revised on 31 August 2006)

1. Introduction

First SGML based mathematical markup languages emerged in 1988-1994 years and were mainly used by scientific publishers. Due to complexity of SGML and lack of good SGML/DSSSL tools these markup languages were not really popular among individual authors. Later SGML and DSSSL were replaced with more simple and much more widespread XML and CSS, that were implemented in many browsers, CSS formatters and authoring tools. However transition from SGML/DSSSL to XML/CSS appeared to be lethal for existing mathematical DTDs. One problem came from style language. DSSSL with special set of properties intended to format mathematical formulae was replaced with more simple Cascading Style Sheets that had no math oriented extensions at all. Another problem came from meta markup. SGML that allowed users to omit redundant tags and to cut down verbosity of markup languages in this way, was replaced with more simple and strict eXtensible Markup Language that did not admit optional elements and optional end tags. To address issue W3C decided to develop new XML based mathematical markup language and to add special math module to CSS3. Unfortunately from the very beginning working group responsible for development of mathematical markup ignored technical restrictions coming from XML/CSS and produced markup which turned out to be completely unsuitable for usage in XML/CSS publishing framework. Later CSS unfriendly design of mathematical markup played crucial role in failure of CSS3 math module, work on which was nearly abandoned as working group failed to merge existing markup into XML/CSS model.

As a result need in new markup language, that would fit well in the scope of CSS2.1 visual formatting model and thus would be suitable for rendering of mathematical formulae in wide range of web browsers and CSS formatters, emerged and was partly addressed by designing XML MAIDEN markup (XML Manuscript Authoring, Interchange and Delivery Environment). Scope of XML MAIDEN is roughly aligned with the scopes of ISO 12083 and AAP mathematical DTDs, but unlike these DTDs it admits universal CSS style sheet that makes markup suitable for rendering in browsers and formatters that support CSS2.1 or later. Universal CSS2.1 style sheets can handle arbitrary complex math formulae obtained by combining and nesting of subscripts, superscripts, prescripts, under and over scripts, fractions, operators, matrices, vectors, determinants, cases, fences, radicals and other mathematical expressions. Apart of XML/CSS framework which XML MAIDEN markup was designed to be used in, one can use XML MAIDEN in less widespread XML/XSL and XML/DSSSL publishing models.

2. Content Model

In XML MAIDEN most of the mathematical elements with mixed content model may carry inline mathematical content that may be sequence of character data and one of the following elements: apply, bold, cases, det, fence, fenced, float, fraction, group, inf, italic, matrix, ope, opgroup, over, overline, radical, sqrt, strike, sub, sup, sur, under, underline, vector. In addition to inline mathematical content formula element may contain line breaks (elements line and wrap) and subformula elements. Content model of each element is specified below. The following entities are used in DTD:

```
Relevant DTD Fragment:
<!ENTITY % innermath "(#PCDATA | apply | bold | cases | det | fence | fenced | float |
fraction | group | inf | italic | matrix | ope | opgroup | over | overline |
radical | sqrt | strike | sub | sup | sur | under | underline | vector)*">
<!ENTITY % blockmath "(#PCDATA | apply | bold | cases | det | fence | fenced | float |
fraction | group | inf | italic | line | matrix | ope | opgroup | over | overline |
radical | sqrt | strike | sub | subformula | sup | sur | under | underline | vector | wrap)*">
```

3. Mathematical Formulae

XML MAIDEN 2.1 reserves four elements used to mark inline-equations, displayed equations, equation arrays and subformulae. These are math, formula, formulae and subformula elements. In addition there is general purpose inline container used to group mathematical expressions.

Definition of math element: math element is primary container that encloses inline mathematical expressions (those formulae that are part of normal text flow). Aural equivalent of mathematical expression may be specified via pronounce attribute. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ATTLIST math pronounce CDATA #IMPLIED>
<!ELEMENT math %innermath;>
```

Markup example:

```
Dirac's construction provides an explicit local expression for the traverse Poisson
structure of a Poisson manifold <math>(M, W)</math> at any of its points.
```

Formatted example:

Dirac's construction provides an explicit local expression for the traverse Poisson structure of a Poisson manifold (M, W) at any of its points

Definition of formula element: formula element is basic block level container that marks displayed equations (block level formulae). Large formulae may be broken into several block level parts using subformula element. Multiple related formulae may be grouped using formulae element. Aural equivalent of mathematical expression may be specified via pronounce attribute. In addition to inline mathematical content element may contain may contain line breaks (elements line and wrap) and subformula elements.

```
Relevant DTD Fragment:
<!ATTLIST formula pronounce CDATA #IMPLIED>
<!ELEMENT formula %blockmath;>
```

Markup example:

```
Any function <math>F</math> commuting with Hamiltonian
<formula>{F, H} = 0</formula> is first integral of Hamiltonian system.
```

Formatted example:

Any function F commuting with Hamiltonian

$$\{F, H\} = 0 \qquad (1)$$

is first integral of Hamiltonian system.

Definition of formulae element: formulae element is used to group several related formulae (for example equations in system of equations). Element may contain two or more formula elements.

```
Relevant DTD Fragment:
<!ELEMENT formulae (formula, formula+)>
```

Markup example:

```
Being linear operation,
Poisson bracket satisfied properties
<formulae>
<formula>{F, G + H} = {F, G} + {F, H}</formula>
<formula>{F, GH} = {F, G}H + {F, H}G</formula>
</formulae>
```

Formatted example:

Being linear operation, Poisson bracket satisfied properties

$$\{F, G + H\} = \{F, G\} + \{F, H\} \quad (2)$$

$$\{F, GH\} = \{F, G\}H + \{F, H\}G \quad (3)$$

Definition of subformula element: subformula element may be used to break large formulae into several block level parts. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT subformula %innermath;>
```

Markup example:

```
<formula>
<subformula>{F, (G + H)S} = {F, GS} + {F, HS}</subformula>
<subformula>= {F, G}S + {F, H}S + (G + H){F, S}</subformula>
</formula>
```

Formatted example:

$$\{F, (G + H)S\} = \{F, GS\} + \{F, HS\} \quad (4)$$
$$= \{F, G\}S + \{F, H\}S + (G + H)\{F, S\}$$

Definition of group element: group element is general purpose inline container that may be used to group expressions. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT group %innermath;>
```

4. Subscripts and superscripts

In XML MAIDEN 2.1 the elements sup and sub are used to mark subscripts and superscripts respectively. They may be nested and combined to produce complex indices. Note that indices produced via sub and sup elements are applied to preceding Unicode character (if any). Otherwise base is undefined, in particular note that sup and sub elements do not apply to fences (element fence), matrices (elements matrix, determinant and vector) and operators (element ope). In XML MAIDEN 2.1 matrix, fence indices (markers) and operator limits are considered to be intrinsic part of matrix, fence and operator constructions, extra elements are reserved for this purpose (operator, fence, matrix and vector markers).

When positioning indices (elements sup, sub), XML MAIDEN 2.1 implementations may base vertical-alignment offsets on font-size value, like it is done in case of XHTML subscripts, superscripts (elements sup, sub) and CSS vertical-align:super, sub property values, while markers (elements marker, submarker) should be positioned with respect to base that they are applied to.

Markup example:

```
<formula>ax<sup>2</sup> + bx + c = 0</formula>
<formula>u<sub>t</sub> = u<sub>xxx</sub> + uu<sub>x</sub></formula>
```

Formatted example:

$$ax^2 + bx + c = 0 \quad (5)$$

$$u_t = u_{xxx} + uu_x \quad (6)$$

Indices may be nested, combined and may carry complex mathematical expressions like fractions, operators etc. When subscript and superscript are combined they may be formatted in three different ways.

1. Subscript may precede superscript
2. Subscript may succeed superscript
3. Subscript and superscript may be stacked (floated to common base)

The difference between these three cases is not purely presentational and in certain cases position of indices may change meaning of mathematical expression therefore XML MAIDEN 2.1 distinguishes these cases as follows:

1. A_X^Y is treated as:

$$A_X^Y \tag{7}$$

2. A^Y_X is handled as:

$$A^Y_X \tag{8}$$

3. A<float>^Y_X</float> is formatted like:

$$A^Y_X \tag{9}$$

Note that float element is used to stack indices.

Definition of sub element: sub element is used to mark subscripts in XML MAIDEN. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT sub %innermath;>
```

Definition of sup element: sup element is used to mark superscripts in XML MAIDEN. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT sup %innermath;>
```

Definition of float element: float element is used to group stacked indices. Element may have one of the following content models:

1. sup element followed by sub element
2. sur element followed by inf element

```
Relevant DTD Fragment:
<!ELEMENT float ((sup,sub)|(sur,inf))>
```

Definition of submarker element: submarker element is used to mark subscripts that are associated with operators (lower limits), fences (fence markers), matrices (matrix markers) or vectors (vector markers). Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT submarker %innermath;>
```

Definition of marker element: marker element is used to mark superscripts that are associated with operators (upper limits), fences (fence markers), matrices (matrix markers) or vectors (vector markers). Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT marker %innermath;>
```

Definition of markers element: markers element is used to group markers. Element may contain marker element followed by submarker element

```
Relevant DTD Fragment:
<!ELEMENT markers (marker, submarker)>
```

5. Prescripts

The elements inf (inferior) and sur (superior) are similar to sub and sup but are used to mark prescripts (sub and superscripts that precede their base). They may be nested and combined to produce complex prescripts. Note that prescripts produced via inf and sur elements are applied to succeeding Unicode character (if any). Otherwise base is undefined.

Markup example:

```
<formula>
<inf>13</inf>Al<sup>27</sup> + <inf>2</inf>He<sup>4</sup>
= <inf>15</inf>P<sup>30</sup> + n
</formula>
```

Formatted example:

$$ {}_{13}\mathrm{Al}^{27} + {}_{2}\mathrm{He}^{4} = {}_{15}\mathrm{P}^{30} + \mathrm{n} \qquad (10) $$

When inferior and superior are combined they may be formatted in three different ways. XML MAIDEN 2.1 treats combined prescripts as follows:

1. <sur>Y</sur><inf>X</inf>A is treated as:

$$ {}^{Y}_{X}\mathrm{A} \qquad (11) $$

2. <inf>X</inf><sur>Y</sur>A is handled as:

$$ {}_{X}^{Y}\mathrm{A} \qquad (12) $$

3. <float><sur>Y</sur><inf>X</inf></float>A is formatted like:

$$ {}^{Y}_{X}\mathrm{A} \qquad (13) $$

float element is used to stack indices.

Definition of inf element: inf element is used to mark subscripts that precede their base. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT inf %innermath;>
```

Definition of sur element: sur element is used to mark superscripts that precede their base. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT sur %innermath;>
```

6. Fractions

Markup for fractions resembles that used in ISO 12083.

Markup example:

```
<formula>
<fraction>
<num>tanh(x) + tanh(y)</num>
<den>1 + tanh(x)tanh(y)</den>
</fraction>
</formula>
```

Formatted example:

$$ \frac{\tanh(x) + \tanh(y)}{1 + \tanh(x)\tanh(y)} \qquad (14) $$

Definition of fraction element: fraction element is used to mark fractions. Element must contain num element followed by den element.

```
Relevant DTD Fragment:
<!ELEMENT fraction (num, den)>
```

Definition of num element: num element is used to mark numerator of fraction. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT num %innermath;>
```

Definition of num element: den element is used to mark denominator of fraction. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT den %innermath;>
```

7. Under scripts and over scripts

XML MAIDEN 2.1 uses element ker to mark base (kernel) of over or under scripted expression. Element sat is used for under and over scripts (satellites), while over and under elements are used to compose over or under scripted expressions by combining arbitrary number of over script or under scripts with base. Note that over expression is not expected to produce diacritical marks/accents. See accents section below.

Markup example:

```
<formula>
<over>
<sat>over script</sat>
<ker>BASE</ker>
</over>

<under>
<over>
<sat>over script</sat>
<ker>BASE</ker>
</over>
<sat>under script</sat>
</under>

<under>
<ker>BASE</ker>
<sat>under script</sat>
</under>

<under>
<over>
<sat>over script</sat>
<ker>BASE</ker>
</over>
<sat>under script-1</sat>
<sat>under script-2</sat>
</under>
</formula>
```

Formatted example:

$$\overset{\text{over script}}{\underset{\text{under script}}{\text{BASE}}} \quad \overset{\text{over script}}{\underset{\text{under script}}{\text{BASE}}} \quad \underset{\text{under script}}{\text{BASE}} \quad \overset{\text{over script}}{\underset{\substack{\text{under script-1}\\ \text{under script-2}}}{\text{BASE}}} \tag{15}$$

Definition of over element: over element is used to mark expression with over scripts and/or over braces. Element may have one of the following content models:

1. one or more sat elements followed by optional overbrace element and followed by one of the elements from the following list: ker, ope, under
2. overbrace element followed by zero or more sat elements and followed by one of the elements from the following list: ker, ope, under

```
Relevant DTD Fragment:
<!ELEMENT over (((overbrace, sat*)|(sat+, overbrace?)),(ker|ope|under))>
```

Definition of under element: under elements are used to mark expression with under scripts and/or under braces. Element may have one of the following content models:

1. One of the elements from the following list: ker, ope, over, followed by underbrace and followed by zero or more sat elements
2. One of the elements from the following list: ker, ope, over, followed by one or more sat elements and followed by optional underbrace element

```
Relevant DTD Fragment:
<!ELEMENT under ((ker|ope|over),((underbrace, sat*)|(sat+, underbrace?)))>
```

Definition of ker element: ker element is used to mark base of under and/or over scripted expression. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT ker %innermath;>
```

Definition of sat element: sat element is used to mark under and over scripts. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT sat %innermath;>
```

Under and over scripted expressions may involve under and over braces that outline expression which scripts apply to. Such expressions are produced using empty elements underbrace and overbrace

Definition of overbrace element: overbrace element represents over braces. Element is empty.

```
Relevant DTD Fragment:
<!ELEMENT overbrace EMPTY>
```

Definition of underbrace element: underbrace element represents under braces. Element is empty.

```
Relevant DTD Fragment:
<!ELEMENT underbrace EMPTY>
```

Markup example:

```
<formula>
<over>
<sat>k times</sat>
<overbrace/>
<ker>(m, m, ... , m)</ker>
</over>

<under>
<ker>(m, m, ... , m)</ker>
<underbrace/>
<sat>k times</sat>
</under>
</formula>
```

Formatted example:

$$\overbrace{(m, m, \dots, m)}^{k \text{ times}} \underbrace{(m, m, \dots, m)}_{k \text{ times}} \tag{16}$$

8. Operators

Element ope marks mathematical operators such as sum, product, unification, intersection, integral etc. In case if operator is accompanied by over script and/or under scripts one may use over and under elements to associate over/under scripts with operator

Markup example:

```
<formula>
A =
<under>
```

```
<over>
<sat>n</sat>
<ope>U</ope>
</over>
<sat>s = 1</sat>
</under>
A<sup>(s)</sup>
</formula>

<formula>
A =
<under>
<ope>U</ope>
<sat>s</sat>
</under>
A<sup>(s)</sup>
</formula>
```

Formatted example:

$$A = \bigcup_{s=1}^{n} A^{(s)} \tag{17}$$

$$A = \bigcup_{s} A^{(s)} \tag{18}$$

Instead of under and over scripts one may associate markers that are positioned near top and bottom edges of operator's core. Element 'opgroup' is reserved for this purpose

Markup example:

```
<formula>
A =
<opgroup>
<ope>U</ope>
<markers>
<marker>n</marker>
<submarker>s = 1</submarker>
</markers>
</opgroup>
A<sup>(s)</sup>
</formula>

<formula>
A =
<opgroup>
<ope>U</ope>
<submarker>s</submarker>
</opgroup>
A<sup>(s)</sup>
</formula>
```

Formatted example:

$$A = \bigcup_{s=1}^{n} A^{(s)} \tag{19}$$

$$A = \bigcup_{s} A^{(s)} \tag{20}$$

Definition of opgroup element: opgroup element is used to associate markers (limits) with operators. Element may have one of the following content models:

1. ope element followed by marker element
2. ope element followed by submarker element
3. ope element followed by markers element

```
Relevant DTD Fragment:
<!ELEMENT opgroup (ope, (marker | markers | submarker))>
```

Definition of ope element: ope element is used to mark resizable operators like integrals and N-ary mathematical operators. Element may contain one or more characters representing mathematical operator.

```
Relevant DTD Fragment:
<!ELEMENT ope (#PCDATA)>
```

9. Matrices, vectors and determinants

To represent matrices, vectors and determinants, XML MAIDEN 2.1 uses elements matrix, vector, det (determinant), row, cell and entry. Matrices and determinants are structured row by row, they should contain at least two rows and should be at least two column wide

Markup example:

```
<formula>
M =
<matrix>
<row><cell>A</cell><cell>B</cell></row>
<row><cell>C</cell><cell>D</cell></row>
</matrix>
</formula>
```

Formatted example:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{21}$$

In determinants number of row must be equal to number of columns.

Markup example:

```
<formula>
det(M) =
<det>
<row><cell>A</cell><cell>B</cell></row>
<row><cell>C</cell><cell>D</cell></row>
</det>
= AD - BC
</formula>
```

Formatted example:

$$\det(M) = \begin{vmatrix} A & B \\ C & D \end{vmatrix} = AD - BC \tag{22}$$

Vectors should contain at least two entries.

Markup example:

```
<formula>
V =
<vector>
<entry>X</entry>
<entry>Y</entry>
</vector>
</formula>
```

Formatted example:

$$V = \begin{bmatrix} X \\ Y \end{bmatrix} \tag{23}$$

Element apply may be used to associate either marker or submarker with matrix or vector. Matrix (or vector) marker (submarker) is placed near top (bottom) of right fence that encloses matrix (vector).

Markup example:

```
<formula>
M<sup>T</sup> =
<apply>
<matrix>
<row><cell>A</cell><cell>B</cell></row>
<row><cell>C</cell><cell>D</cell></row>
</matrix>
<marker>T</marker>
</apply>
</formula>
```

Formatted example:

$$M^{T} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{T} \tag{24}$$

Definition of matrix element: matrix element represents matrices. Element should contain two or more row elements.

```
Relevant DTD Fragment:
<!ELEMENT matrix (row, row+)>
```

Definition of det element: det element represents determinants. Element should contain two or more row elements.

```
Relevant DTD Fragment:
<!ELEMENT det (row, row+)>
```

Definition of vector element: vectors element represents vectors. Element should contain two or more entry elements.

```
Relevant DTD Fragment:
<!ELEMENT vector (entry, entry+)>
```

Definition of row element: row element marks row in matrices and determinants. Element should contain two or more cell elements.

```
Relevant DTD Fragment:
<!ELEMENT row (cell, cell+)>
```

Definition of cell element: cell element marks individual cells (entries) in matrices and determinants. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT cell %innermath;>
```

Definition of entry element: entry element marks individual entries (components) in vectors. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT entry %innermath;>
```

Definition of apply element: apply element is used to associate markers with matrices and vectors. Element may have one of the following content models:

1. matrix element followed by marker element
2. matrix element followed by submarker element
3. vector element followed by marker element
4. vector element followed by submarker element

```
Relevant DTD Fragment:
<!ELEMENT apply ((matrix | vector),(marker | submarker))>
```

10. Cases

In case when expression has different values under different conditions, one may use cases construction. For each case value element specifies value of expression, while element scope specifies condition under which the value is valid.

Markup example:

```
<formula>
F(x) =
<cases>
<case><value>x<sup>3</sup></value><scope>if x > 0</scope></case>
<case><value>x<sup>2</sup></value><scope>otherwise</scope></case>
</cases>
</formula>
```

Formatted example:

$$F(x) = \begin{cases} x^3 & \text{if } x > 0 \\ x^2 & \text{otherwise} \end{cases} \qquad (25)$$

Definition of cases element: cases element marks set of possible values of certain expression. Element should contain two or more case elements.

```
Relevant DTD Fragment:
<!ELEMENT cases (case, case+)>
```

Definition of case element: case element marks individual values from the set of possible values of expression. Element should contain value element followed by scope element.

```
Relevant DTD Fragment:
<!ELEMENT case (value, scope)>
```

Definition of value element: value element marks one of the values of expression, value is valid in domain specified by following sibling scope element. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT value %innermath;>
```

Definition of scope element: scope element marks domain in which, value specified by preceding sibling value element is valid. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT scope %innermath;>
```

11. Fences

Element fence is used to produce large brackets and fences. Type of fence is specified via attributes left and right. Fence markers are produced via marker, markers and submarker elements. In XML MAIDEN fence marker is expression aligned near the top (or bottom) of right fence. To attach markers to base fence fenced element is used. The marker element carries top-right marker.

Markup example:

```
<formula>
<fenced>
<fence>
EXPRESSION IN SQUARE BRACKETS
</fence>
<marker>TOP MARKER</marker>
</fenced>
</formula>
```

Formatted example:

$$\left[\text{EXPRESSION IN SQUARE BRACKETS}\right]^{\text{TOP MARKER}} \qquad (26)$$

The submarker produces bottom-right marker.

Markup example:

```
<formula>
<fenced>
<fence left="double" right="double">
EXPRESSION IN DOUBLE BARS
</fence>
```

```
<submarker>BOTTOM MARKER</submarker>
</fenced>
</formula>
```

Formatted example:

$$\left\| \text{EXPRESSION IN DOUBLE BARS} \right\|_{\text{BOTTOM MARKER}} \tag{27}$$

The markers element produces right markers.

Markup example:

```
<formula>
<fenced>
<fence left="none" right="solid">
EXPRESSION WITH RIGHT RULER
</fence>
<markers>
<marker>TOP MARKER</marker>
<submarker>BOTTOM MARKER</submarker>
</markers>
</fenced>
</formula>
```

Formatted example:

$$\text{EXPRESSION WITH RIGHT RULER} \Big|^{\text{TOP MARKER}}_{\text{BOTTOM MARKER}} \tag{28}$$

Definition of fence element: fence element encloses its content in large brackets or fences. Element has attributes left and right. Attribute left may have values none, curly, dashed, double round, solid and square with square being default value. Attribute right may have values curly, dashed, double round, solid and square where square is default value. Value none indicates that fence should be omitted, curly produces large brace {curly bracket}, round marks large parenthesis (round bracket), square is used for large bracket [square bracket], dashed produces large dashed bar, solid is used for large single bar fence like |absolute value| and double for large double bar fence like ||norm|| Current list of fence types is not comprehensive and may be extended in future versions of XML MAIDEN. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ATTLIST fence left (none | curly | dashed | double | round | solid | square) "square"
right (curly | dashed | double | round | solid | square) "square">
<!ELEMENT fence %innermath;>
```

Definition of fenced element: fenced element is used to associate markers with fences. Element may have one of the following content models:

1. fence element followed by marker element
2. fence element followed by submarker element
3. fence element followed by markers element

```
Relevant DTD Fragment:
<!ELEMENT fenced (fence, (marker | markers | submarker))>
```

12. Radicals

Markup for radicals resembles that used in ISO 12083.

Markup example:

```
<formula>
<radical><radix>3</radix><radicand>x<sup>3</sup> + y<sup>3</sup> + z<sup>3</sup></radicand></radical>
</formula>
```

Formatted example:

$$\sqrt[3]{x^3 + y^3 + z^3} \tag{29}$$

For square roots there is separate element.

Markup example:

```
<formula>
<sqrt>b<sup>2</sup> - 4ac</sqrt>
</formula>
```

Formatted example:

$$\sqrt{b^2 - 4ac} \qquad (30)$$

Definition of radical element: radical element marks radicals. Element must contain radix element followed by radicand element.

```
Relevant DTD Fragment:
<!ELEMENT radical (radix, radicand)>
```

Definition of radix element: radix element is used to mark radical index and is placed before the radical sign. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT radix %innermath;>
```

Definition of radicand element: radicand element is used to mark radical content and is placed under the radical sign. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT radicand %innermath;>
```

Definition of sqrt element: sqrt element is used to mark square roots. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT sqrt %innermath;>
```

13. Bold and Italic

Elements bold and italic are used to produce bold and italic expressions. In mathematical formulae bold and italic are sometimes applied to individual glyphs that may denote some variables, vectors etc. Note however that in such cases Unicode consortium recommends to use bold, bold italic and italic math alphanumeric symbols located in Unicode Plane 1:

1. Latin bold math characters (𝐀-𝐳)
2. Latin bold italic math characters (𝑨-𝒛)
3. Latin italic math characters (𝐴-𝑧)
4. Greek bold math characters (𝚨-𝛡)
5. Greek bold italic math characters (𝜜-𝝕)
6. Greek italic math characters (𝛢-𝜛)
7. Bold face numbers (𝟎-𝟗)

Set of math alphanumeric symbols located in Unicode Plane 1 is specially shaped and kerned for usage in mathematical formulae, as letters that represent mathematical variables are slightly slanted and look smoother then ordinary Latin/Greek glyphs, while kerning of characters used in mathematical expressions differs from kerning in Latin/Greek text.

Definition of bold element: bold element produces bold text. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT bold %innermath;>
```

Definition of italic element: italic element produces italic text. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT italic %innermath;>
```

14. Under, over lines and strike

Elements overline and underline are reserved for over and underlined expressions. In case if one needs to apply under or over bars to individual glyphs it is better to use precomposed Unicode characters or combining diacritical marks.

Definition of underline element: underline element produces underlined math expression. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT underline %innermath;>
```

Definition of overline element: overline element produces overlined math expression. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT overline %innermath;>
```

Element 'strike' generates line through. In the same time it may be used to mark cancelled or deleted terms. Note that element does not produce new glyphs like h-bar (appropriate Unicode characters should be used instead).

Definition of strike element: strike element produces math expression with line through. Element carries inline mathematical content.

```
Relevant DTD Fragment:
<!ELEMENT strike %innermath;>
```

15. Line breaks

Automated line breaks inside mathematical expressions are usually prohibited. However line breaks can be specified manually using empty element line that generates line break, or wrap element that marks place (wrap point) where line breaks may be generated by browser (depending on page width browser may decide whether to use this opportunity).

Markup example:

```
<formula>
{F, (G + H)S} = {F, GS} + {F, HS}<wrap/>
= {F, G}S + {F, H}S + (G + H){F, S}
</formula>
```

Formatted example:

Depending on available width example may be formatted as

$$\{F, (G + H)S\} = \{F, GS\} + \{F, HS\} = \{F, G\}S + \{F, H\}S + (G + H)\{F, S\} \quad (31)$$

or

$$\{F, (G + H)S\} = \{F, GS\} + \{F, HS\} \quad (32)$$
$$= \{F, G\}S + \{F, H\}S + (G + H)\{F, S\}$$

Definition of line element: line element produces line feed. Element is empty.

```
Relevant DTD Fragment:
<!ELEMENT line EMPTY>
```

Definition of wrap element: wrap element marks places (wrap points) where line breaks are allowed. Element is empty.

```
Relevant DTD Fragment:
<!ELEMENT wrap EMPTY>
```

16. Core attributes

Core attributes are shared by most of XML MAIDEN 2.1 elements. There are three core attributes: xml:id, token and role.

Definition of xml:id attribute: xml:id is ID type attribute used to assign unique identifier to element. In XML MAIDEN 2.1 xml:id attribute may be applied to any element excluding empty elements line and wrap.

Definition of token attribute: token is general purpose attribute, it can be used to enrich document structure by grouping/dividing elements into classes. Its value is of NMTOKEN type. Attribute may be applied to any XML MAIDEN element.

Definition of role attribute: role attribute is used to enrich document structure with semantics, and may specify semantical role of the given element. In XML MAIDEN 2.1 role attribute may be applied to any element excluding empty elements line and wrap.

17. Accents

XML MAIDEN 2.1 provides no markup for diacritical marks (accents like over dots, hats, tildes, caps) as rendering of diacritical marks is governed by Unicode standard (via combining diacritical marks and precomposed glyphs). Unicode based solution is preferable as it is more universal (may be used in many XML based markup languages and even plain Unicode text), does not put extra burden on style sheets, admits more accurate positioning of accents based on font metrics and allows multiple diacritical marks per character. In the same time, note that Unicode standard in its current form does not provide mechanism for stretching diacritical marks to enclose several characters (there are some for pairs, however). See also document Unicode Support for Mathematics issued by Unicode consortium

18. XML MAIDEN 2.1 Math DTD

XML MAIDEN 2.1 Math DTD specifies allowed nesting of XML MAIDEN 2.1 math oriented elements, allowed attributes and attribute value types. DTD may be used for validation or error tracking purposes. It is not required to attach DTD to each XML MAIDEN document explicitly. In absence of DTD, XML MAIDEN 2.1 markup may be identified by namespace.

```
Relevant DTD Fragment:
<!ENTITY % attr "role CDATA #IMPLIED token NMTOKEN #IMPLIED xml:id ID #IMPLIED">

<!ATTLIST apply %attr;>
<!ATTLIST bold %attr;>
<!ATTLIST case %attr;>
<!ATTLIST cases %attr;>
<!ATTLIST cell %attr;>
<!ATTLIST den %attr;>
<!ATTLIST det %attr;>
<!ATTLIST entry %attr;>
<!ATTLIST fence %attr; left (none | curly | dashed | double | round | solid | square) "square"
right (curly | dashed | double | round | solid | square) "square">
<!ATTLIST fenced %attr;>
<!ATTLIST float %attr;>
<!ATTLIST formula %attr; pronounce CDATA #IMPLIED>
<!ATTLIST formulae %attr;>
<!ATTLIST fraction %attr;>
<!ATTLIST group %attr;>
<!ATTLIST inf %attr;>
<!ATTLIST italic %attr;>
<!ATTLIST ker %attr;>
<!ATTLIST line token NMTOKEN #IMPLIED>
<!ATTLIST marker %attr;>
<!ATTLIST markers %attr;>
<!ATTLIST math %attr; pronounce CDATA #IMPLIED>
<!ATTLIST matrix %attr;>
<!ATTLIST num %attr;>
<!ATTLIST ope %attr;>
<!ATTLIST opgroup %attr;>
<!ATTLIST over %attr;>
```

```
<!ATTLIST overbrace %attr;>
<!ATTLIST overline %attr;>
<!ATTLIST row %attr;>
<!ATTLIST radical %attr;>
<!ATTLIST radicand %attr;>
<!ATTLIST radix %attr;>
<!ATTLIST sat %attr;>
<!ATTLIST scope %attr;>
<!ATTLIST sqrt %attr;>
<!ATTLIST strike %attr;>
<!ATTLIST sub %attr;>
<!ATTLIST subformula %attr;>
<!ATTLIST submarker %attr;>
<!ATTLIST sup %attr;>
<!ATTLIST sur %attr;>
<!ATTLIST under %attr;>
<!ATTLIST underbrace %attr;>
<!ATTLIST underline %attr;>
<!ATTLIST value %attr;>
<!ATTLIST vector %attr;>
<!ATTLIST wrap token NMTOKEN #IMPLIED>

<!ENTITY % innermath "(#PCDATA | apply | bold | cases | det | fence | fenced | float |
fraction | group | inf | italic | matrix | ope | opgroup | over | overline |
radical | sqrt | strike | sub | sup | sur | under | underline | vector)*">
<!ENTITY % blockmath "(#PCDATA | apply | bold | cases | det | fence | fenced | float |
fraction | group | inf | italic | line | matrix | ope | opgroup | over | overline |
radical | sqrt | strike | sub | subformula | sup | sur | under | underline | vector | wrap)*">

<!ELEMENT apply ((matrix | vector),(marker | submarker))>
<!ELEMENT bold %innermath;>
<!ELEMENT case (value, scope)>
<!ELEMENT cases (case, case+)>
<!ELEMENT cell %innermath;>
<!ELEMENT den %innermath;>
<!ELEMENT det (row, row+)>
<!ELEMENT entry %innermath;>
<!ELEMENT fence %innermath;>
<!ELEMENT fenced (fence, (marker | markers | submarker))>
<!ELEMENT float ((sup, sub) | (sur, inf))>
<!ELEMENT formula %blockmath;>
<!ELEMENT formulae (formula, formula+)>
<!ELEMENT fraction (num, den)>
<!ELEMENT group %innermath;>
<!ELEMENT inf %innermath;>
<!ELEMENT italic %innermath;>
<!ELEMENT ker %innermath;>
<!ELEMENT line EMPTY>
<!ELEMENT marker %innermath;>
<!ELEMENT markers (marker, submarker)>
<!ELEMENT math %innermath;>
<!ELEMENT matrix (row, row+)>
<!ELEMENT num %innermath;>
<!ELEMENT ope (#PCDATA)>
<!ELEMENT opgroup (ope, (marker | markers | submarker))>
<!ELEMENT over (((overbrace, sat*)|(sat+, overbrace?)),(ker|ope|under))>
<!ELEMENT overbrace EMPTY>
<!ELEMENT overline %innermath;>
<!ELEMENT row (cell, cell+)>
<!ELEMENT radical (radix, radicand)>
<!ELEMENT radicand %innermath;>
<!ELEMENT radix %innermath;>
<!ELEMENT sat %innermath;>
<!ELEMENT scope %innermath;>
<!ELEMENT sqrt %innermath;>
<!ELEMENT strike %innermath;>
```

```
<!ELEMENT sub %innermath;>
<!ELEMENT subformula %innermath;>
<!ELEMENT submarker %innermath;>
<!ELEMENT sup %innermath;>
<!ELEMENT sur %innermath;>
<!ELEMENT under ((ker|ope|over),((underbrace, sat*)|(sat+, underbrace?)))>
<!ELEMENT underbrace EMPTY>
<!ELEMENT underline %innermath;>
<!ELEMENT value %innermath;>
<!ELEMENT vector (entry, entry+)>
<!ELEMENT wrap EMPTY>
```

19. XML MAIDEN namespace

XML MAIDEN 2.0 and 2.1 math elements belong to http://xml-maiden.com namespace. Most of XML MAIDEN math attributes are not bound to namespaces, exception is xml:id attribute which is bound to http://www.w3.org/XML/1998/

20. User Agent Conformance

XML MAIDEN does not impose any specific UA conformance requirements apart of those defined by host environment (like XML/CSS, XML/XSL, XML/DSSSL).

21. Document Conformance

For basic conformance of math formulae to XML MAIDEN 2.1 Math DTD, XML MAIDEN elements should be bound to http://xml-maiden.com namespace, while each of normalized XML subtrees, obtained from subtrees spanned by XML MAIDEN math, formula or formulae elements, by stripping element namespace prefixes and corresponding namespace declarations (if any), should validate against XML MAIDEN 2.1 Math DTD. Attaching DTD to the XML document explicitly is not required. Note that formatting of XML MAIDEN documents is governed by style sheets, therefore associating appropriate style sheet(s) with XML document is crucial for proper rendering of math formulae.

References

[1] B. Beeton et al., *Unicode support for mathematics*, 2003
[2] B. Bos et al., *Cascading Style Sheets, level 2 revision 1*, 2006
[3] T. Bray et al., *Extensible Markup Language (XML) 1.0*, 2004
[4] T. Bray et al., *Namespaces in XML*, 1999
[5] J. Clark, *Associating Style Sheets with XML documents*, 1999
[6] J. Marsh et al., *xml:id Version 1.0*, 2005