Cahiers GUTenberg

PRODUIRE DU MATHML ET AUTRES… ML À PARTIR D'Ω : Ω SE GÉNÉRALISE

Yannis Haralambous, John Plaice

Cahiers GUTenberg, nº 33-34 (1999), p. 173-182.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1999___33-34_173_0>

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

Yannis HARALAMBOUS

Atelier Fluxus Virus, 187 rue Nationale, F-59800 Lille, France,
<yannis@fluxus-virus.com>

John PLAICE

Université UNSW, Sidney, Australie,
<plaice@cse.unsw.edu.au>

Résumé. Le système de typographie Ω permet aujourd'hui non seulement la typographie fine 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.

Abstract. *Nowadays, the Ω typesetting system not only lets you generate typographically excellent documents in many scripts, but you can also use it to transform the input in your Ω files into* SGML. *In particular, mathematics expressions will be translated automatically into MathML while through a redefinition of the LaTeX macros any kind of* SGML *tags can be obtained, thus turning the editor into a powerful system.*

Mot-clés : SGML, MathML, Omega.

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 maintenant vrai aussi : le système Ω [1] est un outil de choix pour générer des fichiers SGML, y compris du MathML [2].

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 Ω ;
– de nouvelles primitives permettant la génération de balises SGML.

Ce document-ci est une documentation concise de cette fonctionnalité d'Ω, permettant une compréhension des grandes lignes du systéme actuel.

2. Fontes

Quand Ω ouvre un fichier de métriques de fonte (fichier `.tfm` ou `.ofm`), il enlève le suffixe numérique 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, une fois que le fichier de métriques de fontes est chargé. En théorie, on pourrait mettre n'importe quoi dans ce fichier, mais, normalement, on devrait mettre des commandes définissant les entités qui se trouvent dans la famille de fontes. 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 ligne `\SGMLFontEntity` définit un caractére. Elle prend cinq arguments :

- le nom de la famille de fontes ;
- la position numérique du caractère dans la fonte ;
- le nom de l'entité correspondante ;
- un de `mi`, `mn` ou `mo`, correspondant à *math identifier*, *math numeric* ou *math operator* ;
- des attributs supplémentaires.

Dans la définition des entités, les macros `\SGMLno` et `\SGMLname` sont utilisées régulièrement. Le premier permet de désigner des entités SGML numériques : par exemple, `\SGMLno{64256}` devient `&#64256;`. Le second permet de désigner des entités SGML textuels : par exemple, `\SGMLname{Gamma}` devient `&Gamma;`.

On voit deux primitives `SGMLampersand` et `\SGMLhash` qui sont utilisées dans la définition de `\SGMLno`. Elle permettent de taper les caractères « spéciaux » sans qu'il n'y ait de problèmes d'interprétation par le constructeur de jetons Ω. Nous avons aussi :

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

La première ligne du fichier `cmr.onm` reste à être définie :

```
\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 dans les fichiers logs qui seront inambiguës.

3. Mathématiques

MathML est une spécification signée W3C disponible sur le web à l'adresse `http://www.w3c.org/Math/`. Elle consiste en deux parties, une première dite de « présentation » et une deuxième 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 de faire un peu d'interprétation des formules afin de générer du MathML qui a un sens. 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 :

```
<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>
```

La partie

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

a 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 interne mathématique. Pour démarrer tout le processus, une primitive `\MMLmode` a du être tapée. Le texte est envoyé dans un fichier avec le suffixe `.mml`, mais cela deviendra bientôt une option.

LATEX et AMS-LATEX fournissent, en plus des mathématiques de base fournies par TEX, toute une panoplie de macros permettant de définir des entités mathématiques compliquées. Quand on veut générer du MathML, on peut souvent redéfinir ces ma-

cros pour qu'elles génèrent directement la structure nécessaire. Considérons, par exemple, l'expression

```
\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>
```

Mais 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 une redéfinition assez simple, il est possible de faire passer beaucoup de formules et de faire sortir quelque chose de raisonnable. Néanmoins, ce n'est pas toujours ce que l'on souhaiterait. L'expression

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

génère le MathML suivant, manquant beaucoup de structure :

```
<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. Autres points

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. De plus, il y a deux autres primitives, \SGMLwrite et \SGMLwriteln, qui n'attendent pas qu'un groupe soit complété pour imprimer dans le fichier de sortie.

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 document test fourni par Sebastian Rahtz, nous avons redéfini les macros `\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 suites de 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. Conclusions

Le système Ω permet maintenant la génération de documents codés en SGML et soutient directement la forme dite de présentation de MathML. Actuellement, nous travaillons pour qu'il devienne plus robuste et plus paramétrisable.

Bibliographie

[1] Yannis HARALAMBOUS et John PLAICE. « Ω, une extension de TEX incluant Unicode et des filtres de type Lex. »
Cahiers GUTenberg, n° 20, pages 55–80, mai 1995.
Voir également l'URL `http://www.gutenberg.eu.org/omega/`.

[2] World Wide Web Consortium, Patrick ION et Robert MINOR (rédacteurs).
Mathematical Markup Language (MathML), 1.0 Specification.
`http://www.w3.org/TR/REC-MathML`.