The tbook DTD[*]

Torsten Bronger
bronger@users.sourceforge.net

September 30, 2002

Abstract

The tbook DTD described here is supposed to imitate the core functions of LaTeX's book and article classes. Basic letter support is also included. XSLT stylesheets are provided to convert tbook XML documents to high-level LaTeX or XHTML. Table of contents, bibliography and index are created similarly to LaTeX. Mathematical formulas can be entered via MathML or – for simple ones – in some sort of LaTeX style. Graphics and diagrams are converted automatically for the necessary output formats (PS, PDF, HTML).

Contents

[*] This file has version number v1.4, last revised 2002/09/05.

The documentation driver file

The next bit of code contains the documentation driver file for TeX, i. e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the `docstrip` program. Since it is the first code in the file one can alternatively process this file directly with LaTeX 2_ε to obtain the documentation.

```
⟨*driver⟩
\documentclass{ltxdoc}

```

If you comment out the following command, you get a list of all entities and their LaTeX representation, too.

```
\OnlyDescription

```

These definitions are provisional.

```
\newenvironment{template}[1]{\begin{macro}{#1}}{\end{macro}}
\newenvironment{variable}[1]{\begin{macro}{#1}}{\end{macro}}
\newenvironment{element}[1]{\begin{environment}{#1}}{\end{environment}}
\let\DescribeTemplate=\DescribeMacro
\let\DescribeVariable=\DescribeMacro
\let\DescribeElem=\DescribeEnv

\usepackage{url}
\providecommand*{\url}[1]{\texttt{\mbox{#1}}}

\RecordChanges
\CodelineIndex
\setcounter{StandardModuleDepth}{1}
\DoNotIndex{\begin,\end,\def}

\begin{document}
  \DocInput{tbookdtd.dtx}
  \PrintChanges
  \PrintIndex
\end{document}
⟨/driver⟩
```

1 The tbook document type definition

The code of this section will eventually be in the file `tbook.dtd` and is used to validate and to transform XML documents that conform with the **tbook** DTD to LaTeX, XHTML or other XML.

```
⟨*tbookdtd⟩
```

1.1 Header definitions

```
<!-- PUBLIC "-//Torsten Bronger//DTD tbook 1.3//EN"
            "http://tbookdtd.sourceforge.net/tbook13.dtd" -->

```

1.2 Shortcuts for the content models

Most elements have the following attributes.

```
<!ENTITY % basic        "id    ID #IMPLIED
                         class CDATA #IMPLIED
                         style CDATA #IMPLIED">

```

Most non-empty elements have this language attribute.

```
<!ENTITY % i18n         "xml:lang NMTOKEN #IMPLIED">

```

Declaration of parameter entities for structuring the DTD.

```
<!ENTITY % fontchange   "em|visual|verb">
<!ENTITY % miscinline   "url|hspace|unit|relax|wrap">
<!ENTITY % xref         "cite|pageref|ref|vref|mathref">
<!ENTITY % preformat    "verbatim|verse">
<!ENTITY % mathblock    "theorem|proof">
<!ENTITY % index        "ix|idx|indexsee">
<!ENTITY % blockinline  "math|ch|latex">
<!ENTITY % inline       "%fontchange;|m|%blockinline;|%miscinline;|%xref;
                         |footnote|%index;|graphics">
```

`%blockinline;` is both block and inline; however, in absence of inline it must be included explicitly.

If an inline element is added here, it must possibly also be added to `parent-can-contain-inline-math` in `tbcommon.xsl`.

```
<!ENTITY % list         "description|enumerate|itemize">
<!ENTITY % block        "dm|quote|tabular|%preformat;|%list;|p|multipar">
<!ENTITY % bigblock     "figure|table|%mathblock;">


```

1.3 The top level declarations

1.3.1 `book` – root element

book

```
<!ELEMENT book          (frontmatter,mainmatter,backmatter?)>
<!ATTLIST book          %basic; xml:lang NMTOKEN "en">
```

This element embraces all other elements of a *book*. If a `class` attribute is given, and its contents ends with '`.sty`', the whole thing is interpreted as a LaTeX style file that's included instead of `tbook-pl.sty`. However, an explicit style file as an XSLT parameter still has higher precedence.

frontmatter

1.3.2 `frontmatter` – titlepage info

```
<!ELEMENT frontmatter   (title,author+,subtitle?,date?,keywords?,
                           year?,city?,graphics?,typeset?,legalnotice?)>
```

Everything that is normally mentioned before the table of contents. Only title and one author is a must, the rest if you wish, but the given order is (unfortunately) mandatory. The first mentioned author will also be printed after the copyright.

`<author>` has to be simple, i. e. it mustn't contain any additional information such as institute or email address. If it does, that is ignored. (In contrast to in articles.)

mainmatter

1.3.3 `mainmatter` – contents

```
<!ELEMENT mainmatter    ((part|chapter)*,appendix?)>
```

All parts and chapters of a *book*.

backmatter

1.3.4 `backmatter`

```
<!ELEMENT backmatter    (references?,index?)>
```

Bibliography and index. Both optional, but pay attention to order.

article

1.3.5 `article` – root element

```
<!ELEMENT article       (title,author+,date?,keywords?,year?,abstract?,
                          (%block;|%blockinline;|%bigblock;)*,section*,
                          references)>
<!ATTLIST article       %basic; xml:lang NMTOKEN "en">
```

Embraces all elements of an *article*. Title and one author are mandatory. Here – in contrast to book – the `<author>` tag can contain `<newline>` and `<footnote>` elements for e. g. insitute or email address.

As for the `class` attribute, see the `<book>` element.

letter

1.3.6 `letter` – root element

```
<!ELEMENT letter        (city,date,to,subject,opening,
                          (%block;|%blockinline;)*,closing)>
<!ATTLIST letter        %basic; xml:lang NMTOKEN "en"
                        from CDATA #REQUIRED
                        formal (true|false) #IMPLIED>
```

Embraces all elements of a *letter*.

`from` can be e. g. `"Torsten Bronger"`. It should be *your* personal id and should be globally unique, as far as one can guarantee that. Because this attribute is used by the stylesheets to ensure that they are responsible for you (correct letter head etc).[1] Moreover the `owner` attribute in the address book must match it.

If you don't set `formal` explicitly, the default is taken from the address book entry (even if you gave an explicit "to"-address and so overruled the address book entry). If even that fails, `"false"` (private letter) is assumed.

1.4 Structuring

The following may stand between the heading and the first following section.

```
<!ENTITY % sectbegin    "(%block;|%blockinline;|%bigblock;)* ">
```

heading

1.4.1 heading

```
<!ELEMENT heading       (#PCDATA|%inline;)*>
<!ATTLIST heading       %basic; %i18n;>
```

All stucturing elements (`<part>`, `<chapter>`, `<section>` etc.) begin with this element. It contains of course the title of that section.

part

1.4.2 part

```
<!ELEMENT part          (heading, chapter*)>
<!ATTLIST part          %basic; %i18n;>
```

Embraces a part of a book.

chapter

1.4.3 chapter

```
<!ELEMENT chapter       (heading, aphorism?, %sectbegin;, section*)>
<!ATTLIST chapter       %basic; %i18n;
                        kind (preface|introduction|
                        acknowledgements|colophon) #IMPLIED>
```

A chapter of a book. It's always included into the TOC (i. e., there is no star'ed version). In `<aphorism>` you can let a nice witty quote by a famous person being printed above the chapter beginning.

`kind` marks special chapters that are not included into the TOC. A `"preface"` chapter is printed *before* the TOC; so far, there is no difference between `"acknowledgements"` and `"colophon"` yet.

section
subsection
subsubsection
paragraph
subparagraph

1.4.4 section, subsection, subsubsection, paragraph and subparagraph

```
<!ELEMENT section       (heading, %sectbegin;, subsection*)>
<!ATTLIST section       %basic; %i18n;>
<!ELEMENT subsection    (heading, %sectbegin;, subsubsection*)>
<!ATTLIST subsection    %basic; %i18n;>
<!ELEMENT subsubsection (heading, %sectbegin;, paragraph*)>
<!ATTLIST subsubsection %basic; %i18n;>
<!ELEMENT paragraph     (heading, %sectbegin;, subparagraph*)>
<!ATTLIST paragraph     %basic; %i18n;>
<!ELEMENT subparagraph  (heading, %sectbegin;)>
<!ATTLIST subparagraph  %basic; %i18n;>
```

These elements can't have aphorisms, apart from that, they are totally analogous to `<chapter>`. `<section>` and `<subsection>` wander in any case in the TOC.

appendix

1.4.5 appendix

```
<!ELEMENT appendix      (chapter*)>
<!ATTLIST appendix      %basic; %i18n;>
```

This element encloses all appendix chapters. They are unusual only as far as their numbering is concerned. So this element is very similar to LaTeX's `appendix` environment.

1.5 Everything that can go into a paragraph

p

1.5.1 p – one paragraph

```
<!ELEMENT p             (#PCDATA|%inline;|%block;)*>
<!ATTLIST p             %basic; %i18n;
                        skip (small|med|big) #IMPLIED>
```

Embraces *one* paragraph. Empty lines produces a warning and are ignored.

multipar

1.5.2 multipar – many paragraphs

```
<!ELEMENT multipar      (#PCDATA|%inline;)*>
<!ATTLIST multipar      %basic; %i18n;>
```

This one can contain plain text and inline elements. The idea is that every '`*`' ends a paragraph and begins a new one, so you don't have to type all this `<p>...</p>` stuff every time. Empty lines are ignored, but produce no warning.

newline

1.5.3 `newline` – line break

```
<!ELEMENT newline       EMPTY>
<!ATTLIST newline       vspace CDATA #IMPLIED>
```

Inserts a line break. In `vspace` you can give a following skip.

footnote

1.5.4 `footnote`

```
<!ELEMENT footnote      (#PCDATA|%inline;)*>
<!ATTLIST footnote      %basic; %i18n;>
```

Inserts a footnote at the current position. It embraces the footnote text.

1.6 Fontchanges

em

1.6.1 `em` – emphasize

```
<!ELEMENT em            (#PCDATA|%inline;)*>
```

Like `\emph{...}` in LaTeX.

visual

1.6.2 `visual` – visual markup

```
<!ELEMENT visual        (#PCDATA|%inline;)*>
<!ATTLIST visual        markup (nm|rm|it|sc|bf|sf|sl|tt|vs) #REQUIRED>
```

Allows font style/variant change. The attribute `markup` is the same as in LaTeX's `\text??` commands. `"nm"` switches to "normal" without changing the family. `"vs"` ("Versaliae") is intended for all-uppercase acronyms like "ENIAC".

1.7 Cross-references

The following is the format of (almost) all cross references. This ensures that the validator has the opportunity to see whether or not a `refid` points to nothing.

```
<!ENTITY % idref        "refid IDREF #REQUIRED">
```

ref

1.7.1 `ref` – basic cross reference

```
<!ELEMENT ref           (#PCDATA|%inline;)*>
<!ATTLIST ref           %idref;>
```

[1] In a letter system that is better configurable than the current, this attribute is used to read in the correct parameters such as the letter head.

This does the same as the LaTeX command `\ref`. The referenced object, that is determined by `refid`, thus must be a figure/table, section etc., simply something that can bear a number. The contents of `<ref>` is the textual label that is put immediatly before it, e. g.

```
<ref refid="MainTable">table</ref>
```

This yields something like "`table~2.1`" for LaTeX, or for HTML "`table 2.1`" which is *completely* displayed as the link, and not only the "`2.1`".

`<ref>`s to equations put automatically parentheses around the number.

vref

1.7.2 `vref` – cross reference with page number

```
<!ELEMENT vref          (#PCDATA|%inline;)*>
<!ATTLIST vref          %idref;>
```

See `<ref>`, but here we use LaTeX's `varioref` package which means that possibly the page number is included into the reference. For HTML, there is no difference between `<ref>` and `<vref>`.

pageref

1.7.3 `pageref` – reference to a page number

```
<!ELEMENT pageref       (#PCDATA|%inline;)*>
<!ATTLIST pageref       %idref;>
```

Inserts the page number of the object with the `id` of `refid`. In HTML a "`[here]`" is inserted (which you can click on), in LaTeX a possibly given contents of the `<pageref>` is inserted directly before the page number. This means that

```
See <pageref refid="GUTformula">page</pageref>.
```

yields "`See page~42`" in LaTeX and "`See [here].`" (clickable `[here]`) in HTML.

cite

1.7.4 `cite` – bibliography reference

```
<!ELEMENT cite          (#PCDATA|%inline;)*>
<!ATTLIST cite          refid NMTOKENS #REQUIRED
                        kind (text|paren|imparen|nocite) #IMPLIED>
```

Inserts a citation reference to `refid` (which can also be a space separated list of more than one citation). The attribute `kind` determines the kind of insertion. `kind="text"` would mean

```
Einstein et. al. (1921)
```

whereas `"paren"` makes

```
(Einstein et. al. 1921)
```

`"imparen"` (“implicit parentheses”) produces

```
Einstein et. al. 1921
```

You know this probably from the `natbib` package, that’s behind all this. *Important*: So that this also works in HTML, the `key` entries of the BibTeX file must look like `"text"`.

`"nocite"` includes nothing in the text, just in the references list at the end. By the way,

```
<cite refid="-" kind="nocite"/>
```

works like `\nocite{*}` LaTeX. (You can’t use ‘`*`’ because `refid` is of type `NMTOKENS`, and such attributes mustn’t contain ‘`*`’.)

The contents of the `<cite>` element becomes the optional parameter of LaTeX’s `\cite` command. Thus

```
<cite refid="Einstein1921"><em>first</em> chapter</cite>
```

yields: “`Einstein et. al. (1921, \emph{first} chapter)`”

The default value for `kind` is `"text"` except where the `<cite>` is directly surrounded by parentheses in the document. Then it’s `"imparen"`.

`mathref`

1.7.5 `mathref` – odd formula reference

```
<!ELEMENT mathref       (#PCDATA|%inline;)*>
<!ATTLIST mathref       refid CDATA #REQUIRED> <!-- "CDATA" to allow
                                                  e.g. "(2.1)" -->
```

See `<ref>`, but `<mathref>` only works for equations that have been labeled using the first column of an `<mlabeledtr>`. Actually such an equation should also have a real `id`, so you don’t need `<mathref>`. It’s just to make MathML support a little bit more complete.

(I needed an extra element for this kind of reference because I really wanted to be able to point to such equations, but for validation issues `refid` can’t be of type `ID`, but must be `CDATA`.)

1.8 Math constructs

`m`

1.8.1 `m` – simple inline math

```
<!ELEMENT m             (#PCDATA)>
<!ATTLIST m             %basic;
                        xmlns CDATA #FIXED "">
```

Corresponds to `$...$` in LaTeX, but uses a somewhat different expression syntax. *Attention:* If it has an `id`, it’s becoming a *displayed* Formula, with a number.

The explicit namespace is for being able to use `<m>` inside `<math>`. It’s not a clean solution, clean would be to

- set the namespace explicitly in the XML document to `""` (at least where it occurs inside MathML), or
- set a `#FIXED` namespace for *all* `tbook` elements.

Additionally, for validation you need a very slightly modified version of MathML's DTD: The parameter entity `%pgenschema;` is extended by `m`, `ch` and `unit`. So the same applies to `<ch>` and `unit`.

In principle, you can use `<m>` wherever you can use `<mrow>`. However, sometimes it may be necessary to enclose `<m>` and friends with `<mrow>`, because they may expand to multiple MathML elements.

dm

1.8.2 `dm` – simple display math

```
<!ELEMENT dm           (#PCDATA)>
<!ATTLIST dm           %basic;>
```

Corresponds to `\[...\]` in LaTeX, but uses a somewhat different expression syntax.

ch

1.8.3 `ch` – chemical formula

```
<!ELEMENT ch           (#PCDATA)>
<!ATTLIST ch           %basic;
                       display (inline|block) #IMPLIED
                       xmlns CDATA #FIXED "">
```

Corresponds to `<m>` or `<dm>` (according to `display`), but produces chemical formulas with e. g. unslanted chemical element symbols. E. g.

```
<ch>Al_xGa_{1-x}As</ch>
```

yields "$\mathrm{Al}_x\,\mathrm{Ga}_{1-x}\,\mathrm{As}$", i. e. tiny skips between the elements, upshape elements, but all subscripts are treated as mathematics.

For the explicit namespace see `<m>` above. It makes it possible to be used inside MathML's `<math>`.

theorem

1.8.4 `theorem` – mathematical theorem etc.

```
<!ELEMENT theorem      (#PCDATA|heading|subject|%inline;|%block;)*>
<!ATTLIST theorem      %basic; %i18n;
                       countlike CDATA #IMPLIED
                       layout CDATA #IMPLIED>
```

Inserts a (mathematical) theorem, corollar, lemma, definition, or a remark, a note, an exercise, or an example. It is allowed everywhere where you could insert a floating figure.

For this element, the `class` attribute is vital. It is interpreted as the kind of the theorem. E. g., you could say

```
<theorem class="Remark">A short mathematical remark.</theorem>
```

If `class` is ommitted a default is used.

The `countlike` contains another `<theorem>` class with which the current one should share numbering. Otherwise, every `<theorem>` class gets its own counting. For example,

```
<theorem class="Corollary" countlike="Lemma">
  <subject>Streetmentioner's Corollary</subject>
  A short mathematical corollary.  If it's corollary
  number 4, the next lemma will have number 5.
</theorem>
```

If you want to supress counting, set `countlike` to `"(none)"`. If you don't want to have the chapter number included into the theorem number, set `countlike` to `"(global)"`.

The name of the theorem is the same as its class. If you want another name, include a `<heading>` element.

Within `<subject>`...`</subject>`, you can include e. g. a special name for the theorem. This is printed within brackets after the word “Theorem” (or whatever).

With the `layout` attribute, you can choose another style. Prefefined are the $\mathcal{A}\!\mathcal{M}\!\mathcal{S}$-TEX styles `plain`, `definition`, and `remark`. If you want to add another one, you have to give its definition in a LaTeX package.

Important: Only the very first occurence of a certain `<theorem>` class is allowed to have `<heading>`, `countlike` or `layout`.

proof

1.8.5 proof – mathematical proof

```
130 <!ELEMENT proof          (#PCDATA|heading|%inline;|%block;)*>
131 <!ATTLIST proof          %basic; %i18n;>
```

Inserts a mathematical proof. It is allowed everywhere where you could insert a floating figure.

If `<heading>` is given, it is used instead of the default word “Proof” at the beginning.

1.8.6 Use of simple math constructs

`<m>` and friends use simplified LaTeX syntax. For example, matrices don't work (yet). But you may use roots, fractions, standard functions like “sin”, stretchable braces, sub- and superscripts and accents. You may nest these structures so deep until your XSLT processor complains. And you can use these elements *inside* MathML to get the best of both worlds. (However HTML output only contains valid MathML.)

Roots work with `\sqrt`, fractions with `\frac`, just as in LaTeX. Human text is included via `\text` that is known from AMS-TeX. Standard functions are typed without a `\`. Stretchable braces are all braces immediately within a `{...}` grouping. Sub- and superscrips as in LaTeX, but always a possible subscript *before* the superscript. Accents are just written *immediately* before the accented variable or group, they're made wide accents if necessary. If you make a space between accent and anything that follows, the accent is treated as an operator. (So, a `\vec` becomes a `\to`.)

Here an example:

```
<m>&Hat;{1-x_{\text{eff}}} &ne; {( &int;_0^&infin; sin(&tilde;x)
                                   \frac{\sqrt[3]{1/e}}&beta; dx )}
                              &ne; lim_{x &rarr; &infin;}\frac1x</m>
```

becomes

$$1 \widehat{-x}_{\text{eff}} \neq \left(\int_0^\infty \sin(\tilde{x}) \frac{\sqrt[3]{1/e}}{\beta} dx \right) \neq \lim_{x\to\infty} \frac{1}{x}$$

For HTML output, the responsible stylesheet produces:

```
<mrow><mtext>eff</mtext></mrow></msub></mrow>
<mo>&Hat;</mo></mover><mo>&ne;</mo><mrow><mo>(</mo>
<munderover><mo>&int;</mo><mn>0</mn><mo>&infin;</mo></munderover>
<mi>sin</mi><mo stretchy="false">(</mo><mover accent="true">
<mi>x</mi><mo>~</mo></mover><mo stretchy="false">)</mo>
<mfrac><mrow><mroot><mrow><mn>1</mn><mo>/</mo><mi>e</mi></mrow>
<mn>3</mn></mroot></mrow><mi>&beta</mi></mfrac><mi>d</mi><mi>x</mi>
<mo>)</mo></mrow><mo>&ne;</mo><munder><mi>lim</mi><mrow><mi>x</mi>
<mo>&rarr;</mo><mo>&infin;</mo></mrow></munder><mfrac><mn>1</mn>
<mi>x</mi></mfrac></math>
```

Lucky us.

Notice that you can use `<m>`, `<ch>` and `<unit>` *within* MathML constructs. If you use these elements within a MathML equation *array*, you can generate alignment markers (in LaTeX known as '`&`' signs) with '`#`' signs, because this is shorter than `&amp;`. Put them where they would be in LaTeX. Although MathML requires such a marker at the very beginning of an equation row, this is not true for LaTeX, and not true for **tbook**.

1.8.7 MathML

In this context some words about MathML. You may use it if you want, for more complicated formulas you must use it, unfortunately. It's always enclosed by `<math>...</math>`, but only presentation markup can be transformed to LaTeX yet.

Equation array: **tbook** treats a `<math>` element as an equation array, if it consists of only *one* `<mtable>`, with a `groupalign` attribute *or* one or more `<mlabledtr>` rows. If you set `groupalign="right center left"`, this leads to an eqnarray in LaTeX, and where '&' are in LaTeX, you have to use <maligngroup/> in MathML. Else the equations are just stacked and not aligned. But as already mentioned, you can also use <m> inside <math>, which is very helpful for equation arrays:

```
<math>
  <mtable groupalign="right center left">
    <mtr>
      <mtd id="test"> <m> 1+1 #=# 2 </m> </mtd>
      <mtd> <m> 4 #=# 2 &CenterDot; 2 </m> </mtd>
    </mtr>
  </mtable>
</math>
```

which is the same as LaTeX's

```
\begin{eqnarray}
  1+1 &=& 2 \label{Test} \\
  4   &=& 2 \cdot 2 \nonumber \\
\end{eqnarray}
```

and you don't want to see the HTML/MathML output **tbook** must create for that. By the way, being the only child element of an <mtd>, <m> is implicitly surrounded by an <mrow> which is necessary in this context.

Equation numbers: A tricky point is equation labelling and numbering. **tbook** supports three ways of giving an equation a label:

1. A <math> element has an id attribute. Plain and simple.
2. An <mtd> element with an id within an equation array (see above).
3. An <mlabledtr> element with an id within an equation array, and the contents of the first <mtd> element of such an <mlabeledtr> row.

You may use the <ref> element to refer to such equations, but for the contents of the <mtd> element in the third case, you have to use <mathref>. I would recommend you to use only 1. and 2. See the MathML specs at the W3C for more information.

132

1.9 Miscellaneous

url

1.9.1 url

```
133 <!ELEMENT url            EMPTY>
134 <!ATTLIST url            name CDATA #REQUIRED>
```

The name in printed with typewriter style and is linked with its own contents. Just as \url{...} would do in hyperref mode.

hspace

1.9.2 `hspace` – horizonal skip

```
<!ELEMENT hspace        EMPTY>
<!ATTLIST hspace        dim CDATA #REQUIRED>
```

Makes a horizontal skip, as the LaTeX macro of the same name.

```
<hspace dim="1em"/>
```

should do the same as one `\quad`.

relax

1.9.3 `relax`

```
<!ELEMENT relax         EMPTY>
```

Does nothing. Had a meaning between `<cite>`s in former times, when `<cite>` was defined differently. But I didn't want to abandon it. Who knows what it still can be good for. And I dislike input languages without a `<relax>`.

unit

1.9.4 `unit` – physical quantity

```
<!ELEMENT unit          (#PCDATA)>
<!ATTLIST unit          xmlns CDATA #FIXED "">
```

Inserts a physical quantity.

```
<unit>3 m</unit>
```

yields in LaTeX "`$3$\,m`". So it guarantees a neat skip between number and unit, and for configurations with different fonts for number in- and outside mathematics it chooses the correct one. Further advantage: Things like

```
The gravitational constant is
<unit>6.672&middot;10^{-11} m^3 kg^{-1} s^2</unit>.
```

(notice the spaces!) yields

The gravitational constant is $6.672 \cdot 10^{-11}\,\mathrm{m}^3\,\mathrm{kg}^{-1}\,\mathrm{s}^2$.

So, units in upshape with small skips inbetween. You must assure that the *first* space is between the number and the unit, or alternatively you must put a "~" between number and unit.

For the explicit namespace see `<m>` above. It makes it possible to be used inside MathML's `<math>`.

latex

1.9.5 latex – direct LaTeX input

```
<!ELEMENT latex         ANY>
<!ATTLIST latex         code CDATA #REQUIRED
                        desperate (true|false) "false">
```

This is some sort of `\special` command: It inserts `code` if we're producing LaTeX output, and interprets the element's contents else. It's totally ignored if `desperate` is `"true"`.

The idea behing `desperate` is that the trafo **tbook** → LaTeX can be forced to interpret even `<latex>`es with `deperate="true"`. This is usually done in the last phase of production, maybe for having better page breaks. Another example is the "LaTeX" logo which can be achieved by

```
<!ENTITY LaTeX "<latex code='\LaTeX{}'>LaTeX</latex>">
```

in the XML preamble and using it with "`&LaTeX;`".

wrap

1.9.6 wrap – inline wrapper

```
<!ELEMENT wrap          (#PCDATA|%inline;)*>
<!ATTLIST wrap          %basic; %i18n;>
```

This doesn't format, it encloses inline elements that then get an `id` or a language via `xml:lang`. It can be useful empty. For example

```
<wrap id="NicePosition"/>
```

is equal to LaTeX's `\label{NicePosition}`.

1.10 Quoted and verbatim material

quote

1.10.1 quote

```
<!ELEMENT quote         (#PCDATA|%inline;)*>
<!ATTLIST quote         %basic; %i18n;>
```

Inserts a displayed quotation.

verb

1.10.2 verb – preformatted material

```
<!ELEMENT verb          (#PCDATA)>
```

Basically the same as LaTeX's `\verb`.

verbatim

1.10.3 `verbatim` – preformatted material

```
<!ELEMENT verbatim      (#PCDATA|em|visual|%index;)*>
<!ATTLIST verbatim      %basic; %i18n;>
```

Corresponds to LaTeX's `verbatim` environment. In this context XML's `<![CDATA[...]]>` is sometimes useful. Please note that – in contrast to LaTeX – some formatting elements are allowed. So you may print things in bold face, for example.

verse

1.10.4 `verse`

```
<!ELEMENT verse         (#PCDATA|%inline;)*>
<!ATTLIST verse         %basic; %i18n;>
```

Formats its contents in a way that line breaks are conserved, which is significant for e. g. lyrics.

aphorism

1.10.5 `aphorism`

```
<!ELEMENT aphorism      (#PCDATA|%inline;|caption)*>
<!ATTLIST aphorism      %basic; %i18n;>
```

Embraces a little witty quote for the beginning of a chapter. `<caption>` contains the origin, typically the name of a more or less famous person. There must be up to *one* `<caption>` and it must come *last* within `<aphorism>`.

1.11 Lists

description

1.11.1 `description` – glossary like list

```
<!ELEMENT description   ((term,item)*)>
<!ATTLIST description   %basic; %i18n;>
```

A glossary like list of `<term>`–`<item>` pairs.

enumerate

1.11.2 `enumerate` – numbered list

```
<!ELEMENT enumerate     (item*)>
<!ATTLIST enumerate     %basic; %i18n;>
```

A numbered list of `<item>`s.

itemize

1.11.3 `itemize` – list with bullets

```
<!ELEMENT itemize       (item*)>
<!ATTLIST itemize       %basic; %i18n;>
```

A not numbered list of `<item>`s.

term

1.11.4 `term` – list element

```
<!ELEMENT term          (#PCDATA|%inline;)*>
<!ATTLIST term          %basic; %i18n;>
```

The term that sould be explaind by the respective `<item>` within a `<descrition>`.

item

1.11.5 `item` – list element

```
<!ELEMENT item          (#PCDATA|%inline;|%block;)*>
<!ATTLIST item          %basic; %i18n;>
```

One item of a list. (See above for the three types of lists.)

1.12 "Floats" and their contents

figure

1.12.1 `figure`

```
<!ELEMENT figure        (graphics,caption?)>
<!ATTLIST figure        %basic; %i18n;>
```

Encloses a figure and possibly a caption, that is then places as a (numbered) float object. If you want to refer to a graphics by an `id`, give the `<figure>` element that `id`.

table

1.12.2 `table`

```
<!ELEMENT table         (tabular,caption?)>
<!ATTLIST table         %basic; %i18n;>
```

Encloses a table (`<tabular>`) and possibly a caption, that is then places as a (numbered) float object. If you want to refer to a table by an `id`, give the `<table>` element that `id`.

graphics

1.12.3 `graphics`

```
<!ELEMENT graphics      (psfrag*)>
<!ATTLIST graphics      %basic; %i18n;
                        file     CDATA #REQUIRED
                        scale    CDATA #IMPLIED
                        basefontsize CDATA #IMPLIED
                        kind (vector|bitmap|overlay|diagram) #REQUIRED>
```

This includes a graphics that is taken from `file` (without file name extension). `kind` is interpreted as follows:

`"vector"` is an EPS file with correct bounding box.

`"bitmap"` is a JPEG bitmap with correct dpi resolution information.

`"overlay"` is a JPEG bitmap like `"bitmap"` with an equally big EPS vector image that is printed over the bitmap as a label layer. The EPS file has the file name `file` plus an '`l`'.

`"diagram"` is a LaTeX fragment read in directly. It may be e. g. Gnuplot output.

Eventually the XML processor must see how to interpret `kind`. I explain here the way my current **tbook** tools go. `kind` could be made implicit if the processor can guess which graphics type is meant, but I find it clearer this way.

`basefontsize` may be `"10pt"`, `"11pt"` or `"12pt"`. Sometimes one changes the global font size in a document which may make all Psfrag labels look ugly, namely too big or too small. Or one graphics migrate from one document to another with a different main font size. With `basefontsize` you can switch locally to the old font size. Of course, you can also use this attribute to change the label size for a certain graphics.

`caption`

1.12.4 `caption`

```
<!ELEMENT caption       (#PCDATA|%inline;)*>
<!ATTLIST caption       %basic; %i18n;>
```

The caption of a float object or the origin of an aphorism. If you use it within a float, i. e. a `<figure>` or a `<table>`, you can leave it empty; in this case the float only gets a number. It doesn't get a number just because it has an `id`!

`psfrag`

1.12.5 `psfrag` – graphics label creation

```
<!ELEMENT psfrag        (#PCDATA|%inline;)*>
<!ATTLIST psfrag        %basic; %i18n;
                        tag CDATA #REQUIRED
                        number (true|false) #IMPLIED
                        contrast (boxed|inverse) #IMPLIED
                        relsize (normal|large|small) "normal"
                        interval CDATA #IMPLIED
                        align (left|center|ccenter|right) "left">
```

The `tag` is searched in the surrounding vector graphics and the contents of the `<psfrag>` is substituted for it:

```
<psfrag tag="x-axis"><m>x</m>-axis</psfrag>
```

For this I use of course the fantastic Psfrag package. Is `<psfrag>` empty, `tag` is substituted for itself, which means that only font and size is adjusted to the main document:

```
<psfrag tag="Diagram"/>
```

If you want to erase something from an image, you have to replace it with white-space:

```
<psfrag tag="was dull"> </psfrag>
```

`align` determines alignment relatively to the replaced `tag`:

`"left"`: left on the same baseline.

`"right"`: right on the same baseline.

`"center"`: centered on the same baseline.

`"ccenter"`: horizontally and vertically centered.

`number` is `"true"` by default, if `tag` is obviously a number, or if `interval` is given, else `"false"`.

`constrast="boxed"` sets the substitution on a white rectangle. `constrast="inverse"` shows the substitution in white colour. One of both may be necessary for too dark/chaotic backgrounds.

`relsize` should be clear.

`interval` consists, if given, of three semicolon separated numbers: start, end and step. Thus `interval="0;10;1"` yields automatically Psfrag substitutions for all numbers between 0 and 10. This is very convenient for EPS files with a labeled axis. By the way, `tag` plays in this case the role of a pattern for the numbers in the EPS file. This works somehow accoring to the DecimalFormat routine of Java 1.1, if anybody knows this.

Some examples:

```
<psfrag tag="#" interval="1;10;2"/>
```

replaces 1, 3, 5, 7 und 9 by itself (i. e., it changes the font only). Now for something more complicated:

```
<psfrag tag="#.0" interval="-4.5;-6;-0.5">#,0</psfrag>
```

replaces `-4.5`, `-5.0`, `-5.5` and `-6.0` by the same numbers, but with a comma instead of a point (for our non-English friends). Additionally,

```
<psfrag tag="#.0" interval="-4.5;-6;-0.5">#,#</psfrag>
```

does the same, but in the output post-comma digits (and the comma) are omitted where they are zero anyway. Last example:

```
<psfrag tag="0.0" interval="-1.5;1;0.5">#,#</psfrag>
```

does the following substitutions: `-1.5` $\rightarrow$ −1,5, `-1.0` $\rightarrow$ −1, `-0.5` $\rightarrow$ −0,5, `0.0` $\rightarrow$ 0, `0.5` $\rightarrow$ 0,5 and `1.0` $\rightarrow$ 1.

1.13 Tabular material

tabular

1.13.1 `tabular` – an actual table

```
<!ELEMENT tabular       (tabhead?,tabbody)>
<!ATTLIST tabular       %basic; %i18n;
                        preamble CDATA  #IMPLIED>
```

The `preamble` looks like a simple LaTeX tabular preamble. `"lcc"` means “three columns, one left aligned, then two centered”. Except `l`, `r` and `c` nothing is allowed (yet), in particular no vertical bars, because in very most cases they are bad style.

Is no preamble given, `c` for all columns is assumed.

Attention: In a tabular, the three standard horizontal lines of LaTeX’s `booktabs` package are always present, so you don’t have (must not) add them by yourself.

tabhead

1.13.2 `tabhead` – header of a table

```
<!ELEMENT tabhead       (hline|row|srow)*>
```

Here every column gets a description.

tabbody

1.13.3 `tabbody` – table’s main part

```
<!ELEMENT tabbody       (hline|row|srow)*>
```

Contains the actual data rows of a table.

row

1.13.4 `row`

```
<!ELEMENT row           (cell)*>
<!ATTLIST row           %basic; %i18n;>
```

One row in a table.

srow

1.13.5 `srow` – simple table row

```
<!ELEMENT srow          (#PCDATA)>
<!ATTLIST srow          %basic; %i18n;>
```

One table row. The columns are separated by ‘`|`’ characters. This is intended for simple rows that don’t need special formatting. It saves you from typing this `<cell>...</cell>` stuff.

hline

1.13.6 `hline` – horizontal table line

```
<!ELEMENT hline         EMPTY>
<!ATTLIST hline         from  NMTOKEN  "1"
                        to    NMTOKEN  #IMPLIED
                        trim  (lr|l|r|no)  #IMPLIED>
```

Inserts a horizontal line in a table. `from` is the starting column, `to` the ending column. By default, a line spans the whole width. Also by default, a line ending is trimmed (shortened) if the line ends *within* the table. The `trim` can change this. `"lr"` means “trim left and right”, `"l"`, `"r"` accordingly, and `"no"` means “keep full length under all circumstances”.

`cell`

1.13.7 `cell`

```
<!ELEMENT cell          (#PCDATA|%inline;)*>
<!ATTLIST cell          %basic; %i18n;
                        colspan NMTOKEN "1"
                        align   (left|center|right) #IMPLIED>
```

Encloses one rectangular row/column field in a table. `colspan` corresponds to the `\multicolumn` macro in LaTeX, and `align` is clear, I think.

1.14 Things that go in the frontmatter

`title`

1.14.1 `title`

```
<!ELEMENT title         (#PCDATA|%inline;|newline)*>
<!ATTLIST title         %basic; %i18n;>
```

Title of the document. Appears on the title page and in the title bar of the browser window.

`author`

1.14.2 `author`

```
<!ELEMENT author        (#PCDATA|newline|footnote)*>
<!ATTLIST author        %basic; %i18n;>
```

Name of *one* author. It is automatically split up into first- and lastname, so that it can be mirrored for some cases. The point between first- and lastname is normally the *last* space, but if you give an explicit '`|`', this is used for the distiction.

`<newline>` and `<footnote>` are interpreted only for `<article>`, for `<book>` only the first text node is used.

`subtitle`

1.14.3 `subtitle`

```
<!ELEMENT subtitle      (#PCDATA|%inline;|newline)*>
<!ATTLIST subtitle      %basic; %i18n;>
```

Works like the title and is used for the title page.

`typeset`

1.14.4 `typeset` – the artist

```
<!ELEMENT typeset       (#PCDATA)>
<!ATTLIST typeset       %basic; %i18n;>
```

Name of the typesetter and maybe also the used font, program (TeX) etc.

date

1.14.5 date

```
<!ELEMENT date          (#PCDATA)>
<!ATTLIST date          %basic; %i18n;>
```

Date of print. The format is free, i. e. it will we printed unchanged as you've give it here.

keywords

1.14.6 keywords

```
<!ELEMENT keywords      (#PCDATA)>
<!ATTLIST keywords      %basic; %i18n;>
```

Keywords decribing the document. This element is exclusively used for meta information. For HTML, it creates `<meta>` elements, in PDF files it's used for the Acrobat Reader "Summary" field.

year

1.14.7 year

```
<!ELEMENT year          (#PCDATA)>
<!ATTLIST year          %basic;>
```

Year(s) for the copyright. If you omit it, the **tbook** stylesheets use the current year.

city

1.14.8 city

```
<!ELEMENT city          (#PCDATA)>
<!ATTLIST city          %basic; %i18n;>
```

City of coming into existence. Also for the copyright. For a letter, this is the city printed directly before the date in the header.

legalnotice

1.14.9 legalnotice

```
<!ELEMENT legalnotice   (#PCDATA|%inline;|p)*>
<!ATTLIST legalnotice   %basic; %i18n;>
```

A disclaimer or things like that. If you give it as an empty element `<legalnotice/>`, a default legal notice is used, which you may not agree with.

abstract

1.14.10 abstract

```
<!ELEMENT abstract      (p+)>
<!ATTLIST abstract      %basic; %i18n;>
```

Summary of an *article*.

1.15 Bibliography

references

1.15.1 references

```
<!ELEMENT references    (%block;|%blockinline;)*>
<!ATTLIST references    %basic; %i18n;
                        bibfile CDATA #IMPLIED>
```

Inserts a references list. The contents of this element is printed as a preamble to it.

`bibfile` denotes the bibliography file. A possibly included file name extension is ignored. The default is the value of `bib-filename`, a parameter that can be given to the XSLT stylesheet when calling the XSLT processor. If even that is not given, `"biblio"` is used.

1.16 Index

The following three shortcuts are used in the intex related elements.

```
<!ENTITY % sortingkey "sortkey CDATA #IMPLIED">
<!ENTITY % indexstyle "kind (emph|bold|italic|start|end) #IMPLIED">
<!ENTITY % indexinline "%fontchange;|m|%blockinline;|%miscinline;|
                        graphics">
```

The attribute `sortkey` is the explicit sorting key. For MakeIndex, this was the text before the '`@`'. `kind` was in MakeIndex everything after a possible '`|`'. This changes the formatting of the page number for that entry, or it starts or ends a text reagion that should completely be connected with the index entry term. `"emph"` and `"italic"` will probably mean the same, their use is a matter of taste. But see below for examples.

index

1.16.1 index

```
<!ELEMENT index         (%block;|%blockinline;)*>
<!ATTLIST index         %basic; %i18n;>
```

Inserts an index. The contents of this element is printed as a preamble to it.

ix
ix2

1.16.2 ix and ix2 – index entry

```
<!ELEMENT ix            (#PCDATA|%indexinline;|ix2)*>
<!ATTLIST ix            %i18n; %sortingkey; %indexstyle;>
<!ELEMENT ix2           (#PCDATA|%indexinline;)*>
<!ATTLIST ix2           %sortingkey;>
```

One index entry. There mustn't be more than one `<ix2>` element within `<ix>`, and that must come last.

You don't have to watch out for special characters. Everything is escaped if necessary. Most latin letters with diacritic symbols are sorted properly.

The optional <ix2> contains the entry on the second level. To sum it up, the old MakeIndex commad

```
\index{S"anger!Twopac@2~Pac|emph}
```

looks in **tbook** like this:

```
<ix kind="emph">Sänger<ix2 sortkey="Twopac">2&nbsp;Pac</ix2></ix>
```

(Sänger = singer in German.)

idx

1.16.3 idx – simple index entry

```
<!ELEMENT idx          (#PCDATA|%indexinline;)*>
<!ATTLIST idx          %i18n; %sortingkey; %indexstyle;>
```

Does the same as <ix>, but aditionally inserts its contents at the current text position.

indexsee

1.16.4 indexsee – cross reference within index

```
<!ELEMENT indexsee     (ix,ix+)>  <!-- Actually (ix+,ix), but it
                                       doesn't want it -->
```

This creats cross referneces within the index. Apparently it has to contain at least two <ix> elements. The last in the row is allways the one all the others are pointing to.

1.17 Letter elements

to

1.17.1 to – recipient

```
<!ELEMENT to           (#PCDATA|%inline;|newline)*>
<!ATTLIST to           %basic; %i18n;
                       nickname NMTOKEN #IMPLIED>
```

Encloses the recipient. Single lines can be separated with <newline/>s. If the nickname attribute is given *and* there is contents, the contents of this element has higher priority. So, if you want to use the address book, leave it empty:

```
<to nickname="Knuth"/>
```

subject

1.17.2 subject

```
<!ELEMENT subject      (#PCDATA|%inline;)*>
<!ATTLIST subject      %basic; %i18n;
                       silent (true|false) #IMPLIED>
```

The subject of a letter. This element is mandatory, but is suppressed for informal letters (attribute formal="false"). For formal letters, it's printed. With silent you may change that behaviour explicitly.

opening

1.17.3 opening – begin of letter

```
<!ELEMENT opening       (#PCDATA|%inline;)*>
<!ATTLIST opening       %basic; %i18n;>
```

Corresponds to `\opening{...}` in LaTeX. It begins the letter text.

closing

1.17.4 closing – end of letter

```
<!ELEMENT closing       (#PCDATA|%inline;)*>
<!ATTLIST closing       %basic; %i18n;
                        kind (above|below|signature) #IMPLIED>
```

It ends the letter with "Sincerely yours ..." or something like that.

The default for `kind` is `"signature"` for formal letters and `"above"` else. `"above"` means that the contents of `<closing>` is printed above the space for the signature, `"below"` means, well, below, and `"signature"` means above, but with the name of the author below.

So, if you only want to have your name below the signature, you have to say:

```
<closing kind="signature"/>
```

```


```

1.18 Basic XML entities

These entities must be declared in a complete XML DTD.

```
<!ENTITY lt             "&#60;">     <!-- "<" -->
<!ENTITY gt             "&#62;">     <!-- ">" -->
<!ENTITY amp            "&#38;#38;"> <!-- "&" -->
<!ENTITY apos           "&#39;">     <!-- "'" -->
<!ENTITY quot           "&#38;#34;"> <!-- '"' -->

```

1.19 Obsolete code

This code was important at a time when I used alternative entity definitions to map them to LaTeX constructs. As I use an external filter program now, this here is pointless.

```
<!ENTITY % tblatex "IGNORE">
<!ENTITY % tblatex.ent SYSTEM "tblatex.ent">
<![ %tblatex; [
 %tblatex.ent;
]]>

<!--==========================-->
<!--Ersatz Basic XML entities --> <!-- obsolete -->
<!--==========================-->
```

```
<!--ENTITY lt              "<"-->  <!-- no ersatz for this -->
<!--ENTITY gt              ">"-->  <!-- no ersatz for this -->
<!--ENTITY et              "&amp;">
<!ENTITY dq             "&quot;">
<!ENTITY backslash      "\">
<!ENTITY dollar         "$">
<!ENTITY number         "#">
<!ENTITY caret          "^">
<!ENTITY underscore     "_">
<!ENTITY percent        "&#37;"-->


```

1.20 Inclusion of external DTDs and entity files

First, I include **t**book's standard entities (see section below).

```
<!ENTITY % tbook.ent SYSTEM "tbook.ent">
 %tbook.ent;

```

Unfortunately there is nothing comparable to SGML catalog files in the XML world – yet. Till it comes, I have to comment out the following clean code — (proceed below)

```
<!-- Until we have no XML Catalog ...
<!ENTITY % MathML.dtd PUBLIC "-//W3C//DTD MathML 2.0//EN"
                             "mathml2.dtd" [
<!ENTITY % pgenschema
     "%mrow.qname; | %mfrac.qname; | %msqrt.qname; | %mroot.qname;
      | %menclose.qname; | %mstyle.qname; | %merror.qname;
      | %mpadded.qname; | %mphantom.qname; | %mfenced.qname;
      | m | ch | unit" >
]>

 %MathML.dtd;

<!ENTITY % HTMLlat1 PUBLIC "-//W3C//ENTITIES Latin 1//EN//HTML"
                           "HTMLlat1.ent">
 %HTMLlat1;

<!ENTITY % HTMLsymbol PUBLIC "-//W3C//ENTITIES Symbols//EN//HTML"
                             "HTMLsymbol.ent">
 %HTMLsymbol;

<!ENTITY % HTMLspecial PUBLIC "-//W3C//ENTITIES Special//EN//HTML"
                              "HTMLspecial.ent">
 %HTMLspecial;

 and instead: -->

```

— and have to use this rubbish instead. In the file `hmml2dst.dtd` are all entities from MathML and HTML together with the complete MathML 2 DTD. It's pathetic, but I don't have to bother about path names so much now.

```
<!ENTITY % MathML-and-HTML-distillated.dtd
       PUBLIC "-//Torsten Bronger//DTD MathML 2.0 distillated and HTML 4.0 entities//EN"
              "hmml2dst.dtd">
%MathML-and-HTML-distillated.dtd;
⟨/tbookdtd⟩
```

2 Human languages

tbook supports four human languages so far.

`"en"`	English
`"de"`	German
`"fr"`	French
`"it"`	Italian
`"en-US"`	American
`"en-GB"`	Britisch
`"de-DE"`	FR German
`"de-AT"`	Austrian
`"de-1901"`	German, trad. orthography
`"de-1996"`	German, new orthography
`"de-DE-1901"`	FR German, trad. orthography
`"de-AT-1901"`	Austrian, trad. orthography
`"de-DE-1996"`	FR German, new orthography
`"de-AT-1996"`	Austrian, new orthography

(Language tags are case in-sensitive.) If a subtag is omitted, the LaTeX output stylesheet tries to guess the subvariant by a primitive heuristic method. So please give it as explicitly as you can. The standard fallbacks are `"de-DE-1996"` and `"en"`, the latter being interpreted as "American" by LaTeX and probably most browsers.

The German year subtags are still non-standard, but the odds are that they become standard one day. (I introduced them in the responsible IETF mailing list. Unfortunately the process somewhat inert.)

2.1 The decimal point

Where English people write "3.5", Germans and Frechmen behave according to ISO and write "3,5". In numerous places this must be payed attention to, e. g. for the default of the `number` attribute in `<psfrag>`, the output of numbers in LaTeX formulas (the `{,}` thing) or the decision whether an `<m>` contents token should be `<mi>` or `<mn>` in MathML output. tbook does so.

3 The BibTEXML DTD

Warning! There are two ways of bringing a bibliography in **tbook** files, this sections refers to the *old* way. This way has some advantages, however at the moment I recommend to use just `makebib`, which simply calles BibTEX, for both LaTEX and HTML. I. e., you don't need to create bibliography XML files.

This is basically the same DTD as created by the BibTEXML project, so I have to refer to their project homepage at `http://bibtexml.org`.

I changed two things:

1. `xml:lang` for bibliography and bibitem (bibliography: `'en'` default)
2. All elements are in a unique namespace `http://bibtexml.org/STSCHEMA`

```
⟨*tbbibdtd⟩
<!-- PUBLIC "-//Torsten Bronger//DTD bibliography 1.0//EN" -->
<!ENTITY % bibtexml.xmlns "http://bibtexml.org/STSCHEMA">
<!ENTITY % bibtexml.xmlns.attrib
     "xmlns        CDATA          #FIXED '%bibtexml.xmlns;'">

```

`<bibliography>` is the root element of a BibTEXML file.

```
<!ELEMENT bibliography (macro | preamble | bibitem | comment)*>
<!ATTLIST bibliography
        %bibtexml.xmlns.attrib;
        version CDATA #IMPLIED
        author CDATA #IMPLIED
        created CDATA #IMPLIED
        lastmodified CDATA #IMPLIED
        xml:lang NMTOKEN "en"
>
<!ELEMENT macro (#PCDATA | tex)*>
<!ATTLIST macro
        %bibtexml.xmlns.attrib;
        alias CDATA #REQUIRED
>
<!ELEMENT preamble (#PCDATA)>
<!ATTLIST preable
        %bibtexml.xmlns.attrib;
>
<!ELEMENT comment (#PCDATA)>
<!ATTLIST comment
        %bibtexml.xmlns.attrib;
>
<!ELEMENT bibitem (address?,annote?, author*, booktitle?, chapter?, crossref?,
edition?, editor?, howpublished?, institution?, journal?, key?, month?, note?, number?,
organization?, pages?, publisher?, school?, series?, title?, type?, volume?, year?,
notstandard*,URL?,ISSN?,ISBN?,abstract?,contents?)*>
```

Every book/article in a BibTeXML file is enclosed by a `<bibitem>` element.

```
<!ATTLIST bibitem
        %bibtexml.xmlns.attrib;
        type CDATA #REQUIRED
        label ID #REQUIRED
        xml:lang NMTOKEN #IMPLIED
>
<!ENTITY % entry "(#PCDATA | tex | abbrev)*">
<!ELEMENT author ANY>
<!ATTLIST author %bibtexml.xmlns.attrib;>
<!ELEMENT address %entry;>
<!ATTLIST address %bibtexml.xmlns.attrib;>
<!ELEMENT annote %entry;>
<!ATTLIST annote %bibtexml.xmlns.attrib;>
<!ELEMENT booktitle %entry;>
<!ATTLIST booktitle %bibtexml.xmlns.attrib;>
<!ELEMENT chapter %entry;>
<!ATTLIST chapter %bibtexml.xmlns.attrib;>
<!ELEMENT crossref %entry;>
<!ATTLIST crossref %bibtexml.xmlns.attrib;>
<!ELEMENT edition %entry;>
<!ATTLIST edition %bibtexml.xmlns.attrib;>
<!ELEMENT editor %entry;>
<!ATTLIST editor %bibtexml.xmlns.attrib;>
<!ELEMENT howpublished %entry;>
<!ATTLIST howpublished %bibtexml.xmlns.attrib;>
<!ELEMENT institution %entry;>
<!ATTLIST institution %bibtexml.xmlns.attrib;>
<!ELEMENT journal %entry;>
<!ATTLIST journal %bibtexml.xmlns.attrib;>
```

The `<key>` element must always be set and must be of the form “`Author (year)`”.

```
<!ELEMENT key %entry;>
<!ATTLIST key %bibtexml.xmlns.attrib;>
<!ELEMENT month %entry;>
<!ATTLIST month %bibtexml.xmlns.attrib;>
<!ELEMENT note %entry;>
<!ATTLIST note %bibtexml.xmlns.attrib;>
<!ELEMENT number %entry;>
<!ATTLIST number %bibtexml.xmlns.attrib;>
<!ELEMENT organization %entry;>
<!ATTLIST organization %bibtexml.xmlns.attrib;>
<!ELEMENT pages %entry;>
<!ATTLIST pages %bibtexml.xmlns.attrib;>
<!ELEMENT publisher %entry;>
<!ATTLIST publisher %bibtexml.xmlns.attrib;>
<!ELEMENT school %entry;>
<!ATTLIST school %bibtexml.xmlns.attrib;>
<!ELEMENT series %entry;>
<!ATTLIST series %bibtexml.xmlns.attrib;>
```

```
<!ELEMENT title %entry;>
<!ATTLIST title %bibtexml.xmlns.attrib;>
<!ELEMENT type %entry;>
<!ATTLIST type %bibtexml.xmlns.attrib;>
<!ELEMENT volume %entry;>
<!ATTLIST volume %bibtexml.xmlns.attrib;>
<!ELEMENT year %entry;>
<!ATTLIST year %bibtexml.xmlns.attrib;>

<!-- not bibtex element-->
<!ELEMENT abstract %entry;>
<!ATTLIST abstract %bibtexml.xmlns.attrib;>
<!ELEMENT contents %entry;>
<!ATTLIST contents %bibtexml.xmlns.attrib;>
<!ELEMENT ISBN %entry;>
<!ATTLIST ISBN %bibtexml.xmlns.attrib;>
<!ELEMENT ISSN %entry;>
<!ATTLIST ISSN %bibtexml.xmlns.attrib;>
<!ELEMENT URL %entry;>
<!ATTLIST URL %bibtexml.xmlns.attrib;>
<!ELEMENT notstandard %entry;>
<!ATTLIST notstandard
        %bibtexml.xmlns.attrib;
        name CDATA #REQUIRED
>
<!ELEMENT tex (#PCDATA)>
<!ATTLIST tex
        %bibtexml.xmlns.attrib;
        code CDATA #REQUIRED
>
<!ELEMENT abbrev EMPTY>
<!ATTLIST abbrev
        %bibtexml.xmlns.attrib;
        alias CDATA #IMPLIED
>
<!ELEMENT lastname (#PCDATA | tex)*>
<!ATTLIST lastname %bibtexml.xmlns.attrib;>
<!ELEMENT firstname (#PCDATA | tex)*>
<!ATTLIST firstname %bibtexml.xmlns.attrib;>
<!ELEMENT middlename (#PCDATA | tex)*>
<!ATTLIST middlename %bibtexml.xmlns.attrib;>
<!ELEMENT suffix (#PCDATA | tex)*>
<!ATTLIST suffix %bibtexml.xmlns.attrib;>
⟨/tbbibdtd⟩
```

3.1 Example BibTEXML file

Here is an example BIBTEXML file (with two entries):

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<!DOCTYPE bibliography PUBLIC "-//Torsten Bronger//DTD bibliography 1.0//EN"
                              "/usr/local/lib/tbook/tbbib.dtd">
<bibliography>
  <bibitem type="book" label="findelmayer:34">
    <key>Findelmayer (1934)</key>
    <author>
      <firstname>Heinrich</firstname>
      <lastname>Findelmeyer</lastname>
    </author>
    <author>
      <firstname>Ruetli</firstname>
      <lastname>Obermeyer</lastname>
    </author>
    <editor>Petri Heil</editor>
    <title>Models of Rhaetian Scales</title>
    <publisher>Berner Verlag</publisher>
    <year>1934</year>
    <note>The long forgotten Findelmeyer hypothesis</note>
  </bibitem>
  <bibitem type="mastersthesis" label="bronger:01" xml:lang="de">
    <author><firstname>Torsten</firstname><lastname>Bronger</lastname></author>
    <key>Bronger (2001)</key>
    <title>Morphologische und optische Studien an V-Graben Quantendrähten:
       Einfluß der Quantentopfgeometrie auf die Quantenzustände im Draht</title>
    <school>RWTH Aachen</school>
    <year>2001</year>
  </bibitem>
</bibliography>
```

4 The address book DTD

The titchy address book DTD has been designed especially for **tbook**. I know of an address book DTD project at OASIS, but it is not suitable for the task (it's way too complicated).

```
⟨*tbadrdbdtd⟩
<!-- PUBLIC "-//Torsten Bronger//DTD address book 1.0//EN" -->
<!ENTITY % tbadrdb.xmlns
     "http://tbookdtd.sourceforge.net/addressbook">
<!ENTITY % tbadrdb.attrib
     "xmlns CDATA #FIXED '%tbadrdb.xmlns;'">

```

4.1 The elements

addresslist

4.1.1 addresslist – root element

```
<!ELEMENT addresslist   (addressentry*)>
<!ATTLIST addresslist   %tbadrdb.attrib;
                        owner           CDATA           #REQUIRED
```

```
                          private        (true|false)    #IMPLIED>
```

`<addresslist>` encloses all elements in an address book file. `owner` the the unique ID of the author of the letters using this address book. It it compared with the `from` attribute in the `<letter>` element.

`private` delivers a default value for all missing `private` values in this address book.

addressentry

4.1.2 addressentry – address book entry

```
<!ELEMENT addressentry  EMPTY>
<!ATTLIST addressentry  %tbadrdb.attrib;
                        nickname        ID              #REQUIRED
                        sex             (male|female)   #IMPLIED
                        titles          CDATA           #IMPLIED
                        firstnames      CDATA           #IMPLIED
                        recipient       CDATA           #REQUIRED
                        addressdetails  CDATA           #IMPLIED
                        street          CDATA           #IMPLIED
                        postalcode      CDATA           #IMPLIED
                        city            CDATA           #IMPLIED
                        country         CDATA           #IMPLIED
                        countrycode     NMTOKEN         #IMPLIED
                        phone           CDATA           #IMPLIED
                        fax             CDATA           #IMPLIED
                        email           CDATA           #IMPLIED
                        www             CDATA           #IMPLIED
                        private         (true|false)    #IMPLIED
                        notes           CDATA           #IMPLIED
                        keywords        CDATA           #IMPLIED>
```

This contains one possible recipient in the address book. The element itself contains nothing, everything is realised via attributes. Only two attributes are mandatory: `nickname` which must match `nickname` of the `<to>` element of `<letter>`, and `recipient` that contains the surname of the person, the company's name etc. The rest should be clear.

In `addressdetails`, every '`*`' begins a new line.

4.2 Address book example

Here now a tiny address book:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE addresslist PUBLIC "-//Torsten Bronger//DTD address book 1.0//EN"
                             "tbadrdb.dtd">
<addresslist owner="Bugs Bunny &lt;bugs.bunny@warner.com&gt;">
  <addressentry nickname="Duffy"
                sex="male"
                firstnames="Duffy"
```

```
                  recipient="Duck"
                  street="ACME Street 125"
                  city="Toontown"
                  postalcode="0001"
                  country="Toonland" countrycode="us"
                  phone="001-555-5555"
                  email="duffy.duck@warner.com"
                  private="yes"/>
</addresslist>
```

⟨/tbadrdbdtd⟩

5 The **tbook** Standard Entities

The following entities are not defined by HTML or MathML2, but I think they serve very well in LaTeX and so they should do too in XML. Eventually this goes into the file `tbook.ent`.

⟨*tbookent⟩

The first two entities are included for maybe somewhat sentimental reasons. They produce, of course, the logos of LaTeX and TeX.

```
<!ENTITY LaTeX         "<latex code='\LaTeX{}'>LaTeX</latex>">
<!ENTITY TeX           "<latex code='\TeX{}'>TeX</latex>">
<!ENTITY tbook         "<latex code='\tbook{}'><visual markup='sf'><visual
                        markup='bf'>t</visual>book</visual></latex>">

```

The entity name `&sf;` stands for "space factor". It is used *before* full stops to mark them as "end of sentence", if the last letter of the sentence is in upper case. And it's used *after* full stops to mark them as "abbreviations dots". This ensures correct skips in English texts. For German or French this is not critical. Example:

```
Dr.&sf; Pauling loved vitamin C&sf;.
```

```
<!ENTITY sf "<latex code='\@'/>">

```

The following entities are vital for the traditional orthography of German, in order to get proper hyphenation. Here an example:

```
Die Schi&ff;art ist langsam wie eine Schne&ck;e.
```

this becomes for LaTeX output something equivalent to

```
Die Schi"ffart ist langsam wie eine Schne"cke.
```

```
<!ENTITY ck            "<latex code='{\ck}'>ck</latex>">
<!ENTITY CK            "<latex code='{\CK}'>CK</latex>">

<!ENTITY ff            "<latex code='\dfk{f}'>ff</latex>">
<!ENTITY FF            "<latex code='\dfk{F}'>FF</latex>">
```

```
<!ENTITY ll           "<latex code='\dfk{l}'>ll</latex>">
<!ENTITY LL           "<latex code='\dfk{L}'>LL</latex>">
<!ENTITY mm           "<latex code='\dfk{m}'>mm</latex>">
<!ENTITY MM           "<latex code='\dfk{M}'>MM</latex>">
<!ENTITY nn           "<latex code='\dfk{n}'>nn</latex>">
<!ENTITY NN           "<latex code='\dfk{N}'>NN</latex>">
<!ENTITY pp           "<latex code='\dfk{p}'>pp</latex>">
<!ENTITY PP           "<latex code='\dfk{P}'>PP</latex>">
<!ENTITY rr           "<latex code='\dfk{r}'>rr</latex>">
<!ENTITY RR           "<latex code='\dfk{R}'>RR</latex>">
<!ENTITY tt           "<latex code='\dfk{t}'>tt</latex>">
<!ENTITY TT           "<latex code='\dfk{T}'>TT</latex>">

```

Nor for entities that are also mostly used in German, but are actually of general interest. `&bph;` is equivalent to `german.sty`'s `""` macro, and `&nbhy;` to `"~`.

```
<!ENTITY bph          "&#x0082;">  <!-- Break permitted here -->
<!ENTITY nbhy         "&#x2011;">  <!-- Non-break hyphen -->
⟨/tbookent⟩
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.