INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

Project IST-2001-33562 MoWGLI

Deliverable n. D6.d
Final MoWGLI Prototype

Main Authors:
all partners

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

Contents

1 Overview

The Final MoWGLI Prototype is an enhancement of the Advanced MoWGLI Prototype meant to implement the new functionalities and enhancements resulting from the validation phases.

2 Introduction

The MoWGLI project aims at exploiting the possibilities offered by a content-oriented, machine understandable encoding of mathematical knowledge in the creation of a digital library of mathematics. Currently, in accordance with the timetable of the workplan, all the functionalities expected in the final version of the prototype have been implemented as prototypes that have been tested and refined during the evaluation period.

The library comprises documents from the Coq standard library as well as from several other Coq contributions, and TEX documents that have been processes with the LATEX-to-MathML converter tool (deliverable D6.c). These two kinds of documents (formal mathematical proofs and TEX articles) have been chosen because they represent two opposite categories: highly structured, automatically generated pieces of mathematical knowledge and handwritten scientific articles in a widespread typesetting markup language.

In the following sections we briefly summarize the functionalities that have been added or enhanced in the advanced prototype as a result of the validation phase.

3 Browsing and searching

3.1 Enhancement to the browsing interface

The browsing interface has been completely re-designed and partly re-implemented to improve usability and accessibility of information (see Figure 1):

- The new interface limits the use of frames, which hindered the possibility of linking specific theorems and caused several technical problems with the JavaScript code embedded in the HTML pages in the previous prototype.
- For each object in the library, the following clearly distinct sections are available in the rightmost "menu", from top to bottom:
  1. the profile (decribed later);
  2. the name of the object;
  3. a set of *actions*, including links to the HTML and MathML renderings, a link to the proof tree (if available), a link that causes the proof checker to validate the object, links to the pages showing the mutual dependencies between objects;
  4. a set of *documents* related to the displayed object representing the various encodings (CIC XML, OMDoc with embedded MathML Content, XHTML with embedded MathML Presentation, XHTML) and the metadata (in RDF and Dublin Core formats).
- The main frame displays the object along with its full URI where the fragments of the URI are separate links enabling a quick navigation across sections.

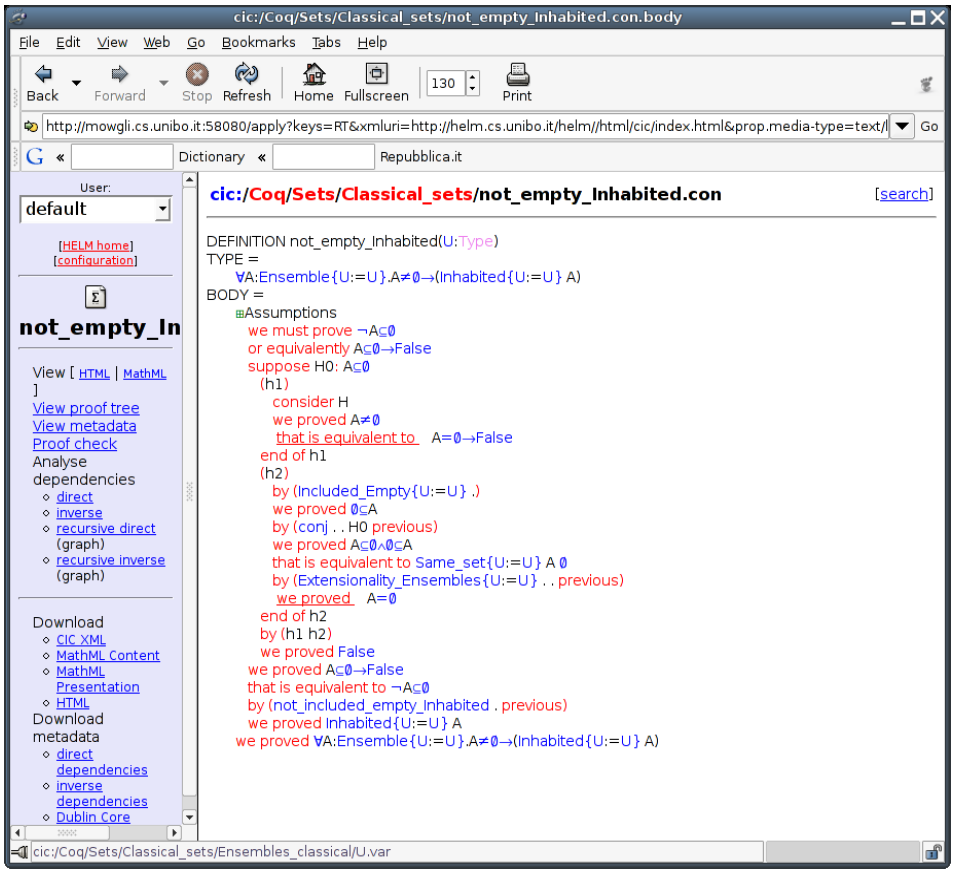

Figure 1: The re-newed browsing interface

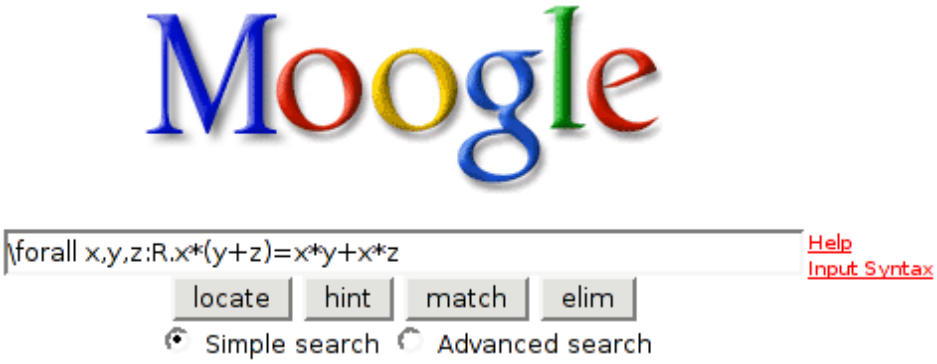

Figure 2: The re-newed searching interface.

- A notion of *user profile* has been added into the UWOBO Web service (see Section 3.2).

3.2 User profiles in UWOBO

The UWOBO Web service now supports a notion of *user profile*, consisting of a simple database of pairs of the form (*key*, *value*). The basic idea is that a number of parameters that affect browsing and rendering of the library should be customizable even though they do not change often. Hence, an user of the library may create her own profile and identify herself by means of a simple profile identifier (for instance, `default`). This solution has the effect of shortening the URLs for accessing the library (previously all the parameters were mangled in the URL, sometimes reaching the its maximum length limit) and providing a mechanism that can be more easily extended in case of future enhancements requiring more parameters to be passed around.

Appropriate Web pages have been set up for remotely changing the information stored in profiles. In order to avoid accidental or malicious modifications in the profile data, a simple password-based protection mechanism has been implemented. Although trivial and not completely secure, this mechanism is sufficient for the purposes of the prototype.

Currently, profiles are used for storing the URLs of the several Web services used in the MoWGLI architecture, as well as other parameters affecting the transformation phase (use of Unicode fonts, natural language presentation of displayed proof objects, and so on). Since data stored in the profiles must be accessible during the transformation phase, and in particular during the execution of XSLT transformations, upon request the UWOBO Web service can provide profile information as an XML document, thus enabling subsequent processing and scanning of the information therein contained by means of the usual XPath navigation facilities.

3.3 Enhancements to the searching interface

The Web interface to the Search Engine, which we temporarily named "Moogle", has been redesigned. The interaction with the user has been simplified. As can be seen in Figure 2, Moogle looks like a typical search engine, offering four different kind of queries, one for each of the queries made possible by the MoWGLI metadata set:

locate retrieve a mathematical object (theorem, definition, lemma, ...) from its short name;

match allows to retrieve all mathematical objects matching a pattern provided by the user (possibly containing wild cards);

elim takes in input a datatype and gives back the list of its elimination principles;

hint takes in input a (closed) statement and gives back a list of theorems of the repository which can be applied to the conclusion of the statement in the attempt of proving it in a backward fashion.

Locate and hint queries have been improved. Hint is now able to score results. Two criteria are used for scoring: number of symbols present in the statement given by the user which are also present in results (the more, the better) and number of goals that results application will leave open (the less, the better). Locate now supports shell-like wildcards ("?" meaning "any character", "*" meaning "any sequence of character") in names given by the user.

Input syntax has been revised as well. User can now input mathematical formulae using a TEX-like syntax. Handling of results has not been changed. Moogle is still a Web service which returns the list of URIs matching user query. Those URIs are then processed by UWOBO in order to render their conclusion. As an additional feature, pagination of results has been added: no more than 10 results are shown in a single page.

The implementation of interface changes and query improvements did not require changes in set of collected metadata.

4 Feedback from Validation 1: Education

We briefly describe in this section the major enhancements with respect to the advanced MoWGLI prototype that have been introduced as a result of the validation phase for education. An extensive account of the rationales and a more detailed description of the enhancements can be found in **Deliverable D6.a**.

4.1 Links

The linking mechanism that allows MoWGLI documents to refer to other documents has been extended as follows:

- It is now possible to have (relative or absolute) links to documents that are not part of the MoWGLI library. These relative links have to be relative to where the "raw" document is (where the Getter gets it from) and not relative to the URL seen by the browser, namely an UWOBO service request. They thus have to be rewritten to absolute URLs for the browser.
- Links to documents in the library can now be specified more or less anywhere in the markup, not just within `HREF` attributes.

As a consequence, the URI-to-URL rewriting engine has been adapted to act only on an explicit request, in form of a `helm:helm_link` or `helm:external_link` attribute to any node; the value of this attribute gives the name of the attribute of the same node that needs to be rewritten. In this way, any attribute of any XML node can undergo URI-to-URL rewriting. The Getter now puts an `xml:base` tag in the document giving where it got it from, and the rewriting engine uses that information to unrelativise relative links.

4.2 Coercions handling

The treatment of all coercions has been factored into a generic template, resulting in a simpler and more maintainable code.

4.3 In-line Rendering

In order to enable a more flexible mechanism for the embedding of formal definitions, it is now possible to have a rendering that

- integrates in a sentence, as a word or phrase, instead of being its own paragraph. We call this the *in-line* rendering.
- Misses / contains some parts of the definition of the object that are included / not included by default, such as the name of the object, the body of the definition or the notation for the object.

These choices are all independent, although not all combinations make sense and will give a reasonable result. Syntactic shortcuts have been added for common combinations such as "only the notation for the object, in a way that integrates in a sentence", i.e. the notation for the object as a single word, without any paragraph break.

4.4 Automatic generation of the notational stylesheets

The feedback coming from the validation experience resulted in some adaptations that affect the definition of the input XML language, the definition of the intermediate XML language, the meta-stylesheet from CIC-XML to MathMLContent, the meta-stylesheet from MathML-Content to HTML, and the arrangement of the input files.

5 Feedback from Validation 2: Smart Card Security

Trusted Logic explored the use of Mowgli's prototype as a support for the certification of IT products based on the Common Criteria standard (CC). This international standard is presently recognized in several countries of the European Union, like France, Germany, Spain and the UK. This is a short account of the feedback resulted from this validation phase. A more detailed account can be found in **Deliverable D6.b**.

5.1 Extracting meta-data from Coq source files

Mowgli's prototype was tested on the Java Card formal model, which contains more than 3750 formal definitions and 2000 theorems along 4MB of Coq source code. The intended purpose was to test Mowgli as a support for explaining those models to the evaluator. Three main observation arisen from this experiment:

1. The installation of Mowgli's prototype is not a simple task. This is the consequence of at least three implementation choices. First, it is based on an heterogeneous implementation based on several different languages: ocaml, Perl, MySql, XSLT, PXP, etc. Second, it depends on several (unstable) Linux packages developed by the free software community. Third, it is based on a completely open architecture, where information can

be exchanged and accessed with no restrictions, a vision that cannot be easily integrated to firewalls and other control access functions.

2. The rendering that is actually available in Mowgli's prototype is better suited for explaining Mathematics than Software. For instance, there is no support for the correct rendering of record types, functions are presented from the lamba-term perspective (and not as programs) and indentation is not always as expected. More important, there is no support for simplifying rendering modification, like an intermediate language of "boxes".

3. Mowgli's prototype takes as input the compiled format of Coq files, so all comments present in the Coq sources are lost. This is annoying, because the CC standard requires formal definitions to be accompanied by explanatory text.

4. Other information present in Coq sources that could improve readability and be exploited by rendering tools is also missed, like the coercions and the implicit arguments.

As a contribution to the point (1) above, Trusted Logic collected and packaged all the libraries that are required for compiling Mowgli's prototype an write a short installation guide to help other users. Concerning the points (3) and (4), Trusted Logic developed a tool for retrieving source information as meta-data. This includes several kinds of information that present in the sources but that is not part of logical terms, like comments on the formal definitions included in the source files, coercions, implicit arguments, derived constants, etc.

5.2 Security Policy document

One of the main conclusions from this experience is that Mowgli's prototype could be enriched with an XSLT transformation for displaying formal definitions in English. Actually, when writing an SPM model (or a scientific article describing a proof in Coq), most of the authors re-phrase in English the formal definitions in Coq with the aim of clarifying its meaning to the reader. Similarly to what Mowgli's prototype actually does with formal proofs, definitions could also be translated into structured English.

6 Feedback from Validation 3: Journal interface

All the currently published articles in the Living Reviews in Relativity collection have been converted to XML+MathML using Hermes. This phase provided a testbed for constructing the Hermes macros corresponding to the major LaTeX packages used.

No major structural changes were necessary in the software architecture since the extended prototype version.

A lot of improvements have been made, all of them beyond the original MoWGLI requirements: recognition of more document metadata, a larger number of TeXfonts are mapped to Unicode, two supplementary macro sets have been added (for AMSTeX and AMSLaTeX) to support the transformation of these, major packages specific, mathematical regions (structures like multline, gathered, split, etc.), used at authoring time, into the corresponding MathML macro-structures.

The gradual semantic annotation and document structure continue to belong to the 'beta' stage as they evolve slowly according to the new users' needs.

The content macros (responsible for creating MathML-content) in the distribution are unchanged, user interest seems to be low in this direction, mainly because the benefits are not obvious and the lazy approach of converting an already authored paper is less tedious than using a new vocabulary at authoring time. We expect that this interest will grow, slowly, once the benefits of expressing mathematics in XML become more popular.

7 Status of XMLCVS (Version Control for Structured Mathematical Documents)

Existing content management systems (like CVS or Subversion) treat documents as text files, i.e. as a sequence of text lines. Changes of documents are represented as the list of text lines that have been changed. The level of granularity on which changes can be maintained in CVS or Subversion is the level of text lines. Mathematical documents, however, are typically organized in a tree-like or even graph-like structure. For example, mathematical terms are trees or proofs are encoded by proof trees consisting of the individual inference steps. Thus, there is a need for describing changes of mathematical documents in terms of trees. In order to describe that an argument of a function has been changed we have to deal with subtrees of a document that is considered as a tree. Therefore mathematical entities are represented in an XML-language and the content management system has to be aware of the intrinsic tree structure of such documents. Since there is no adequate way to encode and deal with changes of trees as changes of text lines and vice versa, we had to build up CVS-like operations DIFF. PATCH and MERGE operating on trees instead of sequences. In order to make use of the large effort spent in developing Subversion, we tried to convince the developers to restructure their program to support generic DIFF. PATCH and MERGE operations. While in theory the overall management system should be independent of the concrete instantitations of DIFF. PATCH and MERGE, it turned out that the overall implementation is heavily based on the assumption that changes are encoded as sequences of text lines. Thus in order to satisfy the time restrictions of the Mowgli-project we had to build our own content management system, which we call XML-CVS.

Despite the additional amount of work, we provide a demonstrator version in alpha status of the XML-CVS tool (available for download at `http://www.mathweb.org/releases/xmlcvs/`). It is based on the three basic programs for semantic DIFF, PATCH and MERGE of XML documents. The algorithms are parameterized over the language specific description of the semantics. Lacking a declarative language for such descriptions, the parameterization is only visible at the level of the source code of these algorithms and the actual implementations are only instantiated for the OMDOC language. For the representation of conflicts we designed a special language to represent subtree conflicts inside the XML document tree structure. This eases the automatic detection of documents containing unresolved conflicts and commits of such documents can easily be prevented. Furthermore, the representation of conflicts as XML elements allows to use standard XML editing tools for the visualization of XML documents containing conflicts.

The structure of XML-CVS is a standard repository structure which allows for concurrent working on local copies. A difference to CVS is that all DIFF and MERGE operations are performed on the client side which allows to mainly rely on the exchange of patch descriptions rather than complete XML documents. The cascading of repository, as planned in the design of XML-CVS, could not be implemented yet. However, this feature will be implemented after

completion of the Mowgli project, in order to have it available for ongoing and future projects in which we plan to use the XML-CVS tool.

Projects in which we plan to use the XML-CVS tool are: The OMEGA project in the Collaborative Research Center "Resource-adaptive Cognitive Processes" (SFB 378) at Saarland University[1], the long-term research project VERISOFT[2]), and the development of the ActiveMath-tool.[3]

[1]See `http://www.coli.uni-sb.de/sfb378/`, funded by the German DFG

[2]See `http://www.verisoft.de` funded by the German BMB+F

[3]See `http://www.activemath.org`; funded, between others, by the EU in the FP6 project LeActiveMath `www.leactivemath.org`