Multidirectional Typesetting in XSL-FO
Jean-Michel Hufflen

To cite this version:
Jean-Michel Hufflen. Multidirectional Typesetting in XSL-FO. BachoTeX 2009 Conference, 2009, Poland. pp.37–40. hal-00661890

HAL Id: hal-00661890
https://hal.science/hal-00661890
Submitted on 20 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Multidirectional Typesetting in XSL-FO

Jean-Michel HUFFLEN
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
jmhufflen@lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/home/~jmhufflen

Abstract
XSL-FO texts use an XML-like syntax that aim to describe high-quality print outputs. This article complements the introduction to XSL-FO EuroBachoTEX 2007. We show how XSL-FO allows users to typeset texts belonging to different writing systems: from left to right, from right to left, . . . We compare this implementation to TEX-like typeset engines, e.g., Xe\TeX{}.

Keywords multidirectional typesetting, Unicode, Xe\TeX{}.

Streszczenie
XSL-FO są tekstemi XMLowymi, których celem jest opisywanie wydruków o wysokiej jakości. Niniejszy artykuł jest uzupełnieniem wprowadzenia do XSL-FO przeprowadzonego na konferencji EuroBacho\TeX{} 2007. Pokażemy, jak XSL-FO pozwalają użytkownikowi składać teksty w różnych systemach pisma: z lewej do prawej, z prawej do lewej, . . . Porównamy tę implementację do maszyn \TeX{}owych takich jak np. Xe\TeX{}.

Słowa kluczowe drukowanie wielokierunkowe, Unicode, Xe\TeX{}.

0 Introduction
This article continues the introduction to XSL-FO\textsuperscript{1} given in [7]. Let us recall that this language aims to describe high-quality print outputs using XML\textsuperscript{2} syntax. Reading this article requires basic knowledge of XSL-FO, as introduced in [7], and of Unicode [12]. More complete introductions to XSL-FO’s general features are [1, 9, 10], the official document being [15]. At first glance, texts written using XSL-FO may seem to be very verbose, but let us recall that this language is not devoted to direct use. The ‘normal’ way to build an XSL-FO text is to get it as an output file of XSLT\textsuperscript{3}, the language of transformations used for XML texts\textsuperscript{4}. We demonstrated this point in [7, § 2], but do not use it in the present article.

Here we focus about multidirectional typesetting in XSL-FO and explain the principles guiding it. Let us recall that some natural languages—e.g., Hebrew and Arabic—are typeset from right to left, some—e.g., traditional Japanese—uses vertical writing. Our demonstrations\textsuperscript{5} will be performed by using Apache FOP\textsuperscript{6}, which yields PDF\textsuperscript{7} or PNG\textsuperscript{8} figures.

1 Layout and writing mode
Let us consider the XSL-FO text given in Figure 1. The layout, given by the \texttt{fo:simple-page-master} element, defines a paper format and its margins, where anything cannot be printed. It also defines \texttt{regions}, as shown in Figure 2. The localisation of pages’ margins is \textit{physical}, that is, the top margin will be always... at the top of the page, the bottom one will be always at the bottom, and so

\textsuperscript{1}Title in Polish: \textit{Skład wielokierunkowy w XSL-FO}.
\textsuperscript{2}eXtensible Stylesheet Language — Formatting Objects.
\textsuperscript{3}eXtensible Markup Language. Readers interested in a general introductory book to this formalism can refer to [11].
\textsuperscript{4}Besides, some XSL-FO processors—e.g., Apache fop [2]—allow users to join transformations performed by means of XSLT and typesetting XSL-FO texts. But they are still based on XSLT’s ’old’ version (1.0) [14], introduced in [5, 6].
\textsuperscript{5}Passive \TeX{} [3] is a \TeX{}-based engine that may be used as an XSL-FO processor, but it does not provide multidimensional typesetting.
\textsuperscript{6}Formatting Objects Processor.
\textsuperscript{7}Portable Document Format.
\textsuperscript{8}Portable Network Graphics.
Jean-Michel HUFFLEN

<?xml version="1.0" encoding="ISO-8859-2"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="page-simple" page-height="297mm" page-width="210mm"
      margin-top="10mm" margin-bottom="20mm" margin-left="25mm"
      margin-right="25mm" writing-mode="lr-tb">
      <fo:region-body margin-top="20mm" margin-left="20mm" margin-right="20mm"/>
      <fo:region-before extent="20mm"/>
      <fo:region-after extent="20mm"/>
      <fo:region-start extent="15mm" reference-orientation="90"/>
      <fo:region-end extent="15mm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="page-simple" font-family="serif" font-size="18pt"
    text-align="left" xml:lang="po">
    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="center" font-size="10pt">Pieśń ludowa z opolskiego</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="xsl-region-start">
      <fo:block text-align="center" font-size="10pt">III Symfonia</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block>Ej, ćwierkejcie mu tam,),<fo:block>
      <fo:block>wty ptosecki boze,),<fo:block>
      <fo:block>kiedy mamulicka</fo:block>
      <fo:block>znaleźć go nie moze.</fo:block>
      <fo:block>A ty, boze kwiecie,<fo:block>
      <fo:block>kwitnijze w około,<fo:block>
      <fo:block>niech się synockowi</fo:block>
      <fo:block>choć lezy wesolo.</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>

**Figure 1**: Henryk Mikolaj Górecki’s 3rd Symphony, 3rd movement, last stanza.

---

on. On the contrary, regions’ placement depend on a writing mode. This information specifies two directions: the first is the inline-progression-direction, which determines the direction in which words will be placed, and the second is the block-progression-direction, which determines the direction in which blocks and lines are placed one after another. This information is expressed within the page model by means of a writing-mode attribute, that can be set to:

- lr-tb for ‘left-to-right, top-to-bottom’ (the default value), suitable for languages such as English, Polish, etc.;
- rl-tb for ‘right-to-left, top-to-bottom’, suitable for languages such as Hebrew and Arabic;
- tb-rl for ‘top-to-bottom, right-to-left’, suitable for traditional Japanese;
- lr-alternating-rl-tb inline components within the first line are stacked left-to-right, within the second line they are stacked right-to-left, continuing in alternation; lines and blocks are stacked top-to-bottom;
- lr-alternating-rl-bt like the previous value, but lines and blocks are stacked bottom-to-top;
- lr-inverting-rl-bt like the previous value, but in addition, inline components within even lines are inverted.

---

9 This writing system is called boustrophedon: from Greek etymology, that means ‘turning like oxen in ploughing’. Some archaic Greek inscriptions used this system.
10 Some pre-Columbian scripts in Classical Quechuan — the Inca Empire’s language — used this system.
11 This reverse boustrophedon was discovered in some undeciphered scripts in Rapa-Nui.
Multidirectional Typesetting in xsl-fo

Figure 2: A page’s regions for lr-tb writing mode.

Figure 3: A page’s regions for rl-tb writing mode.

tb-lr-in-lr-pairs text is written in two-character, left-to-right, pairs; these pairs are then stacked top-to-bottom to form a line; lines and blocks are stacked left-to-right\(^{12}\).

Other analogous values are\(^ {13}\):

\begin{align*}
tb-lr & \quad bt-lr \quad bt-rl \quad lr-bt \quad rl-bt \\
lr-inverting-rl-tb &
\end{align*}

rl, tb are shorthands for lr-tb, rl-tb, tb-rl.

In regions, something can be printed as a static content—e.g., a page number—or a flow—for a text possibly printed on regions belonging to several successive pages. Regions’ names do not express ‘geographical’ directions, such as ‘top’ or ‘bottom’, but come from a relative frame reference depending on a writing mode\(^ {15}\). The ‘before’ and ‘after’ (resp. ‘start’ and ‘end’) regions are always related to the block-progression-direction (resp. inline-progression-direction). A writing-mode attribute set to ”lr-tb” yields to what is pictured in Figure 2. Other region placements are given in Figures 3 & 4 for rl-tb and tb-rl writing modes. If the writing mode specifies an inline-progression-direction of left-to-right for odd-numbered lines, and right-to-left for even-numbered lines—that is, when the writing mode is ‘lr-alternating-rl-...’ or ‘lr-inverting-rl-...’—the region-start (resp. region-end) is located at the left (resp. right). If the writing mode is tb-lr-in-lr-pairs, the placement rules are the same than for tb-lr.

In fact, the definitive placement of regions results from a combination of a writing mode with a reference-orientation attribute, which is an angle whose value is given in degrees and defaults to zero. More detailed examples can be found in [10, Fig. 3.4]. Each region can define its own writing mode and reference orientation, otherwise these attributes are inherited from parent elements, as done usually in xsl-fo. As an example, we define such a reference orientation in Figure 1 for the ‘start’ region, the result is pictured in Figure 5.

2 Present and future

As mentioned in [15, §§ 1.2.5 & 5.8], an xsl-fo processor must implement the Unicode bidirectional algorithm [13]. Let us recall that this algorithm only applies to paragraphs, that is, blocks w.r.t. xsl-fo’s terminology. In addition, this algorithm requires some adaptations to fit into the xsl-fo processing model. The inline-progression-direction is determined at the block level by the writing mode—possibly refined by means of a fo:block-container attribute.
element [15, § 6.5.3] — and within the inline formatting objects within a block — by refining the writing mode of a fo:inline-container element [15, § 6.6.8] or by means of the attributes direction and unicode-bidi associated with a fo:bidi-override element [15, § 6.6.2]. Character properties, as they are defined by the standard Unicode [12], are implicit or may be refined by means of the fo:character element [15, § 6.6.3]. The inline-progression-direction of each glyph is used to control the stacking of glyphs.

To sum up, xsl-fo bidirectional algorithm is not controlled by functionalities related to precise natural languages, as did in XEP — by means of natural-language-oriented environments such as arab and urdu — but is superseded by informations about writing modes.

At the time of writing, this algorithm is not fully implemented yet by xsl-fo processors. Besides, most processors difficulty deal with non-Latin characters. However, some interesting effects can be observed by using the attributes writing-mode and direction, even on texts writing using the Latin alphabet. As two additional examples:

- the source text of Figure 6 is obtained from Figure 1’s text by changing the writing-mode attribute of the fo:root element to "rl-tb";
- the source text of Figure 7 is obtained from Figure 1’s text by applying the following changes:
  - the writing-mode attribute of the root element, fo:root, is set to "tb-rl";
  - the font size used for the start-region’s contents has become smaller:
    <fo:block text-align="center" font-size="9pt">
      III Symfonia
    </fo:block>
  - the second fo:block element is replaced by:
    <fo:block-container writing-mode="rl-tb">
      <fo:block text-align="right">
        wty ptosecki boze,
      </fo:block>
      <fo:block>
        </fo:block-container>

More examples will be demonstrated at the BachoTEX 2009 conference.

3 Acknowledgements

Many thanks to Jerzy B. Ludwichowski, who has translated the abstract and keywords in Polish very quickly.
Figure 6: Formatting a text in tb-rl writing mode (page’s upper part).

References


