CAHIERS GUTenberg

PASSER DE LaTeX À XSL-FO

Jean-Michel Hufflen

Cahiers GUTenberg, nº 51 (2008), p. 77-99.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2008___51_77_0>

PASSER DE LaTeX À XSL-FO

Jean-Michel Hufflen

Résumé. — Cet article se propose d'introduire les utilisateurs de LaTeX à XSL-FO. Son but n'est pas de donner une vue exhaustive de XSL-FO, mais de faciliter la prise en main. Nous montrons quelles sont les ressemblances et dissemblances entre ces deux approches pour la composition de textes.

Abstract. — This article aims to introduce LaTeX users to XSL-FO. It does not attempt to give an exhaustive view of XSL-FO, but allows a LaTeX user to get started. We show the common and different points between these two approaches of word processing.

Zusammenfassung. — Dieser Artikel dient als eine Einführung in XSL-FO für LaTeX-Benutzer. Wir versuchen nicht, eine vollständige Übersicht für diese Sprache zu geben, vielmehr helfen wir LaTeX-Benutzern beim Einstieg in XSL-FO. Darüberhinaus zeigen wir die gemeinsamen und unterschiedlichen Aspekte dieser zwei Satzsysteme.

0. Introduction

Même si le format de XML [1] est largement utilisé pour l'échange et le traitement de structures qui confinent aux bases de données, ce méta-langage trouve de lointaines origines dans la gestion de documents textuels. Aussi n'est-il pas surprenant que la boîte à outils de XML fournisse un langage — XSL-FO [2] — pour décrire des compositions typographiques de très haute qualité. XSL-FO et LaTeX ont de nombreux points communs : en particulier, ni l'un ni l'autre n'est interactif [3].

1. *Extensible markup language*. Dans cet article, nous supposerons connues les bases de ce méta-langage. Une bonne référence pour une initiation à XML est [20].

2. *Extensible stylesheet language—Formatting objects*.

3. Ils ne sont pas WYSISWG (*What you see is what you get*) comme on dirait outre-Manche.

Cahiers GUTenberg n° 51 — Octobre 2008, p. 77-99.

De bons ouvrages d'apprentissage de XSL-FO existent : p. ex., [2, ch. 8] en français, [19] en anglais, [17] en allemand [4]. Mais il nous a semblé qu'il est relativement aisé pour un utilisateur de LaTeX d'apprendre les bases de XSL-FO, c'est le propos du présent article [5]. Comme nous allons le voir, ce format est extrêmement verbeux, mais en réalité, il n'a pas été pensé en vue d'une écriture directe par un utilisateur. C'est néanmoins ce que nous allons faire avec un petit exemple [6] pour en donner les bases au § 1. Ensuite, nous montrerons brièvement au § 2 comment a été prévue l'écriture de documents dans plusieurs langues, voire dans plusieurs systèmes d'écriture. Enfin, dans une optique de séparation de la forme et du fond, les textes en XSL-FO sont souvent construits en considérant un fichier XML ne contenant qu'un texte « brut », sans aucune indication de formatage, et en lui appliquant une feuille de style XSLT [7] [27], le langage utilisé pour les transformations de textes XML, nous aborderons ce point au § 3. Bien sûr, cette modeste introduction ne saurait remplacer ni les ouvrages didactiques plus volumineux cités précédemment, ni le document de référence du W3C [26], auxquels nous invitons le lecteur curieux à se reporter pour plus de détails. Quant aux commandes LaTeX que nous mentionnons dans le courant de cet article, elles sont documentées dans le *LaTeX Companion*, en version originale [15] ou en traduction française [16].

1. Prise en main

1.1. Notions de base

Les utilisateurs de LaTeX sont familiers avec la notion de classe de document, la notion équivalente que fournit XSL-FO est celle de **modèle de page**. Celui que nous proposons à la figure 1 est très simple : une seule

4. Tous ces documents se rapportent à la version 1.0 de XSL-FO [24], alors que la recommandation la plus récente du W3C (*World Wide Web consortium*) est la version 1.1 [26]. Les changements sont toutefois mineurs. Voir [26, § E]. Signalons également que le W3C a commencé à travailler sur une révision du langage [29], dont les raisons d'être sont présentées dans [28].

5. Qui est une traduction et une révision de [8].

6. Tous les textes et programmes de cet article, ainsi que quelques versions alternatives et des exemples supplémentaires, sont disponibles sur le *Web* à l'adresse `http://lifc.univ-fcomte.fr/home/~jmhufflen/texts/xsl-fo/`.

7. *Extensible stylesheet language transformations.*

```
<fo:layout-master-set xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:simple-page-master master-name="simple-page"
                         page-height="297mm" page-width="210mm"
                         margin-top="10mm" margin-bottom="20mm"
                         margin-left="25mm" margin-right="25mm">
    <fo:region-body margin-top="20mm" margin-bottom="20mm"
                    margin-left="20mm" margin-right="20mm"/>
                         <!--  (… éventuellement l'attribut column-count
                                  pour écrire sur plusieurs colonnes.)  -->
    <fo:region-before extent="20mm"/>
                 <!--  On donne seulement la largeur des autres régions.  -->
    <fo:region-after extent="20mm"/>
                         <!--  Si l'on ne définit pas de marges pour la  -->
                         <!--    région « body », cela crée des effets de  -->
                         <!--  superposition avec les régions voisines.  -->
    <fo:region-start extent="15mm"/>
    <fo:region-end extent="15mm"/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

FIGURE 1. — Exemple de modèle de page en XSL-FO.

construction de page, spécifiée par l'élément `fo:simple-page-master`, dont les attributs donnent le format de la feuille et les dimensions des marges, où rien n'est imprimé. Viennent les définitions des **régions**, leur disposition étant montrée par le dessin de gauche de la figure 2. Bien noter que la terminologie évite à dessein de parler de « haut », « bas », « gauche », « droite » en ce qui concerne les régions, car leur placement dépend en réalité du système d'écriture[8] : ainsi, le sens de lecture va de la région « start » à la région « end ». À titre d'exemples, nous avons reproduit dans la figure 2 le placement des régions pour les écritures latine et sémitique (utilisée par des langues telles que l'hébreu ou l'arabe). L'ordre des déclarations pour les régions est fixe : `fo:region-body`, puis `fo:region-before`, `fo:region-after`, `fo:region-start` et `fo:region-end`, tous ces éléments étant

8. Donné par l'attribut `writing-mode`, dont nous reparlerons au § 2.

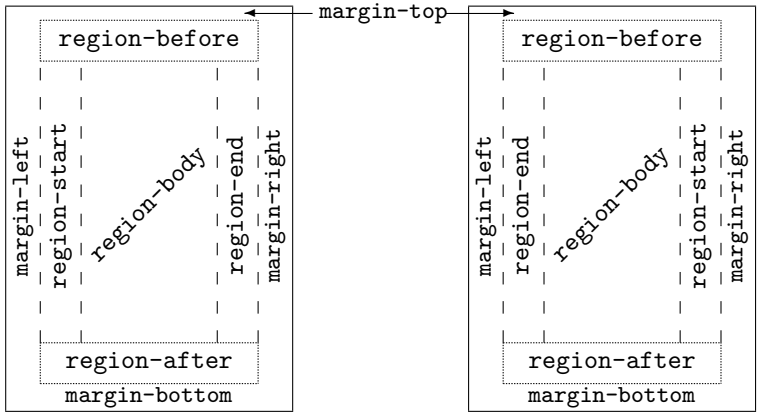

FIGURE 2. — Régions définies par XSL-FO pour les écritures latines et sémitiques.

facultatifs, sauf le premier[9]. L'élément `fo:simple-page-master` admet en outre un attribut `reference-orientation`, qui est une mesure d'angle en degrés : elle permet en particulier de réaliser une présentation en mode paysage lorsque cet attribut est lié à la valeur `90` au lieu de `0`, la valeur par défaut[10].

Lorsqu'on ne définit qu'un seul modèle de page — comme dans la figure 1 —, cela signifie que toutes les pages du document, en nombre quelconque, seront formatées suivant ce modèle. La définition de *modèles de séquences*, par l'élément `fo:page-sequence-master` permet la limitation du nombre de pages d'un document, ainsi que des combinaisons variées de plusieurs modèles de page, il est par exemple possible de combiner deux modèles pour les pages paires et impaires, ou de définir des modèles séparés pour les première et dernière pages d'un document.

9. Ce qui contredit [8]. En fait, la figure 2 de cet article fut correctement traitée par quelques processeurs de XSL-FO qui étaient tolérants sur ce point, elle est néanmoins invalide, l'ordre correct étant donné dans la figure 1 du présent article.

10. Une explication détaillée des effets des combinaisons diverses des attributs `writing-mode` et `reference-orientation` figure dans [19, p. 37-39].

```
<!DOCTYPE root [<!ENTITY en-dash "&#x2013;">
                <!ENTITY layout SYSTEM "layout.fo">]>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  &layout;
  <fo:page-sequence master-reference="simple-page" font-family="serif"
                    font-size="12pt" text-align="start">
    <fo:static-content flow-name="xsl-region-after">
      <fo:block text-align="center" line-height="14pt" color="green"
                font-size="10pt" font-family="serif" xml:lang="it">
        Ballo delle Ingrate (<fo:page-number/>)
      </fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block font-family="sans-serif" font-size="18pt"
                font-variant="small-caps" padding-top="3pt"
                text-align="center" color="white"
                background-color="blue" space-after="15pt" line-height="24pt">
        Ballo delle Ingrate
      </fo:block>
      <fo:block font-family="sans-serif" font-size="14pt" space-after="18pt"
                border-style="solid" border-width="0.5mm" border-color="blue"
                padding="4mm" start-indent="80mm" end-indent="4mm">
        <fo:block text-align="end">
          Ottavio Rinuccini
          <fo:inline font-style="italic">(1562&en-dash;1621)</fo:inline>
        </fo:block>
      </fo:block>
      <fo:block space-before.minimum="10pt" space-before.optimum="11pt"
                space-before.maximum="12pt">
        De l'implacabil Dio
      </fo:block>
      <fo:block>Eccone giunt'al Regno,</fo:block>
      <fo:block>Seconda, O bella Madre, il pregar mio.</fo:block>
      <fo:block (espacement comme précédemment) >Non tacerà mia voce</fo:block>
      <fo:block>Dolci lusinghe e prieghi</fo:block>
      <fo:block>Finche l'alma feroce</fo:block>
      <fo:block>Del Re severo al tuo voler non pieghi.</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

FIGURE 3. — Exemple de texte source en XSL-FO.

La figure 4 donne le résultat du traitement du texte source de la figure 3. Nous pouvons y voir que les modèles de page n'y sont pas donnés par la référence à un fichier comme dans LaTeX. Si l'on souhaite qu'un modèle de page soit réutilisé pour plusieurs documents, la solution est d'utiliser une *entité externe* [20], ce qui implique l'introduction d'une balise `DOCTYPE` artificielle car ne se rapportant pas réellement à un type de document [11].

Comme le montre la figure 3, la racine d'un texte XSL-FO est l'élément `fo:root`, dont les fils sont un modèle de pages, puis une séquence de pages [12]. Une séquence de pages définit *quoi* écrire et *où*. Nous montrons dans la figure 3 comment associer un *contenu statique* — ici, un titre, suivi du numéro de la page — à un pied de page, et comment définir un *flux*, s'étalant sur des régions qui peuvent appartenir à plusieurs pages successives. Un flux est lié à une région au moyen de l'attribut `flow-name` se rapportant à une région. Très souvent, ce sont des conventions implicites qui sont utilisées, aussi la définition de la région « body » de la figure 1 équivaut à :

```
<fo:region-body region-name="xsl-region-body"/>
```

1.2. Formater du texte

En première approximation, un élément `fo:block` de XSL-FO s'utilise pour un alinéa, terminé en TeX par la commande `\par`. En fait, ces « blocs » se rapprochent plutôt de l'environnement `minipage` de LaTeX dans la mesure ils peuvent être imbriqués. Les attributs `color` et `background-color` sont utilisés pour la couleur du texte et du fond. D'autres attributs — `border-style`, `border-width`, `border-color`, ... — se rattachent au tracé d'une boîte autour d'un bloc. Par défaut, `border-style` est lié à une dimension nulle et aucune boîte n'est tracée. Des attributs « `padding-...` » sont utilisés pour gérer l'espacement entre le texte et le bord de la boîte, tracée ou non.

11. Ce qui est un contournement. Une méthode plus propre consiste en l'emploi de balises appartenant à XInclude [25], mais beaucoup d'outils actuels ne savent pas encore les interpréter.

12. Entre ces deux éléments obligatoires peuvent s'intercaler deux éléments facultatifs, pour des déclarations globales [26, § 6.6.3] (ce sont en fait des déclarations de couleurs) et pour la spécification de *signets* [26, § 6.11.1].

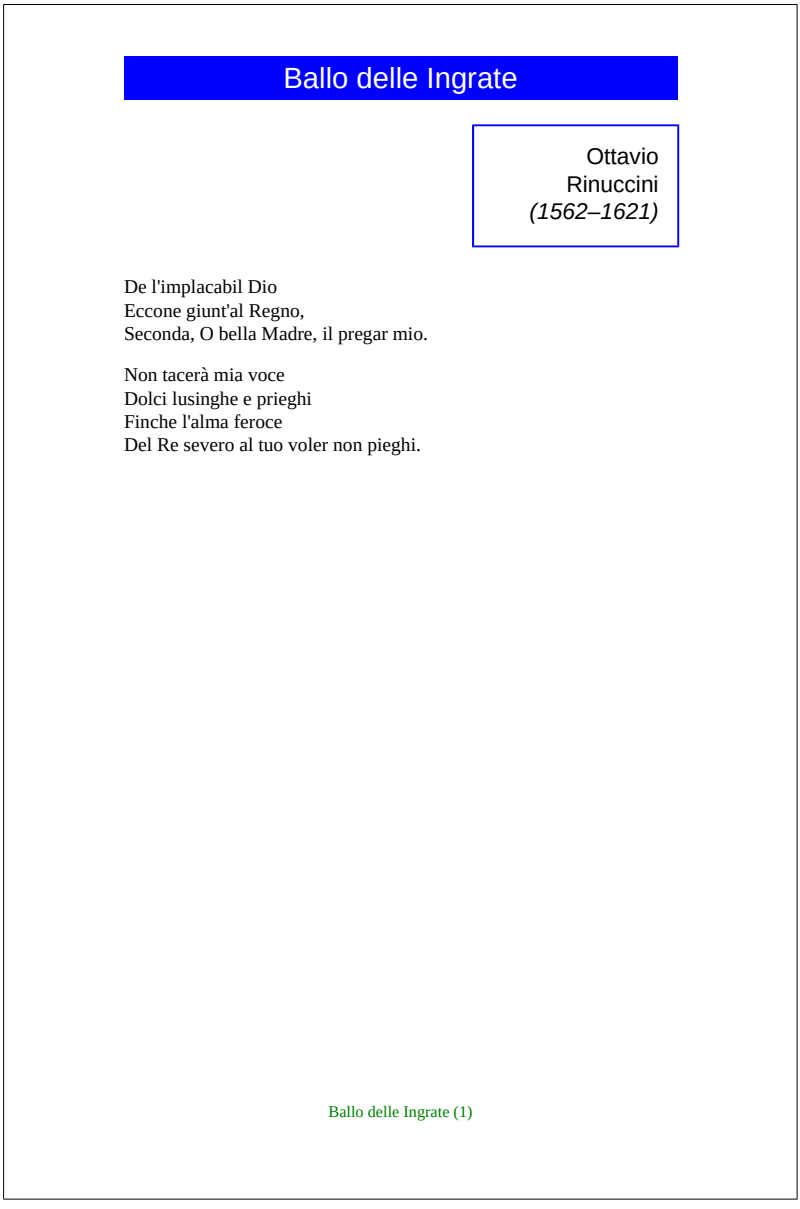

FIGURE 4. — Le texte de la figure 3, formaté.

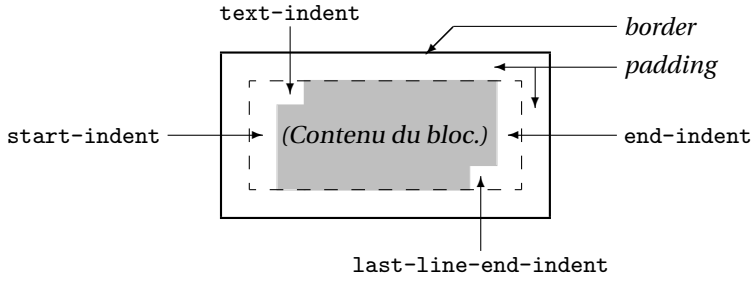

FIGURE 5. — Visualisation des parties d'un bloc.

L'organisation des différentes zones d'un bloc est donnée dans la figure 5. L'attribut `start-indent` (resp. `end-indent`) donne la distance entre le premier (resp. dernier) caractère et le bord initial (resp. terminal) de la boîte. Ne pas perdre de vue que ces notions se rattachent au *sens* de l'écriture, il n'est donc pas toujours pertinent de parler de « bord supérieur gauche » et de « bord inférieur droit » comme si nous étions toujours dans le cas des écritures latines. De même, les conventions prises pour les valeurs de l'attribut `text-align` — `justify` pour un texte justifié, `center` pour un texte centré, `start` et `end` pour un texte serré sur la région de même nom — évitent de trop se rapporter à l'écriture latine, quoique les valeurs `left` et `right` soient acceptées et respectivement équivalentes à `start` et `end`[13]. Un autre attribut, `text-align-last`, concerne le formatage de la dernière ligne du bloc ; sa valeur par défaut est `relative`, ce qui équivaut à `start` lorsque `text-align` est lié à `justify`, à la valeur de ce dernier attribut dans les autres cas[14]. Les valeurs `left` et `right` sont elles aussi acceptées pour l'attribut `text-align-last` avec la même convention que pour l'attribut `text-align`. Les deux autres attributs montrés dans la figure 5 sont des dimensions : `text-indent` donne le renfoncement

13. Par compatibilité avec CSS (*Cascading style sheets*, voir [14] pour une bonne introduction à ce langage), qui emploie lui aussi une directive `text-align`.

14. Et correspond au formatage « traditionnel » d'un alinéa. Notons également que c'est l'ensemble de l'alinéa qui doit être formaté de manière aussi harmonieuse que possible. Aussi, un changement de l'attribut `text-align-last` peut affecter la composition de l'ensemble de l'alinéa.

TABLE 1. — Valeurs possibles pour la plupart des attributs liés aux polices.

Attribut	Défaut	Autres valeurs possibles
`font-family`	`serif`	`sans-serif`
`font-size`		Tailles absolues : `xx-small`, `x-small`, `medium`, `large`, `x-large`, `xx-large` Tailles relatives : `smaller`, `larger` Dimensions, p. ex. : `10pt`
`font-stretch`	`normal`	`wider`, `narrower`, `ultra-condensed`, `extra-condensed`, `condensed`, `semi-condensed`, `semi-expanded`, `expanded`, `extra-expanded`, `ultra-expanded`
`font-weight`	`normal`	`bold`, `bolder`, `lighter`
`font-style`	`normal`	`italic`, `reverse-normal`, `reverse-oblique`
`font-variant`	`normal`	`small-caps`

traditionnel au début d'un alinéa, `last-line-text-indent` est utilisé pour un renfoncement en fin d'alinéa. À noter que ces deux attributs ne sont pas toujours pris en compte : `text-indent` n'influence la composition que si l'alinéa est justifié ou serré à gauche, tandis que `last-line-text-indent` n'est pertinent que pour un texte serré à droite.

En cas d'imbrication d'éléments de XSL-FO, les attributs non redéfinis sont hérités : p. ex., le bloc introduisant l'auteur du poème de la figure 3 utilise la même police que le bloc qui l'englobe. L'élément `fo:inline` permet la redéfinition temporaire d'attributs sans ouvrir un nouveau bloc, c'est-à-dire à l'intérieur du même alinéa. Quant aux attributs régissant les polices, nous les avons donnés dans la table 1. On pourra remarquer que les noms de la plupart de ces attributs coïncident avec des noms de propriétés utilisés dans les feuilles de style en CSS [15]. En LaTeX, les changements concernant le *look* d'une police sont exprimées par des

15. Notons également qu'à l'instar de CSS, quelques propriétés de XSL-FO peuvent être exprimées sous forme abrégée : p. ex., les attributs `font-size` et `font-family`, relatifs à la police utilisés avec l'élément `fo:page-sequence` de la

combinaisons de commandes telles que `\textbf` ou `\textit`, alors que les attributs de XSL-FO sont davantage « typés » (notions de *chasse*, de *style*, ...). Cela peut sembler artificiel à un utilisateur habitué à LaTeX, mais montre clairement toutes les combinaisons possibles. Quant à l'expression de la taille, la solution largement adoptée dans beaucoup de textes en XSL-FO consiste en l'utilisation de dimensions exprimées en points, ce qui complique évidemment l'adaptation de ces textes si tous les caractères doivent être grossis ou réduits. L'utilisation de tailles relatives étant quelque peu restrictive (cf. table 1), ce sont les tailles absolues qui se rapprochent le plus de la « philosophie » de LaTeX : lorsqu'une taille est exprimée en points, elle correspond dans tous ses sous-éléments à la taille `medium` et toutes les autres tailles s'en déduisent, comme `\small`, `\footnotesize`, `\large`, etc., s'adaptent en LaTeX par rapport à la taille `\normalsize` de la classe du document, éventuellement précisée par une option de la commande `\documentclass`.

Revenant aux éléments `fo:block`, l'espacement entre deux blocs consécutifs est contrôlé par les attributs `space-before` et `space-after`. Les **composants** permettent en particulier la réalisation en XSL-FO des *longueurs élastiques*[16] de TeX. Ce qui est exprimé pour la première strophe de la figure 3, c'est que la valeur idéale pour l'espacement vertical avant ce bloc est de 11 points — le composant `optimum` de l'attribut `space-before` — cette valeur devant se situer entre 10 et 12 points — les composants `minimum` et `maximum`. Spécifier :

```
<... space-before="11pt" space-before.minimum="10pt">
```

définit le composant `minimum`, les autres composants de l'attribut étant liés à la valeur `11pt`. XSL-FO fournit deux autres composants pour la gestion de l'espacement : `conditionality` indique si la spécification prend effet (valeur `retain`) ou non (valeur par défaut, `discard`) au début ou à la fin d'une surface de référence[17] — le début (resp. la fin) d'une page pour l'attribut `space-before` (resp. `space-after`) associé à l'élément

figure 3 peuvent être remplacés par « `font="12pt serif"` ». Nous signalons ce point — cf. [26, §§ 5.2 & B.3] pour plus de détails — sans l'utiliser car la plupart des processeurs XSL-FO actuels ne le traitent pas.

16. D'après la terminologie de [16], « *rubber lengths* » en version originale.
17. « *Reference area* » [26, § 4], selon la terminologie de XSL-FO.

`fo:block`, ou le début et la fin d'une ligne pour les mêmes attributs associés à l'élément `fo:inline` ; `precedence` influence le calcul de l'espacement entre la fin d'une surface de référence et le début de la surface suivante, c'est soit un entier (0 par défaut), soit la valeur `force`, considérée comme plus grande que tout entier [18].

Mentionnons brièvement deux attributs pour les blocs et les textes « inline » : `text-decoration`, utilisé pour le tracé d'une ligne au-dessus, en-dessous, ou à travers un texte [26, § 7.17.4], et `baseline-shift` pour le placement des indices et des exposants. Par contre, les seuls calculs applicables à un texte sont donnés par l'attribut `text-transform`, qui peut prendre les valeurs `uppercase` (conversion en lettres capitales [19]), `lowercase` (conversion en bas de casse), `capitalize` (la première lettre de chaque mot est mise en capitale, les autres sont en bas de casse), `none` (par défaut). Pour préciser ce que nous entendons par « calcul », mentionnons tout simplement que le fragment :

```
\iflanguage{french}{Chanson italienne}{Italian song}
```

— cf. [15, § 9.2.1] au sujet de la commande `\iflanguage` — n'a pas d'équivalent en XSL-FO.

D'autres attributs liés à l'élément `fo:block` concernent la répartition du texte lors de la composition des blocs : `keep-with-next` (resp. `keep-with-previous`), qui indique que le contenu d'un alinéa doit être placé sur la même page que le précédent (resp. suivant) ; `keep-together`, qui spécifie que le contenu de l'alinéa doit être placé sur une seule page. Chacun de ces trois attributs possède trois composants — `within-line`, `within-column`, `within-page` — et les valeurs possibles sont des entiers indiquant l'importance de ces contraintes, comparables à l'argument facultatif des commandes `\linebreak` et `\pagebreak` de LaTeX. Deux notations sont également utilisées : `auto` (par défaut) et `always` pour les niveaux minimal et maximal. Forcer le commencement d'une nouvelle page ou colonne est possible au moyen des attributs `break-before` et `break-after`, dont les valeurs possibles sont `auto` (par défaut), `column`, `page`, `even-page`, `odd-page`, les deux

18. Les règles précises de ce calcul sont données dans [26, § 4.3.1].

19. Ne pas oublier qu'il existe des écritures *monocamérales* — p. ex., les écritures d'Extrême-Orient —, où cette notion n'a pas de sens, c'est pourquoi cet attribut `text-transform` est de moins en moins utilisé.

dernières indiquant un saut à la prochaine page paire ou impaire qui suit. Voir [19, p. 70-72] pour plus de détails.

1.3. D'autres éléments

La mention des quelques éléments qui suivent vise à donner une idée de la puissance d'expression de XSL-FO. L'élément racine pour les listes, comparable à l'environnement `list` de LaTeX, est `fo:list-block`. En ce qui concerne les tables, l'élément racine est `fo:table` [19, p. 104-110] et l'organisation est analogue aux tables du langage HTML[20]. Les notes en bas de page sont réalisées par l'élément `fo:footnote`, mais leur numérotation et l'éventuelle ligne horizontale qui sépare le bas du texte et le haut de ces notes est à la charge de l'utilisateur [11]. Comme en HTML, les références croisées sont réalisées par des *hyper-liens*, vers d'autres parties du même document ou vers des documents externes, l'élément utilisé étant `fo:basic-link`. Comme LaTeX, XSL-FO permet la manipulation d'*objets flottants*, — voir [26, § 6.12.2] au sujet de l'élément `fo:float` — et d'*index*, au moyen d'éléments et d'attributs [26, § 7.24] analogues à l'environnement `theindex`. Pour terminer, mentionnons qu'il n'y a pas de mode mathématique en XSL-FO.

2. Gestion du multilinguisme

XSL-FO fournit des attributs pour contrôler la division des mots lors du formatage d'un bloc [26, § 7.10]. En premier, l'attribut `hyphenate`, lié par défaut à `false`. Dans le cas contraire, d'autres attributs[21] comprennent la spécification d'un caractère de coupure (`hyphenation-character`), d'une langue (`language`), d'un pays (`country`), ... En pratique, l'attribut prédéfini `xml:lang` — voir ses deux occurrences dans la figure 6 — peut être utilisé comme raccourci pour ces deux dernières informations : rappelons que c'est une langue qui est spécifiée par ses deux premières lettres, éventuellement suivies de deux autres désignant un pays [1]. Ce qui manque, par contre, à la version présente de XSL-FO, c'est la spécification de divisions possibles de façon *ad hoc*[22], comme le peuvent en LaTeX les commandes `\-` et

20. *Hypertext markup language*. Une bonne introduction à ce langage est [18].

21. Que pour l'instant, beaucoup de processeurs XSL-FO ne traitent pas.

22. Ce point sera très probablement corrigé dans la prochaine version de XSL-FO au moyen de l'attribut `hyphenation-exceptions` [29, § 4.7.4].

`\hyphenation`. De même, il n'y a pas de réel équivalent de la commande `\discretionary`, qui permet en TEX de spécifier des coupures plus « exotiques », par exemple, le fait qu'en allemand, le groupe « ck » se divisait en « k-k »[23] — `\discretionary{k-}{k}{ck}` — si ce n'est un attribut `script` permettant de spécifier un programme de coupure [26, § 7.10.3].

Comme nous l'avons entrevu précédemment au § 1.1, le langage XSL-FO n'est pas limité aux langues s'écrivant avec l'alphabet latin. Les divers éléments définissant les modèles de page — p. ex., `fo:simple-page-master` — admettent un attribut `writing-mode`, donnant deux *directions* : d'abord, la direction des lignes, puis la direction des blocs[24]. Par défaut, cet attribut `writing-mode` vaut `lr-tb`, pour « left-to-right, top-to-bottom », ce qui signifie que chaque ligne du texte se lit de gauche à droite, plusieurs lignes successives se lisant de haut en bas. Voici toutes les autres valeurs que peut prendre cet attribut (le processeur Apache FOP[25] [3] réalise quelques-uns de ces modes d'écriture[26], ainsi que nous l'avons montré dans [10]) :

`rl-tb` de droite à gauche, puis de haut en bas : c'est le mode adapté aux langues sémitiques ;

`tb-rl` de haut en bas, puis de droite à gauche : c'est le mode de l'écriture japonaise traditionnelle ;

`tb-lr` de haut en bas, puis de gauche à droite, utilisé dans de très vieux écrits en mongolien ;

`lr-alternating-rl-tb` la première ligne est lue de gauche à droite, la deuxième de droite à gauche, et ainsi de suite, les lignes se succédant de haut en bas ; ce système, utilisé dans quelques inscriptions en grec archaïque, est appelé *boustrophédon*[27] ;

23. Nous disons bien « se divisait », car depuis la réforme orthographique de 1996, on coupe avant le groupe « ck » [5, K 165].

24. « Inline-progression-direction » et « block-progression-direction », d'après la terminologie de XSL-FO.

25. *Formatting objects processor*.

26. Mentionnons [6] à l'attention des lecteurs intéressés par l'histoire des systèmes d'écriture.

27. Ce mot d'étymologie grecque (βουστροφηδόν) se rapporte aux mouvements d'un bœuf dans un champ.

`lr-alternating-rl-bt` analogue au précédent, mais les lignes se succèdent de bas en haut ; quelques écrits précolombiens dans la langue de l'Empire Inca utilisaient ce système ;

`lr-inverting-rl-bt` analogue au précédent, mais les caractères des lignes se lisant de droite à gauche présentent une rotation de 180 degrés ; ce « *boustrophédon inversé* » a été observé dans l'écriture Rongo-rongo de l'Île de Pâques, non encore déchiffrée ;

`tb-lr-in-lr-pairs` le texte est fractionné en paires de caractères dont les éléments sont écrits de gauche à droite, ces paires sont ensuite disposées de haut en bas pour former une ligne, les lignes successives se présentant de gauche à droite ; ce système a été utilisé pour le langage maya.

Le *modus operandi* lié aux autres valeurs possibles de cet attribut se déduit aisément :

```
bt-lr   bt-rl   lr-bt   rl-bt   lr-inverting-rl-tb
```

et terminons en mentionnant que `lr`, `rl` et `tb` sont des raccourcis pour `lr-tb`, `rl-tb` et `tb-rl`. Lorsqu'un bloc est traité, la direction des lignes pour une suite de caractères peut être implicitement déterminée à partir de la base de données des caractères d'Unicode [21], ce qui permet la mise en œuvre de l'algorithme bidirectionnel [22]. Notons cependant que l'information donnée par l'attribut `writing-mode` l'emporte sur l'information implicite : en d'autres termes, il est par exemple possible d'écrire de droite à gauche en utilisant des caractères latins si `writing-mode` est lié à une valeur telle que `rl-tb`. Comme nous le montrons dans [10], la redéfinition de cet attribut s'opère au moyen des éléments `fo:block-container` et `fo:inline-container`, dont les fils sont des éléments `fo:block` et `fo:inline`.

3. Production de textes XSL-FO par XSLT

Nous allons à présent montrer comment a été réellement pensée l'utilisation de XSL-FO dans une optique de séparation de la forme et du fond. Une formulation en XML de notre poème italien est reproduite à la figure 6. Quant aux figures 7 à 10, elles donnent *in extenso* une transformation XSLT 2.0 dont le résultat est un texte en XSL-FO, celui de la

```
<!DOCTYPE song-0 SYSTEM "song-0.dtd" [<ENTITY en-dash "&#x2013;">]>
<song-0 xml:lang="it">
  <preamble>
    <title>Ballo delle Ingrate</title>
    <author>
      <firstname>Ottavio</firstname>
      <lastname>Rinuccini</lastname>
      <birth-year>1562</birth-year>
      <death-year>1621</death-year>
    </author>
    <note xml:lang="fr">Musique de Claudio Monteverdi (1567-1643).</note>
  </preamble>
  <body>
  <stanza role="Amore">
      <verse>De l'implacabil Dio</verse>
      <verse>Eccone giunt'al Regno,</verse>
      <verse>Seconda, O bella Madre, il pregar mio.</verse>
    </stanza>
    <stanza role="Venere">
      <verse>Non tacerà mia voce</verse>
      <verse>Dolci lusinghe e prieghi</verse>
      <verse>Finche l'alma feroce</verse>
      <verse>Del Re severo al tuo voler non pieghi.</verse>
    </stanza>
    <stanza role="Amore">
      <verse>Ferma, Madre, il bel piè, non por le piante</verse>
      <verse>Nel tenebroso impero,</verse>
      <verse>Che l'aer tutto nero</verse>
      <verse>Non macchiass'il candor del bel sembiante:</verse>
      <verse>Io sol n'andrò nella magion oscura,</verse>
      <verse>E pregand'il gran Re trarotti avante.</verse>
    </stanza>
    <stanza role="Venere">
      <verse>Va pur come t'agrada. Io qui t'aspetto,</verse>
      <verse is-followed-by="Sinfonia">Discreto pargoletto.</verse>
    </stanza>
  </body>
</song-0>
```

FIGURE 6. — L'exemple de poème, *in extenso*.

```
<!DOCTYPE stylesheet [<!ENTITY layout SYSTEM "layout.fo">
                     <!ENTITY en-dash "&#x2013;">]>
<xsl:stylesheet version="2.0" id="song-0-2-fo"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                exclude-result-prefixes="xsd">
  <xsl:output method="xml" indent="yes"/>
  <xsl:strip-space elements="*"/>     <!-- Écarter les nœuds blancs. -->
  <xsl:template match="song-0" as="element(fo:root)">
    <fo:root>
      <xsl:apply-templates select="@xml:lang"/>
      &layout;
      <fo:page-sequence master-reference="simple-page" font-family="serif"
                        font-size="medium" text-align="start">
        <xsl:variable name="preamble" select="preamble"/>
        <xsl:apply-templates select="$preamble/title" mode="in-footer"/>
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates select="$preamble,body"/>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  <xsl:template match="preamble" as="element(fo:block)+">
    <fo:block font-family="sans-serif" font-size="x-large" padding-top="3pt"
              text-align="center" color="white" background-color="blue"
              space-after="15pt" line-height="24pt">
      <xsl:apply-templates select="title,note"/>
      <xsl:apply-templates select="collection">
        <xsl:with-param name="left" select="'('" as="xsd:string" tunnel="yes"/>
        <xsl:with-param name="right" select="')'" as="xsd:string" tunnel="yes"/>
      </xsl:apply-templates>
    </fo:block>
    <xsl:variable name="both" select="'4mm'" as="xsd:string"/>
    <fo:block font-family="sans-serif" font-size="large" space-after="18pt"
              border-style="solid" border-width="0.5mm" border-color="blue"
              padding="{$both}" start-indent="40mm" end-indent="{$both}">
```

FIGURE 7. — Transformation XSLT générant un texte XSL-FO (début).

```
    <xsl:apply-templates select="author"/>
  </fo:block>
</xsl:template>

<xsl:template match="title | collection" as="xsd:string">
  <xsl:call-template name="put-text"/>
</xsl:template>

<xsl:template match="note" as="element(fo:footnote)">
  <fo:footnote>
    <xsl:call-template name="put-footnotemark">
      <xsl:with-param name="size" select="'large'" as="xsd:string"/>
    </xsl:call-template>
    <fo:footnote-body>
      <fo:block text-align-last="justify" color="black">
        <fo:leader leader-pattern="rule"/>
      </fo:block>
      <fo:block font-family="serif" font-size="small" text-align="justify"
                color="black">
        <xsl:apply-templates select="@xml:lang"/>
        <xsl:call-template name="put-footnotemark"/>
        <xsl:apply-templates/>
      </fo:block>
    </fo:footnote-body>
  </fo:footnote>
</xsl:template>

<xsl:template match="author" as="element(fo:block)">
  <fo:block text-align="end">
    <xsl:value-of select="firstname,lastname"/>
    <fo:inline font-style="italic">
      <xsl:value-of
        select="' (',birth-year,
                if (//song-0/@xml:lang = 'fr') then '-' else '&en-dash;',
                death-year,')'"
        separator=""/>
    </fo:inline>
  </fo:block>
</xsl:template>

<xsl:template match="body" as="element(fo:block)+">
```

FIGURE 8. — Transformation XSLT générant un texte XSL-FO (suite).

```
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="stanza" as="element(fo:block)">
  <fo:block space-before.minimum="10pt" space-before.optimum="11pt"
            space-before.maximum="12pt">
    <xsl:apply-templates select="@role,verse"/>
  </fo:block>
</xsl:template>

<xsl:template match="verse" as="element(fo:block)+">
  <fo:block><xsl:apply-templates/></fo:block>
  <xsl:apply-templates select="@is-followed-by"/>
</xsl:template>

<xsl:template match="@role" as="element(fo:block)+">
  <fo:block font-weight="bold" keep-with-next.within-page="always">
    <xsl:call-template name="put-text">
      <xsl:with-param name="left" select="'['" as="xsd:string" tunnel="yes"/>
      <xsl:with-param name="right" select="']'" as="xsd:string" tunnel="yes"/>
    </xsl:call-template>
  </fo:block>
</xsl:template>

<xsl:template match="@is-followed-by" as="element(fo:block)">
  <fo:block font-style="italic" text-align="center"
            keep-with-previous.within-page="always">
    <xsl:call-template name="put-text">
      <xsl:with-param name="left" select="'('" as="xsd:string" tunnel="yes"/>
      <xsl:with-param name="right" select="')'" as="xsd:string" tunnel="yes"/>
    </xsl:call-template>
  </fo:block>
</xsl:template>

<xsl:template match="@xml:lang" as="attribute(xml:lang)"><xsl:copy/></xsl:template>

<xsl:template match="title" mode="in-footer" as="element(fo:static-content)">
  <fo:static-content flow-name="xsl-region-after">
    <fo:block text-align="center" line-height="14pt" color="green"
              font-size="small">
      <xsl:value-of select="concat(.,' (')"/>
      <fo:page-number/>
      <xsl:text>)</xsl:text>
```

FIGURE 9. — Transformation XSLT générant un texte XSL-FO (suite).

```
      </fo:block>
    </fo:static-content>
  </xsl:template>

  <xsl:template name="put-text" as="xsd:string">
    <xsl:param name="left" select="''" as="xsd:string" tunnel="yes"/>
    <xsl:param name="right" select="''" as="xsd:string" tunnel="yes"/>
    <xsl:value-of select="$left,.,$right" separator=""/>
  </xsl:template>

  <xsl:template name="put-footnotemark" as="element(fo:inline)">
    <xsl:param name="size" select="'xx-small'" as="xsd:string"/>
    <fo:inline font-size="{$size}" vertical-align="super">*</fo:inline>
  </xsl:template>

</xsl:stylesheet>
```

FIGURE 10. — Transformation XSLT générant un texte XSL-FO (fin).

figure 3 en étant une version simplifiée [28]. Nous n'avons pas joint d'explications au texte de ces figures — du reste, le propos de cet article n'est pas XSLT, mais XSL-FO ; plus d'informations à propos des techniques employées peuvent être trouvées dans [7, 9], la référence pour un manuel didactique de XSLT 2.0 étant [12].

Nous avons utilisé la version la plus récente de XSLT parce qu'elle présente une meilleure expressivité que la version précédente (1.0) [23] pour ce genre de tâche et permet en particulier la mention — ainsi que la vérification à l'exécution — des types des paramètres et du résultat d'un *template* au moyen de l'attribut `as`, que nous avons intensément utilisé dans nos figures. L'emploi de deux espaces de noms définis par préfixes, `xmlns:xsl` et `xmlns:fo` [29], ce qui sépare clairement les balises qui sont évaluées lorsque le programme XSLT est en action (`<xsl:...>`) des balises qui constituent le résultat de la transformation (`<fo:...>`).

28. En particulier, nous pouvons remarquer que le texte obtenu montre comment spécifier en XSL-FO une note en bas de page (cf. fig. 8 & 10).

29. Rappelons que l'information qui identifie un espace de nom particulier n'est pas le préfixe lui-même, en tant qu'identificateur, mais la valeur qui lui est associée, p. ex., `http://www.w3.org/1999/XSL/Transform` pour un programme en XSLT. XInclude (voir note 11, en bas de la p. 82) introduit un autre espace de noms pour réaliser des inclusions de fichiers [25].

Notons pour clore cette section que XSL-FO ne fournit pas d'outils pour construire automatiquement une table des matières ou un index, mais cette tâche n'est pas très difficile lorsqu'un texte XSL-FO résulte de l'application d'une transformation en XSLT : des exemples sont donnés en [19, p. 149-150].

4. Conclusion

Nous voici arrivés au terme de notre initiation à XSL-FO, ces premiers pas permettant déjà la réalisation d'effets typographiques non triviaux. En ce qui concerne les processeurs disponibles, la *XSL-FO's home page* (`http://www.w3.org/Style/XSL/`) en donne une liste. Parmi eux, nous avons plus spécialement expérimenté :

— PassiveTEX [4] : c'est une adaptation de TEX pour le traitement de textes en XSL-FO, le résultat est un fichier DVI [30] (resp. PDF [31]) produit par la commande `xmltex` (resp. `pdfxmltex`) ;

— Apache FOP [3] : écrit en Java, plus complet, le résultat peut être un fichier PDF ou PostScript, d'autres formats étant possibles [32].

Ajoutons à cette très brève liste qu'aucun format particulier de sortie n'est requis par la recommandation du W3C [26], même si la quasi-totalité des processeurs génèrent des fichiers PDF, ce format étant par ailleurs devenu un standard de fait. D'autre part, à notre connaissance, aucun processeur actuel ne réalise la totalité de la recommandation du W3C, même si ces processeurs sont prêts à traiter la plupart des textes en pratique. En outre, quelques-uns — dont Apache FOP — permettent d'enchaîner à une transformation XSLT le formatage du résultat : notons toutefois qu'il s'agit de transformations utilisant la version 1.0 de XSLT [33].

Nous pensons toutefois qu'il est intéressant de suivre l'évolution de XSL-FO car ce langage repose sur un concept de base analogue à celui

30. *Device independent*.

31. *Portable document format*, le format d'Adobe.

32. En particulier le format RTF (*rich text format*) utilisé par les outils en interface avec Microsoft Word. Nous avons utilisé Apache FOP pour produire la figure 4 à partir du texte source de la figure 3.

33. En réalité, les deux langages appartenaient au même projet à l'origine. Plus tard, une clarification des tâches a débouché sur une séparation bien distincte entre XSLT et XSL-FO.

qu'ont adopté TEX & Cie : le recours à une compilation pour construire un format de sortie. Il s'agit néanmoins d'une version alternative — voire très différente sur certains points de mise en œuvre — de ces concepts. Une version plus récente, aussi, même si dans le domaine du multilinguisme et de la gestion des divers systèmes d'écriture, des outils puissants et complémentaires tels que le *package* babel de LATEX 2ε [15, ch. 9] ou le nouveau moteur XETEX [13] ont vu le jour. Nous pensons quant à nous qu'une reprise du projet Passive TEX pourrait déboucher sur une intéressante coordination des efforts des communautés de TEX et de XSL-FO.

Remerciements

En premier lieu, je remercie les auditeurs des présentations que j'ai déjà données sur XSL-FO, dans le cadre de cours à l'Université de Franche-Comté ou à l'occasion de journées organisées par des groupes d'utilisateurs de TEX. Le souvenir des questions qu'ils ont alors posées m'a permis d'améliorer substantiellement quelques points de cet article. Merci également à Thierry Bouche, pour sa patience lorsque cet article était en cours d'écriture, et sa collaboration aux mises au point finales. Merci enfin à Alexander Heinisch, qui a relu le résumé en allemand.

Bibliographie

1. Harald Tveit Alvestrand. — *Request for Comments: 3066. Tags for the Identification of Languages*. UNINETT, Network Working Group, March 1995. `http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc3066.html`.

2. Bernd Amman et Philippe Rigaux. — *Comprendre XSLT*. Éditions O'Reilly France, 2002.

3. *Apache Fop*, 2009. `http://xmlgraphics.apache.org/fop/`.

4. David Carlisle, Michel Goossens and Sebastian Rahtz. — « De XML à PDF avec xmltex, XSLT et PassiveTEX ». Actes du congrès GUTenberg 2000, Toulouse, mai 2000. *Cahiers GUTenberg*, nº 35-36, p. 79-114.

5. Duden. — *Die deutsche Rechtschreibung. Band 1*. 22. Auflage. Bibliographisches Institut, Mannheim, 2000.

6. James G. Février. — *Histoire de l'écriture*. 2e édition. Payot. 1984.

7. Jean-Michel Hufflen. — « Introduction to XSLT ». *Biuletyn GUST*, Vol. 22, p. 64. In *BachoTEX 2005 conference*, April 2005.

8. Jean-Michel Hufflen. — « Introducing LaTeX users to XSL-FO ». *TUGboat*, Vol. 29, no. 1, p. 118-124. EuroBachoTEX 2007 proceedings. 2007.

9. Jean-Michel Hufflen. — « XSLT 2.0 vs XSLT 1.0 ». In: *Proc. BachoTEX 2008 Conference*, p. 67-77, April 2008.

10. Jean-Michel Hufflen. — « Multidirectional Typesetting in XSL-FO ». In: Tomasz Przechlewski, Karl Berry and Jerzy B. Ludwichowski (eds.), *Proc. BachoTEX 2009 Conference*, p. 37-40, April 2009.

11. Jean-Michel Hufflen. — « Processing 'Computed' Texts ». GUIT 2009 meeting proceedings, October 2009. *ArsTEXnica*, Vol. 8, p. 102-109.

12. Michael H. Kay. — *XSLT 2.0 Programmer's Reference*. 3rd edition. Wiley Publishing, Inc. 2004.

13. Jonathan Kew. — « XeTeX in TeX Live and beyond ». EuroBachoTEX 2007 proceedings. *TUGboat*, Vol. 29, no. 1, p. 146-150.

14. Eric A. Meyer. — *CSS. La référence*. 2e édition. Éditions O'Reilly France. Traduction française de *CSS: the Definitive Guide* par Hervé Soulard, avril 2005.

15. Frank Mittelbach and Michel Goossens, with Johannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod. — *The LaTeX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts, August 2004.

16. Frank Mittelbach et Michel Goossens avec Joannes Braams, David Carlisle, Chris A Rowley, Christine Detig et Joachim Schrod. — *LaTeX Companion, 2e édition*. Pearson Education France. Traduction française de [15] par Jacques André, Benoît Bellet, Jean-Côme Charpentier, Jean-Michel Hufflen et Yves Soulet, octobre 2005.

17. Manuel Montero Pineda und Manfred Krüger. — *XSL-FO in der Praxis. XML-Verarbeitung für PDF und Druck*, März 2004.

18. Chuck Musciano et Bill Kennedy. — *HTML. La référence*. 4e édition. Éditions O'Reilly France. Traduction française de *HTML: The Definitive Guide* par James Guérin, février 2001.

19. Dave Pawson. — *XSL-FO*. O'Reilly & Associates, Inc, August 2002.

20. Erik T. Ray. — *Introduction à XML*. Éditions O'Reilly France. Traduction française de *Learning XML* par Alain Ketterlin, octobre 2001.

21. The Unicode Consortium. — *The Unicode Standard Version 5.0*. Addison-Wesley, November 2006.

22. The Unicode Consortium, `http://unicode.org/reports/tr9/`: *Unicode Bidirectional Algorithm*. Unicode Standard Annex #9, March 2008.

23. W3C. — *XSL Transformations (XSLT). Version 1.0*. W3C Recommendation. Edited by James Clark, November 1999. http://www.w3.org/TR/1999/REC-xslt-19991116.

24. W3C. — *Extensible Stylesheet Language (XSL). Version 1.0*. W3C Recommendation. Edited by James Clark, October 2001. http://www.w3.org/TR/2001/REC-xsl-20011015/.

25. W3C. — *XML Inclusions (XInclude) Version 1.0*, 2nd edition. W3C Recommendation. Edited by Jonathan Marsh, David Orchard, and Daniel Veillard, November 2006. http://www.w3.org/TR/2006/REC-xinclude-20061115/.

26. W3C. — *Extensible Stylesheet Language (XSL). Version 1.1*. W3C Recommendation. Edited by Anders Berglund, December 2006. http://www.w3.org/TR/2006/REC-xsl11-20061205/.

27. W3C. — *XSL Transformations (XSLT). Version 2.0*. W3C Recommendation. Edited by Michael H. Kay, January 2007. http://www.w3.org/TR/2007/WD-xslt20-20070123.

28. W3C. — *Extensible Stylesheet Language (XSL) Requirements Version 2.0*. W3C Working Draft. Edited by Klaas Bals, March 2008. http://www.w3.org/TR/2008/WD-xslfo20-req-20080326/.

29. W3C. — *Design Notes for Extensible Stylesheet Language (XSL) 2.0*. W3C Working Draft. Edited by Liam R E Quin, September 2009. http://www.w3.org/TR/2009/WD-xslfo20-20090929/.

◖ Jean-Michel Hufflen
LIFC (EA CNRS 4269)
Université de Franche-Comté.
16, route de Gray
25030 Besançon Cedex — France
jmhufflen@lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/home/~jmhufflen