Mathematical Information Retrieval Powered by Theory Management

by Immanuel Normann

International University Bremen
Email: i.normann@iu-bremen.de

November 23, 2004

1 Introduction [tmp macros:]

Visions once enthusiastically proclaimed by an intellectual genius, then seemingly buried forever, and suddenly after hundreds of years brought back to life are fascinating. G.W. Leibniz was a far-sighted mind with such a vision. His vision consisted in the invention of a formal language, the *lingua characteristica* capable to express any assertion one can think of, together with formal argumentation rules on it the *calculus ratiocinator* with which the truth value of such assertions could be determined once and forever. Any argumentation about the truth of any assertion would be replaced by his motto “Calculemus - Let’s simply calculate it”.

1.1 The Leibniz Project

Though Leibniz’ vision was not restricted to mathematics the *Leibniz project* (LP) restricts itself to mathematics with the following theoretical and practical goal:

1. Invention of a *universal formal system* comprising
   - a lingua characteristica where any mathematical assertion can be expressed, and
   - a calculus ratiocinator on it.
2. The *actual formalization* of the most important areas of mathematics; i.e.
   - translating all important mathematical statement into this language, and
   - transforming each proof of any important mathematical assertion into a sequence of applied inference steps of the calculus.

In 1879 about 200 years after Leibniz’ vision G.Frege presented his most influential “Begriffsschrift” [9] which was the creation of what is nowadays called *first order logic* (FOL). This can be regarded as a partial accomplishment of the theoretical part of LP, since FOL is sufficiently powerful to express and prove most of classical mathematics.

In 1931 K.Gödel [10] proved that there is no complete calculus for any complete language. Any sufficiently powerful language capable to express all mathematics cannot have a calculus with which one could deduce the truth value of all its mathematical statements. Let us call the maximal set of assertions formulated in a complete language such that a given calculus can determine the truth value of its assertions the *domain of the calculus*. Gödel proved that the initial goal of LP, i.e. to find a calculus whose domain equals a complete language, must be dropped. Nevertheless, a weakened goal to find a calculus whose domain is as extensive as needed can be still pursued. In particular, whenever an assertion of interest is outside the domain of the calculus then one can extend the calculus (or invent a new calculus) suitably. Theoretically, this process never ends. Practically, we already reached an end, because powerful calculi (higher order logics HOL)[1] were invented that cover all known assertions of current mathematics. Furthermore, many logicians are convinced that there don’t exist ”relevant” mathematical assertions outside the domain of HOL or alternatively axiomatic set theories based on

1. Actually HOL denotes only a certain set of such powerful calculi, but since there is no common handy name for the set of all these calculi let us call them HOL for convenience.

FOL. So the creation of HOL can be regarded as a step in the direction of LP. Russell and Withehead verified (1910-1913) with their enormous opus “Principia Mathematica” (PM)[17] that classical mathematics can be indeed formalized into HOL, that is a partial accomplishment of the practical part of LP.

Unfortunately, this result is shadowed by a pragmatic failure apparent from the PM: Any formalization of mathematics, whether in FOL or in HOL, is - due to the verbosity of formal languages - extremely tedious and unintuitive for the understanding of nontrivial mathematical content. Up to this point the LP put the focus on formal rigor, but for the sake of actual deployment it had to consider the usability of formal languages. Until today there is no formal language that would satisfy the usability needs of many working mathematicians.

As a consequence the great majority of mathematicians isn’t eager to strictly formalize their content. Instead they still content themselves with different degrees of rigor. Let us denote a rigorous mathematical language *quasi-formal* if - in the opinion of the majority of mathematicians - it could be translated to a strictly formal language. Such a language was established by Bourbaki [3], an association of young mathematicians founded in France around 1930. They started the ambitious project of writing down all important areas of mathematics (providing proofs for every theorem) in their quasi-formal language and they accomplished an impressingly large part of it. Again this can be regarded as a great success in the sense of LP: Even though mathematics wasn’t strictly formalized, its quasi-formal rigor allows an immediate and unambiguous strict-formal rigor on demand.

With the invention of the computer the LP got yet another impact. Church’s famous thesis [?] states that a computer can compute anything that is computable. Hence, if proving is just a matter of computing a truth value of a formal expression, then a computer should be able to prove. Creation and verification of massively verbose proofs that are infeasible for humans because of their slowness and unreliability in manual processing of much data, could be calculated (as Leibniz would say) by a computer with an enormous speed and precision. In 1993 an association of researchers in the field of automated theorem proving founded the QED project convinced that - thanks to the computer - the Bourbaki project could be accomplished with strict-formal rigor. This target had to be pursued [?] in order to dispel any doubt about the correctness and completeness of proofs. Quasi-formal proofs never met this strong demand. That it can be effectively fulfilled is proclaimed in the first paragraph of QED Manifest[15] published in 1994:

> ”*QED is the very tentative title of a project to build a computer system that effectively represents all important mathematical knowledge and techniques. The QED system will conform to the highest standards of mathematical rigor, including the use of strict formality in the internal representation of knowledge and the use of mechanical methods to check proofs of the correctness of all entries in the system.*”

At that time some projects had succeeded in the development of ever growing large libraries[2] of theorems with strict formal proofs. Thus it seems to be just a question of time until we will finally have the most important parts of current mathematics formalized.

1.2 Mathematical Information Retrieval on Formal Digital Libraries

Unfortunately, almost nobody outside the community of automated theorem proving benefits from these libraries. The reason for this deficiency is these LP followers’ one-sided attitude to draw all attention to formal correctness. In other words, they prove assertions and subsequently they forget about them which is clearly unsatisfying. For the majority of mathematicians and in particular other researchers involved in mathematics formal correctness of proofs is only a very limited value on its own. This majority would only benefit from the content of these libraries if they effectively obtained more information than the mere fact that all the content is correct. Mathematicians not only could but also we claim would consult these libraries if they had an appropriate information retrieval tool.

2. One of the pioneer system was Mizar developed by A. Trybulesc since about 1973.

Information is a set of facts which helps an interpreter accomplishing a certain task. The dependency should be emphasized: Information is not just pure data, but data depending on an interpreter and a task. We call a system an *information service* if it is able to supply the requested information which is formulated in a defined *information request language*. Finally, an *information retrieval system* is an information service that retrieves information by searching or inferring data from a data base. The effectiveness of an information retrieval system depends heavily on the expressiveness of the request language as well as on the data stored in the underlying database. A *mathematical information retrieval system* (MIRS) is dedicated to interpreters and tasks related to mathematics - i.e. *mathematical information retrieval* (MIR) . Its request language must enable a mathematician to express mathematical information requests.

For instance a math student's main task might be trying to understand a theorem. When he comes across a symbol he hasn't seen before, he looks up its definition. As the definition is too abstract to his taste he looks for simple properties of the defined symbol, etc. A request language should therefore support requests such as: `show-definition-of-symbol`, `list-basic-properties-of-object`, `show-assumptions-used-to-proof-assertion`, etc.

1.3 Mathematical Theories

Typically the meaning of an object is not simply determined by a symbol or a formula, but its meaning is tied to a certain context. The request "What is the definition of the symbol $\supset$?" posed to a logician would yield perhaps the answer "$\supset$ means logical implication" whereas a set theorist would answer "$\supset$ denotes the subset relation". Both answers are correct in their specific context. If the context isn't obvious then one would modify the request to "What is the definition of the symbol $\supset$ in logic?". Concerning formulae there are typical mathematical categories of meanings: *definition, axiom, example, lemma, theorem, proof,* etc. A formula is never per se an axiom, theorem, or what ever, but always only with respect to a particular context. For instance the formula "$\forall a, b, c.\ a + (b + c) = (a + b) + c$" is a lemma in number theory whereas in group theory it is an axiom. The words "number theory", "group theory", "set theory", etc. assign mathematical contexts to names. Let us define *theory* as a collection of formulae each tied to certain mathematical category of meaning. Definitions and axioms constitute a theory while assertions (lemma, theorems, etc.) hold in a theory if they can be deduced from these constitutive formulae. Theories are not just flat objects, but they are nested via abstraction or specialization. For instance we find that all axioms of group theory are lemmata in number theory. Hence, all theorems of group theory are also theorems in number theory.

Relating and connecting theories to each other is an important mathematical task and often triggers new research and results. In order to support a mathematician in this respect the request language of a MIRS should support requests about theories. There are many interesting information requests one can think of: "Show me all theories which are built on theory $\mathcal{T}$", "Show me all theories sharing a given set of formulae as axioms", "Show me a theory whose axioms are theorems in other theories", etc.

1.4 Management of Mathematical Theories

From a extreme formalist's point of view a theory is an arbitrary collection of formulae, where some of them are regarded as assumptions and all others as assertions. The only constraint that assumptions should be consistent and assertions must be proven from these assumptions, does determine neither the collection of assumptions nor the selection of theorems. Practically theories are results of mathematical discourse with various intentions. They are intended to build a meaningful unit and thereby subject to human goals. As these goals change over time theories must be reorganized too. Distributing or redistributing content over theories and defining or proving logical implications between theories is called *mathematical theory management* (MTM). The purpose of mathematical theory management is at first to adapt the organization of theories to the user's tasks. The *information value of a theory* is the degree of the adaption. A *theory management system* (MTMS) is a system to organize or reorganize theories preserving logical consistency and its purpose is to increase their information value.

Mathematical theory management can be used to increase the information value of formal digital libraries. Since a significant part of these libraries share common content, a merge of overlapping content would accomplish many potential user goals. For instance a most prominent goal would be to collect all theorems deducible from a set of axioms in a single theory containing these axioms instead of distributing them over many theories sharing the same axioms.

In order to find overlapping content in a large formal library mathematical information retrieval will be very helpful. Successful theory management in turn improves mathematical information retrieval too. Ideally the process of increasing the theory information value increases the understanding of theories itself.

The research proposed in this paper is the specification and implementation of a mathematical information retrieval and theory management system (MIRTMS).

1.5 Structure of this document

The rest of this proposal is organized as follows. In section 2 we first describe systems related to the MIRTMS. Section 3 identifies the deficiencies of those systems with respect to those that should be covered by a MIRTMS. Section 4 concretises the idea of the MIRTMS and provides an elaborated fictional case study demonstrating the interaction with such a MIRTMS. Section 6 identifies the research tasks to be attacked for the realization of the MIRTMS. Section 8 sketches the work plan followed finally by summary for this research project.

2 Systems Related to MIR and MTM

After illustrating what I mean by mathematical theory management in practice we have to scan the landscape of math systems for systems related to a MIRTMS which serve the needs of these users to some extend.

2.1 ActiveMath

ActiveMath [14] is a web-based learning system that features high user adaptivity to students' learning goals, preferences, capabilities, and knowledge. Most web-based learning systems can be considered as HTML management systems, i.e. content is stored presentation oriented and assembled by hand as online courses. ActiveMath's approach is different: Authors supply semantically instead of merely presentational marked up content and in addition they supply content independent pedagogical rules. Moreover ActiveMath gathers information about the user during a session and stores it for subsequent sessions. These data enable the system to create user adapted online courses.

Semantic content, pedagogical rules, and knowledge about the user are used by the system to present a preferably user adapted online course.

A closer look at ActiveMath's architecture reveals how different modules act on different tasks. Essentially there are four modules: the content database, a user model, a course generator, and a session manager.

- content database: stores semantic content in an extended OMDoc format. The extension concerns some pedagogical attributes such as: `depends-on`, `definition-for`, `example-for`, `motivation-for`, etc.
- user model: gathers and stores information about the user's learning goals, preferences, capabilities, and knowledge. This information is fed by the user either explicitly by filling out knowledge questionnaires or implicitly by analysis of the user's interaction with the system. This analysis comprises for instance the evaluation of interactive exercise sessions, or the user's reading behaviour recorded via a simple eye tracker.

- course generator: retrieves information about the user from the user model and subsequently takes appropriate content from the database to compose an course in accordance with the pedagogical rules.
- session manager: takes the OMDoc course material from the course generator and generates presentational output, e.g. HTML, PDF, SVG, LaTeX, etc.

ActiveMath can be considered as a MIRS in the following sense: A user wants to study a certain field of mathematics and therefore requests the system to supply the appropriate material. This material can be considered as information, since it is data that helps the user accomplishing a learning task. Besides this request to the course generator ActiveMath offers also dictionary. All symbols occurring on the HTML pages generated by the course generator are hyperlinked to an entry in the dictionary. A dictionary entry is a meaningful unit such as a definition, a theorem, a proof, an example, an exercise, a motivation, etc. Simple string matching queries in the dictionary is also possible. The result list of items where this string occurs is sorted by these categories of entries.

2.2 Mizar

The Mizar Language [5][m-intro][mizar-outline] is a formal language derived from mathematical vernacular, i.e. the quasi formal language typically used in nowadays math textbooks. The principal idea of its author A. Trybulec was to design a language readable for mathematicians and, simultaneously, sufficiently rigorous to enable processing and verifying by computer software, in particular mechanical verification of correctness. Logically Mizar is based on a natural deduction calculus with a notion of an obvious inference instead of using some fixed set of inference rules.

As a long term project (starting at 1973) Mizar aims to formalize the whole mathematics into the Mizar Mathematical Library (MML)[mml]. A text fragment written according to the Mizar Syntax is called a Mizar Article. These articles can be published afterwards in the Journal of Formalised Mathematics. Thus (as far as I know) MML is the only formalised mathematical content actually published in a journal that is dedicated to most recent mathematical research. In contrast to other formalised content from projects described below MML is mathematically grounded on just one fundamental theory namely Tarski-Grothendieck set theory.[tarski]

Mizar environment framework. Mizar articles are written as text files and consist of two parts, each processed by a separate program: the environment declaration is processed by the so called accommodator and the so called "text proper" by the verifier.

The accommodator reads directives of the environment declaration and imports from the data base the requested information into working files. There are three types of information units in MML: article, vocabulary, and built-in feature. Directives request information from these units by just listing their names. The vocabulary directive requests from vocabularies, the requirements directive from built-in features, and all the others request from articles. Let's give a short description of the sort of information each directive requests:

- `vocabulary`: lexical declaration to be use in the article, i.e. a symbol as identifier, its category (function, predicate, type, etc.), and its precedence (as numerical value)
- `constructor`: the definition patterns of symbols in use. This includes arity, type, and formatting information.
- `notation`: the 'meaning' of symbols.
- `clusters`: correctness conditions about relationships among adjectives (associative, commutative, etc.), function symbols, and types.
- `definitions`: definientia that can be used in proving by definitional expansion without mentioning the definition's name.
- `theorems`: enables referring to theorems and definitional theorems from the listed articles.

- `schemes`: the same for theorems with second order free variables.
- `requirements`: gives access to built-in features like basic arithmetic.

After the accommodator has built the local environment of the article the verifier checks the correctness of the text proper without any direct connection to the MML but the local environment.

Information retrieval inside MML. All content is accessible over the internet and each content symbol is hyperlinked to a page that contains the symbol's introduction. A static page presenting a graph of depending structures in the MML enables a top down navigation. Besides mere navigation through hyperlinked articles, Mizar provides its own query language [2] that allows querying all kind of items listed above. Most recently G. Bancerek and J. Urban [11] developed a tool using this query service for semantic browsing of the MML inside Emacs.

2.3 Formal Digital Library Navigator for NuPRL and PVS

A navigator for browsing and manipulating formal content has been designed by the group at the department of computer science at Cornell-University.

The *formal digital library* (FDL)[13] is a mathematical and logical database emerged from the NuPRL [?] and PVS [?] digital libraries. All library contents are represented by common basic data structure called objects. There are objects for definition, theorem with references to proof objects, inference rule, tactic, algorithm, comment and article, presentation information, and objects to organise other objects in a theory and directory. All library objects are kept in a persistent library which operates like a data base with an immediate commit on every theory modification and a transaction mechanism including the undo operation. The declared features are:

- browse and search object names through the library
- create, delete, and rename library objects of various
- arrange objects in folders and theories
- edit library objects by invoking a term editor or proof editor
- check the validity of objects and theories
- export and import theories
- print library objects and theories to text and LaTeX files.

The user interacts with the library through an editor process that communicates with the library application interface. It generates the user's work space and communicates the modification to a transaction manager, which in turn performs the actual modifications to the library. This make it possible to restrict access to certain parts of the library, to hide the abstract representation of data, and to present to the user a consistent view of the library.

Every object has a unique name and belongs to exactly one theory, although they may be linked to from other theories or by a different name within the same theory. Theories can be nested like Unix directories and may depend on each other which then forms a partial order. Within each theory, objects are ordered linearly. Each object may be associated with a reference environment which is involved in theory management. Operations on objects are analogous to operations on files in a filesystem: Objects can be saved, moved, removed, renamed, copied, and symbolically linked within the theory hierarchy. In order to experiment with alternate versions of a definition or theorem they can be activated or deactivated such that they become visible or invisible for tactics during a proof session.

Theory management in FDL supports a dependency checking mechanism for theories to avoid circular chains of lemma references in its proofs. Each theory has its "reference environment" which specifies a list of: reference environments to inherit from, abstract definitions to add, lemmas to add, and some arcane update code that updates values of reference variables. Subtheories are just subdirectories without any reference environment.

Commands to examine the reference environments are supplied, which shows the reference environment specification as an association list of reference variables an indices. Another command shows some indication of data that is bound to that variable/index pair.

To check consistency after a theory is modified the system provides a utility to replay the proofs. Hereby the user can control the depth of the theory check, i.e. how much of the subtheories or referenced theories should be taken into account.

Theories can be exported and imported to or from dump files. For importing and migrating data from external libraries into FDL's data repository additional utilities are provided including transforming objects and filtering by name, kind, and creation time.

2.4 Imps

Imps [6][7] is an Interactive Mathematical Proof System for a rigorous development of mathematics. The intention of Imps is to provide a system which serves students as a general mathematics laboratory as well as a useful adjunct in mathematics research. Imps contains an extensible database for mathematical objects and an interactive theorem prover. Its user interface is an elaborated mode for XEmacs with Xdvi as an additional viewer for high quality formula presentation.

The invention of Imps' theory framework [?] is the initial step towards MTM which is in the focus of this section. The theorem prover which might be considered as the core of Imps won't be discussed here, because it contributes neither to MIR nor to MTM directly.

The mathematical database of Imps is a file based database [?], hence extending or updating means inserting, editing, or removing items representing mathematical objects in so called theory files. All top level items are declared in a fixed set of so called def-forms (with meaningful names like def-theorem, def-constant, etc.). These are labeled tree structures. Below the most prominent objects together with their most important child nodes are listed (optional child nodes are listed in rectangular brackets):

- def-section: id, section, file-names
- def-theory: id, parent-theories, language, constants, axioms
- def-translation: id, source-theory, target-theory, sort-, constant-pairs.
- def-theorem: id, formula, target-theory, [source-theory, translation, proof]

def-section is used to conveniently load packages of knowledge into working space. Def-theory allows multiple theory inheritance. The language of a theory is the union of the explicit named language and the languages of the parent theories. Thus theories can be built up in a hierarchy such that the whole mathematics of the Imps database is based on some fundamental theory (e.g. Zermelo-Fraenkel set theory). The Imps developers call this paradigm "big theory" as opposed to the so called "little theory" paradigm. The "little theory" does not force the mathematician to stick to one particular hierarchy, but allows to relate theories of different languages via translations. A translation is a top level object defined by a source theory $\mathcal{T}_1$ and a target theory $\mathcal{T}_2$, a mapping from sorts of $\mathcal{T}_1$ to sorts, unary predicates, and sets of $\mathcal{T}_2$, and a mapping of constants of $\mathcal{T}_1$ to expressions of $\mathcal{T}_2$. The goal of the "little theory" paradigm is to interprete one theory in the language of another theory. An *interpretation* of $\mathcal{T}_1$ *in* $\mathcal{T}_2$ is a translation Φ from $\mathcal{T}_1$ to $\mathcal{T}_2$ if $\Phi(\varphi)$ is a theorem of $\mathcal{T}_2$ for each theorem φ of $\mathcal{T}_1$. It can be shown that a translation Φ from $\mathcal{T}_1$ to $\mathcal{T}_2$ is an interpretation if each translated axiom (and additional definedness properties) of $\mathcal{T}_1$ is a theorem in $\mathcal{T}_2$. For every inserted translation Imps tries as much as possible to verify that the translation is an interpretation. If successful it marks the translation as interpretation otherwise it prompts the user to prove the outstanding obligations. This is done by def-theorem where the formula is the obligation in the source theory which is translated then to the target theory and finally verified by an explicit proof provided by the user.

The use of interpretations is that problems difficult to prove in one theory might emerge as the translation of an easy solvable problem from another theory.

The user interface of IMPS is implemented as a special mode in XEmacs where the user interacts with IMPS via special commands or by using the adapted menus. For a high quality formula presentation Xdvi can be launched to show almost any content produced or loaded in the course of an IMPS session.

The strength of the user interface is rather its support for interactive theorem proving then for building or navigating through IMPS' mathematical database. The only feature to browse is to list all def-forms of the current buffer and to display a selected def-form of the requested list.

Recent theoretical development based on IMPS A successor framework of IMPS is the so-called *biform theory* [ffmm] which is simultaneously an axiomatic theory and an algorithmic theory; i.e. it provides a formal context for both deduction and computation. Moreover it is intended to be used with several background logics simultaneously.

2.5 MAYA and the Development Graph

The development graph (DG) [dg][?] is a theory management framework implemented in the MAYA[1] system to maintain the specification and verification of industrial-size software as an evolutionary process. This process can be described as formal life cycle:

1. Write or modify the specification text of the software and translate it into logical representation.
2. Build the graph of logical dependencies and extract local proof obligations.
3. Prove these obligations with a available theorem prover or fix faulty specifications; i.e. go to step 1.

The main notions to understand MAYA are "verification in-the-large" and "verification in-the-small" which are related to the notions of "little theory" and "big theory" of IMPS [?]. Note that MAYA does not come with its own theorem prover, but offers an interface to various external theorem provers for the third step in the formal life cycle. Therefore MAYA abstracts from any specific logic by allowing any logic that satisfies the entailment properties: reflexivity, monotonicity, transitivity, and signature morphism invariance of the deduction relation. This logical abstraction is based on theoretical framework of entailements and institutions [mesg][gog].

The development graph is a graph whose nodes are tuples of signature and local axioms where the edges are annotated with signature morphisms (related to translations in IMPS). There are four types of edges:

- local/global definition link
- local/global theorem link

Each node N of the graph can be interpreted as a theory $T_D(N)$ where D is an acyclic subgraph consisting of definition links: Let Φ_D^N be the union of all local axioms of the nodes linked by D then $T_D(N)$ is the closure of Φ_D^N under the entailment relation. This inheritance mechanism is similar to IMPS except that translation is not intended at this place in IMPS.

Theorem links represent assertions: An acyclic subgraph Δ *satisfies* a link $M \to N$ iff $T_\Delta(M) \subseteq T_\Delta(N)$ and it satisfies a set Γ of links if it satisfies all elements in Γ. A DG is *verified* iff Ψ_D satisfies Ψ_T where D is the subgraph of all definition links and T subgraph of all theory links.

The goal of the formal life cycle is to finally verify the DG. MAYA supports this evolutionary process as it maintains all the verification work already done. This is guaranteed by a special calculus operating on the DG.

The DG calculus provides inference rues to decompose the global links into local links and eliminate redundant links that are already represented by a path of other links. The difference between local and global links is that inheritance of local axioms are limited to the source node of the given edge. Global links in contrast cause recursive inheritance which can be blocked only by local links.

Each edge also contains information about proof obligations which are propagated to local links during the graph decomposition (verification in-the-large). If there are still open proof obligations after the complete decomposition, then these obligations are passed to the external theorem prover (verification in-the-small).

Basic operations on the DG. In order to build complex theories systematically and to relate them (or interpret them in IMPS words) to each other MAYA provides the following basic operations:

- Nodes: insert/delete nodes
- Links: insert/delete global/local definition/theorem links
- Local signature: insert/delete symbols
- Local axioms: insert/delete/change local axioms

After modifying the development graph MAYA runs a difference analysis between the old and the new graph.

The difference analysis aims at the preservation of as many validated conjectures during the transformation of the old proof to the new DG. This means the difference of axioms and signatures (between old and new DG) which were available during proof time must be determined.

The basic operations which change the DG require a notion of *node-equivalence*. As there doesn't exist an optimal solution heuristics are applied based on the number of shared local signature symbols plus similarity of the incoming definition links. Thereby nodes and links of the new and old DG are associated.

2.6 OMDoc

The OMDoc (Open Mathematical Documents) [12] format is an XML-based content markup scheme for (collections of) mathematical documents containing formal and informal content. As such it attempts to provide an infrastructure for the communication and storage of mathematical knowledge. In fact a considerable part of formal digital libraries is already translated to OMDoc and in the long run it is expected that more content of formal digital libraries will be represented in OMDoc than in any other format. This tendency suggests OMDoc also as communication and storage format for MIR and MTM.

The following description restricts itself to the formal part as this is our concern in the spirit of LP[3]. The purpose of the formal part of OMDoc is in short to make the *role* of a *formula* within a *theory* explicit and to *link theories* to each other to make content sharing explicit.

A formula in OMDoc is an XML fragment (enclosed by the `FMP` element) built in accordance to a formula language scheme called OPENMATH [4]. Its syntactic categories are such fundamental that any formal language can be expressed within them[4]: constants, variables, application, and abstraction. For convenience OPENMATH also supplies built-in constants such as integers and numbers, and it becomes extendable by user-defined higher order constants which are called symbols. This allows the construction of formulae with user defined function and relation *symbols*. We call the semantic mark-up with OPENMATH *content mark-up on the formula level*.

3. i.e. we won't discuss features such as narrative text, or presentation issues, meta information about the author etc.

4. according to the Church thesis the λ-calculus is capable to express anything that is considered to computable. Since the formulae of the λ-calculus are just built out of constants, variables, application, and abstraction, these components are supposed to be sufficient for any formal language.

A formula on its own is a pure syntactic object without a fixed meaning. In OMDoc the meaning of a formula is determined by its statement category and its theory where it is located. The statement categories are: *axiom*, *definition* and *assertion*. The assignment of a formula to a statement category is made explicit in XML representation by enclosing the formula inside an element whose tag-name is the corresponding statement category. This sort of mark-up is called *content mark-up on the statement level.*

A theory in OMDoc is logically constituted by a set of symbol declarations, axioms and definitions. All assertions of a theory are supposed to be proven or refuted based on its definitions and axioms. From a purely logical point of view a theory can be considered as an entailment statement $\Gamma \vDash \Delta$, where Γ is the set of definitions and axioms and Δ the conjunction of assertions. The XML representation of a theory in OMDoc is just a `theory` element with `symbol`, `definition`, `axiom`, or `assertion` as child elements. If a formula is part of a theory then each symbol must be explicitely declared in that theory. The listing below illustrates all kind of elements in little example theory.

```
<theory id='group'>
 <imports from='semigroup'/>
  <symbol id='0'/>
  <axiom id='neutral'>
   <FMP>∀x.x + 0 = x</FMP>
  </axiom>
</theory>
```

Listing: A Simple OMDoc Theory

Theories can be linked to each other in order to make content sharing explicit. The mechanism is borrowed from the framework of “little theories” [?] and adapted to the framework of the development graph [1]. All kind of links are realized in XML subelements of `theory`. In the listing above `imports` points to the theory with id `semigroup`. This sort of mark-up is called *content mark-up on the theory level.*

The conceptual strength of OMDoc is that it makes meaning explicit while remaining uncommitted to a particular logical formalization, inference system or implementation. The practical strength is that OMDoc is XML based. So it is open to standard XML transformation methods. In particular it can be easily translated to proprietary formats of third party applications.

2.7 MBase

MBase [8] is a database especially developed for the storage of OMDoc collections, which can be accessed and searched by humans and software systems. Thus MBase provides separate interfaces for these two different types of “clients”: one can browse through the MBase content via the web interface whereas software systems such as theorem provers can access it via an XML-RPC interface. The web interface enables navigation through theory collections, which resembles navigation through a filesystem. The descending level in this hierarchy are: collections, theories in a collection, statements in a theory. In each level the corresponding item is represented by its hyperlinked id which leads you to children of the current item. A mouse click on a statement id returns the pretty printed formula of this statement. Within a statement each symbol is hyperlinked which leads you to the corresponding symbol declaration. For any page presenting a single item (symbol or statement) there are hyperlinks to the theory where this item lives and to the OMDoc code. All pages are dynamically generated from the database.

Items can be queried by their Id allowing regular expressions. Formulae can be retrieved by simple pattern matching. For instance `eq(a(F X Y) a(F Y X))` finds all commutative functions. This valuable feature isn't supported by any other of the described systems.

3 Critical Discussion of the Related Systems

All described systems from above share the domain of operation which is more or less formal mathematical content. Their differences are their functionality and their principal application goal. How far do the functionalities of each system satisfy principal application goal of a MIRTMS?

3.1 MBase

MBase is the only one of the described systems whose principal application goal is mathematical knowledge retrieval. Essentially it provides three retrieval methods: 1) Navigation 2) query by item by id 3) formula pattern matching.

The general flaw of navigating through theories is that you will find only exceptionally all wanted information on a single page. In most cases you have to switch between several pages in order to recall some just found information For instance if you want to retrieve the definition of a symbol occurring in some statement. This definition might contain in turn symbols whose definitions you need to retrieve. Thus you do a lot of forward and backward operations from one page to another. Actually it would be much more convenient to have all information collected on one page.

For such tasks an elaborated query machinery is needed. Yet MBase' query language isn't that powerful. Searching items by their id is too weak to do interesting queries. The formula pattern matcher has a higher potential. However, MBase lacks a powerful combination of query features which would allow queries such as: Show all theorems from a certain set of theories matching a given pattern.

Even formula pattern matching is not the most powerful one can have on the formula level. In some situations (in particular in proof situations) you want to whether there is a theorem in a theory unifiable with a given formula. Thus a more powerful IRS then MBase should also support unification.

3.2 Mizar

Mizar is one of the largest formalized digital libraries with about 37000 theorems and 7000 definitions. Extending this library without reusing the already existing knowledge should be avoided. More and more the developers of the Mizar community become aware of this need. Navigating through hyperlinked Mizar articles is only a very limited help, since each target page of a link presents as much data as the initial page. Retrieving information this way is pretty inefficient. One has to spend most of the time to identify the actual information from the target page of a hyperlink. Also the general limits of navigation described for the MBase case do apply here as well.

The query language for the MML is certainly more valuable for information retrieval. However it requires a deep understanding of the particular Mizar notions of article, vocabulary, notation, cluster, etc. and how they are related to each other. Apparently this is one of the major difficulties even for an expert user of Mizar: "Even for advanced Mizar users, the most troublesome activity is preparing the environment directives, i.e. setting up the imports from the MML" [16]. Understanding how mathematical context is accomplished in Mizar is difficult. A query language based on this understanding is therefore only a limited help for information retrieval.

The very particular and obscure implementation of mathematical context in Mizar does not only affect information retrieval, but theory management too. The relation between article, vocabulary, notation and cluster isn't that clear as symbol, theory and theory import in OMDoc. In contrast to OMDoc Mizar doesn't make these relations explicit instead it implements it in the MML machinery such that it remains hardly comprehensible to the user. A tool to do theory management on the MML in the flavour of MAYA consequently doesn't exist (in particular MML lacks the notion of theory inclusion). Therefore knowledge reuse on the theory level isn't supported by Mizar.

3.3 FDL

Concerning information retrieval, the FDL Navigator provides only poor facilities for navigation and searching the database. As already mentioned theories are organized like Unix directories. Browsing the database merely means navigating through directories. Objects contained in theories aren't hyperlinked to anything else. Searching objects is restricted to string pattern matching of the object names. The search space can be restricted by including or excluding parts of directory tree. Compared to e.g. MML's query language this is a very poor information retrieval support.

Theory management is rather the strength of the FDL-Navigator as the description from above demonstrates. However (similar to Mizar), the concept of theory is quite involved and not explicitly given as in OMDoc. The mechanism of reference environment, essential for FDL's implementation of theory, and updating reference variables isn't very comprehensible. The effect and intention of a user's theory manipulation might therefore differ substantially.

Concerning theory management, FDL's concept of theories lacks a notion of theory inclusion. And FDL's theory import doesn't support theory morphisms. Furthermore, theory management and theory proving is not as well separated as in Maya. The FDL Navigator offers some interesting features, e.g. experimenting with alternate versions of definitions and theorems to prove a conjecture. However, these operations are not controlled by theory management which should keep track of the logical dependencies of theories.

The very tied integration of the theorem prover into theory management can cause even loss of information concerning the knowledge about the minimal assumptions to prove a theorem, since this knowledge is encoded inside powerful tactics of the theorem prover. This approach has to drawbacks: 1) Knowledge encoded in a tactic (i.e. a procedure) is much harder to retrieval than from explicit proof objects as provided in OMDoc. 2) Some tactics require, to be applicable in a proof situation, more assumptions as actually needed to conduct a proof. In such situations too many assumptions are consumed which means a decrease of information value about the minimal set of assumptions for a certain proof.

3.4 Imps and Maya

Imps' information retrieval support is very limited, it restricts itself to query def-forms by name or to list def-forms of the current theory. The system description of Maya [1] doesn't report of any kind of information retrieval support. Both systems certainly have their strength in theory management and Maya is an improvement of Imps in this respect. Therefore only Maya's flaws will be discussed here.

Maya's operations on the development graph enables the mutation of theories by adding, removing or changing symbols and axioms. It doesn't provide an effective mechanism to split or merge theories. Of course this could be done via Maya's basic operations, but carrying out several operations manually which could be done automatically is tedious and a source of error. Splitting theory is useful when a theory has grown to such a scale that parts of it could be already considered as subtheories. Merging theories is in some cases useful when two theories share most of their axioms. For instance one should merge two theories whenever they import each other anyway in order to reduce redundant data.

Operations on the development graph mutating more then one theory simultaneously, such as merging and splitting theories, will be subsumed under the notion of *theory factorization*. Theory abstraction is another example of theory factorization not supported by Maya. The concept of theories will be discussed in more detail in section 4.4. The rough idea is, if we observe certain isomorphic properties (e.g. symmetry, transitivity, has maximal element, etc.) in different theories then this often gives grounds to study these properties on its own; i.e. they become axioms in a new theory.

In general theory factorization is a powerful method to increase the information value of a formal digital library. There are a lot of operation steps involved in theory factorization that could be automatized, build on top of Maya's basic operations. Besides redistributing the content of theories factorization would also affect a consistent reorganization of theory links. Maya doesn't provide these composed operations.

3.5 Content Overlap of Formal Digital Libraries

All the formal digital libraries - Imps, PVS, TPS, Coq, Mizar, Omega, etc. - have there own format. Just looking at the names of objects (theories, articles, files, theorems, structures, etc.) shows that these libraries certainly have a lot of content in common, but just in a different format. This very specific format of each library is still the highest obstacle to compute their mutual content overlap.

OMDoc is most likely powerful enough to express any content of these libraries. In fact a considerable part of these libraries is already translated successfully to OMDoc. It could be more and the translation scripts must be improved in some cases, such that no information gets lost during the translation process. Finally it is time to develop useful applications for these "OMDoctized" libraries - i.e. in particular a system for mathematical information retrieval and theory management. Such a system would help to identify and merge the actual content overlap.

3.6 ActiveMath

ActiveMath is one of the most prominent applications using OMDoc from the beginning. But it makes almost no use of OMDoc's theory features. Only simple theory imports without morphisms are partially supported. The semantics of formulae are only used for exercise interactions with computer algebra systems and the theorem prover Omega. In general ActiveMath does not supply theory management facilities as Maya or Imps.

ActiveMath doesn't make any logical use of theory import and theory inclusion even doesn't exist there. A different notion of theory, however, could be seen in the use of ActiveMath's proprietary attributes such as: `depends-on`, `definition-for`, `example-for`, `motivation-for`, etc. These attributes (applicable in various OMDoc elements) are essential for the user adapted course generation. Thus elements from different places, more or less independent from OMDoc theories, can be connected on purely pedagogical purpose. Let us call it the *pedagogical theory graph* opposed to Imps and Maya's logic oriented development graph, which we want to call *logical theory graph*.

The user (at least on the level of a graduate student) would benefit from mathematical learning environment more if it would support additionally to pedagogical also the logical theory graph. Because something like theory inclusion which also a pedagogical valuable could become visible. Also the user and author could trust the system much more with respect to the logical consistency of the content. This is not guaranteed in ActiveMath since it does not provide any tools for authors to check their content against logical consistency.

Besides the course generator ActiveMath's dictionary works on the pedagogical theory graph as well; i.e. a dictionary query is only a string matching query. There is no expert search e.g. for certain formula pattern in assertions. The query result can't be ordered or filtered by the user. It always shows a list of items ordered by pedagogical dependencies whereas a list of reliable logical dependent items is not available in ActiveMath.

Perhaps the main criticism of ActiveMath's pedagogical theory graph concerns its management. In fact there is no framework to build up or manipulate systematically any kind of dependencies, neither pedagogical nor logical neither for the author nor for the learner. In particular, a framework for the user to reorganise the theory graph (including both pedagogical and logical dependencies) is at the top of the authors wishlist for a learning environment, since reorganising content again and again is an effective learning method.

4 The Idea of a MIRTMS

The principal application goal of the MIRTMS[5] is MIR and MTM on formal digital libraries whose content is represented in OMDoc. MIR is about finding information in a digital library whereas MTM is about modifying content of a digital library coherently. The improvement MIRTMS in comparison to former systems will be on one hand the enhanced query features for MIR and on the other hand theory factorization for MTM. The novelty to be emphasized is the integration of both MIR and MTM in one system.

4.1 The Mathematical Information Retrieval System

MIR means sending a *request* formulated in the MIRL to the MIRS and receiving a *query result*. The whole performance is called *query*. A request in MIRL is a logical expression with some object variable. The intention of a request $R(x)$ is to find all objects x satisfying this request. Mathematically spoken a query result for the request $R(x)$ in the digital library L is the set $\{x \in L | R(x)\}$.

4.1.1 Basic Relations and Properties in the MIRL

The domain of semantic mark-up supported by the MIRS is essentially the formal part of OMDoc which is sufficient to encode all well known formal digital libraries. This restricted OMDoc is already described in section 2.6. Now we are going to present the basic relations and properties available in the MIRL which are derived from this formal part of OMDoc.

Elementary properties A *base type* in MIRL is a property generated by the translation from <*tagname* id='*elem-id*' ...>...</*tagname*> to *tagname*(*elem-id*). The set of most important base types in MIRL comprise: definition, axiom, assertion, theory, proof.

Elementary relations A *reference relation* in MIRL is a relation generated from references encoded in attributes of OMDoc elements. For instance if we have <definition id='def.abs' for='#abs'> ...</definition> that refers to a symbol declaration <symbol id='abs'/> then this translates to the MIRL relation definition-for(def.abs,abs). An *element relation* in MIRL is a relation generated from the parent-child relation in OMDoc. The translation pattern is from <*p* id='*pid*'><*c* id='*cid1*'> ...</c><*c* id='*cid12*'/>...</c>...</p> to *cs*-in-*p*([cid1,cid2,...],pid). This pattern is mainly used to generate axioms-in-theory, definitions-in-theory and assertions-in-theory.

Compound properties and relations From these elementary properties more expressive properties and relations are derived in MIRL. We don't explain how they can be derived explicitly instead we are content with short comments.

- formula(obj) iff obj is anything that can be expressed in OPENMATH.
- symbol(sid,formula) iff sid is an atomic element occurring inside formula[6].
- statement(obj) iff assumption(obj) or assertion(obj).
- proof(P,c) iff assumption(c) and statement(p) for all p ∈ P. (read: there is a proof for c under the premises P)
- import(t1,t2) iff theory(t1) and theory(t2) and there is a definition link (as described in section 2.5) in t1 that points to t2.

5. A recall of all abbreviations that will be used in this section: MIR: mathematical information retrieval; MTM: mathematical theory management; MIRS (MTMS): a system for doing MIR (MTM); MIRTMS: MIRS+MTMS; MIRL: the query language of the MIRS.

6. as a constant, as a function symbol, as a relation symbol, or as symbol for a binding operator like the integral symbol.

- `proven-include(t1,t2)` iff `theory(t1)` and `theory(t2)` and there is a theory link (as described in section 2.5) in `t1` that points to `t2` and for every assumption `ass` in `t1` it holds `proof(A,ass)` where `A` is the set of assumptions in `t2`.
- `postulated-include(t1,t2)` same as `proven-include(t1,t2)` but without these proofs.

4.1.2 Formula Matching and Unification in the MIRL

The relations and properties introduced so far are powerful for MIR on the statement and theory level, but to weak to run interesting queries on the formula level. On that level formula matching and unification are the strength of the MIRS. This means in MIRL:

- `matches(`*formula*`,`*pattern*`)` iff there is a substitution σ such that *formula*$=\sigma($*pattern*$)$.
- `unifies(`*formula1*`,`*formula2*`)` iff there is a substitution σ such that $\sigma($*formula1*$)=\sigma($*formula2*$)$.

A *formula pattern* is a formula with some labeled placeholders where the substitution eventually acts on. For instance assume capital letters stand for labeled placeholders then the pattern $F(X,E)=E$ matches the formula $+(2,0)=0$ and $\circ(a,e)=e$, whereas the pattern $F(X,e)=e$ matches only the second formula.

4.1.3 Example Request in MIRL

Though we haven't defined the concrete syntax of the MIRL, we want to look at an example of a query request how it might look like:

```
search-all obj: (assertion-in-theory(obj,𝒯1) or assertion-in-theory(obj,𝒯2))
                and matches(obj,F(X,E)=E)
```

In natural language this would be read as: "Search all assertion from theory $\mathcal{T}_1$ or $\mathcal{T}_2$ matching $F(X,E)=E$."

4.1.4 Query Results as Views

So far only the built-in relations are discussed. The MIRL will also support user-defined relations, e.g.:

```
def-rel my-rel(obj): (assertion-in-theory(obj,𝒯1)
                      or assertion-in-theory(obj,𝒯2))
                     and matches(obj,F(X,E)=E)
```

Since the MIRS operates on a mutable digital library, the query result is state dependent; i.e. two requests based on one query expression can yield two different results. In order to save a query result the MIRL allows to bind it to a name. We call such a binding a *view*, e.g.:

```
def-view my-view: (assertion-in-theory(obj,𝒯1) or assertion-in-theory(obj,𝒯2))
                  and matches(obj,F(X,E)=E)
```

Both, views and user-defined relations can be used in turn inside requests[7].

4.2 Mathematical Theory Management System

While MIR is just a reading operation, MTM is about manipulating the content of a digital library. All content manipulation operations of the MTMS possibly affecting the coherence of the digital library can be constrained to preserve this coherence.

7. This concept resembles views and nested queries in SQL.

4.2.1 MTM Coherence

Roughly spoken formal objects are coherent if they are really what they claim to be; i.e. a theorem must have a proof, an assertion must have a home theory, a theory link must point to a theory, etc. A complete definition of coherence can't be given at this stage, but some example relations will be sufficient to get the idea of *coherence relation* (or *coherence property*):

- `symbols-declared(`*`formula,theory`*`)` iff all symbols occurring in *`formula`* are declared in *`theory`*
- `proper-import(`*`theory`*`)` iff all import links in theory point to existing theories
- `grounded-proof(`*`proof`*`)` iff all premises of this *`proof`* are either assertions or proper theorems
- `proper-theorem(`*`thm`*`)` iff *`thm`* is an assertion of type theorem and there is a grounded proof object for *`thm`*

Some of these coherence relations are not necessarily relevant for a logical consistent interpretation of a theory. Again it is not possible at this stage to provide a complete list of properties to define theory consistency. Basically a *consistency property* is a property of either an assumption, an assertion, a proof or a theory. For instance `grounded-proof` as defined above is a consistency property.

It must be emphasized that consistency as we introduce it here does not mean the same as consistency in model theory or in any calculus. The point is that the MIRS does not try to verify a proof described by a `proof` object. It just checks whether the premises explicitly given in the `proof` point to existing assertions or theorems. In fact the proof objects might describe incorrect proofs and axioms might be inconsistent from a model theory point of view. These issues are outside the scope of the MIRS and left to theorem provers or human verification.

4.2.2 MTM Operations

The basic operations of MAYA on the development graph (see section 2.5) will be available in the MTMS too. Only additional features, already mentioned as theory factorization, will be discussed here.

With the notation $(\Gamma \vdash \Delta)$ we want represent a coherent theory where Γ represents all its assumptions and Δ all its theorems. To simplify the exposition we disregard possibly involved morphisms.

Intersection of theories: If two theories share a subset of assumptions then this might be a reason to take this subset as set of assumptions for a new theory. More formally:

$$\{(\Gamma_1 \vdash \Delta_1), (\Gamma_2 \vdash \Delta_2)\} \longrightarrow \{(\Gamma_1 \cap \Gamma_2 \vdash \Delta_3'), (\Gamma_1' \vdash \Delta_1'), (\Gamma_2' \vdash \Delta_2')\}$$

with $\Gamma_1 \cup \Gamma_2 = (\Gamma_1 \cap \Gamma_2) \cup \Gamma_1' \cup \Gamma_2'$ and $\Delta_1 \cup \Delta_2 = \Delta_1' \cup \Delta_2' \cup \Delta_3'$.

Abstraction of theories: Sometimes we observe that some properties of objects are so fundamental that they could serve as axioms for a new theory. In particular if we find these properties also in other theories we consider the new theory as abstract theory since it abstracts from the specific definition of the original theories:

$$\{(\Gamma_1 \vdash \Delta_1), (\Gamma_2 \vdash \Delta_2)\} \longrightarrow \{(\Gamma_1 \cup \Gamma_2 \vdash \Gamma_3), (\Gamma_3 \vdash \Delta_3)\}$$

with $\Delta_3 \subseteq \Delta_1 \cup \Delta_2$.

Dissolving of inconsistent theories: As the consistency of an axiom system in a sufficiently expressive logic is not decidable it is necessary to be prepared in a theory management system to handle discovered inconsistence. Namely by elimination of the “harmful” assumption.

$$(\Gamma \cup A \cup B \vdash \Delta \cup \{\bot\}) \longrightarrow \{(\Gamma \cup A \vdash \Delta'), (\Gamma \cup B \vdash \Delta'')\}$$

with $\Delta', \Delta'' \subseteq \Delta$.

4.3 The MIRTMS from a User's Perspective

From a user's point of view the MIRTMS is accessible via a GUI (which will be $\mathrm{T_EX_{MACS}}$) where one can run all the methods of the MIRTMS interface. However, it will be content with a relatively moderate user-friendly GUI since this is not the main interest of my research. The main purpose of the GUI will be to render OMDoc nicely and to have more or less convenient access to the MIRTMS functionality. The input for the MIRTMS methods will be OMDoc fragments and pointer to subparts of preceding MIRTMS output. A mini sample scenario should illustrate the interaction with the MIRTMS: In order to insert a new definition into a theory one would retrieve the intended theory with the corresponding method. The MIRTMS would return an OMDoc fragment which represents this theory (e.g. a theory name). Then the user would apply a definition method which needs as input an OPENMATH formula and a pointer to the theory where the definition should be inserted. In this situation the user would take the last MIRTMS output (i.e. a pointer to that theory) as input for the definition method.

5 A Case Study

For this section let us imagine to be a user of the envisioned MIRTMS. The following fictive interaction with such a system should demonstrate its usefulness to solve various tasks concerning mathematical matter. This will be done best by assuming different degrees of the user's background knowledge and the existence of appropriate mathematical content accessible through the system. Instead of the concrete MIRL syntax introduced in the previous section we are going to describe the interaction rather informally and thereby more readable. Furthermore some new features not explicitly discussed in the previous section will be used in passing; e.g. the ordering of a query result list by a certain criterion.

5.1 Views and Theory Inclusion

Imagine to be an undergraduate student who recently learned some math about sequences and is now going to study the basic concepts of analysis. We haven't heard about any advanced topics as vector- or functional analysis.

A theory $\mathcal{T}_{\text{seq}}$ of convergent sequences and a theory $\mathcal{T}_{\text{func}}$ of convergent functions may be assumed in our knowledge base. And both theories import from a third theory $\mathcal{T}_{\text{base}}$[8] the *field* properties for reals, the *axioms of order*, and the definition of *absolute value function* and its well known properties. After launching the system we would select $\mathcal{T}_{\text{func}}$ as working theory.

Theories such as $\mathcal{T}_{\text{func}}$ contain a whole bunch of assumptions and assertions either as local elements or included or imported from underlying theories. Therefore the system would offer many filters in order to hide those parts of the content which are not of any interest at a certain situation. As we approach this theory for the first time we would try to get a logical ordered view of what is new for us. We accomplish our goal by the following instructions to the system:

- hide all inherited items (i.e. from $\mathcal{T}_{\text{base}}$), because we know them from earlier studies.
- order the visible items logically such that each notion is introduced by axioms or a definition before it is used by other definitions or assertions.

After a first reading we would decide to attack the first conjecture `func.lim+` of the logically ordered list of items in $\mathcal{T}_{\text{func}}$ which could be for instance

`func.lim+`: $\lim(f, x_0, c) \wedge \lim(g, x_0, d) \Rightarrow \lim(f+g, x_0, c+d)$

For a better focus on this problem we would next hide all irrelevant items of the current view of $\mathcal{T}_{\text{func}}$ with respect to `func.lim+`. In this context the command "show only relevant items" could mean: show besides the selected conjecture nothing else but the definitions of symbols occuring in this conjecture. The result view $\mathcal{T}^1_{\text{func}}$ would be:

8. actually this theory would be build out of several theories.

`func.lim`: $\lim(f, x_0, c) :\Leftrightarrow\ \forall\varepsilon.\exists\delta.\forall x.\varepsilon > 0 \wedge 0 < |x - x_0| < \delta \Rightarrow |f(x_0) - c| < \varepsilon$

`func.lim+`: $\lim(f, x_0, c) \wedge \lim(g, x_0, d) \Rightarrow \lim(f + g, x_0, c + d)$

We remember fuzzily the notion of convergence from some other similar theory where somewhat similar theorems were proved. This knowledge at hand should help solving our goal. In order to find these definitions and theorems we could look at first for similar and helpful theories. So we request the system to list all theories which have some knowledge in common ordered by the scale of knowledge overlap. Probably there are many theories based on $\mathcal{T}_{\text{base}}$, hence there is not really a best query result. But at least the search space of knowledge is shrank to a manageable size. Upon this result list of theories further queries can be adjoined.

In this case we decide to activate formula matching as query constraint. As we want to retrieve the notion of convergence from the result list of theories, we request the system to find a definition that resembles `func.lim`. A list of definition formulas will be returned from different theories, for instance *theory of vector analysis* or *theory of complex analysis*:

- $\lim(\vec{v}, \vec{c}) :\Leftrightarrow \forall\varepsilon.\exists\delta.\forall x.\varepsilon > 0 \wedge 0 < |x - x_0| < \delta \Rightarrow \|\vec{v}(x) - \vec{c}\,\| < \varepsilon$
- $\lim(f, g) :\Leftrightarrow \forall\varepsilon.\exists\delta.\forall z.\varepsilon > 0 \wedge 0 < |z - z_0| < \delta \Rightarrow \|f(z) - g\| < \varepsilon$
- ....

Unfortunately the request did not yield a theory familiar to us. So we turn back to our $\mathcal{T}^1_{\text{func}}$ and look for another strategy. It seems that we have to load more information from $\mathcal{T}_{\text{func}}$ into $\mathcal{T}^1_{\text{func}}$. We may ask the system to show other theorems of $\mathcal{T}_{\text{func}}$ containing "lim". An interesting lemma appears:

`func.lim-eqv`: $g = \lambda x.1/x \Rightarrow [\lim_\infty(f, c) \Leftrightarrow \lim(f \circ g, 0, c)]$

We add `func.lim-eqv` to $\mathcal{T}^1_{\text{func}}$ as it states that $\lim_\infty$ is in some sense almost the same as lim. Consequently we prompt the system to extend the view $\mathcal{T}^1_{\text{func}}$ as well by the definition

`func.lim-inf`: $\lim_\infty(f, c) :\Leftrightarrow \forall\varepsilon.\exists x_0.\forall n.\varepsilon > 0 \wedge x_0 > 0 \Rightarrow |f(x) - c| < \varepsilon$

After that we restart again a theory query trying to match `func.lim-inf` instead of `func.lim`. Now a different list of definition formulas will be returned from *theory of sequences*, *theory of series*, and *theory of functional analysis* among others:

- $\lim(a, c) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\varepsilon > 0 \wedge n > m \Rightarrow |a_n - c| < \varepsilon$
- $\lim(a, c) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\varepsilon > 0 \wedge n > m \Rightarrow |\sum_k^n a_k - c| < \varepsilon$
- $\lim(\vec{v}, \vec{c}) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\varepsilon > 0 \wedge n > m \Rightarrow \|f_n - \vec{c}\,\| < \varepsilon$
- $\lim(f, g) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\varepsilon > 0 \wedge n > m \Rightarrow \left\| \sum_k^n f_k - g \right\| < \varepsilon$
- ....

This result together with the first result from above may prompt us to think about a *theory of convergence* on its own. But for now we concentrate on our initial goal, so we look for a similar notion of convergence only in a theory we are familiar with. We select $\mathcal{T}_{\text{seq}}$ and dismiss all other theories for further investigations. In fact $\mathcal{T}_{\text{seq}}$ appears to provide the missing links to solve our problem. This becomes clear after some already filter applications (similar to those above) which yields eventually the view $\mathcal{T}^1_{\text{seq}}$ of $\mathcal{T}_{\text{seq}}$ containing just two items:

`seq.lim`: $\lim(a, c) :\Leftrightarrow \forall\varepsilon.\exists n_0.\forall n.\varepsilon > 0 \wedge n > n_0 \Rightarrow |a_n - c| < \varepsilon$

`seq.lim+`: $\lim(a, c) \wedge \lim(b, d) \Rightarrow \lim(a + b, c + d)$

A visual comparison between $\mathcal{T}^1_{\text{seq}}$ and $\mathcal{T}^1_{\text{func}}$ convinces us that $\mathcal{T}^1_{\text{seq}}$ could help us very well to solve our goal since `seq.lim+` and `func.lim+` seem to be somehow the same by `func.lim-eqv`. But before we can use `seq.lim+` this way, it must be part of $\mathcal{T}^1_{\text{func}}$.

At this point a theory inclusion suggests itself. Remember that for a valid theory inclusion the axioms and definitions of the source theory must be proved to be theorems in the target theory. In our case this means that `seq.lim` must become a theorem in $\mathcal{T}^1_{\text{func}}$. Obviously this should be easy as `seq.lim` should match with `func.lim-inf`. In fact the system provides us the appropriate morphism $\sigma := \{[n, n_0{:}\, \mathbb{N} \hookrightarrow x, x_0{:}\, \mathbb{R}], [\, a{:}\, \mathbb{N} \to \mathbb{R} \hookrightarrow f{:}\, \mathbb{R} \to \mathbb{R}], [\text{conv} \hookrightarrow \lim_\infty\,]\}$. Now we ask the system whether $\mathcal{T}^1_{\text{func}}$ really includes $\sigma\mathcal{T}^1_{\text{seq}}$.

Note that there is actually more to check in order to establish a real theory inclusion: $\mathcal{T}^1_{\text{seq}}$ is a mere theory fragment of $\mathcal{T}_{\text{seq}}$ and its construction by filter operation does not at all guarantee its validity. Of course our system supports this validity check for theory fragments. If we tell our system that we aim to include $\mathcal{T}^1_{\text{seq}}$ into $\mathcal{T}^1_{\text{func}}$ it checks the validity of $\mathcal{T}^1_{\text{seq}}$ in the background; i.e. it adds all assumptions from $\mathcal{T}_{\text{seq}}$ to $\mathcal{T}^1_{\text{seq}}$ needed to solve `seq.lim+`. In our case $\mathcal{T}^1_{\text{seq}}$ is fortunately already a valid theory fragment which means that $\mathcal{T}^1_{\text{func}}$ in fact includes $\sigma\mathcal{T}^1_{\text{seq}}$. In case of invalidity of the source theory fragment all added assumptions would become proof obligations in the process of theory inclusion.

After we have successfully established $\mathcal{T}^1_{\text{func}} \sqsupseteq \sigma\mathcal{T}^1_{\text{seq}}$ we can attack our original goal to prove `func.lim+` which is a rather simple proof with the use of σ`seq.lim+` and `func.lim-eqv`.

Although we are actually done, it would be wasteful not to take all theorems we could get for free: `seq.lim+` is not the only theorem derived by `seq.lim`. So we request the system to add to $\mathcal{T}^1_{\text{seq}}$ all these theorems from $\mathcal{T}_{\text{seq}}$ which can be derived by `seq.lim`.

Such an extension $\mathcal{T}^1_{\text{seq}}$ would not violate $\mathcal{T}^1_{\text{func}} \sqsupseteq \sigma\mathcal{T}^1_{\text{seq}}$. So we would gain besides σ`seq.lim+` all the analogous limit theorems for subtraction, multiplication, division. These in turn would enable us to easily prove just as `func.lim+` all the other corresponding conjectures.

Conclusively, the benefit of the theory inclusion was an extension of $\mathcal{T}^1_{\text{func}}$ by limit theorems from $\mathcal{T}_{\text{seq}}$ without any proof effort and with these theorems in turn a surprising simple solution for limit conjectures of $\mathcal{T}^1_{\text{func}}$.

We considered for this example discussion an undergraduate student as system user. The main reason for this decision was perhaps the content itself which is indeed of undergrad level. However, the interaction with the system, i.e. the actions of theory management could be relevant in many aspects also for a graduates or even researchers, as well as for lecturers. A few arguments to support this claim: For solving a mathematical problem it is a natural practice to consult facts from related theories. An organised arrangement of theories facilitates the retrieval of these facts by an automatic measure of theory overlap as well as formula matching. Such automatic assistance may be regarded as add on to human expert consultancy.

Which of the illustrated operations might be interesting for lecturers or teachers? If we consider a mathematical theory base without views that grows for ever then our example is hardly realistic since the given limit conjecture was already proved by a thousand students. For didactical reasons it is the teachers purpose to present the students a view were all these simple proofs are hidden.

5.2 Theory Abstraction Motivated by Common Facts

This time we imagine to be a student familiar with *integral theory* an we are experts in the *theory of infinite series* $\mathcal{T}_{\text{ser}}$. As we already figured out by our theory queries with formula matching there are a lot of notions of convergence around besides those of $\mathcal{T}_{\text{seq}}$ and $\mathcal{T}_{\text{func}}$. Let's assume we saved the query result. This is an extract of the result list:

- $\lim(\vec{v}, \vec{c}) :\Leftrightarrow \forall\varepsilon.\exists\delta.\forall x.\varepsilon > 0 \wedge 0 < |x - x_0| < \delta \Rightarrow \|\vec{v}(x) - \vec{c}\,\| < \varepsilon$
- $\lim(f, g) :\Leftrightarrow \forall\varepsilon.\exists\delta.\forall z.\varepsilon > 0 \wedge 0 < |z - z_0| < \delta \Rightarrow \|f(z) - g\| < \varepsilon$
- $\lim(a, c) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\varepsilon > 0 \wedge n > m \Rightarrow |a_n - c| < \varepsilon$
- $\lim_\infty(f, c) :\Leftrightarrow \forall\varepsilon.\exists x_0.\forall x.\varepsilon > 0 \wedge x > x_0 \Rightarrow |f(x) - c| < \varepsilon$
- $\lim(f, c) :\Leftrightarrow \forall\varepsilon.\exists x_0.\forall x.\varepsilon > 0 \wedge x > x_0 \Rightarrow |\int_0^x f(x')\,dx' - c| < \varepsilon$
- $\lim(f, g) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\varepsilon > 0 \wedge n > m \Rightarrow \|f_n - g\| < \varepsilon$

- $\lim(f,g) :\Leftrightarrow \forall\varepsilon.\exists m.\forall n.\, \varepsilon > 0 \wedge n > m \Rightarrow \left\| \sum_k^n f_k - g \right\| < \varepsilon$
- ....

And in each theory we can find the theorem $\mathrm{conv}(a) \Leftrightarrow \exists c.\lim(a,c)$ where conv is defined as the Cauchy property:

$$\mathrm{conv}(f) :\Leftrightarrow \forall\varepsilon.\exists n_0.\forall m,n.\, \varepsilon > 0 \wedge n > m > n_0 \Rightarrow |a_n - a_m| < \varepsilon$$

Obviously a human regards these definitions just as variants or instantiations of one particular notion of convergence. How do we make the semantic between these variants explicit? Our approach is to build a new theory containing a single definition of convergence which is then imported by each corresponding theory via the appropriate morphism. The procedure of theory extraction may proceed as follows: We may take for instance the old `seq.lim` from $\mathcal{T}_{\mathrm{seq}}$ and build a new theory $\mathcal{T}_{\mathrm{conv}}$ containing nothing else then this definition. The system is going to find the morphisms from `seq.lim` to all the other convergence definitions. Afterwards in each theory the convergence definition will be replaced by the corresponding theory and an import of $\mathcal{T}_{\mathrm{conv}}$.

Although we have a new $\mathcal{T}_{\mathrm{conv}}$ going to be imported by many other theories it is not an advantage on its own, because there is no knowledge overlap: Instead of the various definitions the theories store the same information as morphisms. A real benefit will occur as soon as $\mathcal{T}_{\mathrm{conv}}$ contains also theorems shared by the theories which import $\mathcal{T}_{\mathrm{conv}}$. Hence our next goal is to search the common theorems.

At first we let the system collect in each theory containing one of these definitions all theorems about convergence. As we are an expert of $\mathcal{T}_{\mathrm{ser}}$ we recall which of the theorems listed by the system were frequently used in proof practise: The *principle of dominance* was the most important:

`ser.dominance`: $\mathrm{conv}(a) \wedge \forall k.\, a_k \geqslant |b_k| \Rightarrow \mathrm{conv}(b)$

And this principle is also present in the *theory of integrals* $\mathcal{T}_{\mathrm{int}}$. The difference is just the definition of conv in $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$ resp.:

`ser.conv`: $\mathrm{conv}(a) :\Leftrightarrow \forall\varepsilon.\exists n_0.\forall m,n.\, \varepsilon > 0 \wedge n > m > n_0 \Rightarrow \left| \sum_{k=m}^n a_k \right| < \varepsilon$

`int.conv`: $\mathrm{conv}(f) :\Leftrightarrow \forall\varepsilon.\exists x_0.\forall b,a.\, \varepsilon > 0 \wedge b > a > x_0 \Rightarrow \left| \int_a^b f(x)dx \right| < \varepsilon$

Now we hope to manage to propagate these principle to our new common base theory $\mathcal{T}_{\mathrm{conv}}$. Informally spoken the first idea is to collect all assumptions needed to derive `ser.dominance` in $\mathcal{T}_{\mathrm{ser}}$ and move them to $\mathcal{T}_{\mathrm{conv}}$ and finally import $\mathcal{T}_{\mathrm{conv}}$ into $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$. Thus `ser.dominance` becomes a theorem in $\mathcal{T}_{\mathrm{conv}}$ which would be also available via import in $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$. $\mathcal{T}_{\mathrm{conv}}$ would be what we name a *theory intersection* of $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$.

Unfortunately we fail to find the matching assumptions in $\mathcal{T}_{\mathrm{int}}$. Instead we make a different interesting observation which leads us to *theory abstraction*: We find by investigation of the proof of `ser.dominance` that it depends only on these properties:

`ser.dom-invar`: $\forall a,b.\ a \geqslant |b| \Rightarrow \sum_k a_k \geqslant \sum_k |b_k|$

`ser.triangle-ineq`: $\forall b.\ \sum_k |b| \geqslant \left| \sum_k b \right|$

`ser.dom-invar` expresses an invariance of dominance with respect to the application of the sum operator. `ser.triangle-ineq` can be regarded as variant of the triangle equation for the sum operator. Similarly the system finds inside $\mathcal{T}_{\mathrm{int}}$ that `int.dominance` depends only on:

`int.dom-invar`: $\forall f,g.\ f \geqslant |g| \Rightarrow \int f \geqslant \int |g|$

`int.triangle-ineq`: $\forall f.\ \int |f| \geqslant \left| \int f \right|$

This gives us grounds to instruct the system to build a new theory $\mathcal{T}_{\mathrm{ineq}}$ were these properties become axioms:

`ineq.dom-invar`: $\forall f, g.\ f \geqslant |g| \Rightarrow \mathcal{O}f \geqslant \mathcal{O}|g|$

`ineq.triangle-ineq`: $\forall f. \mathcal{O}|f| \geqslant |\mathcal{O}f|$

Further more the system finds the morphisms $\sigma_{\mathrm{ser}} := \{[\mathcal{O} \hookrightarrow \sum]\}$ and $\sigma_{\mathrm{int}} := \{[\mathcal{O} \hookrightarrow \int]\}$ such that we have by construction the theory inclusions $\mathcal{T}_{\mathrm{ser}} \sqsupseteq \sigma_{\mathrm{ser}}\mathcal{T}_{\mathrm{ineq}}$ and $\mathcal{T}_{\mathrm{int}} \sqsupseteq \sigma_{\mathrm{int}}\mathcal{T}_{\mathrm{ineq}}$. Consider now $\mathcal{T}_{\mathrm{conv}}$ with its only local definition of convergence:

`conv.conv`: $\lim(f, c) :\Leftrightarrow \forall \varepsilon. \exists n_0. \forall m, n.\ \varepsilon > 0 \wedge n > m > n_0 \Rightarrow |\mathcal{O}_{k=m}^{n} a_k| < \varepsilon$

If we import $\mathcal{T}_{\mathrm{ineq}}$ into $\mathcal{T}_{\mathrm{conv}}$ then

`conv.princ-dom`: $\mathcal{O}f \leqslant c \wedge g \geqslant |f| \Rightarrow \mathcal{O} \leqslant c$

holds obviously, as this is exactly what we observed in $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$ (modulo the morphisms σ_{ser} and σ_{int} resp.). So we add `conv.princ-dom` as theorem to $\mathcal{T}_{\mathrm{conv}}$. In some sense we can interprete now $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$ as instances of $\mathcal{T}_{\mathrm{conv}}$ with $\mathcal{T}_{\mathrm{ineq}}$ imported. We want to do this more explicit in terms of theory imports and includes. More precisely, we want to apply on our situation the idea of *parameterised theories* developed in [12]. As the situation concerns both theories $\mathcal{T}_{\mathrm{ser}}$ and $\mathcal{T}_{\mathrm{int}}$ in the same manner we want to restrict our considerations on $\mathcal{T}_{\mathrm{ser}}$.

In the terminology of parameterised theories $\mathcal{T}_{\mathrm{conv}}$ is the generic theory and $\mathcal{T}_{\mathrm{ineq}}$ its formal parameter. Yet the way of speaking that $\mathcal{T}_{\mathrm{ser}}$ is an instance of $\mathcal{T}_{\mathrm{conv}}$ with $\mathcal{T}_{\mathrm{ineq}}$ is not quite correct so far. We gain a clearer structuring if we split $\mathcal{T}_{\mathrm{ser}}$ into two theories $\mathcal{T}_{\mathrm{ser-base}}$ and $\mathcal{T}_{\mathrm{ser-conv}}$. Hereby $\mathcal{T}_{\mathrm{ser-base}}$ contains those assumptions needed to proof the theorems `ser.dom-invar` and `ser.triangle-ineq`, but it does not contain the theorems themselves as explicit local elements. As for $\mathcal{T}_{\mathrm{ser}} \sqsupseteq \sigma_{\mathrm{ser}}\mathcal{T}_{\mathrm{ineq}}$ it still holds $\mathcal{T}_{\mathrm{ser-base}} \sqsupseteq \sigma_{\mathrm{ser}}\mathcal{T}_{\mathrm{ineq}}$ too.

Furthermore let neither $\mathcal{T}_{\mathrm{ser-base}}$ nor $\mathcal{T}_{\mathrm{ser-conv}}$ contain `ser.conv` as explicit local element. $\mathcal{T}_{\mathrm{ser-conv}}$ even does not contain any explicit local element. Nevertheless $\mathcal{T}_{\mathrm{ser-conv}}$ holds exactly the same knowledge as $\mathcal{T}_{\mathrm{ser}}$ after a global import of $\mathcal{T}_{\mathrm{ser-base}}$ and local import of $\mathcal{T}_{\mathrm{conv}}$. Note that we do not import $\mathcal{T}_{\mathrm{conv}}$ globally as we would import redundant axioms from $\mathcal{T}_{\mathrm{ineq}}$ which are theorems in $\mathcal{T}_{\mathrm{ser-base}}$ anyway. $\mathcal{T}_{\mathrm{ser-base}}$ is called the actual parameter of the formal parameter $\mathcal{T}_{\mathrm{ineq}}$. Thus we say that $\mathcal{T}_{\mathrm{ser-conv}}$ became a theory instance of $\mathcal{T}_{\mathrm{conv}}$ by instantiation of its formal parameter $\mathcal{T}_{\mathrm{ineq}}$ by the actual parameter $\mathcal{T}_{\mathrm{ser-base}}$.

The same procedure is of course analogously applicable to $\mathcal{T}_{\mathrm{int}}$ such that $\mathcal{T}_{\mathrm{int-conv}}$ becomes another instance of $\mathcal{T}_{\mathrm{conv}}$ through the actual parameter $\mathcal{T}_{\mathrm{int-base}}$. After all we have build a constellation shown in ? that involves all theories.

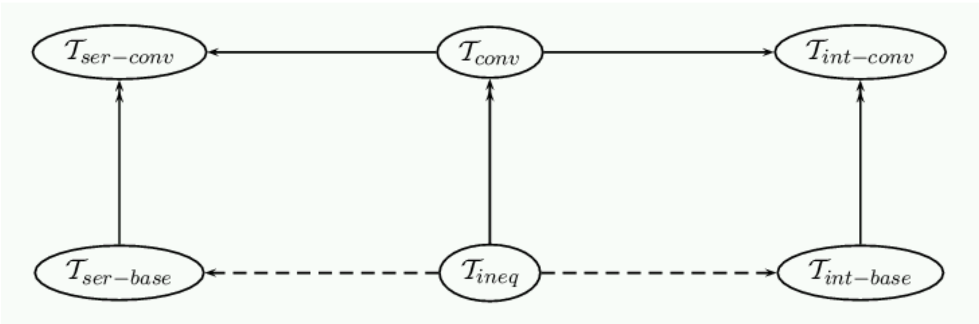


Figure 1.

We conclude again this section with some remarks to the virtual user of this session. Again the reason why we imagined to be graduate student rather then a researcher was the content which is of course not subject of research nowadays. However, abstracting from the content the operations carried out model much more a researchers work then that of a student: The history of mathematical research shows that common properties of objects from distant theories often caused the emergence of a new theory where these properties constitute its axioms. One of the most popular witness is perhaps group theory, or the whole algebra.

What we tried to demonstrate in this section was how to build abstract theories based on already known facts in existing theories. The systems support consisted hereby basically in finding similar theorems, building new theories, and morphisms, moving theorems, turning theorems into axioms, and - most important - keeping the rearrangement of theories consistent.

5.3 Theory Abstraction for Theory Exploration

In contrast to the last section we use the process of theory abstraction not to find a common ground for already known facts in existing theories, but for exploration. The degree of abstraction of $\mathcal{T}_{\text{conv}}$ relatively to $\mathcal{T}_{\text{ser}}$ and $\mathcal{T}_{\text{int}}$ is expressed in the corresponding morphism $\sigma_{\text{ser}} := \{[\mathcal{O} \hookrightarrow \sum\ ]\}$ and $\sigma_{\text{int}} := \{[\mathcal{O} \hookrightarrow \int\ ]\}$. So Σ is the only symbol that makes $\mathcal{T}_{\text{ser}}$ on the signature level an instance of $\mathcal{T}_{\text{conv}}$, and the same holds for $\int$ in $\mathcal{T}_{\text{int}}$. In this sense $\mathcal{T}_{\text{ineq}} \cup \mathcal{T}_{\text{conv}}$ is a minimal abstraction from $\mathcal{T}_{\text{ser}}$ and $\mathcal{T}_{\text{int}}$. We are now curious if we could find further interesting results from more distant theories by a further abstraction of our $\mathcal{T}_{\text{ineq}}$. Let's have a look again at $\mathcal{T}_{\text{ineq}}$:

`conv.dom-invar`: $\forall f, g.\ f \geqslant |g| \Rightarrow \mathcal{O}f \geqslant \mathcal{O}|g|$

`conv.triangle-ineq`: $\forall f. \mathcal{O}|f| \geqslant |\mathcal{O}f|$

We haven't touched the symbols $\geqslant$ and $|.|$ so far, and we have found only two instances for $\mathcal{O}$. But the system's formula matcher will find more related formulas if we tell the system to consider $\geqslant$, $|.|$ and $\mathcal{O}$ as placeholder in a pattern. One result matching `conv.dom-invar` can be found in set theory:

`set.lemma-1`: $\forall A, B. \forall k.\ A_k \supseteq B_k \Rightarrow \bigcup_k A_k \supseteq \bigcup_k B_k$

and another result matching `triangle-ineq-conv` in measure theory $\mathcal{T}_{\text{meas}}$:

`measure.lemma-1`: $\forall B.\ \sum_k \mu(B_k) \geqslant \mu(\bigcup_k B_k)$

Note that there is no morphism from `triangle-ineq-conv` to `measure.lemma-1`, but the other way round: $\sigma_1 := \{[\sum\ \hookrightarrow \mathcal{O}], [\bigcup\ \hookrightarrow \mathcal{O}], [\mu \hookrightarrow |.|]\}$ In this sense `measure.lemma-1` is a special case of `conv.triangle-ineq`. The intuition says that there are more links between $\mathcal{T}_{\text{meas}}$ and $\mathcal{T}_{\text{ineq}} \cup \mathcal{T}_{\text{conv}}$. So we have a closer look at $\mathcal{T}_{\text{meas}}$ and find:

`measure.lemma-2`: $\forall A, B.\ A \supseteq B \Rightarrow \mu(A) \geqslant \mu(B)$

Immediately from `set.lemma-1` and `measure.lemma-2` we follow

`measure.lemma-3`: $\forall A, B. \forall k.\ A_k \supseteq B_k \Rightarrow \mu(\bigcup_k A_k) \geqslant \mu(\bigcup_k B_k)$

which matches almost with `conv.dom-invar`. The only problem is that $|.|$ in $\mathcal{T}_{\text{ineq}}$ which has no correspondence in $\mathcal{T}_{\text{meas}}$. How could we get rid of it? The solution is to revise $\mathcal{T}_{\text{ineq}} \cup \mathcal{T}_{\text{conv}}$. More precisely we revise the proof of `conv.princ-dom`. The goal is to formulate versions $\mathcal{T}'_{\text{ineq}}$ and $\mathcal{T}'_{\text{conv}}$ of $\mathcal{T}_{\text{ineq}}$ and $\mathcal{T}_{\text{conv}}$ that abstract from $|.|$. After many trials and errors of formula pattern matching we get finally $\mathcal{T}'_{\text{ineq}}$:

`ineq.dom-invar`: $\forall b, a.\ b \trianglerighteq a \Rightarrow \forall m, n. \mathcal{O}^n_{k=m} b_k \geqslant \mathcal{O}^n_{k=m} a_k$

`transitivity`: $\forall a, b, c. a > b \wedge b \geqslant c \Rightarrow a > c$

Note that the triangle inequality has vanished. And $\mathcal{T}'_{\text{conv}}$:

`conv.conv`: $\text{conv}(a) :\Leftrightarrow \forall \varepsilon. \exists n_0. \forall n, m. \varepsilon > 0 \wedge n > m > n_0 \Rightarrow \varepsilon > \mathcal{O}^n_{k=m} a_k$

`conv.dominance`: $\text{conv}(a) \wedge b \trianglerighteq a \Rightarrow \text{conv}(b)$

It is quite easy to see that `conv.dominance` follows from $\mathcal{T}'_{\mathrm{ineq}}$ and `conv.conv`.

Again $\mathcal{T}'_{\mathrm{conv}}$ is the generic theory in this scenario and $\mathcal{T}'_{\mathrm{ineq}}$ its formal parameter. What theory instances do appear? Besides $\mathcal{T}_{\mathrm{ser}}$ (and $\mathcal{T}_{\mathrm{int}}$) we have a new candidate with $\mathcal{T}_{\mathrm{meas}}$. Let us step through these theories in the given oder.

For $\mathcal{T}_{\mathrm{ser}}$ we can take just the same splitting into $\mathcal{T}_{\mathrm{ser-base}}$ and $\mathcal{T}_{\mathrm{ser-conv}}$ as we did above. And again we import globally $\mathcal{T}_{\mathrm{ser-base}}$ and with $\sigma_2 := \{[\mathcal{O} \hookrightarrow |\sum|], [\trianglerighteq \hookrightarrow \lambda a, b.\forall k.a_k \geqslant |b_k|]\}$ locally $\mathcal{T}'_{\mathrm{conv}}$. With same morphism σ_2 we convince ourselves that $\mathcal{T}_{\mathrm{ser-base}} \sqsupseteq \sigma_2\mathcal{T}'_{\mathrm{ineq}}$ holds

- σ_2`conv.dom-invar`: $\forall a, b.\forall k.a_k \geqslant |b_k| \Rightarrow \forall m, n.|\sum_{k=m}^n a_k| \geqslant |\sum_{k=m}^n b_k|$

Note that you need this time the triangle inequality to prove the last inequality. So $\mathcal{T}_{\mathrm{ser-base}}$ must have it imported or included from somewhere else. After all $\mathcal{T}_{\mathrm{ser-conv}}$ is a theory instance of $\mathcal{T}'_{\mathrm{conv}}$ and $\mathcal{T}_{\mathrm{ser-conv}}$ the actual parameter. For $\mathcal{T}_{\mathrm{int}}$ we content to note that the same procedure is applicable such that $\mathcal{T}_{\mathrm{int-conv}}$ is a theory instance of $\mathcal{T}'_{\mathrm{conv}}$ and $\mathcal{T}_{\mathrm{int-conv}}$ the actual parameter.

A really new theory instance is $\mathcal{T}_{\mathrm{meas-conv}}$ obtained by the same import and include mechanism but with the morphism $\sigma_3 := \{[\mathcal{O} \hookrightarrow \mu(\bigcup)], [\trianglerighteq \hookrightarrow \lambda A, B.\forall k.A_k \supseteq B_k]\}$ and with the conventional measure axioms as $\mathcal{T}_{\mathrm{meas-base}}$. The definition of convergence would convert to

`measure.conv`: $\mathrm{conv}(A) :\Leftrightarrow \forall \varepsilon.\exists n_0.\forall n, m.\varepsilon > 0 \wedge n > m > n_0 \Rightarrow \varepsilon > \mu(\bigcup_{k=m}^n A_k)$

And the theory inclusion $\mathcal{T}_{\mathrm{meas-base}} \sqsupseteq \sigma_3\mathcal{T}'_{\mathrm{ineq}}$ consisting in the assertion

σ_3`conv.dom-invar`: $\forall B, A.\forall k.B_k \supseteq A_k \Rightarrow \forall m, n.\mu(\bigcup_{k=m}^n B_k) \geqslant \mu(\bigcup_{k=m}^n A_k)$

can be proved easily from the axioms of $\mathcal{T}_{\mathrm{meas-base}}$.

Up to now we were eager to find a new generic theory which can be used for new theory actualisation. Since we performed this business successfully we are curios if we can get just by trial and error even more theory instances. Only sketches of ideas will be presented in the following.

In set theory $\mathcal{T}_{\mathrm{set}}$ we find the subset invariance with respect to the union

`set.dom-invar`: $\forall k.B_k \supseteq A_k \Rightarrow \forall m, n.\ \bigcup_{k=m}^n B_k \supseteq \bigcup_{k=m}^n A_k$

If we take a morphism $\sigma_4 := \{[\mathcal{O} \hookrightarrow \bigcup], [\trianglerighteq \hookrightarrow \lambda A, B.\forall k.A_k \supseteq B_k], [\geqslant \hookrightarrow \supseteq]\}$ we would get for the definition of convergence:

`set.conv`: $\mathrm{conv}(A) :\Leftrightarrow \forall \varepsilon.\exists n_0.\forall m, n.\varepsilon > 0 \wedge n > m > n_0 \Rightarrow \varepsilon > \bigcup_{k=m}^n A_k$

But the question here is: what could be a reasonable meaning of $\varepsilon > \bigcup_{k=m}^n A_k$? One idea could be for instance that ε is a set and $A > B$ for sets could mean "A has higher cardinality then B". This interpretation would also satisfy the transitivity which we get via σ_4

`set.transitive`: $\forall a, b, c\colon \mathrm{set}.a > b \wedge b \supseteq c \Rightarrow a > c$

Remember that transitivity is one axiom of $\mathcal{T}'_{\mathrm{conv}}$ necessary for the proof of `conv.dominance`. So we have in fact all ingredients to make out of $\mathcal{T}_{\mathrm{set}}$ a theory instance of $\mathcal{T}'_{\mathrm{conv}}$. The open question, however, is whether `set.conv` together with the given interpretation of ">" as "greater cardinality" is a reasonable definition: Do there exist non-trivial instances of such convergent sequences of sets? If there is a justifiable objection (and indeed there is) then we still have opportunity to find a more fruitful interpretation for that ">" before we abandon our attempt on the whole to make out of $\mathcal{T}_{\mathrm{set}}$ a theory instance of $\mathcal{T}'_{\mathrm{conv}}$.

A quite similar situation can be found in a theory of deduction $\mathcal{T}_{\mathrm{ded}}$. Here we find an entailment invariance with respect to the conjunction

`set.dom-invar`: $\forall k.B_k \vdash A_k \Rightarrow \forall m, n.\ \bigwedge_{k=m}^n B_k \vdash \bigwedge_{k=m}^n A_k$

This theorem already suggests the morphism $\sigma_4 := \{[\mathcal{O} \hookrightarrow \bigwedge\,], [\trianglerighteq \hookrightarrow \lambda A, B.\forall k.A_k \vdash B_k], [\geqslant \hookrightarrow \vdash]\}$ that could make $\mathcal{T}_{\text{ded}}$ an instance of $\mathcal{T}'_{\text{conv}}$. But the same question is again: what interpretation of $\varepsilon < \bigwedge_{k=m}^{n} A_k$ would make

$$\texttt{ded.conv}: \operatorname{conv}(A) :\Leftrightarrow \forall \varepsilon.\exists n_0.\forall m, n.\varepsilon > 0 \wedge n > m > n_0 \Rightarrow \varepsilon < \bigwedge_{k=m}^{n} A_k$$

a reasonable definition.

After the benefit of the envisioned system is hopefully sufficiently described the ensuing discussion shall point to the involved research questions to be attacked that must preceed any implementation.

6 Research Tasks for the Implementation of the MIRTMS

After the benefit of the envisioned system is hopefully sufficiently described the ensuing discussion points to the involved research questions to be attacked preceeding any implementation.

The main bulk of research questions around theory management originate in a logical trivial fact: For any formula there are infinte many semantical equivalent but syntactically different formulae. Let me define the *logical content* of a set of formulae as this set itself plus all formulae deducable from this set, and the *logical meaning* of a formula as the set of all semantical equivalent formulae (similiarly define the logical meaning of a set of formulae). From a pure logical perspective these definitions are reasonable. But we are certainly on the wrong track if we believe that a MIRTMS should be used to make content and meaning explicit in this sense, because it is simply impossible. It is even not reasonable to make as much meaning and content explicit as possible in this sense. For instance if we have a formula $a + 0 = a$ then we don't gain more information if we make explicit that $(a + 0) + 0 = a$. It is a fundamental fact that some logical equivalences of formulae exhibit an information inflation (because of their triviality) whereas other equivalences represent a high information increase (for instance the equivalence of the set of axioms based on open sets and those based on neighbourhood in topology).

The purpose of a MIRTMS is not to increase the explication of logical content and meaning, but to increase information content! As we already said information depends on an interpreter and a goal. The most important interpreter is the user of the MIRTMS who also defines the current global goal of his activities. But all services of the MIRTMS can be considered also as interpreters of the available data which try to solve subgoals of the current user defined global goal. The prospects of success of the involved content interpreters may depend heavily on the content representation. It is a commonplace that maschine-readable data representation isn't well human-readable and vice versa. Though OMDoc is particularly designed to attack this dilemma it can't be the solution on its own, since OMDoc is only a format but not an interpreter of OMDoc content. The strength of OMDoc is its flexibility that allows to write content either more maschine-readable or more human-readable. On the other hand this flexibility is actually also the problem: Logical content can be represented in arbitrary many variations as OMDoc content. The question is always: Which representation is redundant or preferable and for whom (machine or human)? This central research question will be illustrated now on different levels in the OMDoc format.

6.1 Variations on the Formula Level

The following two formulae are variations of the ε-δ-criterion for convergence and at the same time examples for semantical equivalence:

$$\texttt{conv1}: \forall \varepsilon.\exists \delta.\forall x.\varepsilon > 0 \wedge 0 < \delta \wedge |x_0 - x| < \delta \Rightarrow |f(x) - c| < \varepsilon$$

$$\texttt{conv2}: \forall \varepsilon. 0 < \varepsilon \Rightarrow [\exists \delta. 0 < \delta \wedge \forall x. |x_0 - x| \leqslant \delta \Rightarrow |f(x) - c| < \varepsilon]$$

Of course these two definitions are only two examples of many more variants. A trained mathematician wouldn't take more then a second to see the semantical equivalence between those two formulas. Whereas a machine would have hardly a chance to recognize it. Only some syntactical differences are even not a problem for a machine:

- Prenexform: In `conv1` all quantifiers are gathered in front position as opposed to `conv2`.
- Normalform: `conv1` exhibits the *Hornclause* pattern $\alpha_1 \wedge \alpha_2 \wedge a_3 \Rightarrow \beta$ whereas `conv2` exhibits no normalform $\alpha_1 \Rightarrow (\alpha_2 \wedge a_3 \Rightarrow \beta)$

These problems are pure logical. Other syntactical differences are much harder for the machine:

- Duality principle: `conv1` assumes $\varepsilon > 0$ opposed to $0 < \varepsilon$ in `conv2`
- Inequality variant: `conv1` assumes $|x_0 - x| < \delta$ opposed to $|x_0 - x| \leqslant \delta$ in `conv2`

In these cases semantical equivalence can be only proved from additional theory knowledge.

Suppose `conv1` is and `conv2` isn't stored in the MIRTMS and suppose further a user who wants to retriev from the MIRTMS any formula logically equivalent to `conv2`. This is actually a hard unification problem.

Theory specific unification is a wide research field on its own with a lot of interesting theoretical as well as practical results [refs???]. It will be one task to find out which unification theories or even better implementation could be smoothly integrated into the MIRTMS. This task is many-fold: 1) It is an implementational question. 2) A query language must be designed adapted to the pluged in unification and pattern matching services 3) A conceptual problem concerns the whole framework of theory management: A user of the system wouldn't be an expert of unification theory of course. He considers formula matching as a global, i.e. theory independent, operation. This conflicts with theory dependencies of many unification types. The question is then: How could we satisfy the user's expectations?

A very much related example to that above is the following where the problem involves type theory. $\forall x\colon \mathrm{real}. |x| \geqslant x$ and $\forall x\colon \mathrm{individual}. x \in \mathbb{R} \Rightarrow |x| \geqslant x$ means the same and it is mostly upto the user's taste which expression he prefers. For unification again this is a problem to be tackled.

6.2 Variations on the Theory Level

Related to the variations of formulae with equivalent semantics we have similar variations on the theory level. As example we provide two variations of a theory of *semigroup* in OMDoc notation. Both contain logically the same information. It is easy to transform the first version into the second and vice versa. We explain only one direction, from the first to the second: `semigroup1` contains the axioms `closed` and `assoc`. In `semigroup2` they become simple definitions. Thereby each axiom id of `semigroup1` turns into a symbol in `semigroup2`. The symbols `set` and `op` from `semigroup1` in contrast do not appear any longer in `semigroup2`, because they are now bound in the defining λ-expressions $\lambda f, S.(...)$. Finally the symbol `semigroup` is defined in theory `semigroup2`. There is no corresponding axiom in `semigroup1` rather the whole theory `semigroup1` itself represents something that is just a symbol `semigroup` in theory `semigroup2`.

```
<theory id='semigroup1'>
 <symbol id='set'/>
 <symbol id='op'/>
 <axiom id='closed'><FMP>∀x, y ∈ set.∃x ∈ set.op(x, y) = z</FMP></axiom>
 <axiom id='assoc'><FMP>∀x, y, z ∈ set. op(x, op(y, z)) = op(op(x, y), z)</FMP></axiom>
</theory>
```

Listing: 1

```
<theory id='semigroup2'>
 <symbol id='assoc'/>
 <symbol id='closed'/>
 <symbol id='semigroup'/>
 <definition id='closed.def' for='closed' type='simple'>
   λf, S. ∀x, y ∈ S .∃z ∈ S. f(x, y) = z
 </definition>
 <definition id='assoc.def' for='assoc' type='simple'>
   λf, S. ∀x, y, z ∈ S . f(x, f(y, z)) = f(f(x, y), z)
 </definition>
 <definition id='semigroup.def' for='semigroup' type='simple'>
   λf, S.closed(S, f) ∧ assoc(S, f)
 </definition>
</theory>
```

Listing: 2

Again these two theory variations are only two examples of many more theory variations. It will be one research task how to identify two theories just as variations. One reason for instance are the different definition types: `simple`, `recursive`, and `implicit`. May be a trained OMDoc writer always chooses the most specific type of definition, but we cannot expect it from a novice or from a translation of formal mathematical libraries. These arguments show that we do not just have a unification problem on a formula level. How do we tackle unification on a theory level?

6.3 Variations on the Level of Theory Collections

The issue of theory variations can be even extended. In the semigroup example from above we have varied the assumptions by transforming axioms to definitions. In principle we could go even one step further and transform subformulae of an assertion to assumptions. Suppose a theory $\mathcal{T}_1$ containing nothing else then the assertion $a \wedge b \Rightarrow c$ with any formulae a, b, c. This theory is obviously logically equivalent to a theory $\mathcal{T}_2$ containing one assumption $a \wedge b$ and one assertion c, as well as to theory $\mathcal{T}_3$ containing two assumptions a and b and and one assertion c. In terms of logical semantic $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ is equivalent, because of the fact: $\vDash a \wedge b \Rightarrow c$ iff $\{a \wedge b\} \vDash c$ iff $\{a, b\} \vDash c$. (which corresponds to the $\Rightarrow$-, and $\wedge$-*elimination rules* of sequent calculus).

On the next level of content variations we consider cluster of theories. Suppose theories $\mathcal{T}_a$ and $\mathcal{T}_b$ each containing only one assertion a and b respectively, and the theory inclusion $\mathcal{T}_b \sqsubseteq \mathcal{T}_a$ holds. As we already know this is the case iff any assumption from $\mathcal{T}_b$ can be deduced in $\mathcal{T}_a$. In our case we would have to show $a \Rightarrow b$. But this fact can be expressed in one single theory $\mathcal{T}_{a+b}$ containing the assertion a and the theorem $a \Rightarrow b$. Again the logical equivalence of both variants is evident.

These examples indicate an almost unbounded freedom of coding mathematical content into OMDoc. This freedom is not a benefit on its own, it is only a chance to organize content beneficially. Our research will cover the following questions:

- How is information related to theory size (= number of theory items)?
- Does a theory inclusion always imply information increase? (At a first glance it is, but what if we only include trivial or otherwise useless theorems?)
- Does theory abstraction always lead to information increase?
- Could we increase information automatically by theory factorization?

As we have seen so far mathematical content can be represented in many variations that keep its logical semantic unaffected. In some cases it is obvious which variant bears more information. For instance `simple` type definition conveys more information then a `implicit` type information. However, for the most sorts of variants it is not clear which of them convey the most information. The examples from above exhibit this fact: In case of formula variants take the variants of definitions for convergence `conv1` and `conv2` from ?, and in case of theory variants ? take the variants of `semigroup1` and `semigroup2`. In order to gain real experience about the question how information may depend on the representation variant I will investigate real formal mathematical libraries from some systems described in section?.

6.4 Translation of Mathematical Libraries to OMDoc

As we mentioned in section ? we have large formal mathematical libraries, but unfortunately not in OMDoc representation. For some formats, e.g. Imps and Mizar, there are already translation scripts under development. If OMDoc is really capable to express any mathematical content then the there should be theoretically for each format a translation script that preserves information. Naturally we can't expect such ideal translation scripts. So it will be one task to improve those scripts or implement post processors for these OMDoc outputs. This is not that straightforward, because it is not obvious how to preserve information. For instance it is not clear whether or how the directive mechanism in Mizar can be reasonable used for building theory imports during the translation. In general for all libraries which have no explicit theory framework it is not at all uniquely determined how to enrich them with the theory feature of OMDoc.

It is one research goal to develop semi automated assistance for translation from formal mathematical libraries to OMDoc with a particular emphasis on theory construction. A very important requirement of such tool would be of course that the OMDoc output should have the highest possible information value. Consequently the development of this tool implies some operational notion of information. On the other hand we want to establish the notion of information by investigation of those libraries. In order to break this vicious circle our development is supposed to proceed iteratively with very basic features at the beginning. A very basic approach to reveal information redundancy will be the application of formula pattern matching.

6.5 Implementational Issues

Finally there is a lot to implement. Presumably the MIRTMS can make use of already existing services. Programms that should be considered:

- TeXmacs as GUI
- implemented unification algorithms
- Scripts for translation from formal math libraries to OMDoc
- Maya for theory maintainence
- MBase for content persistence

Since the effort to integrate these programs is unforseeable some integration targets might be dropped. The list above reflects roughly the priority hierarchy.

As implementation language I favour Scheme (a Lisp descendent) for the following reasons:

- Lisp like languages are pretty good for proccessing tree like structures
- elaborated libraries for XML proccessing already exist
- There are APIs for C and Java, and XML-RPC is supported
- Scheme compiler are open-source of course
- finally: Scheme is an extreme powerful language and it is the language where I have gained most experience.

7 Summary

We argue that mathematicians could benefit much more from the already existing large bodies of formal libraries with a system that supports MIR[9] as well as MTM. The effectiveness of MIR depends on the internal structure of the underlying fromal library how it organizes mathematical contexts i.e. mathematical theories. A MTMS is a system suitable to improve a library's theory structure, thus improving MIR. A MIRS is a system that makes information from a formal library effectively accessible and in particular it can be used very beneficially to retrieve those parts of a library which should be reorganized by the MTMS. The mutual benefit of a MIR and a MTM suggests itself to develope an integrated system - a MIRTMS - for both.

None of the existing support MIR and MTM simultanously. Even a restricted support either for MIR or MTM is limited in all existing systems. The limits with respect to MIR in those systems are either an incomprohensable theory encoding in the format of the underlying library (e.g. Mizar) or a query language that does not support context dependent queries (e.g. MBase). Moreover none of the systems integrate unification to their retrieval features. Concerning MTM, MAYA is the most advanced existing system. Its strength is the management of proof obligations generated by extending or changing axioms in theories. However, it does not support theory factorization; i.e. the consistent reorganization of whole theories including splitting and merging theories as well as building theory abstractions.

The proposed research is the specification and implementation of a MIRTMS. This system is intended to work on libraries encoded in the OMDoc format, because most (presumably all) formal libraries can be translated to OMDoc and large bodies of them are already translated. The MIRL of the MIRTMS will allow context dependent and unification supported retrieval of mathematical objects. The MTMS part of the system will focus on theory factorization.

For the implementation of the MIRTMS specialized existing systems wrt. to MIR and MTM (e.g. MAYA) will be integrated as far as possible. Hence, the specification will comprise a lot of interface design[10].

8 Work Plan

The research is divided in five tasks:

Viewer & Validator for restricted OMDoc: A subset of OMDoc relevant for MIRTMS (OMDoc*) will be defined and corresponding validator will be implemented. An configurable format transformer from OMDoc* to $\TeX_{\mathrm{MACS}}$ document trees will be implemented, such that $\TeX_{\mathrm{MACS}}$ can be used as content viewer. This task will be evaluated against all libraries translated to OMDoc* (see last task).

Theory query language A query and filter language in terms of OMDoc* will be defined and implemented excluding the formula level. Basic heuristics for an intellegent hit rating of query results will be explored. A case study with sample queries about each exisisting OMDoc* relation will evaluate this research task.

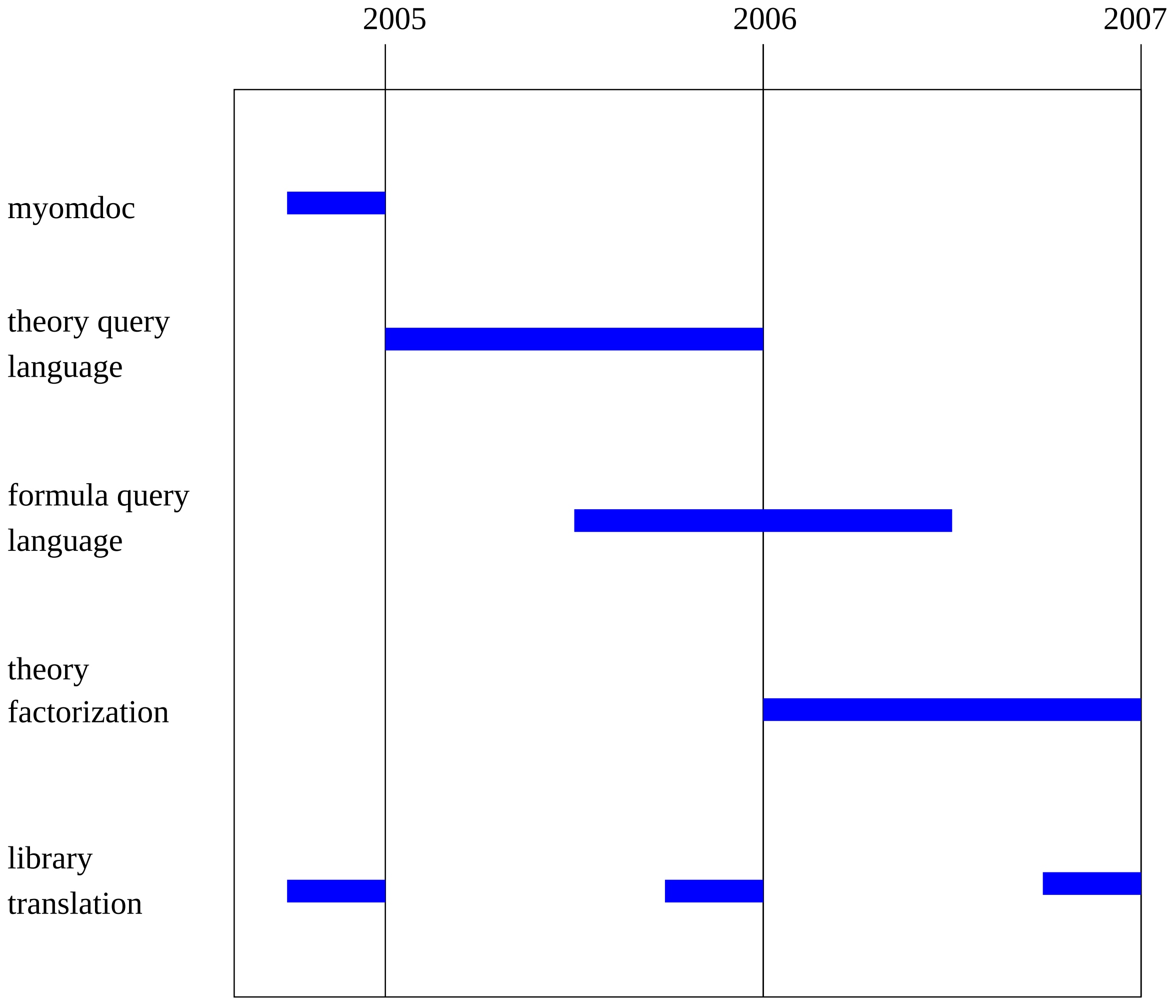
Formula query language After an orientation phase for unification theory and already implemented unification services a query and filter language for formulae will be defined. Already implemented unification services will be integrated as far as possible. The formula query language will be evaluated with respect to its benefit in finding theory morphisms.

Consistent theory factorisation A framework for consistent theory factorization will be worked out. Corresponding operations on theories will be implemented. MAYA will be integrated as far as possible. The connection between redundant content and information will be explored. As case study theory factorisation will be applied to restructure parts of the Mizar library, and to merge parts of different formal libraries (e.g. Mizar and PVS).

9. All abbreviations used in the summary: MIR: mathematical information retrieval; MTM: mathematical theory management; MIRS (MTMS): a system for doing MIR (MTM); MIRTMS: MIRS+MTMS; MIRL: the query language of the MIRS.

10. As such the design of the MTM functionality wrt. OMDoc can be compaired to DOM wrt. XML.

Library translation to OMDoc⋆ Content of several libraries (PVS, Mizar, Coq, TPS, etc.) will be transformed into a single library based on OMDoc⋆. Redundant content will be removed as far as possible.



Bibliography

[1] Serge Autexier, Dieter Hutter, Till Mossakowski, and Axel Schairer. The development graph manager maya (system description). In Hélene Kirchner, editor, *Proceedings of 9th International Conference on Algebraic Methodology And Software Technology (AMAST'02)*. Springer Verlag, 2002.

[2] Grzegorz Bancerek. Mml query description. `http://megrez.mizar.org/mmlquery/description.html`.

[3] Nicolas Bourbaki. Elements of Mathematics. Springer Verlag.

[4] The Open Math standard, version 2.0 public draft 4. Technical report, 2003. The Open Math Society, `http://www.openmath.org/standard/om20-2003-11-24/`.

[5] The Mizar Library Committee. Mizar mathematical library. Web Page at `http://www.mizar.org/library`.

[6] W. M. Farmer, J. D. Guttman, and F. J. Thayer. The IMPS user's manual. Technical Report M-93B138, The MITRE Corporation, 1993.

[7] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, October 1993.

[8] Andreas Franke and Michael Kohlhase. System description: MBase, an open mathematical knowledge base. pages 455–459.

[9] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Louis Nebert, Halle, 1879.

[10] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 37:349–360, 1931.

[11] Josef Urban Grzegorz Bancerek. Integrated semantic browsing of the mizar mathematical library for authoring mizar articles. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *Mathematical Knowledge Management, Proceedings of MKM 2004*, volume 3119 of *LNCS*, pages 44–57. Springer, 2004.

[12] Michael Kohlhase. OMDoc: An open markup format for mathematical documents (version 1.2), 2003. To appear in Springer LNAI; Manuscript at `http://www.mathweb.org/src/mathweb/omdoc/doc/spec/omdoc.ps`.

[13] Christoph Kreitz. The fdl navigator: Browsing and manipulating formal content. `http://www.nuprl.org/documents/Kreitz/03fdl-navigator.html`.

[14] E. Melis, E. Andres, A. Franke, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath system description. In *Artificial Intelligence in Education*, pages 580–582.

[15] The QED manifesto. Internet Report `http://www.rbjones.com/rbjpub/logic/qedres00.htm`, 1995.

[16] Piotr Rudnicki and Anrzej Trybulec. Mathematicel knowledge management in mizar.

[17] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume I. Cambridge University Press, Cambridge, Great Britain; second edition, 1910.