INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

Project IST-2001-33562 MoWGLI

Prototype n. D4.d
LaTeX - based authoring tool
(first prototype)

Author: Romeo Anghelache

Project Acronym: MoWGLI
Project full title: Mathematics On the Web: Get it by Logic and Interfaces
Proposal/Contract no.: IST-2001-33562 MoWGLI

1 Introduction

The automatic (with no manual, content oriented, input from the author) and complete (arbitrary mathematical expressions) conversion into Content-MathML from a presentation oriented authoring system as LaTeX is, in principle, impossible: there are many representations with the same meaning.

The authoring tool described here is created to support automatic generation of Content-MathML. It also can achieve a more general goal: adding and recovering semantic depth and clarity to LaTeX written documents, which makes it relevant to e-publishing and digital libraries, as a method of recovering and storing scientific content that can go beyond mathematics (e.g. it can be extended to cover ChemicalML, SVG etc.). A natural name for a tool with such a goal is **Hermes**, an we will use it from now on.

The first **Hermes** prototype provides, as output, an XML file with a typical structure of an article, containing Content-MathML islands; this structure is entirely determined by the LaTeX source, the semantic depth of the output depends essentially on using the macros **Hermes** provides.

2 Description

Hermes complements the LaTeX system: it enables the authors of scientific articles making explicit the semantics of their work, preserving a high quality rendering. It is written from scratch.

The first prototype implementation of **Hermes** has the following components:

- a set of helper LaTeX macros, which allows the author to disambiguate the meaning of the mathematical expressions he writes, while allowing some choices for the presentation; this set is included by the author in the originally written LaTeX document (it resides in the 'definitions' file in the **Hermes** distribution, detailed in 4.1). A LaTeX run on the macro-enriched document will output a 'semantic **dvi**' file (a **dvi** file containing 'special' annotations of various combinations of graphical and nongraphical symbols in the source).

- a scanner, written in **flex**, which extracts from the resulting **dvi** file the semantic tokens seeded by the macro collection above and sends them to the parser below (the 'hermes.l' file in the **Hermes** distribution, detailed in 4.2).

- a parser, written in **bison**, which is a grammar that performs a semantic action when a structured set of tokens is recognized (the 'hermes.y' file in the **Hermes** distribution, detailed in 4.3); the semantic action is the creation of parts of the XML output; the parser and the scanner compile into a 'semantic **dvi**' compiler called 'the **Hermes** compiler'.

The first **Hermes** prototype handles consistently only those mathematical expressions in LaTeX which contain structures covered by the current Content-MathML standard (MathML version 2.0). A subset of these structures can be and are automatically inferred from the unaltered LaTeX source (at the cost of semantic depth), the rest are provided as supplementary macros; they reside and are briefly commented in the 'definitions' file.

3 Architecture

Hermes as a program needs a well documented input and does not replace nor modify the functionality of the TeX engine, thus, it does not restrict the set of macros used while authoring the original document: it uses the **dvi** format as input.

Hermes is content oriented, therefore an emphasis is put on generating Content-MathML. Generating content requires a high degree of accuracy in fitting the output structures with the authored input as it is intended for machine consumption (search engines, mathematical computation): therefore it has a compiler behaviour (it strictly flags ambiguous input as errors in the process of conversion and stops). The first prototype handles exclusively Content-MathML covered structures, it is in 'beta' stage of development along this direction.

Hermes is also document oriented. It aims at generating the semantic information available typically in a legacy scientific article (text, keywords, references, author information, document structure etc.) or supplementary layers of metadata for the newly created documents. The first prototype is in 'alpha' stage of development along this direction.

Hermes preserves the presentational output of the original source documents. This is a feature of the **Hermes** macros: they should leave the graphical objects unmodified (if they are used for making legacy TeX documents semantically rich) while attaching semantics to them in the background. The first prototype is in 'beta' stage of development along this direction.

Hermes should let the author the freedom to sharpen the meaning of an arbitrary LaTeX chunk, but should also be prepared to convert a legacy source document with no manual intervention. In the latter case only a subset of Content-MathML and only the already specified (e.g. citations, author name, keywords, abstract) metadata subset will be generated; the arbitrary mathematical symbols encountered (e.g. Q^+) will generate only Presentation-MathML if nothing else (no author specified metadata and no **Hermes** macro is explicitely used) makes their meaning precise. This feature enables gradual annotation of scientific work and allows adding semantic depth (e.g. improving its reachability on the Internet or its compatibility with a new mathematical software tool). The first prototype is in 'alpha' stage of development along this direction.

4 Source code distribution

The **Hermes** first prototype's source distribution consists of 3 files: the semantic macros, the token vocabulary and the grammar. The first is necessary to author content-oriented documents, the last pair is necessary to create the **Hermes** compiler, which is aware of the content oriented macros above.

For compiling an example, this source distribution comes also with a stylesheet ('pre.xsl') which prepares the XML output of the **Hermes** compiler for rendering as XHTML with MathML islands (this, in turn, obtained by filtering this output through the generic MathML stylesheet, 'mathml.xsl', from w3.org), and a makefile which automates the creation of the **Hermes** compiler and creates a renderable Content-MathML example out of a LaTeX source example.

4.1 Definitions

Recovering or adding semantics is achieved by leaving appropriate traces into the **dvi** file using the LaTeX 'special' command (at low level, by activating some of the characters or simply prefixing the old LaTeX command with a 'special' string); these traces are enabled by a set of macros residing in the 'definitions' file. The way they should be used is mostly self-explanatory: some of them decorate the corresponding old TeX ones (the author simply uses the same TeX commands), the rest are supplying the structures needed to cover Content-MathML mahematical expressions (the author needs to use these ones if he wants to enable Content-MathML output, they usually start with a capital letter), and all of them are commented.

The semantic traces are tokenized by the scanner.

4.2 Scanner

The scanner uses regular expressions and context conditions to recover the tokens from the **dvi** file; it understands all the **dvi** commands and also keeps track of the current font and coordinates trough an internal stack.

The handled tokens are the ones defined by the macros described above and all the bytecodes typically present in the **dvi** file are dealt with. The first prototype ignores most of the presentation oriented information available in the **dvi**, but does not preclude further enhancements to enable a more accurate rendering too.

The way the scanner source is organized makes it easy to understand the categories of tokens it tackles: basic tokens (e.g. letter 'L'), TEX tokens (e.g. 'PLUS', 'SQRT'), structured tokens (e.g. 'BMoment' and 'EMoment', along with 'BMomentDeg' and 'EMomentDeg' etc.) that come in pairs (prefixes Begin=B, End=E) wrapping the structure inside.

A 'C' variable ('drop') is used in the scanner to decide when to forward the next token to the parser or simply ignore it. This is useful in simplifying the process of writing or reading the content oriented grammar (it is used to neglect some of the graphical glyphs where there is enough semantic information to render it precisely), but it will have to disappear when the presentation oriented code will be implemented, and the burden of handling them will be handed out to the parser.

4.3 Parser

The parser expects various combinations of semantic tokens from the scanner. When a structure is recognized, the appropriate XML output string of characters is built. The first prototype of **Hermes** recognizes LATEX inline or display mathematical areas and builds the corresponding Content-MathML code.

Some of the operators or variables in the source documents are recognized implicitly (e.g. 'VEE' or 'OVER'), in these cases there is no need for any **Hermes** provided macro to create the appropriate Content-MathML code (e.g. `<or/>` or, respectively, `<divide/>`).

Others are provided by **Hermes** as explicit complementary macros (e.g. 'Laplacian' or 'Listl' in the 'definitions') which also have associated with them a specific rendering in a normal (pdf)LATEX run.

The accented letters or greek symbols give a Presentation-MathML code which is embedded in Content-MathML.

The rest of the parser is made of 'C' routines. Some of them put the corresponding XML tags in the right place, based on usual mathematics precedence rules or the nature of the mathematical entity under treatment. Other routines, executed at the end of a structure recognition, prepare the intermediary string for a final ordering; yet other routines are simple helpers for the above or do the pretty printing of the XML output.

5 To do

In the real world, authors need, along beyond the most usual symbols, or **Hermes** provided macros, arbitrary mathematical expressions they feel most appropriate for rendering a particular meaning; some of their choices become de facto standards (e.g. $\sqrt{x}$) so **Hermes** has no difficulty in generating the appropriate content oriented XML, others remain ambiguous from a machine point of view (e.g. Q^+), i.e. there is not enough information for a machine to infer what their meaning were.

There is no realistic (i.e. easily acceptable by the user) alternative solution to the problem above but to convert those arbitrary symbols into Presentation-MathML and let the author complement the source of the arbitrary symbol with simple annotations if he feels the need to do so (and not forcing him to obey a non-standard, external to his way of thinking, set of conventions); these annotated Presentation-MathML structures, along with the Content-MathML, enable a potential reader to locate mathematical expressions on the web by their meaning and not by their particular rendering (which, obviously, cannot be known before accesing the document itself).

Therefore, a truly viable and complete **Hermes** system should go beyond converting from LaTeX to Content-MathML, that is, should be prepared to convert and annotate arbitrary mathematical expressions, not yet covered by the current Content-MathML or OpenMath standards, into Presentation-MathML.

This is how we interpret 'refining and extending' the **Hermes** prototype, as expressed in the MoWGLI proposal.