Omega, OpenType and the XML World

Yannis Haralambous
ENST Bretagne, Brest, France

John Plaice
UNSW, Sydney, Australia

Origins of Ω

- Conceived in February 1993 (Lille, F).
- Public discussion in July 1993 at TUG (Aston, UK).
- Multilingual typesetting extensions to TEX.
- All TEX 8-bit data structures become 16-bit in Ω.
- ΩTP-lists used to prepare input for typesetting.
- Basic TEX typesetting algorithms are unchanged.
- Support for multiple direction typesetting.

Where is Ω Heading?

- Ω version 2 (Free Software)
  - When do we get PDF-XML-HTML-UTF8-OT-e-**Ω**?
  - Two grants from the TEX Users Group.
    - Combining the Extensions of TEX into One System.
    - Using Omega to Generate XML and MathML from TEX Documents.
- Ω version 3 (Research)
  - Redesigning typesetting from the ground up.
  - "A Multidimensional Approach to Typesetting"
    9:00 presentation Wednesday 23 July.

Ω, Version 2

- Upwardly compatible with TEX.
- *Adaptable*, context-dependent.
- "Standards-compliant" (Unicode, XML, OpenType).
- All new code written in C++/STL.
- No fixed-size arrays, 31 bits for characters, glyphs, etc.
- Distribution in 2003.

Participants

- UNSW, Sydney
  - John Plaice
  - Paul Swoboda
  - Lap Yu (Kenneth) Ho
  - Gabriel Christian Ditu
- ENST-Bretagne, Brest
  - Yannis Haralambous
  - Gabór Bella
  - Paweł Grams

Workflow – Input

- Running tree-structured context initialized from command-line and environment variables.
- Entire input is passed through ΩTP-list, possibly empty.
- ΩTP-list interpretation is context-dependent.
- Direct input uses deserialization methods to create internal data structures.
- TEX-style input converts character set, using `iconv`, to UCS-4 (4-byte ISO-10646/Unicode).

Workflow – Output

- Macro-expansion is context-dependent.
- Text is passed through ΩTP-list, possibly empty.
- TEX typesetting algorithms are used.
- Direct output uses serialization methods
  to output internal data structures.
- DVI output is still available.
- Entire output is passed through ΩTP-list, possibly empty.

What are the rôles of XML and OpenType?

Current Plans: XML and OpenType

1. Input filter converting XML → LaTeX.
2. Direct output of XML and MathML.
3. Output filter converting DVI → DVX (XML).
4. Use of OpenType fonts with Ω.

XeLaTeX: XML Input

What is LaTeX?

- A series of *concepts* (boxes, glue, words, paragraphs, pages, footnotes, tables, floating objects, fonts, etc.)
- A series of *methods* (`document`, `itemize`, `enumerate`, `minipage`, `includegraphics`, `usepackage`, etc.)
- A *syntax* (commands, environments, catcodes, etc.)
- The possibility to use *lower level syntax*, whenever necessary (TeX, PostScript).

What is XLaTeX?

- We keep the same *concepts*.
- We keep the same *methods*.
- We change the *syntax*.
- We keep the possibility to use lower level syntax whenever necessary: LaTeX, TeX, PostScript.
- Only a well-formed and valid XLaTeX document can be converted to LaTeX, so say goodbye to LaTeX errors!
  TeX compilation can only go smoothly.

Why is XML a Better Syntax?

- Few special characters: `<`, `>` and `&` (sometimes `'` and `"`).
  These characters all have standard syntax
  (`<` `>` `&` `&apos;` `&quot;`).
- Tag names are well defined and delimited: `<TeX/>`
  (no ambiguity about white space, as in `\TeX`).
- Clear separation between *data*, *meta-data* and *keywords*:

```
<textcolor color="red">A word</textcolor>
```

- Can switch notations using *processing instructions*:

```
<?tex now we are back in \LaTeX?>
```

And What About XML Documents?

- They are *trees* (not limiting; with name spaces and tools such as XLink one can create structures that are not trees).
- Global or partial validation using DTDs or Schemas.
- Well-defined encoding (by default: UTF-8).
- Can carry meta-data (RDF, ontologies).
- Many tools to *edit*, *parse* and *transform* them (only TEX can read TEX, zillions of tools can read XML).
- Have become a standard for information exchange.

Is XLaTeX Yet Another DTD?

- Yes, XLaTeX is YAD.
- However, the element and attribute names *strangely resemble* LaTeX command and environment names.
- The goal is to minimize the *learning curve* of XLaTeX for LaTeXists.

```
<itemize>
<item>Does this look <emph>familiar?</emph></item>
</itemize>
```

XLaTeX Code: Documents

```
<document class="article" opt="12pt,a4paper">
<section id="s1">My first section</section>
...<includegraphics bbox="10 20 100 120"
                    src="toto.eps"/>...
</document>
```

- The document is contained in a `<document>` element.
  The document class and its options are attributes.
- On a first run, the filter can detect if packages are needed
  (`<includegraphics>` needs `graphics` or `graphicx`).
- Explicit `<usepackage>` and `<preamble>` also available.

XLaTeX Code: Labels

```
<figure pos="t"><includegraphics bbox="10 20 100 120"
                                 src="toto.eps"/>
<caption id="cap1">This is figure<nbsp/><ref
id="cap1"/>, on page<nbsp/><pageref~id="cap1"/>.
</caption></figure>
```

- Elements can have `id` attribute, equivalent to `\label`.
- `<includegraphics>` is an empty element (no textual data).
- Non-breakable space: Unicode `0xa0` or empty `<nbsp/>`.
- User-defined entities (such as ` `) can be added.

XLaTeX Code: Verbatim

```
<?verbatim
this is pure verbatim
1 < 2, Y&Y, \end, $x^{}2$
?>
```

- Verbatim code is obtained not by an element, but by a processing instruction.
- Inside the `verbatim` PI, everything is allowed, except `?>`.
- There are also `verbatimstar`, `verb` and `verbstar` processing instructions.

XLaTeX Code: Nested Verbatim

```
<footnote>Believe it or not: <?verbatim
you can put verbatim code into footnotes!
?></footnote>
```

- This works because the code produced by `verbatim` is not a LaTeX `verbatim` environment, but a `quotation` environment.
- Special characters are protected and lines are obeyed.
- All the (notorious) incompatibility problems of `verbatim` environment are gone.

XLaTeX Code: Table of Contents

```
<section id="s1">
<toc>A Short Title.</toc>A Long Title.
</section>
```

- The `id` attribute holds the section label (meta-data).
- The "short version" of the title (for the Table of contents) may contain other mark up. It can only be a sub-element of `<section>`.

XLaTeX Code: Tables

```
<tabular format="|c|c|"><hline/>
A <tab/> B <br/>
C <tab/> D <br/><hline/>
</tabular>
```

- Tables have the same logic as in LaTeX.
- One can also write `<tabular><format>...</format>...`
- `<tab/>` is used both in `<tabular>` and `<tabbing>`, with different productions.

XeLaTeX Code: Shortcuts

Writing:

```
<b>this is <i>code in <em>emphatic</em> style</i></b>
```

is shorter than:

```
<textbf>this is <textit>code in
<emph>emphatic </emph> style</textit></textbf>
```

- We keep the widely known shorter and easy to understand HTML tags: `<i>` and `<textit>` produce the same result.
- There is a `<p>` element to produce an empty line. It is practical for carrying attributes: `<p indent="0pt">`.

XLaTeX Code: Direct TeX Input

If really necessary

```
<?tex One can always return to
\emph{good ol' \LaTeX\ldots} ?>
```

- XML is a hostile environment for you? The LaTeX syntax world is always available. You don't need Mr. Sulu to beam you between worlds, the `<?tex ... ?>` processing instruction is enough.

XLaTeX Code: Namespaces

```
<?xml version="1.0" encoding="iso-8859-1"?>
<document xmlns="http://omega.enstb.org/2003/xlatex"
xmlns:mml="http://www.w3.org/1998/Math/MathML"
xmlns:svg="http://www.w3.org/2000/svg">
...
</document>
```

- XLaTeX document with MathML formulas and SVG figures.
- One can also write mathematics by using a TeX processing instruction: `<?math x^2+y^2=0 ?>`.

XLaTeX Code: Adding New Elements

```
<toto>Some words</toto>
<titi arg1="bla" arg2="bli">and more</titi>
<tata_ optarg="t">Something</tata_>
```

is transformed without validation into:

```
\toto{Some words}
\titi{bla}{bli}\{and more\}
\begin{tata}[t]
Something
\end{tata}
```

Ω Becomes Part of the XML World

- XLaTeX documents can be placed directly on the Web, since XSLT stylesheets can transform them to XHTML.
- One can write XSLT stylesheets to transform DocBook or TEI into XLaTeX.
- Using namespaces virtually any XML tool can be combined with XLaTeX elements.

Direct output of XML and MathML

Getting MathML out of TEX Documents (1)

- Project initiated by American Mathematical Society.
- Inside Ω, new `sgml_node` holds a tagged list.
- Automatic grouping of expressions to form proper `<mrow>`.
- New primitives to generate entities.

```
\def\arccos{\SGMLentityop{mi}{arccos}}
```

Getting MathML out of TEX Documents (2)

- New primitives to redefine math at the macro level

```
\renewcommand{\sqrt}{\@ifnextchar[\sqrttwo\sqrtone}
\newcommand{\sqrtone}[1]{%
    \SGMLstartmathtag{msqrt} #1
    \SGMLendmathtag{msqrt}}
\def\sqrttwo[#1]{\sqrttwoend{#1}}
\newcommand{\sqrttwoend}[2]{%
    \SGMLstartmathtag{mroot} {#2} {#1}
    \SGMLendmathtag{mroot}}
```

Getting XML out of TEX Documents

- New primitives to redefine structural components

```
\def\section#1{%
\@closepar%
\@closesection%
\@startsection%
\refstepcounter{section}%
\SGMLattribute{type}{\@sectiontype}%
\SGMLattribute{n}{\thesection}%
\SGMLstarttexttag{head}#1\SGMLendtexttag{head}%
\@startpar%
}
```

DVX: XML Output

Transforming DVI Directly Into XML

- DTD for DVI, called DVX.

```
<?xml version="1.0"?>
<dvx version="1.0">
   <pre id="2" num="25400000"
        den="473628672" mag="1000"
        string=" Omega output 2003.05.09:2000"/>
   <page id1="1005" id2="0" id3="0" id4="0" id5="0"
         id6="0" id7="0" id8="0" id9="0" id10="0"/>
      <fontdef id="31" checksum="1831058770"
         size="655360" designsize="655360"
         name="cmss10"/>
```

Transforming DVI Directly Into XML

- DTD for DVI, called DVX.

```
<set>Y</set>
<right dim="-18205"/>
<set>ou</set>
<right dim="285661"/>
<set>a</set>
<right dim="-18205"/>
<set>re</set>
<right dim="285661"/>
<set>reading.</set>
```

Moving to OpenType Fonts

Printing OpenType fonts

- Adaptation of `odvips` so that OpenType (and TrueType) fonts are treated as are PostScript fonts.
- The Ω engine still uses `OFM` files.
- OpenType fonts are included in the `psfonts.map` file.

  ```
  Tfmname InternalName </access/path/filename.otf
  ```

- One OpenType font will generate (*many*) Type1 fonts using only those glyphs used.
- Intermediate PostScript Font Container (PFC) file to hold Type 1 descriptions to draw the glyphs.

Current directions

- Get out and polish the distribution.
- Adapt Ω to directly read `.otf` files.
- Develop DTDs so that Ω can ensure round-trip conversion: Ω reads XML and generates the identical XML.
- Longer term:
  - Direct XML input without TEX macro processing.
  - Direct XML (SVG) output, as annotation of input.