CAHIERS GUTenberg

CONVERSION D'ARTICLES EN LaTeX VERS XML AVEC MATHML : UNE ÉTUDE COMPARATIVE

Heinrich Stamerjohanns, Deyan Ginev, Catalin David, Dimitar Misev, Vladimir Zamdzhiev, Michael Kohlhase

Cahiers GUTenberg, nº 51 (2008), p. 7-28.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2008___51_7_0>

CONVERSION D'ARTICLES EN LaTeX VERS XML AVEC MATHML : UNE ÉTUDE COMPARATIVE

Heinrich Stamerjohanns, Deyan Ginev, Catalin David, Dimitar Misev, Vladimir Zamdzhiev, Michael Kohlhase

Résumé. — Depuis une vingtaine d'années, la publication de textes de mathématiques, informatique ou physique théorique s'est faite de façon prédominante à l'aide de systèmes de composition basés sur TeX/LaTeX. Ceci est à la fois un atout — LaTeX est un format à demi sémantique qui capture dans une certaine mesure les intentions de l'auteur — et un problème — TeX est à peu près Turing-complet, ce dont les auteurs abusent en recourant à des centaines d'extensions et à des millions de macros personnelles.

Plusieurs outils ont été développés pour convertir des documents TeX/LaTeX vers des dialectes XML, plus adaptés au web, mais aussi à des traitements automatisés s'appuyant sur la sémantique. Les projets DML utilisent des outils différents, la sélection ne semblant pas reposer sur des critères objectifs. Nous avons passé l'offre actuelle en revue dans le but de fonder ces choix sur des bases plus rationnelles, et d'encourager la compétition ouverte et la convergence vers un socle de fonctionnalités utiles. Dans cet article, nous passons en revue cinq programmes qui transforment des sources TeX/LaTeX en XML en passant les formules en MathML selon trois dimensions : *a*) les paramètres ergonomiques comme la documentation ou la facilité d'installation ; *b*) l'étendue des constructions supportées ; *c*) la qualité du document produit (notamment le code MathML).

Abstract. — Publishing in Mathematics and theoretical areas in Computer Science and Physics has been predominantly using TeX/LaTeX as a formatting language in the last two decades. This large corpus of born-digital material is both a boon—LaTeX is semi-semantic format where the source often contains indications of the author's intentions—and a problem—TeX is Turing-complete and

Cahiers GUTenberg nº 51 — Octobre 2008, p. 7-28.

authors use this freedom to use thousands of styles and millions of user macros.

Several tools have been developed to convert TEX/LATEX documents to XML-based. Different DML projects use different tools, and the selection seems largely accidental. To put the choice of converters for DML projects onto a more solid footing and to encourage competition and feature convergence we survey the market. In this paper we investigate and compare five LATEX-to-XML transformers in three dimensions: *a*) ergonomic factors like documentation, ease of installation, *b*) coverage, and *c*) quality of the resulting documents (in particular the MathML parts).

NOTE. — Ce texte est une version légèrement remaniée d'un article paru sous le titre « MathML-aware article conversion from LATEX — A comparison study » dans *Towards a Digital Mathematics Library*, actes de la conférence DML 2009 (Grand Bend, Ontario, Canada, 8-9 juillet 2009) édités par Petr Sojka et publiés par l'université de Masaryk (Brno, République tchèque). Il est reproduit ici avec l'autorisation de Petr Sojka. La DML dont il est ici implicitement question est la bibliothèque numérique de mathématiques (*Digital mathematics library*), projet en chantier dont le but ultime serait l'archivage raisonné de toutes les mathématiques publiées en un format suffisamment ouvert et bien structuré pour pouvoir envisager un « web sémantique des textes et théorèmes mathématiques ». Traduction : Thierry Bouche.

1. INTRODUCTION

La publication en mathématiques et dans les domaines théoriques de l'informatique ou de la physique s'est faite de façon prédominante à l'aide de systèmes de composition basés sur TEX/LATEX depuis une vingtaine d'années. L'existence de ce corpus nativement numérique substantiel est à la fois un atout — LATEX est un format à demi sémantique qui capture dans une certaine mesure les intentions de l'auteur — et un problème — TEX est quasiment Turing-complet, ce dont les auteurs abusent en accumulant les styles et les macros personnelles. Par ailleurs se développe un mouvement pour diffuser sur internet les publications mathématiques dans des formats mieux adaptés au web que PostScript ou PDF (qui peuvent être produits directement à partir de TEX/LATEX de façon standard). Quoiqu'il existe des alternatives crédibles, MathML [1] semble

être le format de choix pour publier un contenu mathématique non trivial sur le web, car il est exploité par des services de haut niveau comme la lecture à voix haute ou la recherche de formules [10, 12].

De nombreux outils ont été développés pour convertir des documents TEX/LATEX vers du XML dans lequel MathML est le format utilisé pour représenter les expressions mathématiques. Certains de ces outils utilisent le moteur TEX lui-même pour analyser le source LATEX original, d'autres s'efforcent de réimplémenter un parseur TEX complet pour donner un contrôle total sur la conversion. Des projets similaires, notamment dans le contexte de la bibliothèque numérique mathématique, utilisent des outils différents, mais le choix semble être assez largement arbitraire, ou provoqué par des affinités personnelles. Pour en citer deux : le projet arXMLiv [16, 4] de l'université Jacobs (Bremen) utilise LaTeXML [11] pour convertir l'archive d'« ePrints » arXiv [5], désormais à la bibliothèque de l'université de Cornell, tandis que les projets NUMDAM et CEDRAM de la cellule MathDoc à Grenoble [6] s'en remettent à Tralics [18].

Dans cet article, nous tentons de fonder de façon plus rationnelle les choix de convertisseurs LATEX vers XML+MathML, tout en encourageant la compétition et la convergence des fonctionnalités en présentant un panorama des solutions existantes, et une comparaison de cinq systèmes disponibles. Nous commençons par présenter brièvement les cinq systèmes évalués dans la section 2, où ils sont comparés sur des critères ergonomiques comme la documentation ou la facilité d'installation. Dans la section 3 nous étudions la couverture des constructions mathématiques traitées, et la qualité des documents produits (en particulier en ce qui concerne le MathML) en utilisant un échantillon d'articles scientifiques provenant du serveur de prépublications arXiv [5].

Cet article est une version abrégée d'un rapport plus détaillé [15], qui sera maintenu à jour pour tenir compte de l'évolution des logiciels concernés.

Pour conduire nos tests, nous utilisons l'expérience, les ressources, et certaines parties du système de production du projet arXMLiv [16], mais nous avons fait notre possible pour présenter nos résultats de façon neutre. Merci de faire part aux auteurs de tout désaccord sur la description d'un système.

2. Les systèmes

Dans cet article, nous étudions et comparons cinq convertisseurs de LaTeX vers XML qui produisent du MathML : Hermes, Tralics, LaTeXML, TeX4HT et TtM.

Outre ces cinq systèmes nous avons rencontré d'autres traducteurs de TeX vers XML que nous n'avons pas pu inclure dans notre comparatif pour diverses raisons (cf. [22] pour une liste relativement complète et la référence des pages web des différents systèmes).

1. Les outils BlaTeX, itex2mml, RiTeX [1], MathMLStudio Lite convertissent seulement un sous-ensemble des formules TeX/LaTeX en MathML mais ne semblent pas pouvoir fonctionner au niveau du document : ils sont conçus pour des auteurs de documents mathématiques pour le web plutôt que pour convertir des textes complets existant déjà sous un format nativement numérique. C'est pourquoi nous les avons exclus de cette étude, mais notre méthode comparative devrait pouvoir s'appliquer aussi dans leur cas : nous pourrions les inclure dans un travail ultérieur.

2. Les outils HeVeA et LaTeX2HTML transforment des documents LaTeX en HTML, mais semblent produire des images pour les formules plutôt que du MathML.

3. L'université du Western Ontario offre un service en ligne pour traduire des documents en MathML [21]. Le logiciel est écrit en Java. Nous avons demandé le code mais n'avons pas obtenu de réponse.

4. L'outil LXir [14] a été développé par EDP Sciences sous licence GPL. Il prétend transformer LaTeX en XML. Les auteurs n'ont pas réussi à compiler ce logiciel, et ont renoncé à persévérer du fait que les instructions détaillées ne sont disponibles qu'en français.

5. Le développement d'Omega [13] a été interrompu. Certaines de ses fonctionnalités devraient être reprises par LuaTeX qui est une version de pdfTeX embarquant le langage Lua. Le projet a débuté récemment ; il pourrait exploiter les fontes mathématiques OpenType de Microsoft.

Dans ce qui suit, nous allons passer rapidement en revue les systèmes, leur degré de développement, leur stratégie pour réaliser la conversion, leur mise en œuvre. Nous aborderons également leur installation et

1. Pas encore référencé dans [22], voir `http://ritex.rubyforge.org/`.

leur facilité d'utilisation, ce qui joue un rôle important lorsque l'on doit choisir un tel outil pour un projet d'une certaine envergure.

Hermes utilise une grammaire pour traduire un document (AMS)LaTeX vers XML+MathML, recodé en Unicode (utf-8) et enrichi de métadonnées, mais la traduction de documents (AMS)TeX n'est pas possible. MathML est le seul dialecte XML que l'on peut obtenir en sortie actuellement.

Le système est téléchargeable sur son site officiel [3]. Il est diffusé sous licence GNU GPL et s'installe sans difficulté à partir des sources. On peut utiliser Hermes sous Linux, Windows et OS X. Notons cependant que la dernière version du traducteur date de plus de trois ans (v. 0.9.12 publiée le 28 novembre 2006), et que le développement est arrêté. La documentation disponible est minimale : elle ne comporte qu'une courte description de l'installation, du fonctionnement, du format de sortie et de l'utilisation du logiciel.

En théorie, Hermes surcharge sémantiquement une copie du source TeX à traiter, puis lance `latex` et analyse le fichier DVI résultant pour produire la sortie MathML, qui est modelée sur le source TeX. En pratique, cela est obtenu en lançant l'exécutable `hermes` sur le fichier binaire produit par la compilation `latex` du source sémantiquement enrichi, puis en recourant à une transformation XSLT. Une feuille de style XSLT est fournie, mais on peut aussi bien en utiliser une autre. L'interface par ligne de commande est assez austère. Il n'existe pas d'option permettant de paramétrer la conversion (il n'y a même pas d'option d'aide). Hermes ne génère aucune information pendant la conversion à l'exception du log de `latex`, qui n'indique pas si la conversion a été bien menée jusqu'à son terme car elle peut échouer à un stade ultérieur. Traiter un nombre important de fichiers est pratiquement impossible car on ne dispose pas de traces permettant d'évaluer la conversion.

La conversion est lente, principalement du fait de l'utilisation de `latex`, l'exécutable Hermes lui-même est très efficace dans la plupart des cas.

Tralics est conçu pour traduire des sources LaTeX dans un format XML dédié, avec des provisions pour la conversion ultérieure vers PDF ou HTML. Le logiciel est sous licence CeCILL [7], qui est proche de GPL, mais mieux adaptée au droit français. L'objectif initial de Tralics

est la conversion du rapport d'activité annuel de l'INRIA [9], mais de nombreuses adaptations sont possibles à l'aide de fichiers de configuration.

Le système est disponible en ligne [18] et peut être installé directement à partir des sources, ou des binaires compilés pour Linux, Mac OS ou Windows. Une archive de base (« main ») fournit le convertisseur lui-même tandis qu'une archive optionnelle (« extra ») contient les ajouts pour finaliser la conversion vers HTML ou PDF. Une documentation conséquente sur l'utilisation et le paramétrage est disponible en ligne. Ces informations sont plutôt conçues pour des développeurs intéressés par l'adaptation du système à une chaîne de traitement spécifique. Un manuel utilisateur pourrait être fourni, donnant un peu plus de précisions que la description actuelle des options de la ligne de commande. Les feuilles de style XSLT auxiliaires sont expliquées du point de vue de leurs mécanismes de bas niveau, sans exemples clairs d'utilisation basique, ce qui ne facilite pas leur mise en œuvre par un béotien.

Tralics est écrit en C++ et inclut un préprocesseur de macros TEX. Il est par conséquent très rapide. Après avoir interprété le fichier source, l'arbre XML du document est construit, et les mathématiques sont traduites en MathML. Un éventuel fichier BibTEX sera intégré au document produit.

LaTeXML a été développé pour permettre la création de la « bibliothèque numérique des fonctions mathématiques » (Digital Library of Mathematical Functions) [8]. Il est écrit en Perl et tente d'émuler TEX. Un post-processeur prend en charge le format de sortie XML spécifique pour écrire le résultat en HTML ou XHTML+MathML. LaTeXML est téléchargeable librement sur son site [11] et peut être installé à partir des sources sous Linux ou Mac OS. Comme il est écrit en Perl, ça devrait aussi pouvoir se faire sous Windows mais personne ne semble avoir tenté sa chance pour l'instant.

Un manuel très détaillé est disponible en ligne ou sous la forme d'un document PDF de 130 pages. Il fournit des indications précises sur l'architecture du système, mais aussi sur la paramétrisation, l'utilisation des commandes et les post-traitements. Le système LaTeXML comporte un émulateur TEX, un générateur de XML et un post-processeur. Pour lui permettre de traiter des documents LATEX du monde réel, il faut

fournir des définitions LaTeXML au générateur de XML pour les macros sémantiques des *packages* LaTeX utilisés.

L'outil `latexml` peut traiter des documents TeX ou LaTeX. Il charge les définitions LaTeXML correspondant aux *packages* appelés par le source traité et écrit un document LTXML temporaire, qui est une traduction fidèle de la structure du source LaTeX. Le format LTXML permet une représentation XML de tous les concepts centraux de TeX/LaTeX, et fournit la superstructure au sein de laquelle les définitions LTXML vont enrichir le XML produit.

Par la suite, le programme `latexmlpost` permet de transformer ce quasi-LaTeX formulé en XML au format cible. Cela est obtenu par une série de filtres intelligents. Il existe des filtres pour écrire des tables HTML, gérer l'inclusion de graphiques, ou passer les formules en MathML de présentation. D'autres filtres sont en développement, par exemple pour la production de formules en OpenMath ou en MathML de contenu.

La traduction est assez lente, du fait notamment que le processus recharge tous les modules Perl et les définitions pour chaque document.

TeX4HT est l'un des traducteurs basés sur TeX qui est avant tout dédié à l'hypertexte, bien qu'il ne se limite pas à cela. L'une des caractéristiques majeures de TeX4HT est la multiplicité des formats de sortie que le système permet (citons HTML, OpenDocument, OpenOffice, Microsoft Word…).

Le système dispose d'un site officiel où l'on peut le télécharger [17]. Il est en fait inclus dans de nombreuses installations TeX.

La transformation complète d'un fichier source débute par la compilation par `latex` d'un source légèrement modifié, et continue par le lancement d'une série d'utilitaires qui finalisent un fichier .xml au format XHTML+MathML. De ce fait, le système traite toutes les constructions LaTeX valides sur la machine sur laquelle il tourne, et ne nécessite pas de définitions spécifiques. La documentation disponible sur la page web est simple et bien organisée. Comme elle n'est ni simpliste ni hypertechnique, son accès est facile pour tous les utilisateurs.

L'inconvénient le plus fâcheux de TeX4HT est l'absence d'outils pour repérer et corriger les erreurs. Tout d'abord, le recours à `latex` rend le système assez fragile puisqu'il suffit d'une erreur à ce stade pour que tout le processus de conversion soit stoppé. Ceci rend les traitements par

lots assez hasardeux. Même une fois le source LaTeX corrigé, le fichier de sortie a toujours de grandes chances d'être fautif (le cas le plus fréquent étant celui de XML invalide, du fait que TeX4HT ne peut se garantir contre cela). Les fichiers de log n'apportent que très peu de précisions sur les différentes étapes de la conversion à part celui créé par LaTeX.

La triple compilation initiale par `latex` a un impact négatif sur la vitesse de conversion. Les programmes additionnels utilisés par TeX4HT sont écrits en C, ils sont donc rapides.

TtM est un traducteur de TeX vers MathML dérivé de TtH, qui produit du HTML. Il traite les complexités de TeX à l'exception des catcodes et quelques idiosyncrasies qui ne trouvent pas de pendant évident en HTML. Dans la plupart des cas, des documents TeX ou LaTeX sans malice seront traduits instantanément.

TtM existe pour Windows et Linux (cette dernière version étant gratuite), que l'on peut se procurer sur le site officiel [19].

TtM ne repose pas du tout sur le moteur TeX (il tente d'imiter son fonctionnement sur des fichiers utilisateurs standards), et il ne dépend en fait de l'installation d'aucun autre programme sur la machine où il tourne. Il est donc a priori universellement portable, et son installation est aussi simple que l'extraction d'une archive. TtM est écrit en Flex, à partir de quoi un exécutable C est compilé, ce qui explique son extrême rapidité.

La conversion de fichiers TeX, même gros, est une question de (peu de) secondes. Il est donc bien adapté pour produire à la volée du HTML à partir de sources TeX. Une documentation très bien structurée accompagne le logiciel. Le manuel est plutôt simple et court, mais il parvient à couvrir tous les aspects de TtM de façon très abordable. Quasiment toutes les mathématiques connues de TeX sont traitées à l'exception de quelques symboles obscurs qui ne figurent pas dans les fontes habituellement exploitées par les navigateurs web. Le support pour LaTeX couvre à peu près l'ensemble des mathématiques, ainsi que les structures inévitables.

Bien que les macros soient interprétées parfaitement, TtM ignore les catcodes, et ne pourra donc pas se dépêtrer de la grande majorité du code de bas niveau existant. De façon générale, le résultat sera très

satisfaisant pour des fichiers basiques, mais inutilisable en présence de *packages* ou macros personnelles un peu inhabituels.

3. Couverture et rapidité

Dans cette section nous allons évaluer la couverture des convertisseurs. Pour cela nous avons choisi d'utiliser une partie du corpus constitué par les textes déposés sur arXiv car c'est probablement la source la plus complète de documents hétérogènes écrits en TEX/LATEX. On y trouve plus d'un demi-million d'articles scientifiques couvrant des domaines comme la physique, les mathématiques, la biologie et l'informatique, depuis deux décennies. Selon notre expérience, ce corpus fournit une bonne image de la diversité des sources TEX/LATEX d'articles scientifiques en général. Nous ne pouvons évidemment pas lancer chaque convertisseur sur la totalité du corpus (un traitement complet par LaTeXML occupe un processeur pendant un an environ), c'est pourquoi nous nous sommes restreints à un échantillon aléatoire de 1000 documents.

Traiter de la couverture du « format » LATEX par les convertisseurs n'est pas évident car il y a différents degrés d'imperfection, et différentes raisons à cela. En nous appuyant sur l'analyse développée pour le projet arXMLiv [4], nous nous concentrerons sur trois classes d'erreurs qui ont un sens pour chacun des systèmes.

Incomplet. — Le convertisseur n'est pas parvenu au bout de la conversion. Il a planté ou signalé une erreur fatale. Pour quelques systèmes on peut préciser des sous-classes d'erreurs comme « délai d'attente dépassé » ou « erreur fatale ».

Erreurs. — Le convertisseur est parvenu au bout de la conversion mais a émis des messages d'erreurs. Quelques systèmes fournissent des indications sur la nature des erreurs rencontrées, un exemple assez fréquent étant « macro non définie » généralement causée par l'appel d'un fichier de style non supporté.

Succès. — Le convertisseur est parvenu au bout de la conversion sans aucune erreur, mais d'éventuels « avertissements » sont acceptés (ce sont les problèmes que le convertisseur considère comme mineurs).

On notera que l'évaluation de cette section repose uniquement sur les traces produites par les systèmes eux-mêmes — à l'exception de la

rapidité de traitement et des plantages qui sont mesurables de façon objective. Il est évident qu'on ne peut pas faire entièrement confiance à un tel diagnostic : un système peut par exemple considérer qu'il a achevé une conversion avec succès tout en produisant un fichier vide. Mais pour produire les statistiques qui suivent, le nombre de fichiers à traiter était trop grand pour contrôler les conclusions à la main. Nous avons préféré une approche manuelle pour évaluer la qualité du MathML produit sur un échantillon, ce qui est l'objet de la section suivante.

Hermes		
Résultats	**Nombre**	**%**
Incomplet	653	65,3
Erreurs	0	0
Succès	347	34,7

Ces statistiques ont été obtenues en écrivant un script qui lance Hermes sur tous les fichiers. Le fichier converti est vide dans 65,3 % des cas, tous les autres étant considérés comme des succès. Comme Hermes donne très peu d'informations de débogage, les statistiques ne sont pas basées sur ses rapports, mais en faisant passer Validator [20] sur les fichiers produits. Selon cet outil, tous les fichiers non vides écrits par Hermes sont bien formés. Dans la plupart des cas, on rencontre de nombreux avertissements et quelques erreurs quand on lance `latex` sur le source TEX enrichi ; on peut passer outre en forçant le compilateur en *batchmode*, mais 362 des traitements infructueux ne passent pas cette étape. Le traitement par Hermes de tous les fichiers a pris environ 20 minutes.

Tralics		
Résultats	**Nombre**	**%**
Incomplet	0	0
Erreurs	984	98.4
Succès	16	1.6

Du fait de sa couverture initialement réduite, Tralics a rencontré un nombre significatif de macros inconnues, d'où un taux de succès inférieur à 2 %. Du fait que la conversion repose sur des fichiers de configuration qui définissent la traduction XML des commandes LATEX, on ne peut espérer un succès immédiat sur une sélection arbitraire d'articles scientifiques.

De nombreux documents du corpus testé utilisent des *packages* et des classes spéciales, imposées par les standards de la communauté scientifique concernée, ou au gré de chaque auteur. Tralics — une fois installé correctement ainsi que les fichiers de configuration distribués — a rencontré en moyenne 50 commandes inconnues par document. La

conversion n'a pas pris plus de 3 minutes et a produit un fichier XML pour chaque article traité dans tous les cas. Quelque 93 % de ces fichiers étant bien formés, on peut penser que nombre de ces fichiers XML sont utilisables bien que la conversion ait rencontré des erreurs. Lors de tels traitements par lots, on peut opter pour un mode silencieux ou bavard, conserver une trace détaillée de l'exécution dans un fichier log, ou encore choisir ou non d'utiliser MathML pour les formules « triviales », parmi quelques unes des options utiles.

LaTeXML		
Résultats	**Nombre**	**%**
Incomplet	66	6.6
Erreurs	285	28.5
Succès	649	64.9

Il doit être mentionné ici que LaTeXML a bénéficié d'un avantage non négligeable par rapport aux autres systèmes lors de nos tests : de nombreuses définitions permettant de traiter les articles scientifiques du corpus arXiv ont été écrites comme conséquence du projet en cours arXMLiv. De ce fait, LaTeXML réussit à créer un fichier XHTML dans 89 % des cas. Cependant, LaTeXML ne peut pas garantir que le XHTML produit sera affiché correctement dans 35,7 % des cas. La durée du traitement des 1000 documents est assez longue : il faut de l'ordre d'une heure et demi pour en venir à bout.

TeX4HT		
Résultats	**Nombre**	**%**
Incomplet	414	34.3
Erreurs	332	27.5
Succès	450	38.1

Comme cela a été évoqué précédemment, le système TeX4HT devrait être polyvalent car il s'en remet à une passe `latex` sur les fichiers sources, et réalise ensuite la transformation à partir du DVI et d'informations de log. Le programme `latex` est lancé trois fois consécutives pour résoudre à coup sûr les références croisées : cela prend beaucoup de temps sur un nombre de fichiers important et, à cause des erreurs, le processus de conversion est interrompu assez souvent à ce stade. Pour surmonter cette difficulté, nous nous sommes servi d'un script Bash qui assigne un temps maximum pour la conversion de chaque fichier. Une remarque à propos du système TeX4HT est que dans le cas de fichiers importants, il découpe le fichier de sortie en plusieurs morceaux, ce qui produit donc plus de fichiers qu'avec les autres systèmes testés.

Le traitement de tous les documents, avec un temps maximum imparti de 60 s pour la transformation de chaque fichier, a pris en moyenne 90 minutes (ce qui est largement dû aux compilations `latex`).

TtM		
Résultats	**Nombre**	**%**
Incomplet	270	27.0
Erreurs	650	65.0
Succès	80	8.0

En 40 secondes, TtM a réussi à convertir seulement 8 % des 1000 fichiers. Ce taux de succès assez bas est dû pour l'essentiel au fait que TtM n'interprète ni les *catcodes*, ni le recours à des *packages* externes, ce qui le rend inapte à convertir des articles scientifiques arbitraires.

TtM est un outil extrêmement rapide qui convertit un fichier TEX vers XHTML+MathML en moins de 0,04 s en moyenne. Il y a eu 100 avertissements par fichiers, en général dus à des commandes ou environnements inconnus, ou encore à des problèmes de références bibliographiques. Parmi les 651 erreurs signalées, 30 étaient fatales (ce qui signifie que le programme interrompt immédiatement son traitement en laissant un fichier XHTML incomplet).

4. Évaluation de la qualité du MathML

Cette section est consacrée à l'évaluation de la qualité du code MathML produit par les cinq convertisseurs analysés. La méthodologie repose sur un corpus de référence constitué d'un petit nombre de formules non triviales extraites au hasard parmi les articles arXMLiv.

L'étendue de cet article ne permet pas de rentrer dans les détails de l'évaluation de plusieurs formules. Les résultats complets se trouvent dans les annexes A à E du rapport [15]. De façon générale, l'évaluation est faite dans l'optique de la qualité du XHTML+MathML produit (utilisation de CSS, MathML sémantique ou de présentation, structure des formules), notamment en cherchant des réponses à des questions comme : est-ce que le XML produit est valide ? Est-ce que l'on peut régénérer du LATEX ? Est-ce que la sémantique de formules comme $x+y^2$ est explicitée ?

4.1. L'environnement eqnarray

Nous examinerons ici quelques aspects de la conversion de LATEX vers MathML sur l'exemple de la figure 4.1, choisis soit parce qu'ils donnent

$$
\begin{aligned}
4r^2\int_0^{\pi/2}\cos^2\theta\,d\theta &= 4r^2\int_0^{\pi/2}\frac{1}{2}(1+\cos 2\theta)\,d\theta \\
&= 2r^2\theta\Big|_0^{\pi/2} + 2r^2\int_0^{\pi/2}\cos 2\theta\,d\theta \\
&= \pi r^2 + 2r^2(\sin 2\theta)\Big|_0^{\pi/2} \\
&= \pi r^2
\end{aligned}
$$

Figure 1: Une formule destinée à vérifier la qualité des convertisseurs.

$$
\begin{aligned}
4r^2\textstyle\int_0^{\pi/2}\cos^2\theta d\theta &= 4r^2\int_0^{\pi/2}\frac{1}{2}(1+\cos 2\theta)d\theta \\
&= 2r^2\theta|_0^{\pi/2} + 2r^2\int_0^{\pi/2}\cos 2\theta d\theta \\
&= \pi r^2 + 2r^2(\sin 2\theta)|_0^{\pi/2} \\
&= \pi r^2
\end{aligned}
$$

(a) Hermes

$$
\begin{aligned}
4r^2\textstyle\int_0^{\pi/2}\cos^2\theta\,d\theta &= 4r^2\Big[\int_0^{\pi/2}\frac{1}{2}(1+\cos 2\theta)\,d\theta \\
&= 2r^2\theta|_0^{\pi/2} + 2r^2\int_0^{\pi/2}\cos 2\theta\,d\theta \\
&= \pi r^2 + 2r^2(\sin 2\theta)|_0^{\pi/2} \\
&= \pi r^2
\end{aligned}
$$

(b) Tralics

$$
\begin{aligned}
4r^2\textstyle\int_0^{\pi/2}\cos^2\theta\,d\theta &= 4r^2\int_0^{\pi/2}\frac{1}{2}(1+\cos 2\theta)\,d\theta \\
&= 2r^2\theta\Big|_0^{\pi/2} + 2r^2\int_0^{\pi/2}\cos 2\theta\,d\theta \\
&= \pi r^2 + 2r^2(\sin 2\theta)\Big|_0^{\pi/2} \\
&= \pi r^2
\end{aligned}
$$

(c) TeX4HT

$$
\begin{aligned}
4r^2\textstyle\int_0^{\pi/2}\cos^2\theta d\theta &= 4r^2\int_0^{\pi/2}\frac{1}{2}(1+\cos 2\theta)d\theta \\
&= 2r^2\theta|_0^{\pi/2} + 2r^2\int_0^{\pi/2}\cos 2\theta d\theta \\
&= \pi r^2 + 2r^2(\sin 2\theta)|_0^{\pi/2} \\
&= \pi r^2
\end{aligned}
$$

(d) LaTeXML

$$
\begin{aligned}
4r^2\textstyle\int_0^{\pi/2}\cos^2\theta\;d\theta &= 4r^2\int_0^{\pi/2}\frac{1}{2}(1+\cos 2\theta)\;d\theta \\
&= 2r^2\theta|_0^{\pi/2} + 2r^2\int_0^{\pi/2}\cos 2\theta\;d\theta \\
&= \pi r^2 + 2r^2(\sin 2\theta)|_0^{\pi/2} \\
&= \pi r^2
\end{aligned}
$$

(e) TtM

Figure 2: Résultat visuel des différents systèmes avec Firefox.

un rendu différent, soit parce qu'ils révèlent des choix intéressants dans la stratégie de conversion.

Traduction d'eqnarray. — En tant qu'environnement conçu pour représenter une équation sur plusieurs lignes, `eqnarray` nécessite l'utilisation d'une structure tabulaire en XHTML. Les options des traducteurs vont de l'utilisation de l'élément MathML ⟨`mtable`⟩ (Tralics, TeX4HT, TtM) à un structure HTML ⟨`table`⟩ dans laquelle chaque cellule contient une expression mathématique indépendante (LaTeXML), en passant par une combinaison des deux (Hermes).

Opérateurs et symboles. — Pour représenter les opérateurs mathématiques, les traducteurs utilisent basiquement l'élément ⟨`mo`⟩ (Hermes) et peaufinent dans la mesure du possible le résultat à l'aide d'attributs. TeX4HT utilise l'attribut `class` pour améliorer le rendu des symboles, tandis que Tralics utilise l'attribut `form` pour améliorer leur positionnement. LaTeXML utilise en outre l'attribut `movablelimits` pour déterminer le traitement des indices. Il est intéressant de constater que chaque convertisseur apporte des améliorations qui lui sont propres, ce qui donne une idée de ce que l'on pourrait obtenir en les combinant astucieusement.

Espacement. — C'est un sujet difficile pour tous les convertisseurs. Hermes et LaTeXML oublient toute information de ce type lors de la conversion en XHTML, ce dont la lisibilité des formules complexes pâtit. Cependant, LaTeXML tente d'améliorer cette situation en surchargeant la sémantique des formules à l'aide de ses propres heuristiques, par exemple en insérant l'opérateur `&ApplyFunction;` là où ça lui semble approprié. Les résultats sont satisfaisants lorsque cela est fait à bon escient, mais l'aspect paraît souvent incohérent lorsque ces améliorations ne sont pas appliquées partout où elles le devraient. À l'inverse, Tralics et TeX4HT préservent soigneusement les instructions d'espacement placées dans le ficher source, à l'aide d'un élément ⟨`mspace`⟩. TtM a une approche du problème plus générique : il n'utilise que la forme ⟨`mi`⟩` `⟨`/mi`⟩.

Le symbole | et son traitement. — La contrepartie MathML du symbole |, noté `$\mid$` en LaTeX, semble être un mystère pour les convertisseurs. La solution simpliste est d'utiliser l'élément ⟨`mo`⟩ suivi d'un ⟨`msubsup`⟩, ce qui est un concept entièrement différent. TeX4HT est le seul à proposer une alternative : `$\mid$` donne un élément ⟨`mfenced`⟩

avec attribut `open`. De plus, cet élément est inclus dans un ⟨`mstyle`⟩, qui contrôle la taille du trait à l'aide de l'attribut `mathsize`. Le résultat visuel est ainsi plus proche de celui de LaTeX, mais ça n'est pas encore une solution idoine.

Intégrales. — Comme ce sont des constructions mathématiques standard, les intégrales sont traitées de façons très similaires par les convertisseurs. Le symbole lui-même est toujours représenté comme ⟨`mi`⟩`&int;`⟨`/mi`⟩, suivi par un élément ⟨`msubsup`⟩ pour les bornes. LaTeXML est le seul à recourir à ⟨`munderover`⟩ pour les bornes, mais induit un rendu différent de celui de LaTeX, et incohérent d'une intégrale à l'autre.

Bruit. — Un problème qui concerne tous les convertisseurs est la création d'éléments vides comme ⟨`mrow/`⟩, ⟨`mo/`⟩, ⟨`mo`⟩␣⟨`/mo`⟩, etc. On trouve souvent plus d'une dizaine de ces scories par fichier converti, que ce soit dû à des macros inconnues ou à d'autres erreurs. Ces éléments vides sont une nuisance pour les traitements exploitant la sémantique des fichiers XML : il serait souhaitable que les convertisseurs fassent un peu mieux le ménage avant de quitter les lieux.

4.2. Remarques générales

Après cette analyse reposant spécifiquement sur notre exemple de la figure 4.1, nous mettons en exergue quelques différences dans l'utilisation d'éléments MathML apparues au cours de nos travaux.

Groupements avec ⟨`mrow`⟩

L'élément ⟨`mrow`⟩ est conçu pour regrouper en unités logiques des fragments de MathML. Ceci est exploité pour l'analyse de l'arbre XML d'une formule et a par exemple des conséquences sur sa lecture à voix haute aussi bien que l'affichage graphique.

TeX4HT a des difficultés pour identifier la portée des arguments, ce qui peut conduire à l'absence des certains éléments ⟨`mrow`⟩ (comme dans $A = \pi r^2$).

TeX4HT et TtM se servent de ⟨`mrow`⟩ pour encadrer les arguments de la plupart (peut-être de tous ?) les éléments structurels de MathML. Quelques exemples de parents qui traitent leurs enfants de la sorte sont ⟨`msup`⟩, ⟨`msub`⟩, ⟨`msubsup`⟩, ⟨`mfrac`⟩, ⟨`msqrt`⟩, …

Hermes utilise aussi ⟨mrow⟩ pour encadrer tout argument d'un ensemble assez réduit d'éléments : ⟨msup⟩, ⟨msub⟩, ⟨msubsup⟩ et ⟨mfrac⟩. Mais ces éléments sont utilisés systématiquement, quelle que soit la complexité de ce qu'ils encadrent. C'est le seul emploi de ⟨mrow⟩ par ce convertisseur.

TtM inclut dans un ⟨mrow⟩ la totalité des expressions MathML qu'il génère, ce qui est une erreur.

LaTeXML et Tralics utilisent l'élément ⟨mrow⟩ chaque fois qu'il y a plus de trois arguments de même niveau. Par exemple, « $x + y$ » serait encadré, contrairement à « πx ». Même pour des éléments comme ⟨msup⟩ et ⟨msub⟩, ils n'ont recours à ⟨mrow⟩ que lorsque les indices ou exposants comportent plus d'un atome, contrairement à ce que font tous les autres convertisseurs,

Les parenthèses et leur portée

Un autre aspect qui est lié aux groupements est le traitement des parenthèses. MathML propose deux mécanismes pour cela, sans suggérer clairement la méthode préférée : on peut baliser les signes appariés

— soit comme des opérateurs mathématiques avec l'élément ⟨mo⟩ et l'attribut `fence="true"`. Les parenthèses peuvent être rendues étirables en précisant `stretchy="true"` ;

— soit avec l'élément ⟨mfenced⟩. Dans ce cas les parenthèses sont définies comme valeurs des attributs `open` et `close`.

Dans les situations où les parenthèses ne sont pas appariées (comme pour les opérateurs « bra » et « ket » de la physique théorique), elles sont considérées comme des opérateurs mathématiques ordinaires et balisées avec l'élément ⟨mo⟩. Les systèmes diffèrent à nouveau dans leur façon de traiter les parenthèses.

LaTeXML, Tralics et TeX4HT utilisent les deux possibilités en fonction du succès de la détection des appariements. Quand ça marche (comme pour `\left` et `\right` en LaTeX), alors un élément ⟨mfenced⟩ est créé. Sinon, des éléments ⟨mo⟩ ordinaires sont produits. Tralics et LaTeXML ne fournissent pas d'attributs (même dans les cas où `fence="true"` serait de rigueur).

TeX4HT utilise des éléments ⟨mo⟩ dotés d'attributs `class` avec les valeurs `"MathClass-open"` et `"MathClass-close"`. C'est efficace pour la mise en forme par CSS, mais n'est pas très standard (la précision

`fence="true"` qui serait préférable est parfois effectivement ajoutée).

LaTeXML lie la fonction et son argument dans certains cas (dont les fonctions trigonométriques) avec ⟨mo⟩⁡⟨/mo⟩. Si cette structure n'est pas reconnue, c'est ⟨mfenced⟩ qui est utilisé. Mais quand la portée du parenthésage n'a pas pu être déterminée, c'est un ⟨mo⟩ contenant chaque parenthèse qui est écrit.

Hermes et TtM ne donnent pas l'impression de chercher à déterminer ces groupements.

TtM semble ajouter `stretchy="false"` à tous les opérateurs mathématiques sauf s'ils sont précédés de `\left` ou `\right` : ceux-ci ne reçoivent aucun attribut.

Une remarque

Pour clore cette section, nous remarquons qu'Hermes a un défaut surprenant : il ne produit pas de MathML dans les titres, même lorsque le code LaTeX implique le recours à des symboles de taille variable.

5. Conclusions

Ce papier a été écrit pour aider la communauté DML à diffuser les trésors codés en TeX/LaTeX en mettant un peu d'ordre dans le zoo des traducteurs de LaTeX vers XML.

5.1. Synthèse…

Concrètement, nous avons étudié cinq programmes qui transforment des sources TeX/LaTeX en XML en passant les formules en MathML. Nous avons comparé ces systèmes selon trois dimensions : *a*) les paramètres ergonomiques comme la documentation ou la facilité d'installation ; *b*) l'étendue des constructions supportées ; *c*) un test de tous les systèmes sur un échantillon aléatoire de 1000 articles parmi le corpus disponible sur le serveur arXiv. Les résultats sont résumés dans la figure 3.

Notez qu'en ce qui concerne les chiffres de couverture au centre de la figure 3 les « succès » sont ceux dont le convertisseur était satisfait : nous n'avons pas vérifié les résultats. Il y a dans cette classe des avertissements (comme ceux de LaTeXML à propos de parenthèses non appariées). La classe « incomplet » concerne toutes les exécutions qui n'ont pas abouti à une sortie utilisable. C'est probablement le cas le plus général de transformations lancées sur un ensemble de fichiers pouvant utiliser des

Système	Documentation	Installation	**Couverture**	% incomplet	% erreurs	% succès	Qualité	Rapidité	Facilité
Hermes	–	++	**–**	65	0	35	–	o	–
Tralics	+	++	**–**	0	98	2	o	+	– –
LaTeXML	++	+	**+**	7	28	65	+	–	+
TeX4HT	++	++	**o**	34	28	38	+	–	++
TtM	++	++	**–**	27	65	8	o	++	+

Figure 3: Table de comparaison des systèmes

classes ou des extensions LaTeX quelconques. La classe des « erreurs » rassemble tout le reste : les cas où un fichier de sortie a été produit, mais le système n'a pas pu surmonter les problèmes posés par certaines macros.

Nous espérons que notre étude encouragera la compétition, la collaboration, et une certaine convergence en termes de fonctionnalité — après tout, quel projet voudrait obtenir un score « – – » dans la table 3 ?

5.2. ... et recommandations

Une autre motivation pour ce travail était d'aider les projets émergeants à s'orienter dans le choix de leurs outils sur des bases un peu plus solides. Nous allons donc conclure par quelques recommandations sur le choix du convertisseur pour passer une collection de fichiers LaTeX en XHTML+MathML. Nous pensons que chaque système a ses avantages et ses inconvénients et nous allons donc essayer de les résumer ici. Nous commençons par les deux convertisseurs qui utilisent le moteur TeX.

L'avenir de TeX4HT est incertain aujourd'hui car son créateur, Eitan Gurari, s'est éteint récemment. Le système a une couverture assez large car il utilise `latex` lui-même pour interpréter les fichiers et dispose comme options sur la ligne de commande d'une boite à outils très diversifiée. Ceci explique probablement son choix fréquent comme convertisseur approprié pour des conversions de fichiers LaTeX ordinaires, qui n'ont pas été préparés spécifiquement pour être transformés

en XML. C'est aussi notre recommandation. Il n'existe cependant pas de mécanisme pour introduire des changements de bas niveau dans la conversion, donc TeX4HT se cantonne à des conversions standard vers XHTML+MathML, ce qui est suffisant dans de nombreuses situations.

Le convertisseur Hermes semble optimisé pour des documents AMSLaTeX. Il a été mis à profit par le Zentralblatt Math [23] qui utilise un vocabulaire TeX limité. Mais son développement semble interrompu et la documentation ou les outils de débogage sont très limités : il semble difficile de fonder une chaîne de conversion pérenne sur cet outil. C'est tout de même un choix crédible pour des conversions ponctuelles de documents rédigés en LaTeX conventionnel.

TtM est sans conteste le plus rapide des traducteurs qui comportent une interprétation TeX/LaTeX *ad hoc*. Malheureusement, il ne parvient pas à produire un fichier de sortie dans 27 % des cas car il ne comprend pas les catcodes et n'interprète pas les appels d'extensions. Par conséquent, ce convertisseur n'est pas adapté pour la conversion d'articles scientifiques généraux, mais peut se révéler efficace sur le sous-ensemble des commandes LaTeX qu'il traite. Il est donc plutôt à recommander pour des tâches de conversion spécifiques sur un corpus limité ou contrôlé.

Tralics est l'outil qui a donné le meilleur taux de conversions achevées mais la couverture la plus défaillante lors d'un emploi sans configuration particulière sur une collection fortement hétérogène. Cela est dû au fait que les macros prédéfinies concernent un ensemble restreint de styles et qu'il faut un peu de développement spécifique pour interpréter les particularités de chaque fichier source. Mais on est obligé de reconnaître que dans un grand nombre des cas que nous avons regardés, les fichiers produits restent intelligibles : les macros inconnues étaient en général inutiles pour la compréhension du contenu de l'article (utilisées par certaines classes de documents pour enregistrer des métadonnées ou la maquette spécifique d'un journal par exemple). Des exemples typiques sont les extensions XYPic, Pstricks ou PGF. Ce dernier conduit à la perte des images construites de la sorte, mais sans altérer le reste du texte. Cela dit, la documentation de Tralics pourrait donner des indications plus claires sur l'ensemble des étapes nécessaires pour passer de LaTeX à XHTML, plutôt que de s'étendre sur des informations de bas niveau pour développeur. Tralics a été utilisé par le Zentralblatt MATH pour convertir

toutes ses recensions : il a fallu moins de trois semaines pour mettre en place le support pour les macros maison et traiter plus de deux millions de documents relativement uniformes. Tralics est par conséquent un bon choix si l'on dispose des ressources pour développer les adaptations requises, étant donnés la qualité excellente du MathML produit et les possibilités de configuration.

LaTeXML est proche de Tralics pour la grande adaptabilité et la qualité excellente du MathML, mais il va un peu plus loin en fournissant d'origine un vaste ensemble de macros ou extensions préconfigurées. Il permet donc un déploiement rapide avec une large couverture tout en laissant la possibilité de paramétrage fin. C'est un logiciel très bien documenté, aussi bien pour les utilisateurs de base que pour les développeurs. Comme cela a été dit, c'est le convertisseur offrant la meilleur couverture parmi ceux que nous avons testés, ce qui est une conséquence directe du projet arXMLiv. Comme il offre en outre un support pour le MathML de présentation, LaTeXML est le convertisseur le plus polyvalent des cinq testés ici. LaTeXML est un omnivore qui ferait un bon choix aussi bien pour les conversions basiques ne requérant aucune configuration que pour des travaux plus spécifiques sur un corpus particulier.

5.3. Travaux ultérieurs

Il est évident que le travail exposé ici en est à ses débuts : nous devons inclure les réactions des auteurs des systèmes, et augmenter le nombre de systèmes étudiés, notamment ceux qui ne convertissent que des formules et non des documents comme BlahTeX, itex2mml, RiTeX, ou MathMLStudio Lite évoqués dans l'introduction de la section 2.

Bibliographie

1. Ron Ausbrooks et coll. — *Mathematical Markup Language* (MathML) version 2.0 (2e édition). W3C recommendation, World Wide Web Consortium, 2003.

2. Romeo Anghelache. — Hermes discontinued. Page web : `http://humanist.roua.org/2009/01/01/hermes-paused/`, dernière consultation en mai 2009.

3. Romeo Anghelache. — Hermes website. Page du projet : `http://hermes.roua.org/`, dernière consultation en mai 2009.

4. ArXMLiv build system. `http://arxmliv.kwarc.info`.

5. arXiv e-Print archive, dernière consultation en décembre2007. Page web : `http://www.arxiv.org`.

6. Thierry Bouche. — « CEDRICS : When CEDRAM meets Tralics ». In Petr Sojka, editor, *Towards Digital Mathematics Library, Proceedings of the DML 2008 workshop*, p. 153-165. Masaryk University, Brno, 2008. http://dml.cz/dmlcz/702540

7. Cecill license. http://www.cecill.info/, dernière consultation en mai 2009.

8. Digital library of mathematical functions. Page du projet : http://dlmf.nist.gov/, dernière consultation en mai 2009.

9. José Grimm. — Tralics, a LaTeX to XML translator, 2003.

10. Michael Kohlhase & Ioan Şucan. — « A search engine for mathematical formulae ». In Tetsuo Ida, Jacques Calmet, and Dongming Wang, editors, *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*, LNAI nº 4120, p. 241-253. Springer Verlag, 2006.

11. Bruce Miller. — LaTeXML website : http://dlmf.nist.gov/LaTeXML/, dernière consultation en mai 2009.

12. Rajesh Munavalli & Robert Miner. — « Mathfind : a math-aware search engine ». In *SIGIR '06 : Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, p. 735-735, New York, NY, USA, 2006. ACM Press.

13. John Plaice & Yannis Haralambous. — Omega. Site web : http://omega.enstb.org/yannis/.

14. EDP Sciences. — LXiR. Site web : http://www.lxir-latex.org/, dernière consultation en mai 2009.

15. Heinrich Stamerjohanns, Deyan Ginev, Catalin David, Dimitar Misev, Vladimir Zamdzhiev & Michael Kohlhase. — *A comparison study of mathml-aware LaTeX converters*. Kwarc report, Jacobs University Bremen, 2009.

16. Heinrich Stamerjohanns & Michael Kohlhase. — « Transforming the arχiv to XML ». In Serge Autexier et coll., editors, *Intelligent Computer Mathematics, 9th International Conference, MKM 2008 Birmingham, UK, July 28 - August 1, 2008, Proceedings*, LNAI nº 5144, p. 574-582. Springer Verlag, 2008.

17. TeX4HT. Site web : http://www.cse.ohio-state.edu/~gurari/TeX4ht/, dernière consultation en mai 2009.

18. Tralics. Site web : http://www-sop.inria.fr/miaou/tralics/, dernière consultation en mai 2009.

19. TtM. Page du projet : http://hutchinson.belmont.ma.us/tth/mml/, dernière consultation en mai 2009.

20. Validator. Site web : http://homepage.mac.com/rcrews/software/validator/, dernière consultation en mai 2009.

21. Stephen Watt. — MathML at ORCCA. Page du projet : http://www.orcca.on.ca/MathML/, dernière consultation en mai 2009.

22. W3C MATH WG. — Page « Mathml software — converters » : `http://www.w3.org/Math/Software/mathml_software_cat_converters.html`, dernière consultation en mai 2009.

23. Zentralblatt MATH. Page web : `http://www.zentralblatt-math.org`, dernière consultation en octobre 2009.

◖ Heinrich STAMERJOHANNS, Deyan GINEV, Catalin DAVID,
Dimitar MISEV, Vladimir ZAMDZHIEV & Michael KOHLHASE
Computer Science,
Jacobs University Bremen
(Germany)
<first initial>.<last name>@jacobs-university.de