Cahiers GUTenberg

XSL FOS AND TEX : SOME DATA

Chris ROWLEY

Cahiers GUTenberg, nº 39-40 (2001), p. 201-204.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2001___39-40_201_0>

XSL FOs and TEX: some data

Chris ROWLEY

Open University, UK & LATEX3 Project
C.A.Rowley@open.ac.uk

Résumé. La spécification XSL « FO » *(Formatting Objects)* est un projet ambitieux et fécond dont le développement mêle harmonieusement idées profondes et concepts élégants. Comment ne pas se réjouir de voir des propriétés aussi fondamentales que la *visibilité* décrites comme étant de nature « magique » ?

Le moment est venu de nous livrer à l'analyse objective — et passionnante — des hypothèses sur lesquelles repose le modèle XSL de formatage de document. Cet article fournit quelques éléments utiles à cette analyse à partir de la comparaison de certains aspects du modèle XSL et de leurs pendants en TEX/LATEX.

Abstract. *The XSL FO (Formatting Objects) specification is a noble and inventive project that is maturing into a cornucopia of useful insights and intellectual treats. How could it fail to delight when it formally describes basic properties, such as* visibility*, as 'magic'!*

It is therefore a timely, fascinating and pragmatic exercise to analyse the assumptions made by XSL about the process and results of document formatting. This article provides a small amount of the data needed for this analysis by comparing some aspects of the XSL model with that provided by TEX/LATEX.

1. Introduction

In XSL, the classes of 'formatting objects' and 'formatting properties' provide the vocabulary for expressing presentation intent. They hence also define a comprehensive model of the process of document formatting. Likewise, a document formatting system such as LATEX, because it is based on interface and scripting languages, defines such a model. Since both claim to support a wide range of documents and to have commercial and scholarly importance, it will be useful to compare them and to analyse the extent to which they coexist in harmonious creativity.

Any attempt to analyse the use of a TEX-derivative as the basic engine embedded within an XSL-conformant formatter must consider at least the following three possibilities.

– The use of a general purpose LaTeX-like macro package to implement parts of the model using TeX's 'programming features'.
– The development from scratch of an implementation of the XSL model using TeX's 'programming features'.
– A non-TeX application that uses TeX but only to carry out some specific formatting tasks such as line-breaking or mathematical composition.

All of these three can assume the use of as much as is required of TeX's own model and capabilities since they contain the whole TeX system. In all but the last anything not available in basic TeX must be implemented within TeX's programming capability. More radical uses of TeX's abilities are conceivable and they may offer useful insights both into what is required of a TeX-quality formatting engine for the XSL-world and into useful and exciting developments of TeX itself.

1.1. Background

The main section of this article contains some examples of the most obvious differences between the XSL FO model of the formatting process and that of TeX/LaTeX (including the widely available extensions of them such as pdfTeX and Ω). The information about XSL FO on which these examples are based is from Version 1.0 [3].

Some of these differences are more fundamental than others whilst some aspects of the XSL model can be partially emulated by using TeX in quite straightforward ways. It is often claimed that TeX can be used to fully emulate any formatting requirements; even if true this does not imply that there is any sense in doing so. It is far better to learn from these examples how to develop superior systems rather than use them as justification for indulgence in the exquisite torture of pushing TeX/LaTeX beyond sensible limits.

Looking on the brighter side, the TeX model supports many specifications that are not covered by the current version of XSL FO; but the latter is eXtensible, so there is, at least in the publicity hype, hope for convergent evolution of the two, one day!

In most cases it is straightforward, but often tedious, to manufacture a fit to the XSL model by either diminishing or slightly extending the models and data-structures already used by TeX/LaTeX. Such a process is, of course, allowing the tail to wag the dog since XSL is intended only as a standard for a specification language for existing and future formatter capabilities, it is not intended to control the capabilities of those formatters.

The somewhat eclectic contents of the current XSL FO document are the result of various attempts, more or less successful, that have been made to inject

aspects of the LATEX experience into the new standard. It was unfortunately not politically feasible to by-pass this time-consuming process by simply using the strategy available to the US HTML/CSS lobby: to demand that the XSL FO model explicitly incorporate everything from CSS2 however irrelevant, US-biased and confusing.

2. Examples

These examples cover only those areas of XSL that are relevant to the use of TEX-based systems to produce a fully formatted document such as a PDF file [1]. Thus it does not cover much of the material in the specification that is derived from the CSS specification [2] and is thus too closely related to the specific capabilities of HTML and the severe limitations of current main-stream browsers.

2.1. Font selection

This is not strictly part of the formatting model. The XSL (OpenType) model for specifying a 'nominal font' is very similar to that in LATEX thus it is feasible and probably wise for the latter to incorporate that of XSL simply because such a generic standard is better than a specialised ad hoc one.

2.2. Minor differences

1. Hyphenation ladders longer than one line are not identifiable in the TEX model.
2. Many specifications assume that pages have a unique and known binding-side (often called the inside). Making this information available is not directly supported by TEX's galley–pagination model and its asynchronous implementation.

2.3. Substantial differences

1. TEX itself does not support the general concept of 'invisible' graphical content: this is completely formatted, so that its location and size are determined, but it should not be rendered.
2. TEX's paragraph-building and table-building models do not support the strategy of stacking the lines, however high, with constant baseline separation; the line boxes TEX generates always have their natural height and the baseline separation mechanism supports only the minimisation of the leading between baselines.

3. In the XSL model: 'A space-specifier is a compound datatype whose components are minimum, optimum, maximum, conditionality and precedence. Space-specifiers occurring in sequence may interact with each other.' TEX's model of space, based on kerns and glue, does not support those interactions that are based on precedence.
4. Another TEX mechanism that does not support the XSL precedence model is breaking, of lines and pages; here the XSL model supports, in addition, precedence relationships between keep and break conditions. In addition, the XSL distinction between different types of column-break may be difficult to emulate in TEX.
5. Some XSL table column specifications will be very difficult to implement in TEX (and probably in any system) without serious loss of layout quality or processing efficiency.
   TEX's 'halign' mechanism has little to offer for most XSL table properties since its core is a sophisticated glue-based algorithm for determining column widths and cell layouts; but this algorithm does not interact with the paragraph-builder acting on cell contents.
6. The last three points can be subsumed into the statement that TEX's central glue/penalty mechanism cannot directly be used to implement important XSL specifications.

2.4. The major difference

The XSL model assumes that a large variety of flowed material can be split between pages: examples are paragraphs with a visible frame (border) and paragraphs within table cells. This is very difficult to implement using any current applications; it may prove to be, in some sense such as complexity, absolutely difficult. Similar possibilities are allowed for line-breaking.

Bibliography

[1] *Portable Document Format Reference Manual*, Version 1.3, Adobe Systems, 1999, `http://www.pdfzone.com/resources/`.

[2] *Cascading Style Sheets, level 2 (CSS2)*, as amended by Errata document 1999/11/04. W3C Recommendation, 1999, `http://www.w3.org/TR/1998/REC-CSS2-19980512` and `http://www.w3.org/Style/css2-updates/REC-CSS2-19980512-errata.html`.

[3] *Extensible Stylesheet Language (XSL)*, Version 1.0, W3C Candidate Recommendation, 21 November 2000, `http://www.w3.org/TR/2000/CR-xsl-20001121`.