22 Tables, Formulae, and Graphics

Many documents, both historical and contemporary, include not only text but also graphics, artwork, and other images. Although such images could be represented directly with markup, SGML and XML are not primarily designed for that purpose, and it is standard practice to include such information by declaring each image as an external entity encoded in a suitable graphical notation, and then referring to it from within the document.

In addition to graphic images, documents often contain material presented in graphical or tabular format. In such materials, details of layout and presentation may also be of comparatively greater significance or complexity than they are for running text. Indeed, it may often be difficult to make a clear distinction between details relating purely to the rendition of information and those relating to the information itself.

Finally, documents may contain mathematical formulæ or expressions in other formulaic notations, for which no notation is defined in these Guidelines.

These areas (graphics, tabular material, and mathematical or other formulæ) have in common that they have received considerable attention from many other standards bodies or similar professional groups. In part because of this, they may frequently be most conveniently encoded and processed using some notation not defined by these Guidelines. For these reasons, and others, we consider tables, formulæ, and graphics together in this chapter.

As with text markup in general, many incompatible formats have been proposed for the representation of graphics, formulae and tables in electronic form. Unfortunately, no single format as effective as SGML or XML in the domain of text has yet emerged for their interchange, to some extent because of the difficulty of representing the information these data formats convey independently of the way it is rendered.

The additional tag set defined by this chapter defines special purpose ‘container’ elements that can be used to encapsulate occurrences of such data within a TEI-conformant document in a portable way. Specific recommendations for the encoding of tables are provided in section 22.1 *Tables* and recommendations for mathematical or other formulæ in section 22.2 *Formulae and Mathematical Expressions*. Specific recommendations for the encoding of graphic figures may be found in section 22.3 *Specific Elements for Graphic Images*. The rest of the chapter is devoted to general problems of encoding graphic information.

There is at the time of writing no consensus on formats for graphical images, and such formats vary in many ways. We therefore provide (in section 22.4 *Overview of Basic Graphics Concepts*) a brief discussion of the ways in which images may be represented, and (in section 22.5 *Graphic Image Formats*) a list of formal names for those representations most popular at this time. Each one includes a very brief description and (where known) a reference to the formal specification of the notation. These Guidelines recommend a few particular representations as being the most widely supported and understood.

To enable the additional tag set defined by this chapter, the parameter entity *TEI.figures* must be defined with the value `INCLUDE`, as shown in the example below:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN" "tei2.dtd" [
    <!ENTITY % TEI.XML      'INCLUDE' >
    <!ENTITY % TEI.verse    'INCLUDE' >
    <!ENTITY % TEI.figures  'INCLUDE' >
 ]>
```

With this declaration in effect, the TEI elements and attributes described in the following sections are all available. If any of the specialized notations described in sections 22.2 *Formulae and Mathematical Expressions* and 22.3 *Specific Elements for Graphic Images* are used, then an additional *notation declaration* must also be included in the document type declaration subset, as further illustrated below.

The overall structure of the tag set defined in this chapter is as follows:

```
<!-- 22.: Tables, Formulae, Figures-->
<!--declarations from 22.1.1: Tables inserted here -->
<!--declarations from 22.2: Formulae inserted here -->
<!--declarations from 22.3: Figures inserted here -->
<!-- end of 22.-->
```

22.1 Tables

A table is the least ‘graphic’ of the elements discussed in this chapter. Almost any text structure can be presented as a series of rows and columns: one might, for example, choose to show a glossary or other form of list in tabular form, without necessarily regarding it as a table. In such cases, the global rend attribute is an appropriate way of indicating that some element is being presented in tabular format. When tabular presentation is regarded as of less intrinsic importance, it is correspondingly simpler to encode descriptive or functional information about the contents of the table, for example to identify one column as containing names and another as containing dates, though the two methods may be combined.

When, however, particular elements are required to encode the tabular arrangement itself, then one or other of the various ‘table DTDs’ now available may be preferable. The table DTDs in common use generally view a table as a special text element, made up of row elements (or, sometimes, column elements), themselves composed of cells. Table cells generally appear in row-major order, with the first row from left to right, then the second row, and so on. Details of appearance such as column widths, border lines, and alignment are generally encoded by numerous attributes. Beyond this, however, such DTDs differ greatly. This section begins by describing a table DTD of this kind; a brief summary of some other widely available table DTDs is also provided in section 22.1.2 *Other Table DTDs*.

22.1.1 The TEI Table DTD

For encoding tables of low to moderate complexity, these Guidelines provide the following special purpose elements:

<table> contains text displayed in tabular form, in rows and columns. Attributes include:
- **rows** indicates the number of rows in the table.
  - *Values* If no number is supplied, an application must calculate the number of rows.
- **cols** indicates the number of columns in each row of the table.
  - *Values* If no number is supplied, an application must calculate the number of columns.

<row> contains one row of a table. Attributes include:
- **role** indicates the kind of information held in the cells of this row.
  - *Suggested values include:*
    - `label` labelling or descriptive information only.
    - `data` data values.

<cell> contains one cell of a table. Attributes include:
- **role** indicates the kind of information held in the cell.
  - *Suggested values include:*
    - `label` labelling or descriptive information only.
    - `data` data values.
- **cols** indicates the number of columns occupied by this cell.
  - *Values* A number; a value greater than one indicates that this cell spans several columns.
- **rows** indicates the number of rows occupied by this cell.
  - *Values* A number; a value greater than one indicates that this cell spans several rows.

The `<table>` element is defined as a member of the class *inter*; it may therefore appear both within other components (such as paragraphs), or between them, provided that the additional tag set defined in this chapter has been enabled, as described at the beginning of this chapter.

It is to a large extent arbitrary whether a table should be regarded as a series of rows or as a series of columns. For compatibility with currently available systems, however, these Guidelines require a row-by-row description of a table. It is also possible to describe a table simply as a series of cells; this may be useful for tabular material which is not presented as a simple matrix.

The attributes rows and cols may be used to indicate the size of a table, or to indicate that a particular cell of a table spans more than one row or column. For both tables and cells, rows and columns are always given in top-to-bottom, left-to-right order. These Guidelines do not require that the size of a table be specified; for most formatting and many other applications, it will be necessary to process the whole table in two passes in any case.

Where cells span more than one column or row, the encoder must determine whether this is a purely presentational effect (in which case the rend attribute may be more appropriate), whether the part of the table affected would be better treated as a nested table, or whether to use the spanning attributes listed above.

The role attribute may be used to categorize a single cell, or set a default for all the cells in a given row. The present Guidelines distinguish the roles of *label* and *data* only, but the encoder may define other roles, such as “derived”, “numeric”, etc., as appropriate.

The following simple example demonstrates how the data presented as a labelled list in section 6.7 *Lists* might be represented by an encoder wishing to preserve its original appearance as a table:

```
<table rend="boxed" rows="2" cols="2">
   <head rend="it">Report of the conduct and progress
      of Ernest Pontifex.  Upper Vth form &mdash; half
      term ending Midsummer 1851</head>
   <row>
      <cell role="label">Classics</cell>
      <cell>Idle listless and unimproving</cell>
   </row>
   <row>
      <cell role="label">Mathematics</cell>
      <cell>ditto</cell>
   </row>
   <row>
      <cell role="label">Divinity</cell>
      <cell>ditto</cell>
   </row>
   <row>
      <cell role="label">Conduct in house</cell>
      <cell>Orderly</cell>
   </row>
   <row>
      <cell role="label">General conduct</cell>
      <cell>Not satisfactory, on
       account of his great unpunctuality and inattention to
       duties</cell>
   </row>
</table>
```

Note that this encoding makes no attempt to represent the full significance of the “ditto” cells above; these might be regarded as simple links between the cells containing them and that to which they refer, or as virtual copies of it. For ways of representing either interpretation, see chapter 14 *Linking, Segmentation, and Alignment*.

The following example demonstrates how a simple statistical table may be represented using this scheme:

```
<table rows="4" cols="4">
  <head>Poor Man's Lodgings in Norfolk (Mayhew, 1843)</head>
  <row role="label">
    <cell>  </cell>
    <cell>Dossing Cribs or Lodging Houses</cell>
    <cell>Beds</cell>
    <cell>Needys or Nightly Lodgers</cell> </row>
  <row>
    <cell role="label">Bury St Edmund's</cell>
    <cell>5</cell> <cell>8</cell> <cell>128</cell> </row>
  <row>
    <cell role="label">Thetford</cell>
    <cell>3</cell> <cell>6</cell> <cell>36</cell> </row>
  <row>
    <cell role="label">Attleboro'</cell>
    <cell>3</cell> <cell>5</cell> <cell>20</cell> </row>
  <row>
    <cell role="label">Wymondham</cell>
    <cell>1</cell> <cell>11</cell> <cell>22</cell> </row>
  <!-- ... -->
</table>
```

Note the use of a blank cell in the first row to ensure that the column labels are correctly aligned with the data. Again, this encoding does not explicitly represent the alignment between column and row labels and the data to which they apply. Where the primary emphasis of an encoding is on the semantic content of a table, a more explicit mechanism for the representation of structured information such as that provided by the feature structure mechanism described in chapter 16 *Feature Structures* may be preferred. Alternatively, the general purpose linkage and alignment mechanisms described in chapter 14 *Linking, Segmentation, and Alignment* may also be applied to individual cells of a table.

The content of a table cell need not be simply character data. It may also contain any sequence of the phrase level elements described in chapter 6 *Elements Available in All TEI Documents*, thus allowing for the encoding of potentially more useful semantic information, as in the following example, where the fact that one cell contains a number and the other contains a place name has been explicitly recorded:

```
<table>
  <head>US State populations, 1990</head>
  <row><cell> <name>Wyoming</name> </cell>
       <cell> <num>453,588</num> </cell></row>
  <row><cell> <name>Alaska</name> </cell>
       <cell> <num>550,043</num> </cell></row>
  <row><cell> <name>Vermont</name> </cell>
       <cell> <num>562,758</num> </cell></row>
  <row><cell> <name>District of Columbia</name> </cell>
       <cell> <num>606,900</num> </cell></row>
  <row><cell> <name>North Dakota</name> </cell>
       <cell> <num>638,800</num> </cell></row>
  <row><cell> <name>Delaware</name> </cell>
       <cell> <num>666,168</num> </cell></row>
  <row><cell> <name>South Dakota</name> </cell>
       <cell> <num>696,004</num> </cell></row>
  <row><cell> <name>Montana</name> </cell>
       <cell> <num>799,065</num> </cell></row>
  <row><cell> <name>Rhode Island</name> </cell>
       <cell> <num>1,003,464</num> </cell></row>
  <!-- ... -->
</table>
```

In syntactic terms this is little more than a name-change; however, the new names are more useful in that they convey something about the nature and significance of the information, rather than merely suggesting how to display it in rows and columns.

The TEI table elements are defined as follows:

```
<!-- 22.1.1: Tables-->
<!ELEMENT table %om.RR; ((head | %m.Incl;)*, (row, (%m.Incl;)*)+)>
<!ATTLIST table
      %a.global;
      rows CDATA #IMPLIED
      cols CDATA #IMPLIED
      TEIform CDATA 'table'  >
<!ELEMENT row
%om.RO; ((cell|table), (%m.Incl;)*)+>
<!ATTLIST row
      %a.global;
      role CDATA "data"
      TEIform CDATA 'row'  >
<!ELEMENT cell %om.RO; %paraContent;>
<!ATTLIST cell
      %a.global;
      role CDATA "data"
      rows CDATA "1"
      cols CDATA "1"
      TEIform CDATA 'cell'  >
<!-- end of 22.1.1-->
```

22.1.2 Other Table DTDs

Many authoring systems now include built-in support for their own or for public table DTDs. These often provide an enhanced user interface and good formatting capabilities, but are often product-specific, despite their use of a standard markup language such as SGML or XML.

The DTD developed by the Association of American Publishers (AAP) and standardized in ANSI Z39.59 provided a very simple encoding for correspondingly simple tables. This has been further developed, together with the table DTD documented in ISO Technical Report 9537, and now forms part of ISO 12083. The TEI DTD fragment described above has functionality very similar to that defined by ISO 12083.

For more complex tables, the most effective publically-available DTD is probably that developed by the US Department of Defense CALS project. This supports vertical and horizontal spanning and various kinds of text rotation and justification within cells and is also directly supported by a number of existing SGML software systems.

The CALS table DTD is much too complex to describe fully here; information on it can be obtained, among other places, from the Graphic Communications Association in Alexandria, Virginia. The formal name of the CALS SGML requirements is MIL-M-28001A. Tables conforming to the CALS DTD may be incorporated into documents conforming to these Guidelines, but this may require substantial modification of the TEI DTD which should not be undertaken without expert advice.

The apparent complexity of the CALS table DTD has led to simplification efforts, most notably by OASIS in the form of *OASIS Technical Resolution 9503:1995, Exchange Table Model Document Type Definition* (`http://www.oasis-open.org/specs/a503.htm`). An XML version of this is also defined in *OASIS Technical Memorandum TR 9901:1999, XML Exchange Table Model DTD* (`http://www.oasis-open.org/specs/tm9901.html`).

With the ascent of the Web, the HTML table model (`http://www.w3.org/TR/html4/struct/tables.html`) has become very popular and widespread in use. The last few years have seen a growing access to information via devices of varying capabilities ranging from desktop computers to handheld mobile phones. In response to this demand, HTML has been both reformulated into XML and modularized into a family of modules (see `http://www.w3.org/TR/xhtml-modularization`) with semantically related elements. This family includes two Table Modules: the Basic Tables Module (`http://www.w3.org/TR/xhtml-modularization/xhtml-modularization.html`) and the Tables Module (`http://www.w3.org/TR/xhtml-modularization/xhtml-modularization.html`). The Tables Module mimics the functionality of HTML 4 (`http://www.w3.org/TR/html401/`) tables and is a part of the XHTML 1.1 (`http://www.w3.org/TR/xhtml11`) document type. The Basic Tables Module is a 'diluted' version of the Tables Module, retaining only the minimal functionality with the purpose of extending XHTML's reach onto resource-constrained devices. It is a part of the XHTML Basic document type (`http://www.w3.org/TR/xhtml-basic`).

The XHTML table model is based on the HTML table model (`http://www.w3.org/TR/html4/struct/tables.html`). It allows arrangement of data into rows and columns of cells. Table rows and columns may be grouped to convey additional structural information and may be rendered by user agents in ways that emphasize this structure. Support for incremental rendering of tables and for rendering on non-visual user agents (`http://www.w3.org/TR/html4/struct/tables.html`) is also available. Special elements and attributes are provided to associate metadata with tables. They indicate the table's purpose, or are for the benefit of people using speech or Braille-based user agents. It is recommended that tables should not be used purely as a means to layout document content as this leads to many accessibility problems (see further `http://www.w3.org/TR/WCAG10-HTML-TECHS/`). Style sheets provide a far more effective means of controlling layout and other visual characteristics, in both HTML and XML documents.

22.2 Formulae and Mathematical Expressions

Mathematical and chemical formulæ pose similar problems to those posed by tables in that rendition may be of great significance and hard to disentangle from content. They also require access to a wide range of special characters, for most of which standard entity names already exist in the documented ISO entity sets (see further chapters 4 *Languages and Character Sets* and 25 *Writing System Declaration*).

Formulæ and tables are also similar in that well-researched and detailed DTD fragments have already been developed for them independently of the TEI. They differ in that (for mathematics at least) there also exists a richly detailed text-based but non-SGML notation which is very widely used: this is the TeX system, and the sets of descriptive macros developed for it such as LaTeX, AMS-TeX, and AMS-LaTeX.

The AAP and ISO standards mentioned in section 22.1 *Tables* above both provide DTDs for equations as well as for tables, which now form part of ISO 12083. The European Mathematical Trust, an organization set up specifically to enhance research support for European mathematicians, has also defined a general purpose mathematical DTD known as EuroMath (`http://www.dcs.fmph.uniba.sk/~emt/`), for which it provides both software and services.

Most if not all of the functionality provided by these DTDs can now be found in the OpenMath and MathML XML-based systems briefly described below.

As with tables, in all the SGML and XML solutions a tension exists between the need to encode the way a formula is written (its appearance) and the need to represent its semantics. If the object of the encoding is purely to act as an interchange format among different formatting programs, then there is no need to represent the mathematical meaning of an expression. If however the object is to use the encoding as input to an algebraic manipulation system (such as Mathematica or Maple) or a database system, clearly simply representing superscripts and subscripts will be inadequate.

The present Guidelines make no attempt to add to the number of available DTDs for representing formulæ. Instead, we recommend that the user make an informed choice from those already available. The additional tag set described in this chapter makes available only the following element, which should be used to encode any formula, no matter what notation is employed:

<formula> contains a mathematical or other formula. Attributes include:

- **notation** supplies the name of a previously defined notation used for the content of the element.
  - *Values* The name of a formal notation previously declared in the document type declaration.

The legal content of a `<formula>` is determined by two factors. The parameter entity *formulaContent* supplies a content model for the element; while the `notation` attribute specifies what formal *notation* employed within it. Parameter entities formulaContent and formulaNotations may be used to override the default assumptions for these factors. By default, a `<formula>` is assumed to contain only `#PCDATA`, i.e. parsed character data, and may thus use entity references for any special symbols required. [155] For example, an expression such as `a<b`, because it contains the < character cannot be included directly in any XML element; instead, it must be represented by an entity reference:

```
<formula notation="TeX"> a&lt;b </formula>
```

If so desired, the content of the `<formula>` element may be redefined to include elements defined by some other tag set, such as that of ISO 12083, or to use the more recently defined OpenMath or MathML DTDs.[156]

When the content of a `<formula>` element is not expressed in XML or SGML, the notation used should be specified using the `notation` attribute as in the above example. Each notation used by a document must be declared in its DTD subset:

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN"
 "http://www.tei-c.org/P4X/DTD/tei2.dtd" [
  <!ENTITY % TEI.graphics "INCLUDE">
```

[155] In earlier editions of these Guidelines, *formulaContent* was defined by default as `CDATA`, which in SGML means that the only parsing carried out is to search for an end-tag; since XML does not include the `CDATA` element type (it is one of the very few features of SGML that, if used, makes correct parsing in the absence of a DTD intractable), the present edition of these Guidelines defines the content of *formulaContent* as `(#PCDATA)`.

[156] In this case additional redefinitions may also be needed to avoid name clashes with existing TEI elements. For further details see chapter 29 *Modifying and Customizing the TEI DTD*.

```
<!NOTATION TeX PUBLIC
 '-//Donald E. Knuth 1983//NOTATION TeX Mathematical Markup//EN'>
]>
```

With these declarations in force, a document may include formulæ expressed using standard TeX conventions, as in the following example:

```
<p>Achilles runs ten times faster than the tortoise and
gives the animal a headstart of ten meters.  Achilles runs
those ten meters, the tortoise one; Achilles runs that
meter, the tortoise runs a decimeter; Achilles runs that
decimeter, the tortoise runs a centimeter; Achilles runs
that centimeter, the tortoise, a millimeter; Fleet-footed
Achilles, the millimeter, the tortoise, a tenth of a
millimeter, and so on to infinity, without the tortoise ever
being overtaken. . . Such is the customary version.
<!-- ... -->
The problem does not change, as you can see; but I would
like to know the name of the poet who provided it with a
hero and a tortoise.  To those magical competitors and to
the series
<formula notation="TeX">$$
{1 \over 10} +
{1 \over 100} +
{1 \over 1000} +
{1 \over 10,\!000} +
\dots
$$</formula>
the argument owes its fame.</p>
```

The notation attribute supplies the name of a defined notation (“TeX”), which is associated by its declaration in the DTD subset with an external public entity. How that declaration is resolved will depend on the kind of processor in use, and is outside the scope of these Guidelines.

The following SGML- and XML- based notations for encoding formulæ are recommended by these Guidelines:

```
<!NOTATION iso12083 PUBLIC 'ISO 12083:1993//DTD Formulae//EN'>
<!NOTATION euromath PUBLIC '-//Euromath 1992//DTD Euromath equations//EN'>
<!NOTATION mathml2  PUBLIC '-//W3C//DTD MathML 2.0//EN'>
<!NOTATION openmath PUBLIC '-//OpenMath 1999//DTD for OpenMath Objects//EN'>
```

Mathematical Markup Language (MathML) (`http://www.w3.org/Math/`) is a vocabulary for describing mathematical notation, capturing both its structure and content. MathML 1.0, which became a W3C Recommendation on April 7 1998, was the first vocabulary based on XML syntax to reach that status. It was subsequently revised, as MathML 1.01 (W3C Recommendation July 7, 1999), and as the current version MathML 2.0, which became a W3C recommendation on February 21, 2001.

MathML provides two types of markup: Presentation Markup, which captures the **notational** structure of an expression and could be seen as the ‘TeX for the Web’ and Content Markup, which captures the **mathematical** structure of an expression. Most of its content elements correspond with the range of operators, relations, and named functions typically found at the high school level of mathematics. The tortoise example given above in TeX can be re-expressed in MathML as

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
   <mfrac>
      <mrow> <mn>1</mn> </mrow>
      <mrow> <mn>10</mn> </mrow>
   </mfrac>
   <mo>+</mo>
   <mfrac>
      <mrow> <mn>1</mn> </mrow>
      <mrow> <mn>100</mn> </mrow>
   </mfrac>
   <mo>+</mo>
```

```
      <mfrac>
         <mrow> <mn>1</mn> </mrow>
         <mrow> <mn>1000</mn> </mrow>
      </mfrac>
      <mo>+</mo>
      <mfrac>
         <mrow> <mn>1</mn> </mrow>
         <mrow> <mn>10000</mn> </mrow>
      </mfrac>
      <mo>+</mo>
      <mo>&#x2026;</mo>
   </math>
```

MathML 2.0 (http://www.w3.org/TR/MathML2) adds to these some additional elements for MathML Content Markup, thereby further strengthening the support for a 'Semantic Math-Web'. It also provides support for XML namespaces, and other current XML standards, such as XML DOM, OMG IDL, ECMAScript and Java. It also provided a modularized version of the MathML DTD so that MathML fragments 'embedded' in XHTML 1.1 documents as data islands can be correctly validated.

The OpenMath (http://www.nag.co.uk/projects/OpenMath.html) project is coordinated by the OpenMath Society (http://www.openmath.org/)and funded by the European Commission under the Esprit Multimedia Standards Initiative that commenced in September 1997. It is likely to become a key standard for communicating semantically-rich representations of mathematical objects both on and off the Web in a platform-independent manner.

The OpenMath Standard (http://www.openmath.org/V2/standard/index.html) consists of specifications for

1. OpenMath objects, representing the structure of formulas (http://www.openmath.org/V2/standard/objects.html);
2. Content Dictionaries, providing semantic context (http://www.openmath.org/V2/standard/cd.html);
3. Encodings, both binary (http://www.openmath.org/V2/standard/binary.html) and XML (http://www.openmath.org/V2/standard/xml.html).

OpenMath and MathML have certain common aspects. They both use prefix operators, both are XML based and they both construct their objects by applying certain rules recursively. Such similarities facilitate mapping across both standards. There are also some key differences between MathML and OpenMath. OpenMath does not provide support for presentation of mathematical objects and its scope of semantically-oriented elements is much broader that of MathML, with the expressive power to cover virtually all areas of computational mathematics. In fact, a particular set of Content Dictionaries, the 'MathML CD Group', covers the same areas of mathematics as the Content Markup elements of MathML 2.0.

Finally, OMDoc (http://www.mathweb.org/omdoc/) is an extension of the OpenMath standard that supplies markup for structures such as axioms, theorems, proofs, definitions, texts (mixing formal content with mathematical text).

In-line versus block placement for an equation can be distinguished if desired, via the global rend attribute. The global n and id attributes may also be used to label or identify the formula, as in the following (imaginary) example:[157]

```
<p>The volume of a sphere is given by the formula:
<formula id="f12" n="12" rend="inline" notation="mathml">
 <math>
   <mi>V</mi>
   <mo>=</mo>
   <mfrac>
      <mrow> <mn>4</mn> </mrow>
      <mrow> <mn>3</mn> </mrow>
```

[157] We do not show here how the MathML names are to be included in the TEI name space

```
      </mfrac>
      <mi>&#0960;</mi>
      <msup>
         <mrow> <mi>r</mi> </mrow>
         <mrow> <mn>3</mn> </mrow>
      </msup>
   </math>
  </formula>
  which is readily calculated.</p>
  <p>As we have seen in equation
  <ptr target="f12"/>, ... </p>
```

The *formulaContent* and *formulaNotations* parameter entities are defined as follows:

```
<!-- 22.2: Formula Content-->
<!ENTITY % formulaNotations 'CDATA' >
<!ENTITY % formulaContent '(#PCDATA)' >
<!-- end of 22.2-->
```

The formula element itself is defined as follows:

```
<!-- 22.2: Formulae-->
<!ELEMENT formula %om.RR; %formulaContent;>
<!ATTLIST formula
      %a.global;
      notation %formulaNotations; #REQUIRED
      TEIform CDATA 'formula'  >
<!-- end of 22.2-->
```

22.3 Specific Elements for Graphic Images

The following special purpose elements are provided by this tag set to indicate the presence of graphic images within a document:

<figure> indicates the location of a graphic, illustration, or figure. Attributes include:
  entity names the external entity within which the graphic image of the figure is stored.
    Values the name of an external unparsed entity declared elsewhere in the DTD.

<figDesc> contains a brief prose description of the appearance or content of a graphic figure, for use when documenting an image without displaying it.

Inclusion of a graphic image in a marked-up document typically requires three distinct steps:

1. The *notation* employed by the image itself must be defined; this is done with a notation declaration in the document type definition.
2. The external entity in which the image is stored must be defined; this is done with an entity declaration, which refers to the notation declared at step one.
3. Within the document, the `<figure>` element is used to mark the position of the image, which is referenced by name, like any other kind of external entity.

In the TEI scheme, these three functions are carried out as follows.

Declarations for all notations used by a document must be provided within the DTD subset, as described above in section 22.2 *Formulae and Mathematical Expressions*. Many such notations are in common use; for details see section 22.5 *Graphic Image Formats*.

Entity declarations for the entities containing the graphics themselves must be made, using system or public identifiers, within the document's DTD subset, either directly or by including them within a suitable file, as in the example below.

```
<!DOCTYPE TEI.2 PUBLIC "-//TEI P4//DTD Main Document Type//EN"
                       "http://www.tei-c.org/P4X/DTD/tei2.dtd" [
  <!ENTITY % TEI.XML "INCLUDE">
  <!ENTITY % TEI.prose "INCLUDE">
  <!ENTITY % TEI.graphics "INCLUDE">

  <!-- Graphics notations used in this document ... -->
  <!NOTATION svg PUBLIC
    '-//TEI//NOTATION W3C Scalable Vector Graphics Format//EN' >
  <!NOTATION png PUBLIC
    '-//TEI//NOTATION IETF RFC2083 Portable Network Graphics//EN'>
  <!NOTATION jpeg  PUBLIC
    'ISO DIS 10918//NOTATION JPEG Graphics Format//EN' >

  <!-- The file 'figures.ent' contains entity declarations -->
  <!-- for all external entities needed by this document -->
  <!ENTITY % myFigures SYSTEM "figures.ent"> %myFigures;
]>
```

The file *figures.ent* will contain a series of declarations like the following:

```
<!ENTITY Fig1    SYSTEM  "fig1.svg"    NDATA svg>
<!ENTITY Fig1th  SYSTEM  "fig1.jpg"    NDATA jpeg>
<!ENTITY pullman SYSTEM  "pullman.png" NDATA png>
```

the effect of which is to associate the name Fig1 with the external entity fig1.svg, and also to declare that that entity uses the notation called `svg`, which is declared in the DTD subset. In the same way, the external entity fig1.jpg is defined as using the `jpeg` notation, and may be referenced by the name Fig1th (see further below).

Finally, the `<figure>` element is used to indicate the location of the graphic image in the text. For example:

```
<figure entity='Fig1'></figure>
```

Note that an end-tag is always required for this element. Three kinds of content may be supplied: the element `<head>` may be used to transcribe (or supply) a descriptive heading or title for the graphic itself as in this example:

```
<figure entity='Fig1'><head>Figure One: The View from the Bridge</head></figure>
```

Figures are often accompanied not only by a title or heading, but by a paragraph or so of commentary or caption. One or more `<p>` elements following the `<head>` may be used to transcribe any caption or discussion of the figure in the source:

```
<figure entity='pullman'>
  <head>Above:</head>
  <p>The drawing room of the Pullman house, the white and gold saloon
    where the magnate delighted in giving receptions for several
    hundred people.</p>
  <figDesc>The figure shows an elaborately decorated room, at least
    twenty-five feet side to side and fifty feet long, with ornate
    mouldings and Corinthian columns on the walls, overstuffed
    armchairs and loveseats arranged in several conversational
    groupings, and two large chandeliers.</figDesc>
</figure>
```

Here, the paragraph “The drawing room ... several hundred people” is transcribed from the source, while the description is provided by the encoder, for use by applications which cannot display the graphic directly. In documents created in electronic form with the needs of print-handicapped readers in mind, the `<figDesc>` element may be provided by the author rather than a subsequent encoder.

```
<figure entity='Fig1'>
  <head>Figure One: The View from the Bridge</head>
  <figDesc>A Whistleresque view showing four
    or five sailing boats in the foreground, and a
    series of buoys strung out between them.</figDesc>
</figure>
```

Where the graphic itself contains large amounts of text, perhaps with a complex structure, and perhaps difficult to distinguish from the graphic, the encoder should choose whether to regard the graphic as containing the text (in which case, a nested `<text>` element may be included within the `<figure>` element) or to regard the enclosed text as being a separate division of the `<text>` element in which the graphic appears. In this latter case, an appropriate *divn* class element may be used for the text represented within the graphic, and the `<figure>` element embedded within it. The choice will depend to a large degree on the encoder's understanding of the relationship between the graphic and the surrounding text.

Like any other element in the TEI scheme, figures may be given identifiers so that they can be aligned with other elements, and linked to or from them, as described in chapter 14 *Linking, Segmentation, and Alignment*. Some common examples are discussed briefly here; full information is provided in that chapter.

It is often desirable to maintain two versions of an image in an electronic file: one a low resolution or 'thumbnail' version which, when selected by the user, causes the other, high resolution, version to be accessed. In TEI terms, the thumbnail image acts as a *reference* to the other. Referring to the example above, we will assume that the entity `Fig1th` contains a thumbnail version of the full Fig1 entity. We can now embed a reference to that image using the simple `<ref>` element discussed in section 6.6 *Simple Links and Cross References*:

```
<ref target='IM1'>Click here
  <figure entity='fig1th'></figure>
  for enlightenment
</ref>
<!-- ... -->
<!-- elsewhere in the document -->
<figure id='IM1' entity='fig1'></figure>
<!-- other figures here -->
```

Another common requirement is to associate part or the whole of an image with a textual element not necessarily contiguous to it in the text; this is sometimes known as a *callout*. Again, chapter 14 *Linking, Segmentation, and Alignment* should be consulted for the full details of the mechanisms available for this purpose. This example assumes that we wish to associate one portion of the image held as "fig1" with chapter two of some text, and another portion of it with chapter three. The application may be thought of as a hypertext browser in which the user selects from a graphic image which part of a text to read next, but the mechanism is independent of this particular application.

The first requirement is some way of identifying and hence pointing to sub-parts of a graphic image. This is most easily done using the extended pointer syntax discussed in section 14.2 *Extended Pointers*: thus

```
<xptr id='PD1' doc='Fig1' from='space (0 0 9 9)'/>
<xptr id='PD2' doc='Fig1' from='space (40 90 59 119)'/>
```

These `<xptr>` elements identify two areas within the image "Fig1" using the TEI extended pointer syntax. The first (with identifier "pd1") is a square of size 10 by 10, tangent to the origin. The second (with identifier "pd2") is a rectangle of size 20 by 30, starting at the point with co-ordinates (40,90) in the co-ordinate system used by this document.

The next requirement is some way of identifying the parts of the document to which a link is to be made. The most obvious way of doing this is to use the global `id` attribute:

```
<div1 type='chapter' id='C1'>
  <!-- text of chapter one here -->
</div1>
<div1 type='chapter' id='C2'>
  <!-- text of chapter two here -->
</div1>
```

Now, all that is needed to linking these areas to the relevant chapters is a `<linkGrp>` element, as described in section 14.1 *Pointers*:

```
<linkGrp type='callout'>
  <link targets='C1 pd1'/>
  <link targets='C2 pd2'/>
  <!-- ... -->
</linkGrp>
```

Further examples of this technique are provided in chapter 14 *Linking, Segmentation, and Alignment*.

The elements discussed in this section are defined as follows:

```
<!-- 22.3: Figures-->
<!ELEMENT figure %om.RR; ((%m.Incl;)*,
                              (head, (%m.Incl;)*)?,
                              (p, (%m.Incl;)*)*,
                              (figDesc, (%m.Incl;)*)?,
                              (text, (%m.Incl;)*)?) >
<!ATTLIST figure
      %a.global;
      entity ENTITY #IMPLIED
      TEIform CDATA 'figure'  >
<!ELEMENT figDesc %om.RR;  %paraContent;>
<!ATTLIST figDesc
      %a.global;
      TEIform CDATA 'figDesc'  >
<!-- end of 22.3-->
```

22.4 Overview of Basic Graphics Concepts

The first major distinction in graphic representation is that between raster graphics and vector graphics. A *raster image* is a list of points, or dots. Scanners, fax machines and other simple devices easily produce digital raster images, and such images are therefore quite common. A *vector image*, in contrast, is a list of geometrical objects, such as lines, circles, arcs, or even cubes. These are much more difficult to produce, and so are mainly encountered as the output of sophisticated systems such as architectural and engineering CAD programs.

Raster images are difficult to modify because by definition they only encode single points: a line, for example, cannot grow or shrink as such, since it is not identified as such. Only its component parts are identified, and only they can be manipulated. Therefore the resolution or dot-size of a raster image is important, which is not the case with vector images. It is also far more difficult to convert raster images to vector images than to perform the opposite conversion. Raster images generally require more storage space than vector images, and a wide variety of methods exists for compressing them; the variation in these methods leads to corresponding variations in representations for storage and transmission of raster images.

Motion video usually consists of a long series of raster images. Data compression is even more effective on video than on single raster images (mainly owing to redundancy which arises from the usual similarity of adjacent frames). Notations for representing full-motion video are hotly debated at this time, and any user of these Guidelines would do well to obtain up-to-date expert advice before undertaking a project using them.

The compression methods used with any of these image types may be ‘lossy’ or ‘lossless’. Methods for *lossy compression* save space by discarding a small portion of the image’s detail, such as fine distinctions of shading. When decompressed, therefore, such an image will be only a close approximation of the original. In contrast, *lossless compression* guarantees that the exact uncompressed image will be reproducible from the compressed form: only truly redundant information is removed. In general, therefore, lossless compression does not save quite so much space as lossy compression, though it does guarantee fidelity to the original uncompressed image.

Raster images may be characterized by their *resolution*, which is the number of dots per inch used to represent the image. Doubling the resolution will give a more precise image, but also quadruple the storage requirement (before compression), and affect processing time for any operations to be performed, such as displaying an image for a reader. Motion video also has resolution in time: the number of frames to be shown per second. Encoders should consider carefully what resolution(s) and frame rate(s) to use for particular applications; these Guidelines express no recommendation in this matter, save the universal ones of consistency and documentation.

Within any image, it is typical to refer to locations via Cartesian co-ordinate axes: values for x, y, and sometimes z and/or time. These Guidelines provide for this via the `SPACE` keyword of the extended pointing mechanism discussed in section 14.2 *Extended Pointers*. However, graphic notations vary in whether co-ordinates count from left-to-right and top-to-bottom, or another way. They also vary in whether co-ordinates are considered real (inches, millimeters, and so on), or virtual (dots). These Guidelines do not recommend any of these methods over another, but all decisions made should be applied consistently, and documented in the `<encodingDesc>` section of the TEI header.[158]

The way in which the color of an image is rendered also varies greatly. In monochrome images every displayed point is either black or white. In *gray-scale* images, each point is rendered in some shade of gray, the number of shades varying from system to system. In true polychrome images, points are rendered in different colours, again with varying limitations affecting the number of distinct shades and the means by which they are displayed.

22.5 Graphic Image Formats

As noted above, there exists a bewildering variety of different graphics formats, and the following list is in no way exhaustive. Moreover, inclusion of any format in this list should not be taken as indicating endorsement by the TEI of this format or any products associated with it. With the exception of CGM, JPEG, PNG, and SVG, all the formats listed here are proprietary to a greater or lesser extent and cannot therefore be regarded as standards in any meaningful sense. They are however widely used by many different vendors.

The following formats are widely used at the present time, and likely to remain supported by more than one vendor's software:

- BMP: Microsoft bitmap format
- CGM: Computer Graphics Metafile
- GIF: Graphics Interchange Format
- JPEG: Joint Photographic Expert Group
- PBM: Portable Bit Map
- PCX: IBM PC raster format
- PICT: Macintosh drawing format
- PNG: Portable Network Graphics format
- Photo-CD: Kodak Photo Compact Disk format
- QuickTime: Apple real-time image system
- SMIL: Synchronized Multimedia Integration Language format
- SVG: Scalable Vector Graphics format
- SVG: Scalable Vector Graphics format
- TIFF: Tagged Image File Format

Brief descriptions of all the above are given below. Where possible, current addresses or other contact information are shown for the originator of each format. Many formal standards, especially those promulgated by ISO and many related national organizations (ANSI, DIN, BSI, and many more), are available from those national organizations. Addresses may be found in any standard organizational directory for the country in question.

For each format, a sample notation declaration is given, using a formal public identifier constructed from the best information available at the date of publication. It is recommended that such formal public identifiers always be used in the interchange of documents between sites. Unless otherwise noted, however, these formal public identifiers have been formulated by the TEI, and not by the owners of the notation, as is indicated by the use of 'TEI' as the owner identifier. If more recent versions of these formal public identifiers, or versions promulgated by the owners of the notation, are available at the time of document interchange, they should be used in preference to those shown here.

Support for formal public identifiers varies somewhat among existing SGML and XML systems; for local processing, the notation declaration may therefore need to include a system identifier in addition to the formal public identifier. The documentation for the system in use should be consulted for details.

[158] Since no special purpose element is provided for this purpose by the current version of the Guidelines, such information should be provided as one or more distinct paragraphs at the end of the `<encDecl>` element described in section 5.3 *The Encoding Description*.

22.5.1 Vector Graphic Formats

CGM: Computer Graphics Metafile This vector graphics format is specified by an ISO standard, ISO 8632:1987, amended in 1990. It defines binary, character, and plain-text encodings; the non-binary forms are safer for blind interchange, especially over networks. Documentation on CGM is available from ISO and from its member national bodies such as AFNOR, ANSI, BSI, DIN, JIS, etc. Sample declarations:

```
<!NOTATION cgm PUBLIC
    'ISO 8632:1987//NOTATION Computer Graphics Metafile//EN'>
<!NOTATION cgmchar  PUBLIC
    'ISO 8632-2:1987//NOTATION Character encoding//EN'>
<!NOTATION cgmbin   PUBLIC
    'ISO 8632-3:1987//NOTATION Binary encoding//EN'>
<!NOTATION cgmclear PUBLIC
    'ISO 8632-4:1987//NOTATION Clear text
    encoding//EN'>
```

SVG: Scalable Vector Graphics format SVG is a language for describing two-dimensional vector and mixed vector or raster graphics in XML. It is defined by the Scalable Vector Graphics (SVG) 1.0 Specification, W3C Recommendation, 04 September 2001, and is available at `http://www.w3.org/TR/2001/REC-SVG-20010904/`.

```
<!NOTATION SVG PUBLIC
  '-//TEI//NOTATION W3C Scalable Vector Graphics Format//EN' >
```

PICT: Macintosh drawing format This format is universally supported on Macintosh(tm) systems, and readable by a limited range of software for other systems. Documentation is available from Apple Computer Company, Cupertino, California USA.

```
<!NOTATION pict PUBLIC
    '-//TEI//NOTATION PICT:  Macintosh Drawing Format//EN'>
```

22.5.2 Raster Graphic Formats

PNG: Portable Network Graphics format PNG is the only non-proprietary raster format currently widely available. It provides an extensible file format for the lossless, portable, well-compressed storage of raster images. As such, it is a patent-free replacement for GIF and can also replace many common uses of TIFF. Indexed-color, grayscale, and truecolor images are supported, plus an optional alpha channel. Sample depths range from 1 to 16 bits. It is defined by IETF RFC 2083, March 1997.

```
<!NOTATION PNG PUBLIC
  '-//TEI//NOTATION IETF RFC2083 Portable Network Graphics//EN' >
```

TIFF: Tagged Image File Format Currently the most widely supported raster image format, especially for black and white images, TIFF is also one of the few formats commonly supported on more than one operating system. The drawback to TIFF is that it actually is a wrapper for several formats, and some TIFF-supporting software does not support all variants. TIFF files may use LZW, CCITT Group 4, or PackBits compression methods, or may use no compression at all. Also, TIFF files may be monochrome, greyscale, or color. All such options should be specified in prose at the end of the `<encodingDesc>` section of the TEI header for any document including TIFF images. TIFF is owned by Aldus Corporation. Documentation on TIFF is available from them at Craigcook Castle, Craigcook Road, Edinburgh EH4 3UH, Scotland, or 411 First Avenue South, Seattle, Washington 98104 USA.

```
<!NOTATION tiff PUBLIC
   '-//TEI//NOTATION Aldus Tagged Image File Format//EN'>
```

GIF: Graphics Interchange Format Color raster images are widely available in this form, which was created by CompuServe Information Services, but has by now been implemented for many other systems as well. Documentation on GIF is copyright by, and is available from, CompuServe Incorporated, Graphics Technology Department, 5000 Arlington Center Boulevard, Columbus, Ohio 43220 USA.

```
<!NOTATION gif PUBLIC
  '-//TEI//NOTATION
  Compuserve Graphics Interchange Format//EN' >
```

PBM: Portable Bit Map PBM files are easy to process, eschewing all compression in favor of transparency of file format. PBM files can, of course, be compressed by generic file-compression tools for storage and transfer. Public domain software exists which will convert many other formats to and from PBM. Documentation on PBM is copyright by Jeff Poskanzer, and is available widely on the Internet.

```
<!NOTATION pbm PUBLIC
  '-//TEI//NOTATION Jeff Poskanzer, Portable Bit Map//EN' >
```

PCX: IBM PC raster format This format is used by most IBM PC paint programs, and supports both monochrome and color images. Documentation is available from ZSoft Corporation, Technical Support Department, ATTN: Technical Reference Manual, 450 Franklin Rd. Suite 100, Marietta, GA 30067 USA.

```
<!NOTATION pcx PUBLIC
  '-//TEI//NOTATION PCX:  ZSoft IBM PC Raster Graphics Format//EN' >
```

BMP: Microsoft bitmap format This format is the standard raster format for computer using Microsoft Windows (tm) or Presentation Manager (tm). Documentation is available from Microsoft Corporation.

```
<!NOTATION bmp PUBLIC
  '-//TEI//NOTATION BMP:  Microsoft Bitmap Graphics Format//EN' >
```

22.5.3 Photographic and Motion Video Formats

JPEG: Joint Photographic Experts Group This standard is sponsored by CCITT and by ISO. It is ISO/IEC Draft International Standard 10918-1, and CCITT T.81. It handles monochrome and color images with a variety of compression techniques. JPEG per se, like CCITT Group IV, must be encapsulated before transmission; this can be done via TIFF, or via the JPEG File Interchange Format (JFIF), as commonly done for Internet delivery.

```
<!NOTATION JPEG PUBLIC
  'ISO DIS 10918//NOTATION JPEG Graphics Format//EN' >
```

QuickTime: Apple real-time image system QuickTime is a proprietary method introduced by Apple Computer Company to synchronize the display of various data. The data can include frames of video, sound, lighting control equipment, and other things. Viewers for QuickTime productions are available for Apple and other computers. Further information is available from Apple Computer Incorporated, 10201 North de Anza Boulevard MS 23AQ, Cupertino, California 95014 USA.

```
<!NOTATION QuickTime PUBLIC
  '-//TEI//NOTATION Apple QuickTime Video Data Format//EN' >
```

Photo-CD: Kodak Photo Compact Disk format This format was introduced by Kodak for rasterizing photographs and storing them on CD-ROMs (about one hundred 35mm file images fit on one disk), for display on televisions or CD-I systems. Information on Photo-CD is available from Kodak Limited, Research and Development, Headstone Drive, Harrow, Middlesex HA1 4TY, UK.

```
<!NOTATION pcx PUBLIC
  '-//TEI//NOTATION Eastman Kodak Photo-CD Raster Graphics Format//EN' >
```

SMIL: Synchronized Multimedia Integration Language format SMIL is a W3C Recommendation which supports the integration of independent multimedia objects into a synchronized multimedia presentation. It provides multimedia authors with easily-defined basic timing relationships, fine-tuned synchronization, spatial layout, direct inclusion of non-text and non-image media objects, hyperlink support for time-based media, adaptiveness to varying user and system characteristics. SMIL 1.0 (`http://www.w3.org/TR/REC-smil/`) became a W3C Recommendation on June 15, 1998, and was further developed in SMIL 2.0. SMIL

2.0 adds native support for transitions, animation, event-based interaction, extended layout facilities, and more sophisticated timing and synchronization primitives to the SMIL 1.0 language. It also allows reuse of SMIL syntax and semantics in other XML-based languages, in particular those who need to represent timing and synchronization. For example, SMIL 2.0 components are used for integrating timing into XHTML Document Types and into SVG. SMIL 2.0 also provides recommendations for Document Types based on SMIL 2.0 Modules (`http://www.w3.org/TR/smil20/smil-modules.html`). One such Document Type is the SMIL 2.0 Language Profile (`http://www.w3.org/TR/smil20/smil20-profile.html`). It contains support for all of the major SMIL 2.0 features including animation, content control, layout, linking, media object, meta-information, structure, timing and transition effects and is designed for Web clients that support direct playback from SMIL 2.0 markup. SMIL 2.0 (`http://www.w3.org/TR/smil20/`) became a W3C Recommendation on August 7, 2001, becoming the first vocabulary to provide XML Schema support and to have reached such status.

```
<!NOTATION SMIL PUBLIC
    '-//TEI//NOTATION W3C Synchronized Multimedia Integration Language
    Format//EN' >
```

As noted above, the reader will encounter many, many other graphics formats. Other formats are not recommended for data interchange according to the TEI scheme at this time, but may be included in a TEI document without affecting its conformance in other respects, provided that a notation declaration is provided.