Produire du MathML et autres... ML à partir d'Ω : Ω se généralise

John Plaice[1] et Yannis Haralambous[2]

(1) *School of Computer Science and Engineering, University of New Southe Wales, Sydney 2052, Australie,* plaice@cse.unsw.edu.au
(2) *Atelier Fluxus Virus, 187 rue Nationale, F-59800 Lille,*
yannis@fluxus-virus.com

Résumé

Le système de typographie Ω permet aujourd'hui non seulement la fine typographie pour diverses écritures, mais aussi la génération de fichiers SGML à partir de fichiers Ω. Les mathématiques peuvent être traduites automatiquement en MathML, et une redéfinition des macros LaTeX permet une génération de n'importe quelle balise SGML, donnant ainsi une grande puissance au rédacteur.

1. Introduction

Dans la communauté TeX, il est largement reconnu que le système TeX est un outil de choix pour imprimer des documents codés en SGML. Nous prétendons que l'inverse est également vrai, et ceci depuis peu : le système Ω est un outil de choix pour générer des instances SGML, y compris des fichiers contenant du MathML.

Le système Ω, conçu à l'origine pour étendre les capacités de typographie du système TeX afin de permettre la mise en page aisée de l'ensemble des langues et écritures du monde, a récemment été étendu, avec le soutien de l'*American Mathematical Society*, pour générer du SGML.

Ce système consiste en trois niveaux:

- De nouvelles primitives permettant de décrire les entités SGML qui se trouvent dans les fontes qui sont utilisées par Ω.
- Une modification interne au moteur Ω permettant la génération automatique de MathML à partir de mathématiques exprimées en langage TeX ou LaTeX.
- De nouvelles primitives permettant la génération de balises SGML.

Cet article est une documentation concise de cette fonctionnalité d'Ω, permettant une compréhension des grandes lignes du systéme actuel. Il sera étendu dans une version plus complète pour les *Cahiers GUTenberg*.

2. Les fontes

Quand Ω lit un fichier de métriques de fonte (fichier `.tfm` ou `.ofm`), il enlève le suffixe numérique de corps du nom du fichier (par exemple, `cmr10` devient `cmr`), puis cherche un fichier de ce nom avec le suffixe `.onm` (Ω *names*, `cmr.onm` dans ce cas-ci).

Ce fichier, s'il existe, est chargé comme n'importe quel fichier TEX, après le chargement du fichier de métriques de fontes. En théorie, on pourrait mettre toute sorte d'information dans ce fichier, mais normalement, on devrait y mettre uniquement des commandes définissant les entités qui se trouvent dans la fonte en question. Par exemple, les premières lignes de `cmr.onm` sont

```
\gdef\SGMLbold{\SGMLattribute{fontweight}{bold}}%
\gdef\SGMLno#1{\SGMLampersand\SGMLhash#1;}%
\gdef\SGMLname#1{\SGMLampersand#1;}%
\SGMLFontEntity{cmr}{"00}{\SGMLname{Gamma}}{mi}{}%
\SGMLFontEntity{cmr}{"01}{\SGMLname{Delta}}{mi}{}%
\SGMLFontEntity{cmr}{"02}{\SGMLname{Theta}}{mi}{}%
\SGMLFontEntity{cmr}{"03}{\SGMLname{Lambda}}{mi}{}%
\SGMLFontEntity{cmr}{"04}{\SGMLname{Xi}}{mi}{}%
\SGMLFontEntity{cmr}{"05}{\SGMLname{Pi}}{mi}{}%
\SGMLFontEntity{cmr}{"06}{\SGMLname{Sigma}}{mi}{}%
\SGMLFontEntity{cmr}{"07}{\SGMLname{Upsilon}}{mi}{}%
\SGMLFontEntity{cmr}{"08}{\SGMLname{Phi}}{mi}{}%
\SGMLFontEntity{cmr}{"09}{\SGMLname{Psi}}{mi}{}%
\SGMLFontEntity{cmr}{"0A}{\SGMLname{Omega}}{mi}{}%
\SGMLFontEntity{cmr}{"0B}{\SGMLno{64256}}{mo}{}%  ff  lig, U+FB00
\SGMLFontEntity{cmr}{"0C}{\SGMLno{64257}}{mo}{}%  fi  lig, U+FB01
\SGMLFontEntity{cmr}{"0D}{\SGMLno{64258}}{mo}{}%  fl  lig, U+FB02
\SGMLFontEntity{cmr}{"0E}{\SGMLno{64259}}{mo}{}%  ffi lig, U+FB03
\SGMLFontEntity{cmr}{"0F}{\SGMLno{64260}}{mo}{}%  ffl lig, U+FB04
```

Chaque commande `\SGMLFontEntity` définit un caractère. Elle prend cinq arguments:

- le nom de la fonte (sans indication de corps) ;
- la position numérique du caractère dans la table de la fonte ;
- le nom de l'entité correspondante ;

– un mot-clé parmi `mi`, `mn` ou `mo`, correspondant à *math identifier*, *math numeric* ou *math operator* ;
– éventuellement, des attributs supplémentaires.

Dans la définition des entités, on utilise régulièrement les macros `\SGMLno` et `\SGMLname`. La première permet de désigner des entités SGML numériques : par exemple, `\SGMLno{64256}` devient `&#64256;`. La seconde permet de désigner des entités SGML textuelles : par exemple, `\SGMLname{Gamma}` devient `&Gamma;`.

Dans la définition de `\SGMLno` on utilise deux primitives d'Ω : `SGMLampersand` et `\SGMLhash`. Celles-ci permettent la saisie de caractères « spéciaux » sans qu'il n'y ait de problème d'interprétation par le constructeur de jetons Ω. Nous avons ainsi les primitives :

```
\SGMLampersand    &
\SGMLbackslash    \
\SGMLcarret       ^
\SGMLdollar       $
\SGMLhash         #
\SGMLleftbrace    {
\SGMLpercent      %
\SGMLrightbrace   }
\SGMLunderscore   _
```

Décrivons maintenant la première ligne du fichier `cmr.onm` :

```
\gdef\SGMLbold{\SGMLattribute{fontweight}{bold}}%
```

La macro `\SGMLbold` est utilisée, par exemple, dans le fichier `eufb.onm` :

```
\SGMLFontEntity{eufb}{"28}{(}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"29}{)}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"2A}{*}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"2B}{+}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"2C}{,}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"2D}{-}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"2E}{.}{mo}{\SGMLbold}%
\SGMLFontEntity{eufb}{"2F}{/}{mo}{\SGMLbold}%
```

Si une balise est générée pour un de ces caractères, alors l'attribut

```
fontweight="bold"
```

sera ajouté dans la balise ouvrante. Nous verrons des exemples ci-dessous.

Voici la liste des fichiers `.onm` disponibles actuellement :

`cmbsy.onm`	`cmbx.onm`	`cmcsc.onm`	`cmex.onm`	`cmmi.onm`
`cmmib.onm`	`cmr.onm`	`cmsl.onm`	`cmsy.onm`	`cmti.onm`
`ecrm.onm`	`euex.onm`	`eufb.onm`	`eufm.onm`	`eurb.onm`
`eurm.onm`	`eusb.onm`	`eusm.onm`	`msam.onm`	`msbm.onm`

Il est à noter que les entités seront bientôt utilisables pour générer des messages plus facilement lisibles par un humain dans les fichiers log.

3. Les mathématiques

La norme MathML est une norme signée W3C qui est disponible sur le Web à l'adresse `http://www.w3c.org/Math/`. Elle consiste en deux parties : une partie dite de «présentation» et une dite de «contenu». Nous ne nous intéressons pour l'instant qu'à la présentation.

Si toutes les mathématiques de TEX utilisaient une forme préfixe pour les différents opérateurs, alors ce ne serait pas difficile de générer du MathML. Cependant, ce n'est pas le cas: les exemples les plus courants sont les opérateurs infixes `_`, `^` et `\over` (et ses variantes).

De plus, TEX et Ω considèrent que les indices et les exposants se mettent sur le dernier jeton dans le flux, ce qui ne correspond pas du tout à la structure d'une formule.

Donc, Ω est obligé d'interpréter des formules afin de générer du MathML « sensé ». Considérons l'exemple:

```
(x_1+y_1)^2 - \gamma_0^\infty + \mathcal{F}_0^{x^2}
```

qui donne la formule visible :

$$(x_1 + y_1)^2 - \gamma_0^\infty + \mathcal{F}_0^{x^2}$$

Le MathML généré est différent (voir Figure 1).

La partie

```
(x_1+y_1)^2
```

a d'abord été transformée en :

```
{ \left( { x_1+y_1 } \right) } ^ 2
```

Pour générer ce bout de texte, aucun changement n'a du être fait aux macros TEX ou LATEX. Tout a été fait en utilisant les entités définies par les fontes et le fontionnement mathématique interne. Pour démarrer ce processus, la primitive `\MMLmode` est utilisée. Le code MathML est écrit dans un fichier de suffixe `.mml` (ceci deviendra bientôt une option d'Ω).

En plus des mathématiques de base fournies par TEX, les langages LATEX et AMS-LATEX fournissent tout une panoplie de macros permettant de définir des entités mathématiques compliquées. Quand on veut générer du MathML, on doit souvent redéfinir ces macros pour qu'elles génèrent directement la structure nécessaire. Considérons, par exemple, l'expression

```
<mrow>
  <msup>
    <mrow>
      <mo> ( </mo>
      <mrow>
        <msub>
          <mi> x </mi>
          <mn> 1 </mn>
        </msub>
        <mo> + </mo>
        <msub>
          <mi> y </mi>
          <mn> 1 </mn>
        </msub>
      </mrow>
      <mo> ) </mo>
    </mrow>
    <mn> 2 </mn>
  </msup>
  <mo> - </mo>
  <msubsup>
    <mi> &gamma; </mi>
    <mn> 0 </mn>
    <mo> &infty; </mo>
  </msubsup>
  <mo> + </mo>
  <msubsup>
    <mi> &Fscr; </mi>
    <mn> 0 </mn>
    <msup>
      <mi> x </mi>
      <mn> 2 </mn>
    </msup>
  </msubsup>
</mrow>
```

FIG. 1: Traduction de la formule $(x_1 + y_1)^2 - \gamma_0^\infty + \mathscr{F}_0^{x^2}$

```
\sqrt{\zeta+1} + \sqrt[3]{\zeta+1}
```

donnant la forme visible:

$$\sqrt{\zeta + 1} + \sqrt[3]{\zeta + 1}$$

Le MathML généré est :

```
<mrow>
  <msqrt>
    <mi> &zeta; </mi>
    <mo> + </mo>
    <mn> 1 </mn>
  </msqrt>
  <mo> + </mo>
  <mroot>
    <mrow>
      <mi> &zeta; </mi>
      <mo> + </mo>
      <mn> 1 </mn>
    </mrow>
    <mn> 3 </mn>
  </mroot>
</mrow>
```

Comment a-t-il été généré ? En utilisant les macros suivantes :

```
\renewcommand{\sqrt}{\@ifnextchar[\sqrttwo\sqrtone}
\newcommand{\sqrtone}[1]%
  {\SGMLstartmathtag{msqrt} #1 \SGMLendmathtag{msqrt}}
\def\sqrttwo[#1]{\sqrttwoend{#1}}
\newcommand{\sqrttwoend}[2]%
  {\SGMLstartmathtag{mroot} {#2} {#1} \SGMLendmathtag{mroot}}
```

Avec quelques redéfinitions assez simples, il est possible d'interpréter un grand nombre de formules et d'obtenir des résultats raisonnables. Néanmoins, cette méthode ne suffit pas toujours pour obtenir la structure. La formule

$$\sin^2 x + \tan 2\pi x$$

génère le MathML suivant, est structurellement faux :

```
<mrow>
  <msup>
    <mi> &sin; </mi>
    <mn> 2 </mn>
  </msup>
```

```
  <mi> x </mi>
  <mo> + </mo>
  <mi> &tan; </mi>
  <mn> 2 </mn>
  <mi> &pi; </mi>
  <mi> x </mi>
</mrow>
```

4. Le reste

Les primitives `\SGMLstartmathtag` et `\SGMLendmathtag` ont des correspondants pour le texte : `\SGMLstarttexttag` et `\SGMLendtexttag`. Il est donc possible de générer du SGML à partir du texte Ω et pas simplement à partir des mathématiques. En plus, il y a deux autres primitives, `\SGMLwrite` et `\SGMLwriteln`, qui envoient des données au fichier de sortie, sans attendre la fermeture du groupe TEX les contenant.

Considérons l'exemple :

```
\title{Simulation of  Energy Loss  Straggling}
\author{Maria Physicist}
\maketitle
```

avec les redéfinitions suivantes :

```
\renewcommand{\title}[1]%
  {\def\thetitle%
       {\SGMLstarttexttag{title}%
        #1%
        \SGMLendtexttag{title}
        }}
\renewcommand{\author}[1]%
  {\def\theauthor%
       {\SGMLstarttexttag{author}
        #1
        \SGMLendtexttag{author}
       }}
\renewcommand{\maketitle}%
  {\SGMLwrite{<?xml version="1.0"?>}
   \SGMLwriteln
   \SGMLwrite{<!DOCTYPE document SYSTEM "latexexa.dtd" []>}
   \SGMLwriteln
   \SGMLwrite{<document>}
   \SGMLwriteln
```

```
   \SGMLstarttexttag{frontmatter}
   \thetitle
   \theauthor
   \SGMLstarttexttag{date}%
   \today
   \SGMLendtexttag{date}%
   \SGMLendtexttag{frontmatter}
   \SGMLwrite{<bodymatter>}
   \SGMLwriteln
  }
```

Nous avons la sortie suivante :

```
<?xml version="1.0"?>
<!DOCTYPE document SYSTEM "latexexa.dtd" []>
<document>
<frontmatter>
  <title>
    Simulation of Energy Loss Straggling
  </title>
  <author>
    Maria Physicist
  </author>
  <date>
    March 9, 1999
  </date>
</frontmatter>
<bodymatter>
```

Dans un test avec un document fourni par Sebastian Rahtz, nous avons rapidement redéfini les macros pour `\section` et `\subsection`.

```
\def\endsectionprefix@one{}
\def\endsectionprefix@two{%
   \SGMLendtexttag{section}%
}

\def\endsubsectionprefix@one{}
\def\endsubsectionprefix@two{%
   \SGMLendtexttag{subsection}%
}

\let\endsectionprefix=\endsectionprefix@one
\let\endsubsectionprefix=\endsubsectionprefix@one
```

```
\def\section#1{%
\endsubsectionprefix
\endsectionprefix
\SGMLstarttexttag{section}%
\SGMLstarttexttag{stitle}%
#1%
\SGMLendtexttag{stitle}%
\let\endsectionprefix=\endsectionprefix@two
\let\endsubsectionprefix=\endsubsectionprefix@one
}

\def\subsection#1{%
\endsubsectionprefix
\SGMLstarttexttag{subsection}%
\SGMLstarttexttag{stitle}%
#1%
\SGMLendtexttag{stitle}%
\let\endsubsectionprefix=\endsubsectionprefix@two
}
```

Les sections suivantes :

```
\section{Introduction}
\section{Landau theory}
\label{sec:phys332-1}
\subsection{Restrictions}
```

deviennent alors :

```
<section>
  <stitle>
    Introduction
  </stitle>
</section>
<section id="sec:phys332-1">
  <stitle>
    Landau theory
  </stitle>
  <subsection>
    <stitle>
      Restrictions
    </stitle>
  </subsection>
</section>
```

5. Le mot de la fin

Le système Ω permet maintenant la génération directe d'instances SGML, et soutient la forme dite « de présentation » de MathML. Nous travaillons actuellement pour qu'il devienne plus robuste et plus paramétrisable.