CAHIERS GUTenberg

CONVERTIR DU LaTeX EN HTML EN PASSANT PAR XML : DEUX EXEMPLES D'UTILISATION DE TRALICS

José GRIMM

Cahiers GUTenberg, nº 51 (2008), p. 29-59.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2008___51_29_0>

CONVERTIR DU LaTeX EN HTML EN PASSANT PAR XML : DEUX EXEMPLES D'UTILISATION DE TRALICS

José Grimm

Résumé. — Dans cet article nous montrons comment deux types différents de documents écrits en syntaxe LaTeX peuvent être convertis en HTML en utilisant XML comme langage intermédiaire. On commencera par le rapport d'activité de l'Inria : dans ce cas, il n'y a en général pas de conversion directe du LaTeX en PDF, la classe de document LaTeX ne peut être utilisée qu'en mode brouillon, la version faisant foi est la version XML. Le deuxième exemple concerne une thèse de linguistique : dans ce cas la version HTML a été créée, après la soutenance, en modifiant (le moins possible) le source du document.

Abstract. — This paper demonstrates on two examples how a LaTeX document can be converted to HTML using an XML intermediate document. The first example is the Inria's Activity Report, for which the printed reference (the PDF version) is obtained from the XML. The second example concerns a PHD thesis, whose translation to HTML was undertaken after the defence, and needed some adaptations.

1. Le traducteur de LaTeX vers XML

1.1. Introduction

Nous commençons par une brève description de Tralics, un traducteur de LaTeX vers XML utilisé depuis 2002 pour convertir le rapport d'activité de l'Inria (appelé RA ou RaWeb dans la suite) en PDF et HTML. Chaque équipe de recherche [1] rédige chaque année, suivant des règles strictes,

1. Sur les 179 équipes qui ont produit un rapport en 2007, 151 ont le statut d'équipe-projet.

son rapport d'activité et le soumet à la moulinette du RaWeb : on peut, soit taper *make* dans un terminal, soit soumettre l'ensemble de ses textes à un robot [2]. Dans les deux cas, le texte est traduit en XML par Tralics, une feuille de style convertit ce XML en un autre XML (passage de la DTD Tralics à la DTD raweb) ; ce fichier XML est alors converti en PDF et HTML par d'autres feuilles de style. Des traitements supplémentaires sont effectués : des tableaux récapitulatifs facilitent la vérification des auteurs dans la bibliographie. Pour des informations supplémentaires concernant Tralics ou le RA, voir [6, 3, 4, 5].

La version définitive du rapport est transmise (a priori sous la forme XML, mais le source TEX est préféré car plus facile à corriger) à la cellule RaWeb, qui insère les documents XML dans la base de données Ralyx [3], ce qui permet ensuite d'interroger le serveur avec des questions du type : quelles sont les publications du membre X du projet Y pendant la période considérée, ou combien y a-t-il de chercheurs détachés de l'Université dans un thème Inria donné. Cette dernière possibilité n'existe qu'à partir de l'an 2006, suite à un changement de syntaxe du source TEX, décrite plus loin.

Notons que la version imprimable (PostScript ou PDF) du RA n'est pas produite à partir du source TEX, mais bien de la version XML. Une feuille de style convertit le document XML en document XSL-FO, lequel est converti par LATEX en Pdf. Des modifications ont été apportées aux outils de conversion (passiveTEX et autres, voir [2, 8, 1]) notamment pour traiter tout MathML. Nous ne décrirons pas plus en détail cette partie de la chaîne de traitement du RaWeb, mais nous utiliserons à deux occurrences la possibilité qu'a TEX de pouvoir interpréter du XML pour en faire du DVI, et par voie de conséquence du PostScript lequel peut être traduit en image, format PNG, pour inclusion dans du HTML.

La version originale de Tralics comportait, en plus du traducteur proprement dit, un préprocesseur qui vérifiait l'existence (ou l'absence) de certaines commandes ; ce préprocesseur n'a plus lieu d'être, le système étant suffisamment mature : il est remplacé par des tests a posteriori (les tableaux récapitulatifs mentionnés plus haut). En fait, la plupart des limitations du traducteur sont supprimées au fil du temps. Par exemple,

2. Accessible via la page web `http://irabot.inrialpes.fr/`.
3. On peut l'interroger via `http://ralyx.inria.fr`.

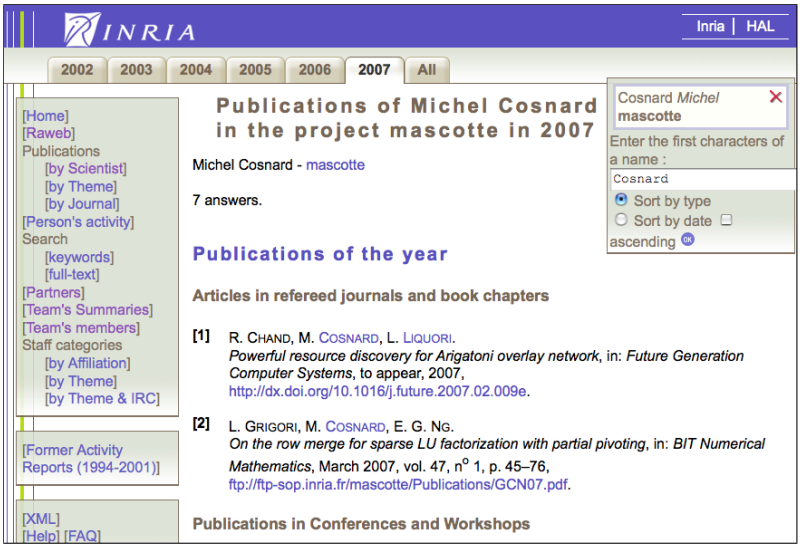


Figure 1. — Exemple d'interrogation du serveur Ralyx.

la commande `\input` était interdite, car le préprocesseur n'était pas récursif. Par défaut `\small` et `\tiny` ont la même traduction car utiliser des fontes trop petites nuit à la lisibilité. Les commandes de sélection de fontes ou de couleur sont mal comprises par le traducteur (la commande `\selectfont` a été ajoutée en décembre 2007, la commande `\color` en novembre 2008) car nous ne voulions pas que certains rapports Inria soient composés en Times et les autres en Computer Modern. Les commandes `\usepackage` et `\documentclass` ont été rajoutées tardivement (en fait, Tralics reconnaît une centaine de *packages*, soit un dixième de l'ensemble des *packages* disponibles sur CTAN, et trop peu de classes).

1.2. Sémantique de la traduction

Quand on écrit un traducteur de source LaTeX, une question importante est : « Quelles sont les commandes reconnues par le traducteur et comment sont-elles traduites ? » Une façon de répondre est celle de

GELLMU [4] ou Hévéa [5] : on utilise une syntaxe proche de celle de LaTeX, mais non compatible (et a priori plus simple). On peut aussi utiliser la technique de TeX4ht [6] ou LXir [7] qui utilise le moteur TeX pour produire du DVI, ce qui assure une compatibilité totale au niveau de la syntaxe. Le choix de Tralics a été de construire un logiciel indépendant de TeX, mais avec la même syntaxe. Toutes les primitives de TeX sont connues par Tralics, y compris `\dump`, mais ne sont pas toujours implémentées ; en particulier, comme Tralics ne fait pas de mise en page, des commandes comme `\mark` ou `\parshape` lisent des arguments mais ignorent la valeur, tandis que d'autres telles que `\ht0` ou `\badness` retournent des valeurs constantes (souvent nulles).

Les commandes TeX se classent en trois catégories, celles qui se développent (par exemple `\ifvoid`), celles qui modifient la mémoire (par exemple `\def`), et celles qui produisent un résultat dans le DVI, par exemple `\halign`, en général sous formes de boîtes imbriquées. Ceci est généralisé dans Tralics : une boîte nommée est traduite en un élément XML, qui peut avoir des attributs. Dans le cas de formules de mathématiques, on respecte les règles de MathML [8] en ce qui concerne les noms des éléments et attributs ; dans la plupart des autres cas, on peut paramétrer le logiciel et choisir les noms, qui sont ceux de la TEI par défaut. Une DTD est une description de l'ensemble des éléments autorisés et de leurs attributs (ainsi que des contraintes sur les attributs). Les deux DTD les mieux adaptées au rapport d'activité étaient DocBook [9] et TEI [10]. On a décidé de choisir une partie de la TEI (il y a trop d'éléments) et de l'étendre (il manque certains concepts dans la TEI). Le résultat est une DTD *ad hoc*, et il n'y a pas moyen de forcer Tralics à respecter un quelconque ensemble de règles.

Les commandes LaTeX sont a priori des commandes qui peuvent se développer, mais utiliser le code de `\chapter` ou de `\section` fait perdre

4. Par William F. Hammond, voir `http://www.albany.edu/~hammond/gellmu`.
5. Par Luc Maranget, voir `http://pauillac.inria.fr/~maranget/hevea`.
6. Par Eitan Gurari, distribué avec TeXLive 2008.
7. Par la Société EDP Sciences, voir `http://www.lxir-latex.org`.
8. Voir `http://www.w3.org/TR/2001/REC-MathML2-20010221`.
9. Voir `http://www.docbook.org`.
10. Voir le site web `http://www.tei-c.org`, et les *Cahiers Gutenberg* nº 24.

toute la sémantique, de sorte que ces commandes sont traitées comme des primitives ; dans le cas d'un traducteur comme TEX4ht, la traduction en HTML de l'argument (le titre) est un `<h1>` ou `<h2>`, il en est de même pour Tralics après passage par XML. Cependant Tralics produit un `<div1>` pour un chapitre et un `<div2>` pour une section, et sait que la fin d'un chapitre (et donc le début du chapitre suivant) entraîne la fin de la section. Des règles similaires s'appliquent à la commande `\item`. Elles sont implémentées par exemple dans ltx2x [11], qui traduit des documents LATEX simples, et qui a servi de maquette pour le RaWeb. Dans le cas des tableaux, l'utilisation du mode mathématique par l'environnement `tabular` ne peut se traduire en XML, car il est incompatible avec MathML, et passer par `\halign` est une perte d'information, c'est pour cela qu'il n'est pas possible d'utiliser `\halign` pour faire des tableaux. Par ailleurs, l'environnement `table` (de même que les autres flottants comme les notes en bas de page) est plus facile à implémenter en Tralics qu'en LATEX.

1.3. Fichiers de configuration

Diverses méthodes permettent de spécifier à Tralics le fichier de configuration à sélectionner, le cas le plus simple étant celui où la classe de document (comme 'ra2008') est associée à un fichier de configuration comme ra2008.tcf, ou est une classe standard comme 'book'. Le fichier de configuration, en plus du nom de la DTD, de macros TEX équivalent à un 'input' placé au tout début du document, ou de commandes définissant la page de titre, peut contenir des lignes de la forme

```
fullsection_vals = "/composition/Team/presentation/\
  Overall Objectives/fondements/Scientific Foundations/\
  domaine/Application Domains"
```

La commande `\tralics@find@config` prend un argument, par exemple « fullsection », son développement est une liste d'égalités, séparées par des virgules (au format keyval). La commande `\tralics@get@config` prend deux arguments, par exemple fullsection et composition, son expansion est Team (ou une erreur, le cas échéant). C'est ce mécanisme qui assure la cohérence entre les rapports des diverses équipes.

11. Par Peter R. Wilson, disponible sur CTAN.

Pour tenir compte des légères variantes d'une année à l'autre (par exemple, la syntaxe de la commande `\pers` a changé en 2006, des nouveaux arguments ayant été ajoutés), un même fichier de style 'ra.plt' est utilisé [12], mais avec des options différentes, par les fichiers de classes 'ra2007.clt' ou 'ra2008.clt' (le fichier de style peut également tester l'année courante, à savoir les 4 derniers caractères du nom du fichier source).

*

* *

Nous considérons dans la suite les documents sources suivant : apics2008.tex et thesis.tex ; il s'agit du rapport d'activité 2008 du projet Inria Apics, et de la thèse de C. Roméro. Nous expliquons dans les deux cas comment ce document est traduit en HTML. Dans le premier cas, la traduction est faite par le rédacteur ou un robot au fil de l'eau ; dans le second cas, nous avons traduit le manuscrit de la thèse, en ajoutant les modifications nécessaires, après la soutenance.

2. Les documents sources

2.1. Les métadonnées du rapport d'activité

Le rapport d'activité du projet Apics est le suivant

```
\documentclass{ra2008} \usepackage{...}
\theme{Num}
\isproject{Yes}
\projet{APICS}{Apics}{Analysis and Problems of Inverse
   type in Control and Signal processing}
\UR{Sophia}
\begin{document}
\maketitle
\begin{composition}
\pers{Laurent}{Baratchart}[Sophia]{Chercheur}{INRIA}
    [DR Inria][Habilite]
\pers{Maxim}{Yattselev}{PostDoc}{UnivFr}[Since September]
```

12. Tralics utilise les extensions clt et plt pour les analogues des fichiers LaTeX cls et sty (classes et *packages*).

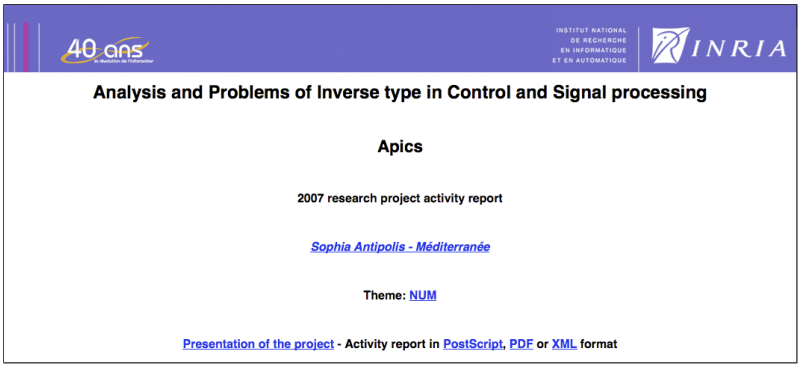


Figure 2. — Page de titre du Raweb 2007 du projet Apics.

```
\end{composition}
...
\loadbiblio
\end{document}
```

Les deux commandes `\maketitle` et `\loadbiblio` ne servent à rien (dans le cas du RaWeb, Tralics crée automatiquement la page de titre et inclut une bibliographie). Les métadonnées du document indiquent qu'il s'agit du rapport d'une équipe, ayant le statut de projet, localisée à Sophia Antipolis, dans le thème Num, de nom Apics, dont le nom développé est Analysis etc. L'argument de `\UR` est une liste de centres de recherche, séparés par des virgules, à choisir exclusivement dans une liste donnée dans le fichier de configuration. Pour des raisons historiques, l'argument peut être `\URxx\URyy` si l'on veut spécifier les centres xx et yy. Par conséquent la commande `\UR` range son argument dans une commande auxiliaire qui est ensuite traitée par la ligne

```
\expandafter\tralics@interpret@rc\expandafter{\ra@UR}
```

Ceci teste la validité, mémorise la liste des centres, et produit un élément XML. La commande `\pers` prend plusieurs arguments : nom, prénom, centre de rattachement, profession, affiliation, suivi de deux arguments optionnels (le dernier indiquant par sa présence si la personne est habilitée à diriger les recherches). Le centre de rattachement est facultatif sauf pour les équipes multilocalisées, auquel cas il faut choisir parmi la liste

des centres spécifiés plus haut ; l'argument est traité par le code suivant :

```
\edef\t@rc{\tralics@get@config{ur}{#1}}
```

Les valeurs de profession et affiliation sont à choisir dans une liste spécifiée par le fichier de configuration. Exemple de traitement de la profession.

```
\edef\t@pro{\tralics@get@config{profession}{#4}}
```

Le résultat est le suivant.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE raweb SYSTEM 'raweb7.dtd'>
<!-- Translated from latex by tralics 2.13.3,
                                date: 2008/11/18-->
<raweb language='english'  year='2008'
   creator='Tralics version 2.13.3'>
<accueil isproject='true' html='apics'>
<theme>num</theme>
<projet>Apics</projet>
<UR><URSophia/></UR>
...
<pers prenom='Laurent' nom='Baratchart'  hdr='Habilite'
   affiliation='INRIA' profession='Chercheur'
   research-centre='Sophia'>DR Inria</pers>
...
</biblio></raweb>
```

2.2. Le corps du rapport d'activité

La philosophie de Tralics est d'interpréter le document une seule fois, il n'y a donc pas de fichiers auxiliaires dans lesquels on peut lire et écrire. Les liens hypertextes (`\label` et `\ref`) se font via des attributs ID uniques (de la forme uid1, uid2, etc.), avec des variantes : l'identificateur unique du second chapitre est toujours cid2, et pour la bibliographie, on utilise bid1, bid2, etc. Par défaut, Tralics ne fait pas appel à `bibtex`, mais convertit lui-même le fichier de bibliographie (en fait les entrées utiles) en bbl, et traduit ceci en XML. De même, l'index est construit par Tralics. Par défaut, l'index et la bibliographie sont placés à la fin du document (en fait, comme le montre l'exemple plus bas, avant le `\endbackmatter` virtuel, ce qui pourrait changer dans une version ultérieure).

Figure 3. — Un module du raweb, montrant la table des matières à gauche, et le début d'un module à droite. Les métadonnées (mots-clés et participants) sont des liens actifs. Notons que le titre du module est placé après les mots clés.

Le rapport d'activité est formé de modules, regroupés en sections, dont voici quelques exemples.

```
\begin{module}{presentation}{pres}{Overall Objectives}
The Team aims at designing constructive methods ...
\end{module}

\begin{module}{fondements}{computer algebra}{Algebraic
 analysis approach to mathematical systems theory}
\begin{participants}\pers{Don}{Knuth}\end{participants}
\begin{motscle} machin, truc,bidule\end{motscle}
Many systems coming from mathematical physics ...
$x^2+y^2=z^2$. See \footcite{Knuth} and section \ref{toto}
\end{module}
```

Un module commence par des métadonnées (participants, mots clés) suivi d'un texte (qui peut être structuré en sous-modules via `\subsubsection`). Il est associé à une page HTML et est censé pouvoir être lu indépendemment des autres modules. Dans l'environnement participants la commande `\pers` a une syntaxe plus simple. La traduction XML du second module est la suivante

```
<module id-text='4' id='uid22'><head>Algebraic analysis
approach to mathematical systems theory</head>
<participants><pers prenom='Don' nom='Knuth'/>
</participants>
<keywords><term>machin</term>...
</keywords><p>Many systems coming from mathematical ...
<formula type='inline'><math xmlns='http://...'><mrow>
<msup><mi>x</mi><mn>2</mn></msup><mo>+</mo>...</formula>.
See <cit rend='foot'><ref target='bid82'/></cit>
and <ref target='uid90'/>
</p></module>
```

L'environnement module prend trois arguments, le nom de la section, le nom du module et le titre. Historiquement, le nom du module définissait le nom de la page web associée (à l'heure actuelle c'est la valeur de l'attribut *id* qui est utilisée). On peut faire référence à un module via son nom, ce qui impose l'unicité. Le fragment de code suivant montre comment Tralics teste cette unicité, le non-emboîtement des modules, et incrémente le compteur de module (la valeur du compteur se retrouve dans l'attribut *id-text* du module).

```
\ifra@inmodule\PackageError{Raweb}{Bad nesting}{}\fi
\ra@inmoduletrue
\refstepcounter{modules}%
\edef\foo{\noexpand\in@{,#3,}{\@allmodules}}\foo
\ifin@ \ClassError{Raweb}{Duplicate module: #3}{}\else
\xdef\@allmodules{,#3\@allmodules}\fi
```

Le premier argument de l'environnement module est le nom symbolique de la section. La commande `\tralics@get@config` considère « fullsection » comme un cas particulier : en plus de convertir « presentation » en « Overall Objectives » comme cela est indiqué dans le fichier de configuration, elle vérifie que les sections sont utilisées dans le bon ordre (présentation doit venir avant fondements) et accepte un argument vide

(cas où le module est dans la même section que le module précédent). L'expansion est vide sauf si on change de section ; dans ce cas il faut commencer une nouvelle section (le cas échéant, terminer la section précédente).

```
\edef\@tmp{\tralics@get@config{fullsection}{#2}}%
\ifx\@tmp\empty\else
  \ifra@firstsection\else \tralics@pop@section\fi
   \global\ra@firstsectionfalse
   \typeout{Translating section #2}%
   \tralics@push@section{#2}
   \ifnum\ra@year>2006 \XMLaddatt{titre}{\@tmp}\fi
\fi
```

2.3. La thèse

Le second document que nous allons considérer est la thèse de Céline Roméro (elle existe en français et en anglais). Pour faire la traduction en HTML, certaines modifications ont été nécessaires. Si le fichier source s'appelle thesis.tex, Tralics charge automatiquement le fichier thesis.ult s'il existe. Celui-ci contient par exemple une définition de `\varnothing` se traduisant en ∅ (la commande n'existait pas à l'époque), une redéfinition de `\item` en `\@item` (l'argument optionnel de la commande `\item` est un attribut au lieu d'être un élément). Nous avons écrit un fichier jghack.tex contenant des redéfinitions conditionnelles. Par exemple on définit

```
\def\jgvag{\va g}
\def\aspfreq{Asp$_{Fr\acute{e}quentatif}$\xspace}
```

et on redéfinit conditionnellement

```
\def\jgvag{^^^^0292}
\def\aspfreq{Asp\textsubscript{Fréquentatif}\xspace}
% Asp$_{Frequentative}$
```

La première ligne montre comment on peut utiliser n'importe quel caractère Unicode 16bits dans Tralics (cette syntaxe apparaît dans Ω, et est reprise par XeTeX et luatex). Quand on exécute Tralics avec l'option trivialmath=7, la formule de math de la troisième ligne est considérée comme triviale, et la traduction est un indice en mode texte. Dans la version française, la formule n'est plus triviale à cause de la commande `\acute`, et c'est pour cela qu'il faut forcer l'utilisation de

\textsubscript. Nous avons modifié le texte pour remplacer tous les \va g par des \jgvag. Nous avons également dû remplacer dans une formule de math la première ligne du code suivant par la seconde. Cette modification n'est plus nécessaire dans la version courante de Tralics, un changement de fonte dans une boîte dans une formule de math n'étant plus une erreur (il n'est pas encore possible de demander une variante de police pour le lettres grecques en mode mathématiques).

```
\mbox{<{\bf $\alpha$}we>} \\
\mbox{<}\alpha\mbox{we>} \\
```

2.4. Les métadonnées de la thèse

Le document est le suivant :

```
\input{jghacks.tex}
\documentclass[12pt,leqno,fleqn]{book}
\toplevelsection{\chapter}
\font\va=cmoer10 scaled 1200
\usepackage{gb4e}
\usepackage{tree-dvips}
\usepackage{titlepage}
...
\begin{document}
\begin{metadata}...\end{metadata}
\frontmatter
\include{Dedication}
\include{Thanks}
\tableofcontents
\include{Abbreviations}
\mainmatter
...
\nocite{*} \bibliographystyle{these} \bibliography{these}
\end{document}
```

Il s'agit d'une thèse, utilisant la classe 'book'. Il n'y pas de \maketitle, les métadonnées seront expliquées plus bas. La troisième ligne indique que le document est découpé en chapitres (il n'y a pas de parties, la traduction d'un chapitre sera donc <div0>). La quatrième ligne définit une commande \va complètement ignorée par Tralics dont on reparlera. Le fichier de style 'gb4e' définit les gloses, tandis que 'tree-dvips' sert à construire les arbres (par exemple celui de la figure 4).

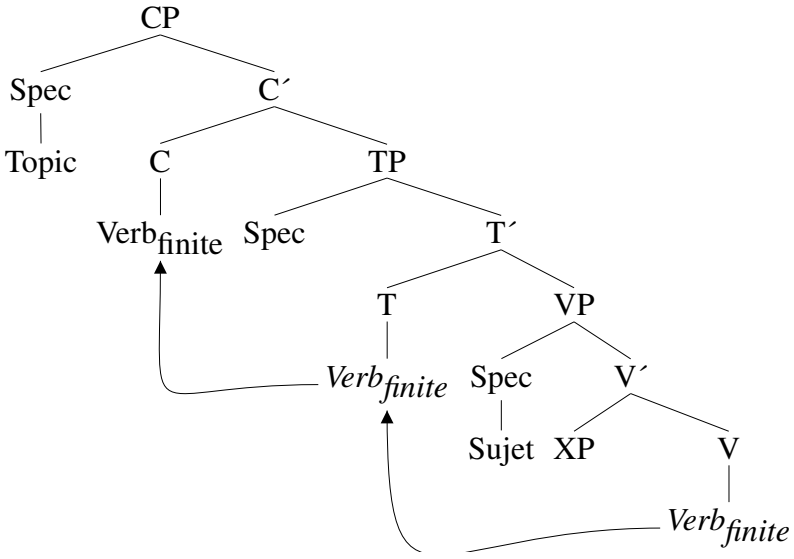


Figure 4. — Un exemple d'arbre.

Il existe plusieurs classes pour produire des thèses sur CTAN, mais à ma connaissance aucune n'a été conçue pour des thèses soutenues en France[13] ; en général les doctorants utilisent une classe standard et composent la page de titre à l'aide de tableaux. Nous avons édité cela pour y ajouter de la sémantique.

```
\begin{metadata}
\begin{center}
\lieuthese{Université Paris III - La Sorbonne Nouvelle}
\lieuthesesuite{U.F.R. d'Anglais}
\typethese{Linguistique}
\doctorant{Céline}{Roméro}
\titrefrancais{L'évolution syntaxique des verbes modaux
dans l'histoire de l'anglais}
```

13. On peut télécharger des classes comme thloria sur le web.

```
\thesetitre{The Syntactic Evolution of Modal Verbs in
the History of English}
\directeur{Jacqueline}{Guéron}
\datesoumission{November, 18th 2005}
\end{center}
\begin{jury}
\membre[President of the Jury, ...]{Mr}{Claude}{Delmas}
\membre[Supervisor, ...]{Mrs}{Jacqueline}{Guéron}
\membre[Lecturer at Paris III]{Mrs}{Annie}{Lancri}
\membre[Research Director ...]{Mrs}{Jacqueline}{Lecarme}
\membre[Prof. University of York]{Mrs}{Susan}{Pintzuk}
\end{jury}
\end{metadata}
```

Toutes les commandes utilisées pour la page de titre sont définies dans le fichier de style 'titlepage' dont voici un extrait :

```
\newcommand\membre[4][]{%
 \xbox{membre}{#1\XMLaddatt{type}{#2}%
   \XMLaddatt{prenom}{#3}\XMLaddatt{nom}{#4}}}
\newenvironment{jury}
    {\begin{xmlelement*}{jury}}{\end{xmlelement*}}
```

Cet exemple montre comment créer des éléments XML (qui peuvent être manipulés comme les boîtes TEX) et modifier les attributs. Nous montrons ici quelques unes de 26 000 lignes produites par Tralics.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE std SYSTEM 'classes.dtd'>
<!-- Translated from latex by tralics 2.13.4,
                                date: 2008/11/18-->
<std flushed-equation='true' equation-number='left'
   chapters='true' part='true'>
<metadata>
<lieuthese>Université Paris III -  ...</lieuthese>...
<doctorant>Céline Roméro</doctorant>...</metadata>
<frontmatter><dedicace>...
</citation></biblio></div0></backmatter></std>
```

2.5. Le corps de la thèse

Il s'agit d'un document LATEX standard, qui ne fait pas appel à un fichier de configuration ni à des astuces pour regrouper les modules en

(10) Đa gewænde seo wydewe ham...
Then went-PRET the-NOM widow-NOM back home...
Then, the widow went back home... (coaelive,ÆLS_[Eugenia]:144.277)

(12)

Đa gewænde seo wydewe ham...
Then went-PRET the-NOM widow-NOM back home...

Then, the widow went back home... (coaelive,ÆLS_[Eugenia]:144.277)

Figure 5. — Un exemple de glose, version originale en PostScript, et sa traduction en HTML.

des sections. Il y a par contre des difficultés, montrées par les lignes suivantes dont la version PostScript et la traduction HTML est donnée sur l'image 5 :

```
\begin{exe}
\ex
\gll {\jgvaD}a gew{\ae}nde seo wydewe ham...\\
  Then went-{\footnotesize PRET} the-{\footnotesize NOM}
widow-{\footnotesize NOM} back~home... \\
\glt {\em Then, the widow went back home...}
(coaelive,{\AE}LS\_[Eugenia]:144.277)
\end{exe}
```

L'environnement `exe` est défini par le *package* 'gb4e'. Il sert à produire des exemples, définis par la commande `\ex` ou `\exj` [14]. On peut considérer ceci comme une énumération où `\ex` joue le rôle de `\item`, mais un compteur global est utilisé. La traduction est

```
<list type='description'>
<item id-text='12' id='uid144' label='12'>
...
<p noindent='true'>
  <hi rend='it'>Then, the widow went back home...</hi>
(coaelive,ÆLS_[Eugenia]:144.277)</p>
</item></list>
```

14. Cette commande correspond à `\item[...]`, qui est mal interprétée par Tralics, son utilisation explique le décalage entre les numéros sur la figure.

L'exemple contient une glose à deux lignes (il y a également possibilité d'utiliser des gloses à trois lignes via `\glll`) terminée par une traduction (introduite par `\trans` ou `\glt`) suivie d'une référence (ici c'est le corpus, Ælfric's Lives of Saints, les chiffres 144 et 277 restent mystérieux pour moi). La première ligne de la glose est une suite de mots en ancien anglais, la seconde est une traduction mot à mot annotée, et la dernière est une traduction intelligible. En fait, pour garantir l'alignement vertical, LaTeX crée une suite de boîtes (contenant par exemple ham au dessus de back home) séparés par des espaces (un retour à la ligne est possible entre deux boîtes). Ceci est traduit par Tralics comme si c'était un tableau [15]. Il n'y a pas de retour à la ligne dans les tableaux, ce qui peut poser problème si on veut imprimer le HTML. L'expansion de la glose est la suivante

```
\begin{tabular}{}
{\jgvaD}a&gew{\ae}nde&seo&wydewe&ham...\\
  Then&went-{\footnotesize PRET}&the-{\footnotesize NOM}
widow-{\footnotesize NOM}&back~home... \\
\end{tabular}
```

Le source du document contenait `{\va D}` qu'on a remplacé par `{\jgvaD}`. On rappelle que la commande `\va` sélectionne une fonte Vieil Anglais, et que lorsque Tralics voit la lettre D, il ne se pose pas la question de savoir quelle est la fonte courante. On pourrait redéfinir la commande `\va` comme `\it` de sorte que la traduction soit de la forme

```
<hi rend='va' font='cmoer10 scaled 1200'>D</hi>
```

Comment alors traduire ceci en HTML, sachant que la fonte cmoer10 est une fonte non standard (variante old english de Computer Modern Roman) ? Nous avons eu de la chance que seuls trois caractères (g, d et t) soient utilisés dans cette fonte, et qu'ils ressemble beaucoup à `\dh` et `\th`. La modification du texte source n'a pas été trop difficile.

L'autre particularité du document est la suivante

```
\begin{exe}
\ex\label{cpv2a}
\begin{tabular}[t]{cccccccc}
&\node{A}{CP}\\
```

15. Ce qui explique pourquoi la glose est centrée et le numéro de l'exemple n'est pas aligné avec la première ligne.

```
\node{B}{Spec} &&\node{a}{C'}\\
\node{BB}{Topic} &\node{b}{C} &&\node{c}{TP}\\
&\node{bb}{Verb$_{finite}$} &\node{d}{Spec} &&...\\
&&&\node{f}{T} &&\node{g}{VP}\\
&&&\node{ff}{{\em Verb$_{finite}$}} &...\\
&&&&\node{hh}{Sujet} &\node{j}{XP} &&\node{k}{V}\\
&&&&&&&&\node{kk}{{\em Verb$_{finite}$}}\\
\nodeconnect{A}{B}
...
\nodeconnect{k}{kk}
\anodecurve[l]{kk}[b]{ff}{2.5cm}
\anodecurve[l]{ff}[b]{bb}{2cm}
\end{tabular}
\end{exe}
```

La traduction XML est donnée plus bas, le rendu HTML sur la figure 4. On notera quelques points intéressants. La plupart des connecteurs sont formés de lignes droites, sauf ceux de kk à ff et ff à bb, qui sont courbes. Cette courbe peut dépasser du cadre du tableau (voir la figure), soit par en bas, soit par la gauche. L'ensemble des connecteurs est dans une unique cellule, dans une ligne en fin de tableau.

```
<list type='description'>
<item id-text='11' id='uid143' label='11'>
<table vpos='t' rend='inline'><row><cell halign='center'/>
<cell halign='center'><node name='A'>CP</node></cell>
...
<cell halign='center'><node name='kk'>
  <hi rend='it'>Verb<hi rend='sub'>finite</hi></hi></node>
</cell>
</row>
<row><cell halign='center'><nodeconnect nameA='A'
   nameB='B' posA='b' posB='t'/>
<nodeconnect nameA='B' nameB='BB' posA='b' posB='t'/>
...
<anodecurve nameA='ff' nameB='bb' posA='l' posB='b'
   depthB='2cm' depthA='2cm'/>
</cell>
</row></table>
</item></list>
```

3. Traduction en HTML

3.1. Principes généraux

La simplicité du langage XML permet d'écrire très facilement des programmes qui lisent ou écrivent du XML, et avec l'avènement de XHTML, on peut considérer HTML comme une variante de XML. Un processeur XSLT (par exemple *xsltproc*, installé par défaut sur un grand nombre de machines), est un mécanisme qui transforme du XML via des feuilles de style, écrites en XML, plus précisément XSL. C'est un langage très verbeux, voir les extraits qui suivent (pour le rapport d'activité, il y 46 feuilles de styles, 1500 lignes au total).

Le principe de XSL est le suivant [16]. Il y a des templates, qu'on appellera règles ou fonctions, selon qu'ils sont définis par l'attribut *name* ou *match*. Les règles s'appliquent à des objets (en général des éléments) suivant un mode et une position. On peut ainsi définir une règle pour le cas où `<ref>` est dans un `<cit>` (traduction par Tralics d'un `\cite`) et une autre règle pour les autres cas (traduction par Tralics d'un `\ref`). Un même élément peut avoir plusieurs traductions. Il y a par exemple cinq modes pour une section : en plus du mode normal, il y a le mode xref (qui produit le numéro de section), le mode xreftitle (qui produit l'attribut *title* dans les liens), le mode xtoc (qui produit une entrée dans la table des matières) et le mode prefix (qui calcule le renfoncement dans la table des matières).

Le résultat de la transformation est l'application de la règle appropriée à l'élément racine. C'est un arbre XML, qui peut être sérialisé de plusieurs façons différentes. Par exemple une feuille de style convertit le XML du rapport d'activité en un autre XML, avec changement de DTD ; pour déterminer l'ensemble des fichiers HTML d'un projet donné on utilise une autre feuille de style, très simple (30 lignes), qui imprime en texte pur, l'attribut *id* de chaque module (dans la nouvelle DTD, c'est un `<subsection>` qui n'est pas dans un `<subsection>`). Dans les exemples qui suivent, le résultat sera un fichier HTML.

Les règles de traduction sont souvent de la forme : X se traduit en Y, les attributs de X sont recopiés dans Y, puis le corps de X est traduit et mis

16. Pour une introduction aux concepts traités ici, on pourra se reporter à : Michel Goossens, « XML et XSL : un nouveau départ pour le web », *Cahiers GUTenberg* nº 33-34 (1999), p. 3-126.

dans l'arbre (donc dans Y), ceci se fait via `apply-templates`, abrégé en AT dans la suite. Lorsque X a une structure connue à l'avance, on peut procéder de façon différente : on considère un arbre Y dans lequel on insère à certains endroits la traduction de parties X1, X2, X3 de X, via `call-template`, abrégé en CT dans la suite. Par exemple la figure 2 est le résultat de la traduction de l'élément `<accueil>`, et l'on voit les éléments `<theme>`, `<UR>`, etc. ; l'élément `<projet>` est utilisé trois fois (texte pur au milieu, sous le nom « presentation of the project » en bas à gauche, et sous la forme d'un lien dans le coin en haut à droite).

Dans le cas d'un titre, la traduction par défaut est vide. Dans le cas d'une section, il n'y pas création d'un nouvel élément pour le contenu entier, il y a un juste un `<h1>` ou `<h2>` pour le titre, qui sera traduit via un AT. Cependant le numéro de section est calculé via CT (dans le cas du RaWeb, ces numéros n'apparaissent pas dans le HTML comme cela est visible sur la figure 3).

Voici un extrait d'une feuille de style. Il définit une règle simple qui s'applique à un élément `<div0>`, en mode fileprefix. Le résultat est la concaténation d'une variable globale, de l'attribut *id* de l'élément courant et d'une chaîne de caractères fixe. Le contenu de l'élément est ignoré.

```
<xsl:template match="div0" mode="fileprefix">
  <xsl:value-of select="concat($prefix,@id,'.html')" />
</xsl:template>
```

Une feuille de style peut importer une autre feuille de style. La feuille de style principale pour la thèse est thesishtml.xsl (et thesehtml.xsl pour la version française, qui définit certains termes en français). Elle utilise la feuille de style thesis.xsl (qui contient les règles applicables aux métadonnées), et cls.xsl, qui contient toutes les autres règles. La feuille de style principale qui sert pour la documentation HTML de Tralics s'appelle tralics-rrhtml.xsl, elle importe RR.xsl pour les métadonnées et cls.xsl pour le reste. Une variante de ce fichier, à savoir clsb.xsl, permet d'exploiter les champs *id-text*.

3.2. Pages multiples

Un unique document XML peut être traduit en un ensemble de pages HTML. Dans le cas du rapport d'activité il y a une page principale (contenant les métadonnées, voir figure 2, et la table des matières) une biblio-

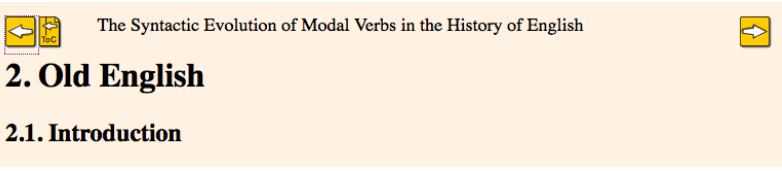


Figure 6. — Début du second chapitre de la thèse, montrant les boutons de navigation.

graphie, et une page par module. Dans le cas de la thèse, la page principale contient les métadonnées, la table des matières, l'introduction, la conclusion, les abréviations. Chaque chapitre est dans un fichier séparé. On peut paramétrer la feuille de style pour avoir une page séparée pour l'index ou les notes. Ces pages ne sont pas indépendantes, dans la mesure où il y a des boutons de navigations pour aller à la page suivante ou la page précédente. De plus, des liens internes peuvent devenir externes. Il faut donc un algorithme de nommage de ces pages. Initialement, le deuxième argument de l'environnement module du RaWeb définissait le nom de la page, maintenant c'est son identificateur unique (uid1, uid3, etc.). Dans le cas de la thèse et de la documentation de Tralics, on utilise l'identificateur unique du chapitre cid1, cid2, avec un préfixe. La règle XSL donnée plus haut calcule le nom du fichier, par exemple thesiscid2.html.

La règle qui suit est appliquée pour l'élément racine de la thèse. Le seul point non trivial concerne la mini-table des matières placée en tête du document (voir figure 7) : elle contient juste des liens vers les chapitres. Elle n'a pas lieu d'être si le document est découpé en petits morceaux (c'est-à-dire si le fichier principal est essentiellement une suite de liens) ou si la table des matières est en tête de document.

```
<xsl:template match="std">
  <html>
    <xsl:CT name="html-meta" />
    <body>
      <xsl:CT name="header"/>
       <xsl:if test="$shorttoc='true' or
          ($shorttoc='maybe' and $split='false')">
         <xsl:CT name="shorttoc"/>
```

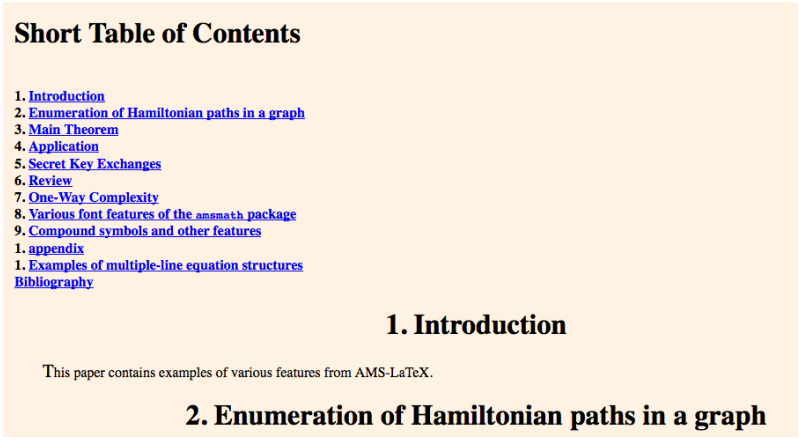


Figure 7. — Fichier de test amsmath, montant la mini-table des matières et le début du texte.

```
      </xsl:if>
     <xsl:AT/>
     <xsl:CT name="footnotes" />
   </body>
  </html>
</xsl:template>
```

Le code qui suit est appliqué lorsqu'un chapitre est traduit en page HTML (à la vérité, c'est du XHTML, au format XML). On notera les boutons de navigation en haut et, en bas de page, le bouton qui permet de revenir vers la page principale. Le début de la page contient le titre et le numéro de chapitre (voir figure 6).

```
<xsl:template match="div0" mode ="xsplit">
  <xsl:variable name="filename">
    <xsl:AT select="." mode="fileprefix" />
  </xsl:variable>
  <xsl:document href="{$filename}"
   encoding='iso-8859-1'
   doctype-public='-//W3C//DTD XHTML 1.0 Strict//EN'
   doctype-system='http://www.w3.org/TR/xhtml1/DTD/...'
   method="xml" >
   <html>
```

```
    <xsl:CT name="html-meta" />
     <body>
       <xsl:AT select="." mode="button" />
       <div class="..."><xsl:CT name="title" /></div>
       <h1 id="@id">
         <xsl:CT name="calculateNumberSpace"/>
         <xsl:CT select="head" mode="full"/>
       </h1>
       <xsl:AT/>
       <xsl:AT select="." mode="button" />
       <xsl:AT name="back.to.main" />
     </body>
   </html>
  </xsl:document>
</xsl:template>
```

Le rapport d'activité utilise des principes similaires ; cependant la commande `<xsl:document>` n'étant pas standard, elle n'est pas utilisée. On récupère comme expliqué plus haut l'ensemble des identifiants des modules, pour chaque identifiant X, on lance le processeur XSLT en lui passant X comme argument. La feuille de style utilise une variable globale Y, qui est l'élément dont l'identifiant est X, et la règle associée à l'élément racine est simplement de mettre la traduction de Y dans un élément `<html>`.

3.3. Liens internes

Décrire le fonctionnement des liens internes dans LaTeX n'est pas simple : dans l'index du LaTeX Companion [7], la commande `\label` apparaît 42 fois et la commande `\ref` apparaît 35 fois. On peut affecter une étiquette à un *élément structurel* que l'on peut utiliser ensuite. Par exemple ce document contient `\label{toto}` à la fin du second alinéa de la section 2.2 page 37. Ceci affecte à *l'élément « actif » courant* l'étiquette toto. La commande `\ref` compose une chaîne identifiant l'élément donné, qui est la *chaîne de référence courante*, qui est « normalement déterminée en début de l'élément et mise à jour lorsqu'il se termine » (les termes en italiques et entre guillemets sont des citations). À chaque étiquette peuvent être associés un texte et un numéro de page utilisable par `\pageref`, et dans le cas d'un document hypertexte une

ancre (position dans une page, qui parfois, malheureusement, n'a rien à voir avec le texte ou le numéro de page).

On se pose ici la question de traduire en HTML un certain élément X, traduction par Tralics d'un certain `\ref`. C'est un `<ref>` qui ne se trouve pas à l'intérieur d'un `<cit>`. Il a une cible, l'attribut *target* (qui correspond à l'étiquette LaTeX, mais XML est moins permissif sur ce point). Il existe un unique élément (appelé « ancre » dans la suite) qui a cette cible comme identifiant unique, comme valeur de l'attribut *id*[17]. Cet élément Y est la traduction de l'élément « actif » courant mentionné plus haut, et son attribut *id-text* est la chaîne de référence courante. Dans le cas de l'exemple de l'arbre, l'ancre est un `\item` et l'argument optionnel est rangé dans l'attribut *label*, qui a priorité en tant que chaîne de référence lors de la conversion en HTML. Si on fait suivre `\chapter*` d'un `\label`, on se trouve dans un cas particulier où l'élément XML courant est le chapitre, qui ne peut pas servir d'ancre, et dans ce cas Tralics utilise la première ancre du document (à savoir uid1, ce qui peut être erroné). Si la traduction de X est XX et celle de Y est YY, on maintient la condition que l'attribut *id* de YY est le même que l'attribut *href* de XX.

En HTML traditionnel, une ancre est un élément `<a>`, et les deux éléments XX et YY sont des ancres ; l'identificateur unique est dans l'attribut *name*. En XHTML on peut utiliser *id* ou *name*, on peut même utiliser les deux, pourvu que la valeur soit la même (il s'agit là d'une exception à la règle XML qui dit qu'un élément peut avoir au plus un attribut de type ID), et beaucoup d'éléments peuvent servir d'ancre. Pour trouver Y connaissant X, on peut parcourir tout l'arbre XML pour y chercher un élément ayant le bon attribut, ou chercher dans une table précalculée (qui associe à tout identifiant unique l'élément adéquat), c'est le cas de la fonction id utilisée ci-dessous. Cela ne fonctionne que si la DTD spécifie que l'attribut *id* de Y est bien de type ID[18].

17. Il est possible d'attribuer à n'importe quel élément n'importe quel attribut, en particulier *id*. On envisage de modifier la création des uid dans Tralics pour tenir compte du problème. Il est aussi possible de dupliquer l'ancre ou de la mettre dans une boîte inutilisée. L'existence et l'unicité de l'ancre ne peuvent pas toujours être garanties.

18. Le fichier classes.dtd ne sert que pour cela.

```
<xsl:template match="ref">
  <a>
    <xsl:CT name="target-to-href"/>
    <xsl:AT mode="xreftitle" select="id(@target)" />
    <xsl:AT mode="xref" select="id(@target)" />
  </a>
</xsl:template>
```

Le code ci-dessus dit que la traduction XX de X est une ancre, un élément `<a>`. Il a un attribut *href*, un attribut *titre* éventuel et un contenu, ces deux dernières informations ne dépendent que de Y. Par exemple, si Y est une équation, le contenu peut être le numéro d'équation. Le titre est le texte affiché par certains navigateurs lorsqu'on passe la souris sur l'ancre, ceci peut être les 100 premiers caractères pour une note en bas de page.

L'attribut *href* est de la forme `#uid17`. Il y a cependant une difficulté : si un unique source XML est découpé en multiples fichiers HTML un lien interne peut devenir externe et il faut ajouter le nom du fichier dans lequel se trouve YY. Ceci se fait en appliquant une certaine procédure à Y, comme le montre le code suivant.

```
 <xsl:template name="target-to-href">
  <xsl:attribute name="href">
    <xsl:AT mode="targetfile" select="id(@target)" />
    <xsl:text>#</xsl:text>
    <xsl:value-of select="./@target"/>
  </xsl:attribute>
</xsl:template>
```

La règle suivante s'applique à un élément Y, elle rend le nom de la page HTML contenant sa traduction YY. Si la variable *split* est fausse, un seul fichier HTML est produit, et tous les liens sont internes. Si la variable split est all, chaque chapitre produit une page HTML, même ceux qui ne sont pas dans un `\mainmatter` (ce qui est le cas typique d'un rapport).

```
<xsl:template match="*" mode="targetfile">
 <xsl:choose>
  <xsl:when test="$split='false'" />
  <xsl:when test="not(ancestor-or-self::div0)">
    <xsl:CT name="mainfile" />
  </xsl:when>
  <xsl:when test="(ancestor::mainmatter) or $split='all'">
```

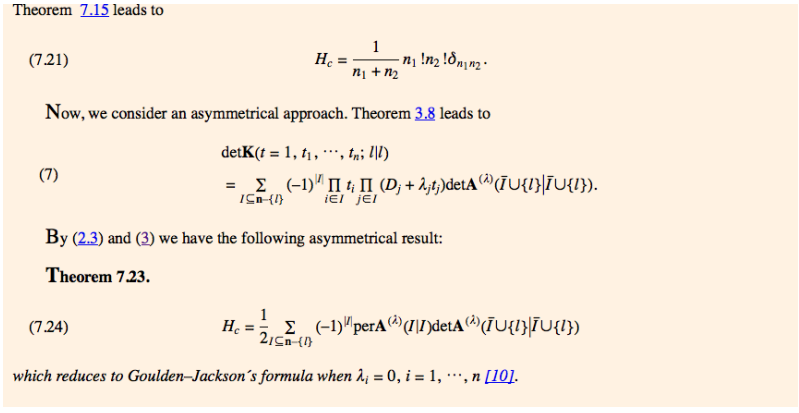
```
    <xsl:AT select="ancestor-or-self::div0"
          mode="fileprefix" />
  </xsl:when>
  <xsl:otherwise>
    <xsl:CT name="mainfile" />
  </xsl:otherwise>
 </xsl:choose>
</xsl:template>
```

Le code suivant calcule le numéro d'une table, en comptant le nombre de tables qui la précèdent. Dans un certain nombre de cas, le flottant (`table` en LaTeX) contient la table (`tabular` en LaTeX) avec une légende. Dans ce cas la traduction est un élément `<table>`; le cas où le flottant contient plusieurs tables, plusieurs légendes, ou autre chose n'est pas traité par défaut. Dans le cas de tables non flottantes, il y a un attribut *rend* qui est inline. L'utilisation de l'attribut *id-text* permet par exemple d'avoir des flottants non numérotés ou d'implémenter proprement la commande `\tag`. Par ailleurs, il est beaucoup plus simple de demander à Tralics de faire suivre le théorème 7.23 par l'équation 7.24 (voir figure 8), que de coder la même chose dans XSL.

```
<xsl:template match='table' mode="xref">
  <xsl:CT name="calculateTableNumber"/>
</xsl:template>
<xsl:template name="calculateTableNumber">
  <xsl:choose>
    <xsl:when test="@id-text">
      <xsl:value-of select="@id-text"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:number count="table[ @rend != 'inline']"
        level="any"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

3.4. Mathématiques et mise en page

Nous commentons ici le code qui produit les figures 8 et 9. En ce qui concerne la partie verbatim, c'est un jeu d'enfant de remplacer l'élément `<latexcode>` (résultat de la commande `\verb`) par `<samp>`, et laisser



Theorem 7.15 leads to

$$H_c = \frac{1}{n_1+n_2} n_1!n_2!\delta_{n_1 n_2}. \tag{7.21}$$

Now, we consider an asymmetrical approach. Theorem 3.8 leads to

$$\begin{aligned}&\det\mathbf{K}(t=1,t_1,\cdots,t_n;I|I)\\&=\sum_{I\subseteq\mathbf{n}-\{I\}}(-1)^{|I|}\prod_{i\in I}t_i\prod_{j\in I}(D_j+\lambda_j t_j)\det\mathbf{A}^{(\lambda)}(\bar{I}\cup\{I\}|\bar{I}\cup\{I\}).\end{aligned} \tag{7}$$

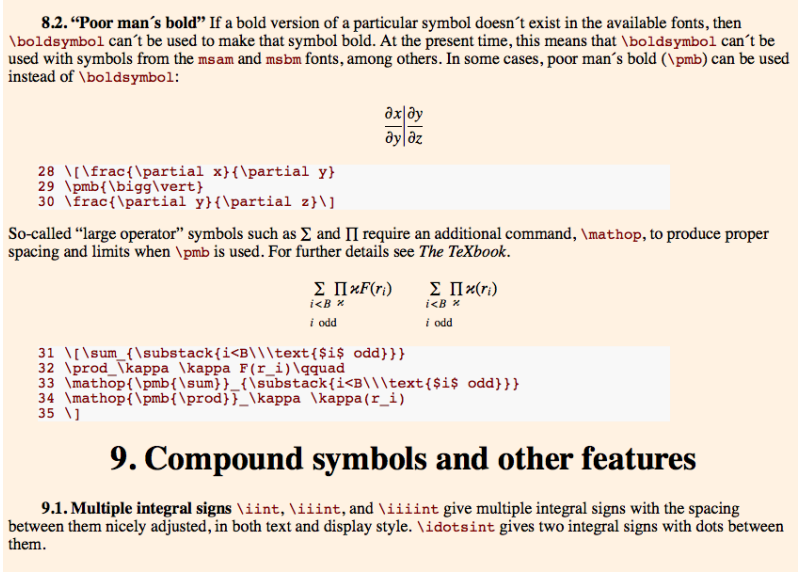
By (2.3) and (3) we have the following asymmetrical result:

Theorem 7.23.

$$H_c = \frac{1}{2}\sum_{I\subseteq\mathbf{n}-\{I\}}(-1)^{|I|}\mathrm{per}\mathbf{A}^{(\lambda)}(I|I)\det\mathbf{A}^{(\lambda)}(\bar{I}\cup\{I\}|\bar{I}\cup\{I\}) \tag{7.24}$$

which reduces to Goulden–Jackson's formula when $\lambda_i = 0$, $i = 1, \cdots, n$ *[10].*

Figure 8. — Exemple de formules de mathématiques. L'affichage est fait par FireFox. Les numéros d'équation et de théorème proviennent de l'attribut *id-text,* lorsque celui-ci est présent.



8.2. "Poor man's bold" If a bold version of a particular symbol doesn't exist in the available fonts, then `\boldsymbol` can't be used to make that symbol bold. At the present time, this means that `\boldsymbol` can't be used with symbols from the `msam` and `msbm` fonts, among others. In some cases, poor man's bold (`\pmb`) can be used instead of `\boldsymbol`:

$$\frac{\partial x}{\partial y}\Big|\frac{\partial y}{\partial z}$$

```
28 \[\frac{\partial x}{\partial y}
29 \pmb{\bigg\vert}
30 \frac{\partial y}{\partial z}\]
```

So-called "large operator" symbols such as $\sum$ and $\prod$ require an additional command, `\mathop`, to produce proper spacing and limits when `\pmb` is used. For further details see *The TeXbook*.

$$\sum_{\substack{i<B\\ i\text{ odd}}}\prod^{\kappa}F(r_i) \qquad \sum_{\substack{i<B\\ i\text{ odd}}}\prod^{\kappa}\kappa(r_i)$$

```
31 \[\sum_{\substack{i<B\\\text{$i$ odd}}}
32 \prod^{\kappa} F(r_i)\qquad
33 \mathop{\pmb{\sum}}_{\substack{i<B\\\text{$i$ odd}}}
34 \mathop{\pmb{\prod}}^{\kappa} \kappa(r_i)
35 \]
```

9. Compound symbols and other features

9.1. Multiple integral signs `\iint`, `\iiint`, and `\iiiint` give multiple integral signs with the spacing between them nicely adjusted, in both text and display style. `\idotsint` gives two integral signs with dots between them.

Figure 9. — Exemple de mise en page, et de code verbatim.

l'élément <pre> intact, la couleur [19] étant définie par le fichier tralics.css. Le titre du chapitre est centré, grâce à un attribut *style*. Le cas de la sous-section est plus subtil : la traduction du titre est a priori vide.

Nous expliquons ici la traduction de l'élément <p>. Dans le cas où l'élément <p> est dans un environnement verbatim, aucun nouvel élément n'est créé (cela supprime le renfoncement et l'espace interparagraphe).

```
<xsl:template match="p">
  <xsl:choose>
    <xsl:when test="parent::pre">
      <xsl:AP/>
    </xsl:when>
```

Cas où l'élément <p> est le second fils d'un théorème (théorème simple) ou le troisième fils d'un théorème, le second étant <alt-head> (théorème avec argument optionnel). Dans les deux cas, le théorème est traité en mode first-par (ce qui a pour effet d'afficher le mot « Theorem », i.e. le premier fils, suivi du numéro an caractères gras). L'argument optionnel est traité le cas échéant.

```
    <xsl:when><!-- th. simple, voir texte--></xsl:when>
    <xsl:when test="parent::theorem and position()=3
        and preceding-sibling::alt_head[1]">
     <p>
      <xsl:AP select=".." mode="first-par"/>
      <xsl:AP/>
     </p>
    </xsl:when>
```

Cas où l'élément <p> est le second fils d'une section. Le traitement est le même que dans le cas d'un théorème, sauf que le numéro précède le titre.

```
   <xsl:when test="parent::div1 and position()=2">
     <p class='nofirst'>
      <xsl:AP select=".." mode="first-par"/>
      <xsl:AP/>
     </p>
    </xsl:when>
```

19. Dans la documentation Tralics on utilise deux couleurs différentes suivant que le verbatim est du LaTeX ou du XML. Cette information de couleur n'est pas visible dans une version noir et blanc.

Autres cas. La traduction est un <p>, avec des attributs (si le texte doit être centré, on utilise l'attribut *style*, si le paragraphe ne doit pas être renfoncé on utilise l'attribut *class* (avec les valeurs nofirst et noindent) et si un attribut *class* est présent on le recopie).

```
    <xsl:otherwise>
      <p>  ...
          <xsl:attribute name="class">
         <xsl:value-of select="@class"/></xsl:attribute>
        </xsl:if>
        <xsl:AP/>
      </p>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Les quatre lignes suivantes du fichier tralics.css expliquent pourquoi les paragraphes sont renfoncés par défaut, et leur premier caractère est plus grand, et comment annuler cet effet.

```
p {text-indent:2em}
p:first-letter {font-size: 130%;}
.noindent{ text-indent:0em}
p.nofirst:first-letter{ font-size:100% }
```

Le traitement des équations de mathématiques est un peu compliqué. Le résultat est un élément avec un attribut *class* qui pourrait être utilisé par un fichier css. Dans le cas d'une équation numérotée, le résultat est un tableau à une ligne et 3 colonnes. La seconde colonne contient la formule, l'une des deux autres contient le numéro d'équation. Via un attribut *class* et un fichier css on demande au navigateur de réserver la même largeur pour chaque colonne, de sorte que la formule soit centrée. Voici par exemple ce qui est prévu pour la colonne de droite

```
.eqno {
    font-style: normal;
    text-align:right;
    width:4em;
    text-indent:0px;}
```

Les deux figures 8 et 9 montrent un document HTML où les formules de mathématiques sont en MathML. Si le fichier s'appelle toto.html, les maths ne sont pas reconnues par Firefox, s'il s'appelle toto.xml, il n'est pas reconnu par Internet Explorer (qui a un plug-in MathPlayer qui permet de visualiser les maths). Ni Safari ni Opera ne reconnaissent pour l'instant le MathML.

Pour cette raison, les formules de math sont traduites en images dans le RaWeb. Il en est de même des arbres de syntaxe de la thèse, voir figure 4. Le processus se fait en deux étapes. On commence par construire un fichier XML contenant tous les objets XML à traduire. Dans le cas de la thèse, c'est fait par une feuille de style qui ne conserve que les éléments `<preview>` et les arbres, en fait tous les éléments `<table>` qui contiennent un élément `<node>`. Dans le cas du RaWeb, il ne faut garder que les formules de math non triviales, et c'est le script Perl décrit plus bas qui s'en occupe. Le tout est mis dans un élément `<fo:block>` dans un `<fo:root>`. Chaque arbre est enrobé par un élément `<tree>` et chaque formule de math par un élément `<formula>`. On demande alors à TEX d'interpréter ce fichier (c'est la même procédure qui convertit le XML en PDF, sauf que l'on produit du DVI, et l'enrobage fait changer de page pour chaque objet). On convertit ce DVI en PostScript (et `dvips` va tracer les connecteurs des arbres). Chaque page du DVI est ensuite convertie en image par le bout de Perl qui suit (la variable *size* contient la taille de l'image, calculée par TEX, rangée dans le fichier de trace, et relue par le script).

```
$cmd = $::pstoimg_cmd;
$cmd .= " -type png -tmp tmpdir";
$cmd .= " -discard -interlace -antialias";
$cmd .= " -depth 1 -scale 1.4 $size";
$cmd .= " -margins 62,41 -crop abls -transparent";
$cmd .= " -out math_image_$name.png tmpdir/$im_name";
```

L'ensemble des commandes exécutées est le suivant

```
prepare_for_latex();
extract;
system("latex", "images");
read_log("images.log");
convert_to_png;
```

Finalement, si on demande d'inclure une image dans le RaWeb, avec par exemple une largeur de 15 cm, il faut soit convertir l'image à cette dimension (par exemple si elle est au format PostScript), soit demander l'inclusion avec une largeur de X pixels (par exemple si l'image est au format PNG). Pour ceci on utilise un script Perl, qui lit le fichier XML, et pour chaque objet spécial fait un traitement, puis remplace l'élément par un autre objet. Le traitement a été expliqué plus haut, la substitution est la suivante :

```
s!<(table|preview).*?</\1>!convert_table($1,$&)!egs;
s/<ressource [^>]*/ handle_one_figure($&)/egs;
s/<formula.*?<\/formula\s*>/ handle_one_formula($&);/egs;
```

Le code suivant explique par quoi on remplace une table. S'il n'y a pas de nœuds, ce n'est pas un arbre, et on rend l'objet, sinon, on rend une référence à une image.

```
sub convert_table {
 my $x = $_[0];
 my $y = $_[1];
 if($x eq "preview" || $y =~ /<\/node>/) {
  $::img_ctr ++;
  return "<img src='images/tree_image_$::img_ctr.png'/>";
 } else { return $y; }
}
```

Le cas de formules de maths est plus compliqué, dans la mesure où une formule simple (par exemple `$x+\alpha$`) peut être remplacée par l'équivalent de `{\it x+}` suivi d'une référence à une image précalculée contenant la lettre grecque, et seules les formules plus compliquées nécessitent la création de nouvelles images.

3.6. Conclusion

Un certain nombre de points peuvent être améliorés dans le traitement. La conversion d'images PostScript en PNG est réalisée par `pstoimg`, qui est un script Perl faisant partie de `latex2html`, et qui nécessite l'installation d'outils comme NetPBM. On peut penser à des alternatives comme `dvipng` (dans toutes les bonnes distributions TEX) ou ImageMagick.

On peut également envisager de ne pas insérer d'images dans le HTML, par exemple en convertissant les connecteurs des arbres au

format SVG. En ce qui concerne les formules mathématiques, on peut envisager une formulation duale, où on peut basculer de la version MathML sans images à la version normale avec images comme cela est fait par le CEDRAM.

Bibliographie

1. D. Carlisle, « XMLTEX : A non validating (and not 100% conforming) namespace aware XML parser implemented in TEX », *TUGboat* **21** (2000), nº 3, p. 193-199.

2. D. Carlisle, M. Goossens & S. Rahtz, « De XML à PDF avec `xmltex` et PassiveTEX », in *Cahiers Gutenberg*, nº 35-36, 2000, p. 79-114.

3. J. Grimm, « Outils pour la manipulation du rapport d'activité », Tech. Report RT-0265, Inria, 2002.

4. ———, « Tralics, a LATEX to XML translator, Part I », Rapport Technique 309, Inria, 2006.

5. ———, « Tralics, a LATEX to XML translator, Part II », Rapport Technique 310, Inria, 2006.

6. P. Louarn, « Une expérience d'utilisation de LaTeX : le Rapport d'activité de l'INRIA », *Cahiers Gutenberg* (1988), nº 0, p. 17-24.

7. F. Mittelbach, M. Goossens, J. Braams, D. Carlisle & C. Rowley, *The LATEX companion, second edition*, Addison Wesley, 2004.

8. S. Rahtz, « Passive TEX », http ://www.tei-c.org.uk/Software/passivetex/, 2003.

◭ José Grimm
Inria Sophia Antipolis Méditerranée
2004 route des Lucioles
Sophia Antipolis Cedex (France)
jose.grimm@sophia.inria.fr