INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

Project IST-2001-33562 MoWGLI

D4.e Refined and extended protoype of the LaTeX - based authoring tool

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 authoring tool described here is created to support automatic generation of XML and MathML from LaTeX sources. We call it **Hermes**: because we interpret it as the carrier of (meaningful) messages. 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.).

The refined **Hermes** prototype generates an XML file with a typical structure of an article, containing 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 to sharpen the semantics of their work while preserving a high quality rendering. **Hermes** is written from scratch.

The refined 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 'ltxdef.tex' 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 'semantic **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**' translator called 'the **Hermes** translator'.

The refined **Hermes** prototype handles mathematical expressions in LaTeX as follows:

1. arbitrary expressions/symbols are encoded first in Presentation-MathML
2. expressions which have a clear meaning are wrapped further in Content-MathML

3 Architecture

Hermes 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, this freedom is a result of using the **dvi** format as input.

Hermes is content oriented: a maximal effort is made 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), so, in those places where the input is too ambiguous, the output will be only Presentation-MathML.

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 refined prototype also handles references to bibliographical items or equations.

Hermes preserves the presentational output of the original source documents. This is a feature of the **Hermes** macros: they leave the graphical objects unmodified (if they are used for making legacy TEX documents semantically rich) while attaching semantics to them in the background.

Hermes lets the author the freedom to improve the meaning of an arbitrary LATEX chunk, but is also be prepared to convert a legacy source document into a renderable XML with no manual intervention.

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 refined prototype is in 'beta' stage of development along this direction.

4 Source code distribution

The **Hermes** refined 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, or to transform a legacy LATEX document into a content-oriented document; the last pair is necessary to build the **Hermes** translator, which is aware of the content oriented macros above.

To help generate an example, this source distribution comes also with a stylesheet ('pre.xsl') which prepares the XML output of the **Hermes** translator for rendering as XHTML with MathML islands (the latter, in turn, is obtained by filtering this XML output through the generic MathML stylesheet, 'mathml.xsl', from w3.org).

The distribution contains also 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 from LATEX sources 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 ensure Content-MathML output, they usually start with a capital letter), and all of them are commented.

The semantic traces are tokenized by the scanner, along with the characters in TEX's fonts.

4.2 Scanner

The scanner uses regular expressions and particular states to recover the tokens from the **dvi** file; it understands all the **dvi** commands and also keeps track of the current font and movements through an internal stack.

The handled tokens are the ones defined by the macros described above and all the byte-codes typically present in the **dvi** file are dealt with. The refined prototype maps each code in the TeX fonts into the corresponding code in Unicode, where there is a Unicode equivalent glyph.

The way the scanner source is organized makes it easy to understand the categories of tokens it tackles: basic tokens (e.g. 'ANY'), TEX tokens (e.g. '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.

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 refined prototype of **Hermes** recognizes LATEX inline or display mathematical areas and builds the appropriate MathML code.

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

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

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 and context 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.