INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

Project IST-2001-33562 MoWGLI

Report n. D1.b
Structure and Metastructure of Mathematical Documents

Main Authors:
G. Goguadze, E. Melis (DFKI), A. Asperti (University of Bologna)

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 Introduction

The MoWGLI partners work on very different mathematics applications, ranging from formal proofs to the publication of informal mathematical texts. In order to take advantage of each other's repositories and knowledge, the knowledge representations that are still heterogeneous have to be structured and annotated in a way that the knowledge can be used by common services that will be built and that will provide an added value for the community.

This report is the basis for developing a common knowledge representation/XML-markup that can serve the tools that will be developed in MoWGLI such as search and retrieval tools, dictionaries, standard APIs, and authoring tools, editors, transformations, and presentation style sheets and those tools developed world-wide that are based on Web-standards.

Therefore, envisioned markup for heterogeneous mathematics applications has to be system-independent and machine-readable. It will have to take into consideration different layers of mathematics: presentation, structure, mathematical ontology, formal content, and logical context.

Similarly to other domains, these information layers have to be reflected in different layers of the markup. Compared with *syntactic* markup in languages such as HTML and LaTeX, *content* markup provides additional information about the structure of a document and the ontology of the domain at hand see §2.1. In addition, modular information that is relevant for a direction of application such as in education can be provided. Since the formal applications of mathematics are based on different logical foundations these have to be additionally introduced into the markup. We shall call this logical context markup. [1]

Content Markup

Standardization of content encoding can be profitably and successfully pursued as already testified by content-MathML and OpenMath. Content markup defines structures, formulas and *names* of mathematical operators. At the content layer, the intended applications include:

- search and retrieval of the content items
- copy and paste of the mathematical expressions to Computer Algebra Systems and to proof systems
- transformation to a suitable presentation format
- semantic refinement that can be envisioned as a system of backward pointers to the logical context encoding of the same item

Context Markup

By *context markup* we mean a description that is used for formal elaboration (computation, theorem proving, proof checking, etc.). This gives raise to requirements for the markup: it should contain enough information to

- allow a logic based formal proof system to use the information as an input

[1] This is somewhat (not fully) similar to the different ontologies developed by different companies for a domain or organization. These ontologies might not map each other totally and may denote the same object differently.

- automatically search and retrieve formal expressions allowing for applying the search heuristics of the underlying logical system
- automatically transform to content markup

2 Structure/Ontology of Mathematical Knowledge

Mathematics has a richly structured language. In this section, we discuss the structured and ontological information in mathematical documents to be encapsulated into a markup.

The ontology comprises different kinds of mathematical items such as theorems, proofs, and examples (see §2.1). On the one hand these items may have a structure themselves. On the other hand they can form new collections/structures in documents such as a version or a particular sub-document.

On examples of particular applications and representation languages that are in the focus of the project (e.g. HELM using COQ [2] or ActiveMath using `OMDoc` [15, 11]) we can see how different applications impose diverse structuring of mathematical knowledge.

2.1 Items of a Mathematical Ontology

Mathematical items are basic components of a library and of a domain ontology (definitions, theorems, conjectures, examples, and so on). Basically, these are the explicitly referentiable objects of a library and can be identified by a Uniform Resource Identifier (URI). The scope of such identifiers is global and differentiates them from other locally referentiable entities such as local variables, assumptions, or proof justifications that live inside a mathematical item.

A generic URI [3] is made of a formatted (structured) string of characters whose intended meaning is associated with the applications managing it. URLs (Uniform Resource Locators) are a particular kind of URIs specifically designed to name resources accessed by means of a standard protocol (for example HTTP). URLs consist of a part identifying the protocol, a host name, and a part to locate the resource on it. URLs can be resolved by standard tools and browsers but suffer from problems of consistency: moving the target document leads to dangling links. Moreover, being physical names, they cannot easily be used to identify a set of copies located on different servers for fault-tolerance and load-balancing purposes. URIs, instead, can be designed as logical names leaving the burden of resolution to physical names to the applications.

A complete definition of a URI mechanism is important for the MoWGLI project. Applications using the MoWGLI encoding will have to convert the URIs to appropriate URLs or user-interface actions and back. Authoring tools will have to encode the URIs in the most portable and readable fashion. Finally validators and search engines will manage URIs and references and will have to resolve them efficiently.

Next we list the atomic structural elements relevant for mathematics, some of their properties and relations. The items or elements that occur in many mathematical documents are **definition**, **conjecture**, **axiom**, **theorem proof**, **proof method**, **algorithm**, mathematical structure such as the real numbers, etc.

In a publication or educational application other items can occur such as **example exercise**, different kinds of **text** (explanations, motivations, introductions etc.), figures, multimedia elements

Elements that are not specified as stand-alone structural elements and therefore, not referred to by a URI but included into those structural elements are, e.g., **formula**, **type**, **expression**.

At the level of items, the distinction between logical context and content is less dramatic. For instance, in the Calculus of Inductive Constructions there are the following basic kinds of items: definitions, axioms and inductive definitions. At the logical level there is no explicit distinction between a definition and a theorem (implicitly it can be "inferred" from its type). Moreover, the notion of inductive definition which is a list of mutually inductive types, is logic-prone with a subtle formal semantics "characteristic".

Coming from different systems and logical foundations, the MoWGLI partners must face the problem of possibly collecting all the basic kinds of objects or try to abstract them to a smaller set of "common" constructions with a pseudo-formal intuitive semantics. Concretely, `COQ` inductive definition do not currently fit within the `OMDoc` categories of "definitions".

2.2 Mathematical Micro- and Macro-Structure

2.2.1 Micro-Structure

Micro-structure refers to the internal structure of mathematical entities such as expressions, formulas, types, theorems, definitions, proofs and other items introduces in the section above.

When defining a representation format for this internal structure, useful for different applications, interoperability problem arises. There are two kinds of representation markups needed: formal - assigning semantics to collections of formal expressions within a mathematical item in order to be able to apply automated reasoning to them, and conceptual - structuring the content of an item due to presentational or pedagogical purposes. For example, the publishing application is concerned with defining nice templates for better presentation, proof assistant application would require some sophisticated formal representation bound to its logical foundation, Computer Algebra system would be satisfied with naive mathematics not digging much into foundation systems, and, finally, a (good) math-education system would require all of these and some additional pedagogical annotations (e.g. mark a part of the text as **important** or link it to another content item etc.)

Therefore, we propose to split the micro-structure of mathematical items into two parts – formal and conceptual. Conceptual part of the micro-structure can consist of informal text, formulas encoded using content level markup, tables, diagrams, other multimedia elements. The conceptual part of the micro-structure should support structuring needs of all applications, in particular, enabling a transformation to suitable presentation formats.

The formal part of the micro-structure should allow for representing structures in different formal foundations. It can consist of collections of formal expressions organized in a particular fashion depending on the logical foundation in use. It is a research question whether designing such a uniform micro-structure is possible, if we do not deal with the concrete system. Even when having systems sharing the same logical foundation, it is sometimes difficult to separate the representation of the content from the syntax of one system and transport it to another.

If the common application-independent representation of a formal part of the micro-structure is not possible, multiple representations should be provided and modularized w.r.t. corresponding systems.

Micro structure of proofs, formulas and their types are the subject of a particular interest and will be discussed separately.

Micro-Structure of Expressions, Formulas, and Types The micro-structure of mathematical expressions is a way of organizing operators, predicates, quantifiers and variables of a mathematical theory in terms and formulas. The basic architecture for building mathematical expressions is provided by such representation formats as MathML and OpenMath (see §3.3).

Even the mere categorization between expressions, formulas, and types can sometimes be completely blurred in the notations of a logical system. This is typical, for instance, for most type-systems relying on the Curry-Howard isomorphism,which have a unique category of "terms" covering all kinds of expressions and their types. In the case of the Calculus of Inductive Constructions, a "formula" may roughly be identified as an object with a particular sort called "Prop", and an expression as an object of sort "Set" or "Type" and symmetrically, a proof as an object of sort "Prop" and an expression as an object of sort "Set" or "Type". This classification is not syntactical (and thus can not be captured by a DTD) and requires typing rules. Similarly, the main logical operator of CIC, the so-called "inductive type", is a delicate primitive construction whose semantics is typical for this calculus and cannot be trivially axiomatized in other systems. All foundational systems (think, for instance of Category Theory) have similar specialized constructions which make their beauty and provide their distinctive features and there is really no point to avoid this richness. However, parts of the encoding that belong to conceptually different layers but are combined for efficiency or elegance have to be separated in a common markup.

The representation of types is a delicate subject as well. OpenMath proposes to 'attribute' the mathematical objects with type information. That means, types can be attached to mathematical expressions or their parts. This mechanism allows for different formal type systems. OpenMath defines two candidate type systems - ECC (based on the Extended Calculus of Constructions) and STS (so called "Small Type System").[2]

`OMDoc` offers its own markup for representation of abstract data types as a shorthand for sets of inductively defined objects and recursive functions on them (see [11]).

Micro-Structure of Proofs Proofs are peculiar and one of the most cumbersome and interesting parts of the structured description of mathematics. The analysis of "real" proofs found in real mathematical document is far too complex to be dealt with in the limited scope of MoWGLI. So, we shall mostly focus our attention on formal proofs (see e.g. [16, 20, 19, 13, 8]) and especially on how proofs are typically dealt with by tools for formal reasoning.

Over the last thirty years, semi-automatic proof assistants have an interactive mode for proving theorems in which commands (tactics) are called that produces new, simpler subgoals to be proved. Usually, the result of an interactive proof is a *script* file in which the commands produced by the user are recorded. Usually, it is easy to check that this file is consistent simply by re-running it with the proof assistant. A script file is system-dependent since the command or tactics are often peculiar to the given proof-assistant. Moreover, the script is barely readable by a user, since it is a sequence of commands issued with respect to a given context, evolving with the script, that was available at proving time but is not explicitly saved in the script.

Some proof assistants save the information in a more primitive, "compiled" representation in the form of a "proof object", e.g., a *proof tree* or a typed (lambda) term of the underlying logical calculus. This representation depends only on the logical system in use, rather than

[2] see `http://nag.co.uk/projects/OpenMath/omstd/#omtype'`

on the application. This representation could be conveniently used as an interchange format between different applications sharing the same logic. However, the logical foundation has to be encoded in the markup because it might influence the micro-structure.

Rendering is more complex than in the case of expressions since it should at least support some kind of structuring, navigation, expanding subproofs on demand, etc. The actual implementation of these dynamic aspects requires pretty complex solutions on the Web, either based on dynamic-HTML (java-scripts), or based on specific constructs (as the MathML-presentation `maction` element).

It is not just the dynamic aspects of the presentation that require a rich micro-structure of proofs but also the information about dependency of steps, different levels of detail for proof steps, the logical context, and its hierarchical structure. The content markup for proof needs a rich internal structure and is not yet sufficiently developed in the existing standard markups which are primarily used for encoding formulas and expressions rather than for grouping collections of formulas according to some rules.

Although proof theory encodes formal proofs just as another category of mathematical expressions this seems to be inappropriate for XML-markup. In particular, representing a proof as an expression looks a bit artificial and mathematically questionable.

In type theory it happens that proofs may be nested inside expressions. For instance, it is customary to define the quotient operator for two integers as a function taking the proof that the denominator is different from 0 as an additional parameter[3] (these kind of problems is often a consequence of the deficiencies of type theory in managing *partial* functions). The above situation may cause problems in case the markup for expressions does not allow nesting of different markup.

Summarizing the above arguments we obtain the following requirements :

- support a human-readable presentation
- annotate the proof script with the context
- navigation in the proof structure e.g. inspection "on-demand" of specific subgoals
- support search for subgoals

2.2.2 Macro-Structure

A macro-structure is any organization of atomic mathematical items that uses relationships between them. There are generic macro-structures as well as relatively arbitrary groups of items. The generic macro-structure provides a general organization of a library that contains the information about the connections between items whereas a group can be a more or less arbitrary collection of mathematical items, manually assembled by an author or automatically assembled for presentational purposes. Those groups range from itemizations to TOCs.

In mathematical documents we can also find structures that are more complex than a single basic item, for instance,

- group of elements such as tables

[3]This kind of situation also gives annoying rendering problems: of course we would like to display the quotient of n and m as the fraction n/m, especially if nested within other expressions; at the same time it would be nice to warn the user that some information as been hidden there, without cluttering the expressions with hideous "handles".

- view
- table of contents
- theory
- module

Depending on the application scenario, different "views" of the same content can use specific dependencies for structuring a collection of mathematical items inherent to the content such as view that contains the definitions only or exercises only, but also a document without proofs.

Following the standards of ontological XML-languages, these structures have also to be identified by a URI. Moreover, in some applications such as educational documents the question arises on how to characterize the properties and usage of those *packages*. A first (preliminary) standardization is available with IMS content packaging. [4] This characterization is key for the reuse in other application contexts. For instance, a course can be characterized by its learning strategy, the appropriate school level, its overall technical requirements.

An element of a macro-structure has to be annotated by relevant information such as :

- technical metadata needed for presentational purposes
- administrative metadata containing information on authors, versions, copyrights, usage rights, etc.
- an ordered graph of elements containing pointers to items i.e. structure of a content package.

According to the general guidelines for XML-languages the following requirements should be met:

- modularization w.r.t. application-dependent criteria
- refinement of the underlying metadata structures w.r.t. application requirements
- extensibility of a packaging mechanism for integrating new resources

Theories

A theory is a special macro-structure that constitutes a logical context and semantics. A theory is a collections of *signature declarations*, *axioms*, and *theorems*. It corresponds to the standard notion of a formal theory that may contain additional items such as definition which are redundant for the formal theory but useful for a human reader. Theories define part of the mathematical ontology by grouping together the concepts that belong to the same mathematical theory and assigning a logical context to them.

Some further sub-structuring of this level could be sensible. One could refine the structure of a theory by introducing so-called modules that represent the maximal sub-theories of a theory. For instance, for each module one introduces some minimal amount of new signature elements and considers the maximal sub-theory of a given theory that does not extend the

[4] see `http://www.imsproject.org/content/packaging/` for the specification

signature. This would make sense, for instance, for the maintainability of a library of a proof assistant : we can access or use a result without requiring the whole theory inside at the same time, but only a minimal needed part of it.

Different theories can be connected by the dependencies of signature elements. The graph of dependencies formed by the hierarchies of theories is called **development graph** in [10]. The `OMDoc` encoding has attempted to implement this paradigm.

2.3 Metadata

To enable and facilitate functionalities such as searching and indexing, additional information has to be associated with documents and their units. It describes properties of the document and its units and relationships among entities. The Web terminology for this information is *metadata*, and typically it covers descriptive information of several kinds. Some metadata are manually provided by author, editor, and so on, while others may be automatically generated from the document itself.

The metadata can annotate every layer of a document structure – atomic pieces of knowledge as well as the whole document and groups of documents.

Let us consider several definitions for metadata provided by different Web-communities and summarize which characteristics of the document are to be described.

- W3C[5]: *Metadata is machine understandable information for the web.*
- FGDC[6]: *Metadata or "data about data" describe the content, quality, condition, and other characteristics of data.*
- UKOLN[7]: *Metadata can be defined literally as "data about data," but the term is normally understood to mean structured data about digital (and non-digital) resources that can be used to help support a wide range of operations. These might include, for example, resource description and discovery, the management of information resources (including rights management) and their long-term preservation.*
- IFLA[8]: *Metadata is data about data. The term refers to any data used to aid the identification, description and location of networked electronic resources.*

Consider the general requirements that a metadata specification should satisfy in order to allow for exchange between different environments on the Web. According to [7], metadata should satisfy the following principles: modularity, extensibility, refinement, and multilingualism (internationalization).

Modularity enables a clear distinction of metadata coming from different resources and provides a way to use existing standards instead of redefining them in the concrete application. The use of name spaces plays an essential role here.

The need of extensibility and refinement is clear: every particular application should be able to extend the metadata set according to its needs as well as refine the already defined structures (e.g. sorting w.r.t. some additional properties or defining particular schemas for value sets).

[5] `http://www.w3.org/Metadata/`
[6] `http://www.fgdc.gov/metadata/metadata.html`
[7] `http://www.ukoln.ac.uk/metadata/`
[8] `http://www.fgdc.gov/metadata/metadata.html`

Some metadata elements in multi-cultural environments can be interpreted differently depending on the country. For example, the date formats differ in North America and Europe, so that in order to store the date information correctly one might consider defining separate fields for day, month, and year rather than representing the full date as a string, where the order of tokens plays a role. Another example is the differences of education systems, that makes it difficult to annotate some pedagogical levels of learning material.

3 XML-Markup

In this section we consider some existing XML markups for both metadata and content of mathematical documents.

Aside of describing the markup the mechanism of specifying the structure of the document is explained. It is to be noted that the structure specification is a central characteristic of such a markup. Such a structure specification is sometimes called a schema, a document type definition, or an ontology.

3.1 Existing Encoding Standards for XML Markup

An XML-document is a combination of elements (constructed using tags) and the attributes of these elements. An XML-language is defined using Document Type Definition (DTD) where some restrictions on the content of elements and value of attributes are put. This restrictions can be validated automatically w.r.t. the DTD. The DTDs themselves are represented in a different language and have restricted syntax that reduces their expressibility. The possibility to restrict the value sets and types of the attribute according to the XML specifications is helpful for implementing basic metadata element sets with fixed value sets. However, the DTDs can represent only extremely restricted set of data types. Data types can be assigned to attributes of elements either in the form of sequences of particular values or as a number of default string formats. When defining an XML-language using a DTD, the exact number of occurrences of elements can not be specified. Another fact, unacceptable for a modularized XML-language is that DTDs have no support for name spaces which are essential for extensibility.

As a solution for these and many other problems of DTDs (see [21]), so-called XML-*schemas* are suggested. A schema is represented in a particular XML language that has itself a schema or a DTD. Schemas can much more flexibly define data structures and provide a stronger control over the structure of the documents. Among others, schemas provide rich support of data types, possibility to define own complex data types and reuse them in a generic way, inheritance of types, control over exact number of child elements (or attributes), support for name spaces, etc. Another limitation of DTDs is that the order of children elements of any defined element is strict (which is not always wanted) and any document using another order will be detected as not valid w.r.t. to the DTD. Schemas eliminate this restriction as well. All these features provide a good basis for writing clean and more machine-understandable XML-documents.

Finally, apart from standard XML-schemas, there exist some alternative and quite advanced schema languages for XML (e.g. RelaxNG[9]).

[9] `http://www.oasis-open.org/committees/relax-ng/`

3.1.1 **Resource Description Framework** RDF

The W3C Resource Description Framework (RDF) [18, 17] provides a general model for representing metadata as well as a syntax for encoding and exchanging these metadata over the Web. This standard approach is domain-neutral, and it does not make any assumption about an application domain. It supports interoperability of independently developed Web-servers and clients, more generally, between applications that exchange machine-understandable information on the Web. Documents described by RDF-metadata can potentially be indexed by search engines in a more effective way.

The basic construct of RDF is a URI, introduced in the section §2.1. RDF introduces the notion of a resource that is anything that has a URI (Uniform Resource Identifier) and way of describing a resource using so-called statements that are triples (*resource*, *property*, *value*) also known as 'subject', 'predicate' and 'object' of a statement. Schematically, one could imagine an RDF document as a labeled directed graph, consisting of nodes and arcs. The nodes of the graph are always resources or the values of the properties assigned to this resources and the arcs represent properties themselves. By definition, a node can be represented by a URI, as a blank node or a string (so-called "literal") and an arc is always labeled by URIs.

There is a straightforward method for expressing the statements in XML that is an intermediate format for interchanging between RDF applications. Namely, the nodes and arcs of the RDF-graph are turned into XML-elements, attributes, element content and attribute values. The URI labels for properties and object nodes are written in XML using name spaces.

The basic syntax of RDF is defined by the RDF-schema. Using this schema one can create documents or define his own RDF languages by specifying a new schema using syntax of RDF.

As in case of XML-schema, RDF-schema provides a mechanism for describing constraints on its elements. The main difference to XML is that RDF focuses on the communication of its *classes* and *properties* using nested elements to simulate the ordered graph of an RDF statement, shading the difference of nature of elements and attributes by freely converting attributes to elements and back, when for XML this difference is of a fundamental character. This means, RDF can RDF class is similar to the class in object-oriented programming. It can have sub/super-classes, instances, and properties. RDF-schema provides a mechanism for specifying constraints on the use of properties and classes in RDF-documents. These constraints are specified by providing domains and ranges of the properties that are the instances of particular classes.

Compared to XML-schemas, RDF-schema is weaker in constraining the XML-structure of a document, and the semantics of the data types defined as abstract classes in RDF depend on the application. RDF does not specify whether or how an application must process the constraint information, so that different applications might use these constraints in different ways.

3.1.2 DAML+OIL

DAML+OIL is a is an RDF-based semantic markup language for encoding Web-resources. It extends the basic RDF-schema with modeling primitives of Description Logic. DAML+OIL provides a semantic interpretation for the parts of an RDF graph that instantiate the DAML+OIL schema.

DAML+OIL originated from DAPRA Agent Markup Language (DAML)[10] expressing more sophisticated RDF class definitions than permitted by RDFS (RDF schema) and Ontology Inference Layer (OIL),[11] another effort using constructs from frame-based AI.

Additional DAML+OIL classes and properties defined in DAML+OIL schema can express far more sophisticated classifications and properties of resources then RDFS and, therefore, provide a powerful representation format for ontologies of machine processable knowledge.

For example, one can express boolean combinations of classes or specify that two classes are disjoint. The `Property` class of RDF has some important refinements. It can be enriched with some qualifiers like *inverseOf* providing relation of one property to another or *TransitiveRelation* providing meta-information on structure of relations. Another important facility that DAML+OIL provides is a property restriction that is a way to restrict classes to a set of resources satisfying particular properties. The cardinality or the values of these properties can be specified.

Another key to the expressivity of DAML+OIL is that apart from the RDF-mechanism for defining types it also allows to use XML-schema data types simply by including their URIs within the DAML+OIL ontology.

3.2 Existing Metadata Standards

The large variety of needs of different information communities makes it impossible to design a generic metadata element set "for everybody". Instead, it is rather desirable to provide an extensible and modularized framework for defining new metadata and for reusing the existing element sets from different sources. This is the goal of studies of the Semantic Web initiative at W3C that suggests to use RDF as a tool for defining interoperable metadata standards.

3.2.1 Dublin Core Metadata Element Set

Dublin Core Metadata Initiative[12] suggests a minimal metadata element set that contains basic metadata needed by most of the Web Applications. It contains administrative information about the document.

According to the version 1.1 of Dublin Core Metadata Element Set, there are 15 metadata elements: **Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights**. Each Dublin Core (DC) element is defined using a set of ten attributes from the ISO/IEC 11179 [ISO11179] standard for the description of data elements. Every metadata element has a `Name, Identifier, Version, Language, Definition` etc.. This meta-metadata specifies some technical properties of the metadata element itself and suggests the usage.

Note that DCMES is not sufficient for describing even it's own elements, i.e. some of the ten attributes used to describe a DC-element are not DC-elements themselves (such as Maximum Occurrence indicating any limit to repeatability of the data element). This, however, does not speak against the aim of the Dublin Core Metadata Initiative that is to provide a minimal set of elements used by an application in order to administrate the data.

Various applications implement DCMES in their data representation formats with a different level of refinement, attributing some of the elements by subproperties such as `role` of

[10] see `http://www.daml.org`

[11] see `http://www.ontoknowledge.org/oil/`

[12] see `http://www.dublincore.org`

Contributor with values “edt” for editor, “trl” for translator etc..

3.2.2 IEEE Learning Object Metadata and Extensions

IEEE Learning Object Metadata (LOM) together with IMS Learning Resource Metadata suggests a metadata element set needed to annotate learning objects. According to the LOM specification,[13] the purpose of this standard is to facilitate search, evaluation, acquisition, and use of learning objects by learners or teachers. Another goal is to facilitate the sharing and exchange of learning objects.

The elements of the Base Scheme of LOM are grouped in nine categories: `General`, `Lifecycle`, `Meta-metadata`, `Technical`, `Educational`, `Rights`, `Relation`, `Annotation` and `Classification`. Each of this categories groups data elements. Every element has an *explanation* that defines this element, *size* indicating the number of values, *order* of this values, *value space*, *data type*, and an illustrative *example*.

Dublin Core element set is represented as a subset of LOM, and some refinements of DC elements are provided. For instance, the element `relation` is refined into separate category, including important properties of relation such as *kind* specifying the nature of relation, *keywords*, or other classificational metadata.

Metadata describing a learning situation are defined. For example, one can specify the type of learning resource (exercise, table, self assessment etc.), intended role of the user (learner, teacher, author etc.), learning context (secondary education, university first cycle, etc.), semantic density and technical difficulty of the content.

Metadata for a measure of interactivity are provided. These are interactivity `type` that is defined according to criteria such as the balance of information flow between the learning object and the user, and naturally the interactivity level for measuring the intensivity of such communication.

Some extensions of IMS such as EML (Educational Modeling Language) [14] are designed to serve not only as static descriptions of properties of learning objects, but also try to annotate the dynamic process of learning and teaching.

3.2.3 IMS content packaging

Packages are collections of documents containing all the information on the organization of these documents as well as their content. Such collections are useful for knowledge exchange between applications sharing common document representation formats.

The IMS Global Learning Consortium has issued a standardization attempt for the organization and distribution of learning materials in the form of so-called content packages. The objective here is to define standardized structures needed in order to exchange content.

According to IMS Content Packaging information model, a *package* represents a unit of reusable content that may itself be a part of another package. It must contain all information needed in order to use it stand-alone.

The package consists of so-called *manifest* that describes the package itself and of the actual content files. The manifest contains metadata about the package, an organization element describing the organization of the content within manifest, resources element consisting of resources that are records of metadata, dependencies and identifiers of the physical resource

[13] see `http://ltsc.ieee.org/doc/wg12/LOM-WD3.html`

[14] `http://eml.ou.nl`

items (files), and possibly one or more (sub)manifests. For metadata any of the IMS metadata elements can be used.

3.3 Existing Approaches for Mathematics

In this section, we describe main features, motivation, and problems of MathML, OpenMath, and its extension `OMDoc`, as well as of the HELM knowledge representation.

3.3.1 MathML

Mathematical Markup Language (MathML)[15] is a general purpose XML-language for the representation of mathematical expressions on the Web. It consists of two parts: *presentation*-MathML aiming to provide a standardized rendering of the mathematical notations on the Web and *content*-MathML - the markup for content-level encoding of mathematical expressions.

The presentation markup of MathML has become a dominant standard for rendering mathematics on the Web and is natively supported by the Mozilla and Amaya browsers.

Content markup of MathML explicitly defines (as elements) a small set of most commonly used mathematical constructs. In addition, it provides a mechanism for associating meaning with new symbols and constructs that are not present in the core tag library. For this purpose the `semantics` element is introduced. It contains a mathematical expression for the presentation of new symbol together with the mapping to its semantics.

The mathematical expression is represented in content-MathML and enriched with additional semantics via providing one or more elements called `annotation` or `annotation-xml`. These elements establish a mapping to one or more external systems that are able to provide semantics to the construct. The value of the attribute *encoding* points to the external mathematical system (e.g. CAS) or representation format (e.g. OpenMath) and the body of this element contains as a string the definition of the symbol encoded in the syntax of the external system. In case, the external system uses an XML-representation format the `annotation-xml` element is used, otherwise - the element `annotation`.

The base set of content-MathML elements is meant to be adequate for simple coding of most of the formulas used in education from Kindergarten up to the first two years of college, that is up to A-Level or Baccalaureate level in Europe. Subject areas covered to some extent in MathML are: arithmetic, algebra, logic and relations, calculus and vector calculus, set theory, sequences and series, elementary classical functions, statistics, and linear algebra.

The meaning of the symbols of content-MathML is fully defined in its specification and the names of mathematical symbols that can be used in the MathML document are hardcoded in a DTD as opposed to OpenMath. For symbols that are not in the base-set, a `csymbol` element has to be used with a URI pointing to something that could be a description or a definition of the symbol. MathML has no architecture to share the definition or description of them, nor has it a formal way to define them.

There exists almost one-to-one mapping between the core content MathML tag libraries to the core Content Dictionaries of OpenMath considered next.

[15] `http://www.w3.org/TR/REC-MathML`

3.3.2 OpenMath

By the definition of the OpenMath community,[16] OpenMath is "an emerging standard for representing mathematical objects with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web".

The mathematical objects are mathematical symbols, expressions and formulas encoded semantically with the help of so called Content Dictionaries (CDs). The symbols to be used as atoms in a formula are introduced in CDs and semantics is assigned to them. A symbol can represent a function or a binding operator or a name for particular mathematical structure. The CDs are sorted thematically and sometimes more then one CD is introduced for a particular context. For instance, for symbols and operators of linear algebra several CDs are defined. Each of the CDs "linalg2" and "linalg3" defines symbol called `vector`. The difference between this two notions of vector is that the first one is used to describe row vectors and the second - column vectors.

However, one can not really speak about a unique semantics here. For instance, if some symbol has a unique representation in mathematical scripts (like the notion of an element of a set), then we assign a unique semantics to it and say that this symbol is always representing the particular object. But, for instance, the symbol for the plus operation that can represent addition of natural numbers as well as the addition of matrices is introduced in OpenMath only as a name for any associative and commutative operation on any structure which is at least a semi-group. The only difference between the symbol `times` for a commutative multiplication operation from "arith2" and symbol `plus` from "arith1" is that it is called times and it is "multiplicative".

Therefore, in order to provide unique and correct semantics for mathematical expressions more symbols have to be defined for particular structures. For example, different symbols should be defined for multiplication operation of integers and complex numbers instead of using `times` from "arith2". As opposed to content-MathML, such an extension is possible in OpenMath by simply introducing a new content dictionary which can be shared between authors or developers.

The conversion of an OpenMath content object to/from its internal representation in a software application is performed by an interface program called phrasebook. In turn, the development of phrasebooks should be directed by Content Dictionaries which provide human-readable (and in the future machine-understandable) descriptions of OpenMath objects. As opposed to HELM markup, `OMDoc` allows for more flexible organization of the items, e.g. it does not have such restrictions as "one item per file".

3.3.3 OMDoc

Typical extension functionalities offered in order to provide more structure to groups of formal expressions by existing languages such as MathML-content or OpenMath have been mostly conceived for adding new "operators" and are too weak for expressing new "grouping constructs".

`OMDoc` extends OpenMath markup which encodes only the micro-structure of mathematical expressions, providing the encoding for definition (formal or not) of mathematical symbols and other items building a mathematical document. Apart from items, it defines the general

[16] `http://www.openmath.org`

structure of mathematical documents and thus an ontology of mathematical knowledge [11]. Most of the mathematical items considered in 2.1 are defined in `OMDoc`.

The micro-structure of an `OMDoc` item consists of two parts that can be provided in parallel - formal expressions and informal explanations. In `OMDoc` the simplest items are all kinds of textual remarks, assertions, and examples. They consist of the building blocks of any item - CMP (Commented Mathematical Property) and FMP (Formal Mathematical Property). FMPs contain only formulas in OpenMath syntax. In CMP, formal expressions are enriched with informal text, diagrams, links etc..

There are `OMDoc` items whose internal structure is more complex. For example, the element `proof` can contain declarations of new symbols as well as some additional structure information for the stepwise representation of a proof (derive, hypothesis, conclude). Proofs are represented as sequences of connected steps that can recursively contain nested subproofs and proof objects.

The element `adt` (abstract data type) that represent short forms for groups of symbols and their definitions has an elaborate internal structure that allows for formal definition of abstract data types. This is relevant for formal software verification.

Except for a few examples, the micro-structure of `OMDoc` items is not expressive enough to support all the needs of many application, since it has to be general enough to serve as a core standard for many applications.

For example a more structured representation of a proof (for instance, a proof plan expanding with different level of detail) would be of interest fpr an educational application or a proof assistant, but it is questionable whether this kind of representation should be a part of core representation and force all applications to use it.

The macro-structure of `OMDoc` documents consists of several parts.

- relations between items
- a grouping element
- theories

`OMDoc` uses the semantical grouping structure `theory` instead of Content Dictionaries for creating ontologies of mathematical concepts. There new symbols are introduced and particular fixed semantics is assigned to them by providing a definition for every symbol. `OMDoc` also provides an inheritance mechanism between theories that allows for introducing hierarchies of mathematical concepts connected via theory imports and morphisms of signature specifying how the semantics of the symbols are mapped. The OpenMath CDs are translated into `OMDoc` theories using an automatic translation tool of M. Kohlhase.

`OMDoc` metadata are present in the macro-structure elements as well as in the mathematical items themselves. `OMDoc` employs Dublin Core Metadata Element Set with the number refinements like introducing attribute *role* for elements `Creator` and `Contributor` etc.. Apart from DC specification, there is an extradata field, where all metadata extensions made within a particular application can be introduced.

Some metadata important for mathematical knowledge manipulation are represented as the XML attributes of mathematical items. An example is the *for* attribute of a mathematical item that can point from a definition to a concept it defines, from the proof to a theorem it proves or from an elaborating text, example, or exercise to a corresponding concept.

Other technical metadata describing different mathematical properties of an item are introduced by attributes of the item. For example, a `definition` has an attribute *type* with

values 'simple', 'inductive', 'implicit', etc., `assertion` can be of a *type* 'theorem', 'lemma', 'corollary' etc.. The fact that these properties are encoded as attributes implanted in the `OMDoc` item is subject of debate. We suggest that all the properties of an item that do not have an impact on the micro-structure of the item should be put inside the metadata field and possibly separated from the content for the better management of the content. This way authors can specify different sets of metadata for the same content items that will allow for reusability in different contexts.

3.3.4 HELM Markup

The project HELM [17] aims at the study and the development of a technological infrastructure for creation and maintenance of a virtual, distributed, hypertextual library of formal mathematical knowledge. The main leit-motivs of the project are the accessibility of the mathematical repositories in an application independent format, as it can be conveniently provided by XML, and the extensive use of XML-technology for implementing generic functionalities such as rendering and searching.

A suitable extension of MathML-content is used as an intermediate content representation between the contextual and the presentational formats. In particular, HELM currently uses MathML-content for encoding both, expressions and proofs. For the level of mathematical items and the macro-structure HELM has its own markup. The markup is experimental and has never been advertised and not adequately documented.

The crucial point is the evolution from the old application-oriented management of information, to a new content-centric design [1].

HELM aims at improving the modularity of the applications and at decoupling all those functionalities which are largely independent from the specific framework used by the application. Schematically, the global architecture of HELM is described in Fig.1.

HELM was tested on the mathematical library of the `COQ` proof assistant. Some preliminary experiments have been also performed with the mathematical repository of the NuPRL system.

3.4 Additional Information Available in and Relevant for the Applications in MoWGLI

Depending on the application, many additional metadata might be useful in a mathematical document.

Historically, for mathematical publications classifications have been used to annotate articles.

Therefore, Mathematical Subject Classification has to be considered since many mathematical documents exist already online, and are annotated by classifiers. Compared to content markup the classifications are abstract and may provide an additional value. So, classification could be a part of the content markup. Note, however, that classification keywords annotate existing documents mostly for document-level purposes, including libraries, search, etc.

Currently, there are three classification systems in use: the Mathematics Subject Classification[18], the Dewey Decimal Classification[19], and the Universal Decimal Classification System[20].

[17] `http://www.http://www.cs.unibo.it/helm/`
[18] MSC 2000, `http://www.ams.org/msc/`
[19] DDC `http://www.oclc.org/dewey/`
[20] UDC, see e.g. `http://www.lib.demokritos.gr/udceng.htm`

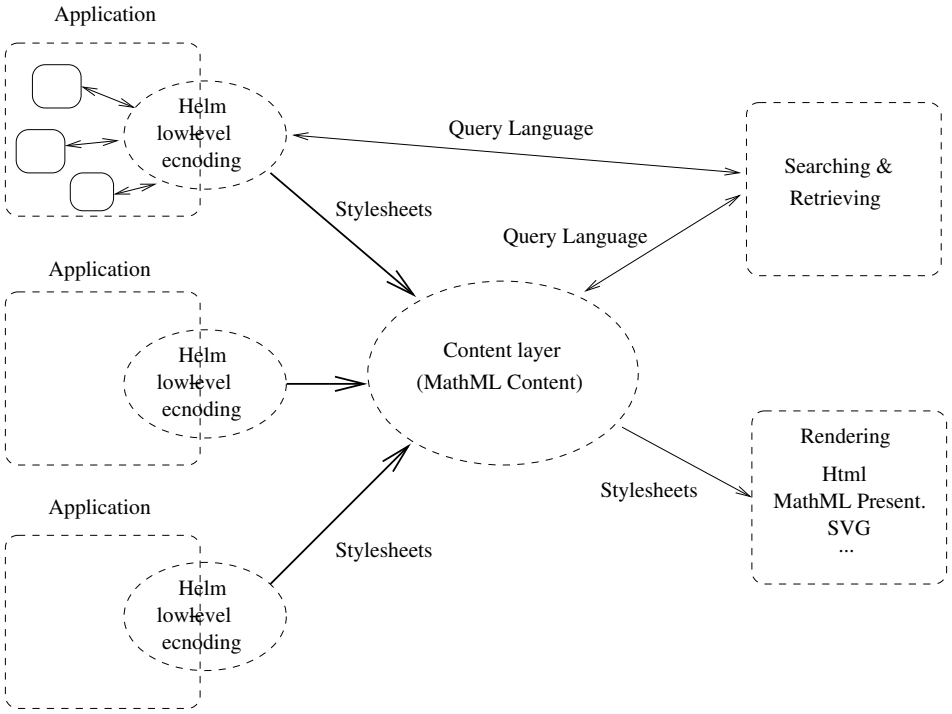

Figure 1: HELM architecture

The latter two are general schemas with mathematics as a part only.

The Dewey Decimal Classification (DDC) was conceived by Melvil Dewey in 1873 and first published in 1876. The DDC is published by Forest Press, a division of OCLC Online Computer Library Center. The DDC system is one the most widely used classification system in the world, it is largely known by librarians but hardly known to mathematicians. At the broadest level, the DDC is divided into ten main classes meant to cover all areas of knowledge. Each main class is further divided into ten divisions and each division into ten sections. Concerning a given area of knowledge such as mathematics, the final classification is not as fine-grained as that of MSC 2000.

The Universal Decimal Classification system (UDC) is a similar "universal" classification schema which was originally inspired, by DDC, and then independently evolved into an international standard. UDC is currently widely spread in Russia but less used than DDC in western countries.

The common standard in mathematics is MSC 2000 which currently is supervised by Jane Kister (editor-in-chief of Mathematical Reviews) and Bernd Wegner (editor-in-chief of Zentralblatt MATH). The MSC is used to categorize items covered by the two reviewing databases, Mathematical Reviews (MR) and Zentralblatt MATH (Zbl). The MSC is broken down into over 5,000 two-, three-, and five-digit classifications, each corresponding to a discipline of mathematics (e.g., 11 = number theory; 11B = sequences and sets; 11B05 = density, gaps, topology).

Certainly a problem of MSCS is that different reviewers may annotate a document with

different classification keywords.

It is to be expected that following the development of MoWGLI research, the organizations maintaining the classifications will gradually gain interest into standardizing symbol sets which will be then used by authors and will then be searched for in databases.

3.4.1 Mathematics Education

In a learning environment such as ActiveMath some additions for educational purposes are important. The metadata provide information for components of a learning environment such as *user model* and the *presentation engine*. In order to be able to reason about the usefulness of a particular piece of content, the piece has to be annotated with metadata describing the learning situation in which it should be used and with difficulty values.

Essential functions of metadata assigned to a content element are :

- they describe administrative/legal characteristics of an element
- they supply information needed to automatically choose the most suitable material in the current learning situation
- they supply information needed to update the user model when the user has worked on the material.

For the first requirement the Dublin Core metadata suffices.

For choosing material depending on the learning situation one has to introduce metadata modeling the learning situation and refine the structure of relations of the mathematical items w.r.t. the representation of learning situation. This includes the education and cultural background of a learning context, e.g. the domain of an example, year of study etc.

The representation of feedback on user's actions is still a subject of research.

3.4.2 Search for Mathematical Expressions

The requirements for effective search techniques for mathematical notions may be highly demanding. In particular, typical queries may require complex elaborations that cannot be trivially compiled into standard data base query languages. Typical examples are:

1. matching "equivalent" notions, such as say n^{-1} and $1/n$. Supporting this feature requires the implementation of some form of "reduction" eventually comprising the unfolding of definitions.
2. taking into account "isomorphic" shapes. When looking for a formula of the shape $A \wedge B$, we would like to match $B \wedge A$ as well.
3. supporting unification, as opposed to "pattern matching".

Complex queries are being pursued by the HELM Project with promising results. Currently, the metadata which have been considered is a list of identifiers at specific, key positions inside the logical items. However, matching "equivalent" notions might become too dependent on the logical foundation of the `COQ` system which is used to proof-check these equivalences. Moreover, it is not clear that such search algorithms are terminating in finite time.

The approach of MBase[12], developed in Saarbrücken, is to support a more general, application-independent query mechanism that uses pattern matching extended by matching

equivalent notions w.r.t. some elementary properties of well-known operators (e.g. associativity and commutativity of equality relation etc.). This mechanism is independent of the particular logical foundation, since the properties considered can be proof-checked by any formal system in use.

Because of the complexity of queries and the dimension of the data base it may be interesting to divide the search into two phases, where we first select a restricted number of "candidate" documents, and then interrogate this subset in a more precise way, to get the answer. The first "selection" step, could be based on metadata automatically generated form the source document. These metadata would essentially provide an approximate description of the content that is explicitly meant for fast searching and retrieving operations.

4 Research Methodology

According to the goals of MoWGLI, we shall determine a common core data model for the heterogeneous applications. For efficiency and readability reasons this includes a modularization of the information relevant to specific applications, see the W3C recommendations[21] It requires to determine which parts of the knowledge representation can be standardized among the partners and in compliance with the existing standards. In case of disagreement the smallest common representation will consist of everything that is needed in a common architecture and common APIs.

In what follows, we list several research problems which have to be investigated in order to design a common data model for the heterogeneous mathematical applications, at least the applications contributing to MoWGLI. A goal of the research is to provide a markup that will be common to all applications and extend existing knowledge representation standards for the Web.

- how to represent logical foundations: natural deduction, sequent calculus, resolution calculus, calculus of inductive constructions etc.
- which macro-structures: view, theory, module, import
- interactive documents (for education, living review..)
- standard for system/service actions, feedback representation
- versions or life cycle information, other technical metadata
- variants of the same item (e.g. with different verbosity, language, etc.)
- sections/chapters and which level content packages are needed
- user-provided keywords (good for bridge to older documents, as abstract markup,)
- hierarchical structures
- how to represent proof
- modules to be stored in separate data bases

[21] `http://www.w3C.org`

- what cannot be standardized?

It is rather unclear whether a standardization of the logical context markup makes sense because it might be too dependent on foundational and methodological issues. If we cannot hope to have a *logic-independent* encoding of the information, we might define a *system-independent* markup in an application-independent format. This encoding would greatly simplify the exploitation of the libraries of formal documents of the application by any interested party for any intended use.

For the purpose of an automatic elaboration as well as for the maintenance of a library it may be convenient to store each mathematical item in a distinguished XML-file. The representation, especially of theorems and proofs may be huge. Parsing complex documents with the current XML-technology is still problematic. Although the solution of storing each mathematical item in a stand-alone XML-file may look natural, it is not customary for proof assistant applications, where information is usually saved in bigger clusters (theories, or sections). The mentioned organization has major drawbacks for the maintainability of the library:

1. a result cannot be accessed or used without requiring the whole theory inside which it is defined. Since these theories are frequently very big, authors often redefine the required results locally which leads to a useless and confusing duplication of information.

2. extending a theory or a section requires a recompilation. As a consequence, the "library" which is the result of the contributions of many different authors who do not have any access to contributions of other authors, tends to be "flat" and badly structured. This hinders its development as a joint and cooperative effort.

The high verbosity of XML-documents and the difference between metadata for different applications could give raise to the need to store metadata information separately from the content and to connect it to the content item via reference to its URI.

For the purposes of manual authoring of the documents, the encoding should, however, allow flexibility in these issues.

References

[1] Asperti, A., Padovani, L., Sacerdoti Coen, C., Schena, I., "Content-centric Logical Environments". Short presentation at LICS'2000, June 26-28, 2000, Santa Barbara, California.

[2] Asperti, A., Padovani, L., Sacerdoti Coen, C., Schena, I., "HELM and the semantic Math-Web". Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLS 2001), 3-6 September 2001, Edinburgh, Scotland.

[3] Berners-Lee, T., "Universal Resource Identifiers in WWW", RFC 1630, CERN, June 1994.

[4] Bray, T., "What is RDF", O'REILY xml.com, January, 2001.
`http://www.xml.com/pub/a/2001/01/24/rdf.html`

[5] Coscoy, Y., Kahn, G., Thery, L., "Extracting Text from Proofs", Technical Report RR-2459, INRIA Sophia Antipolis.

[6] Coscoy, Y. "Explication textuelle de preuves pour le Calcul des Constructions Inductives", Phd. Thesis, Université de Nice-Sophia Antipolis, 2000.

[7] Duval, E., Hodgins, W., Sutton, S., Weibel, S.L., "Metadata Principles and Practicalities", D-Lib Magazine, April 2002, Volume 8, Number 4, ISSN 1082-9873

[8] Girard, J.Y. "Proof Theory and Logical Complexity". Bibliopolis, Napoli, Italy, 1988. ISBN 88-7088-123-7.

[9] Howard, W.A. "The formulae-as-types notion of construction", in "To H.B.Curry: Essays on Combinatory Logic", J.P.Hindley and J.R.Seldin editors, Academic Press, London, pp.479-490, 1980.

[10] Hutter, D., "Reasoning about theories", Deutsches Forschungszentrum für Künstliche Intellligenz (DFKI), 1999

[11] Kohlhase, M., "OMDoc: Towards an OpenMath Representation of Mathematical Documents", Seki Report,FR Informatik, Universität des Saarlandes, 2000.

[12] Kohlhase, M., Franke, A., "MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems", Journal of Symbolic Computation 23:4 (2001), pp. 365 - 402.

[13] Martin-Löf, P. "Intuitionistic Type Theory", Bibliopolis, Napoli, 1984

[14] Mathematical Markup Language (MathML) 2.0 W3C Recommendation, 21 February 2001.
http://www.w3.org/TR/MathML2/.

[15] Melis, E., Büdenbender, J., Andres, E., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M. and Ullrich, C., "ActiveMath: A Generic and Adaptive Web-Based Learning Environment", Artificial Intelligence and Education, Volume 12, Number 4, 2001.

[16] Prawitz, D. "Natural Deduction", Almqvist & Wiksell, Stockholm, 1965.

[17] Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000.
http://www.w3.org/TR/rdf-schema/

[18] Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999.
http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[19] Schütte, K. "Proof Theory", Springer-Verlag, Berlin, 1977.

[20] Takeuti, G. "Proof Theory", North Holland Publishing Company, Amsterdam, 1975.

[21] Walsh, N., "Understanding XML Schemas"', O'REILY, xml.com, 2000,
http://www.xml.com/pub/a/1999/07/schemas/index.html