Rough Concept Analysis – Theory Development in the Mizar System

Adam Grabowski[1] and Christoph Schwarzweller[2*]

[1] Institute of Mathematics, University of Białystok, Białystok, Poland
adam@math.uwb.edu.pl
[2] Department of Computer Science, University of Gdańsk, Gdańsk, Poland
schwarzw@math.univ.gda.pl

Abstract. Theories play an important role in building mathematical knowledge repositories. Organizing knowledge in theories is an obvious approach to cope with the growing number of definitions, theorems, and proofs. However, they are also a matter of subject on their own: developing a new piece of mathematics often relies on extending or combining already developed theories in this way reusing definitions as well as theorems. We believe that this aspect of theory development is crucial for mathematical knowledge management.
In this paper we investigate the facilities of the Mizar system concerning extending and combing theories based on structure and attribute definitions. As an example we consider the formation of rough concept analysis out of formal concept analysis and rough sets.

1 Introduction

The design and the construction of mathematical knowledge repositories is in the core of mathematical knowledge management. Computer-supported processing of mathematics such as knowledge retrieval, distribution over the Internet, or development of lecture material is highly driven by the way mathematical knowledge is represented and maintained. And last, but not least, the acceptance of mathematical knowledge management systems by mathematicians themselves also depends in essence on how the knowledge in such systems is represented and developed.

Mathematicians build up mathematical theories. However, developing a new theory does not (always) mean defining and proving all from scratch: each new mathematical theory uses, refines, combines, or extends previous ones. Definitions and theorems of the constituents are then used without any major reference. So for example field theory is a refinement of group theory, field theory itself is used in the theory of vector spaces, topology and group theory are combined in the theory of topological groups, and so on. Thus almost every theory relies on a number of (more) elementary theories. This phenomenon also occurs in more computer science related areas, see e.g. [Hur03] where the verification of a

* The work of the second author was partially supported by grant BW 5100-5-0147-4.

probabilistic primality test algorithm required proving quite a number of group and number theoretical theorems.

This rather dynamical aspect of theories should be explicitly addressed in mathematical knowledge management systems. Developing a theory for a knowledge repository does not only contribute by introducing new notions, proving some new theorems or applying new proof methods. In principle, each theory also serves as a starting point for further development in the sense that its notions, theorems, and proofs are used to construct other theories. We believe that mathematical knowledge repositories must take this into account by providing language features that enable and support theory development. Furthermore, these should reflect the way theories are dealt within everyday mathematical life.

In this paper we focus on theory development in the Mizar system [Miz04,RT01]. We formalized the beginnings of the theory of rough concept analysis [SD01,Ken96]. Rough concept analysis is a synthesis of rough sets and formal concept analysis. Rough sets [Paw82] approximate objects with respect to an indiscernibility relation. Formal concept analysis [Wil82] is an attribute-based approach for representation and analysis of data. The combination of both thus establishes a theory for representing and analyzing uncertain data.

Both rough sets and formal concept analysis have already been formalized by us in Mizar [Gra03,Sch98], however without any intent to use them in other theories or even to combine them. Hence, rough sets and formal concept analysis provide a good example to investigate the possibilities of Mizar to glue together two independently developed theories.

The plan of the paper is as follows. In the next section we briefly introduce rough sets and formal concept analysis and review their Mizar formalization as done in [Gra03] and [Sch98]. Section 3 describes the combination of these two theories into rough concept analysis. Mizar language features to do so are presented. In the last two sections we discuss the Mizar approach for combination of theories and compare it to other approaches in the literature. Conclusions for the design of mathematical knowledge repositories are drawn.

2 Rough Sets and Formal Concept Analysis

2.1 Approximation Spaces

Computers play a special role in the development of rough sets theory (RST for short). Even if the notion has strong mathematical flavour (although its founder, Zdzisław Pawlak is a computer scientist), it has been used from the beginning to approximate knowledge discovery and data mining process. Such an approach, called KDD-approach (for Knowledge Discovery in Databases) resulted in many practical applications, i.e. Rosetta or RSES, just to name the most important ones. We believe that KDD is also an important issue for the Mathematical Knowledge Management community.

Nobody however – as far as we know – used KDD tools for discovering the knowledge about rough sets themselves. One of the reasons probably is the lack of a proper, sufficiently big, computer-managed repository of RST knowledge. We tried to start with one of them [Gra03], and the results are rather promising, at least in the opinion of the authors and the RS community.

RST is also an interesting subject for machine-oriented formalization for some other reasons. Since the introductory paper [Paw82] back in 1982 numerous generalizations and improvements of the basic notions have been proposed. The community is rather young and fast-growing (the measure of acceptance ratio for presented long papers to the RSCTC 2004 Conference in Uppsala, Sweden was under 20%, there is also a subseries of LNCS, Springer, *Advances in Rough Sets*, devoted to the topic). On the one hand, machine-formalization of the results in such a discipline could be a challenge in itself for many reasons (The results correspond to different disciplines in mathematics and in computer science and offer many possible generalizations which could be discovered automatically etc.). On the other hand, one could reach a research frontier here relatively fast.

The theory of rough sets is a background for a methodology concerned with the analysis and modelling of classification and decision problems involving imprecise, vague, uncertain or incomplete information. As a result, it distinguishes classes of objects rather than the individuals and reflects in a formal way the very basic intuition that concept forming is finding out the similarities and the differences among objects. There are two simple models for RST in the literature. One of them is to have two sets of objects and attributes resp. forming an information system. They determine an external similarity relation clustering some objects together w.r.t. their attributes (properties). Another formal model is an approximation space, the backbone for a whole RST, which is a non-empty finite universe U and the indiscernibility relation R given on U.

Because the latter approach is more often used and methodologically simpler, we have chosen it. One of the key issues was also the possibility of further reusing. The concept of an information system can be also formalized as the descendant of the approximation space in a natural way. At the first sight, the underlying Mizar structure is `RelStr`, which has two fields: the `carrier` and the `InternalRel`, that is a binary relation of the `carrier`. The theory of relational structures has been developed and improved mainly during formalization of the *Compendium of Continuous Lattices*. While in this context `RelStr` was used with attributes `reflexive transitive` and `antisymmetric` to establish posets, we decided to reuse it in our own way. First, we defined two new attributes: `with_equivalence` and `with_tolerance` which state that the `InternalRel` of the underlying `RelStr` is an equivalence resp. a tolerance relation (where a tolerance relation is a total reflexive symmetric relation, see [RS90]). With such defined notions, the basic definitions are as follows:

```
definition
  mode Approximation_Space is with_equivalence non empty RelStr;
  mode Tolerance_Space is with_tolerance non empty RelStr;
end;
```

Formalized theories can be treated as objects (axioms, definitions, theorems) clustered by certain relations based on information flow. The more atomic the notions are, the more is their usefulness. Driven by this idea we tried to drop selected properties of the equivalence relations. Our first choice was transitivity – therefore the use of tolerance spaces – as it seemed to be less substantial than the other two. The generalization work went rather smoothly. As we discovered soon, similar investigations, but without any machine-motivations, were done by Järvinen [Jär01].

What is rather interesting, one of the referees among the RST community complained, that the presentation in [Gra03] is lacking its focus between approximation and tolerance spaces. Because every approximation space is in particular a tolerance space and this is obvious also for the Mizar type checker, all theorems true in more general situations need not (from the viewpoint of the Library Committee of the Association of Mizar Users – even must not) be repeated. Some theorems holding for tolerances however are simply untrue for equivalences, so the apparent lack of focus. For many mathematicians, the uniformity and clarity of the approach is still more important than its generality.

2.2 Rough Sets

As it sometimes happens among other theories (compare e.g. the construction of fuzzy sets), paradoxically the notion of a rough set is not the central point of RST as a whole. Rough sets are in fact classes of abstraction w.r.t. rough equality of sets and their formal treatment varies. Some authors (as Pawlak for instance) define a rough set as an underlying class of abstraction (as noted above), but many authors claim for simplicity that a rough set is an ordered pair containing the lower and the upper limit of fluctuation of the argument X. These two approaches are not equivalent, and we decided to define a rough set in the latter sense.

```
definition let A be Approximation_Space;
           let X be Subset of A;
  mode RoughSet of X means  :: ROUGHS_1:def 8
    it = [LAp X, UAp X];
end;
```

What should be mentioned here, there are so-called *modes* in the Mizar language which correspond with the notion of a type. To properly define a mode, one should only prove its existence. As it can be easily observed, because the above definiens determines a unique object for every subset X of a fixed approximation space A, this can be reformulated as a functor definition in the Mizar language.

If both approximations coincide, the notion collapses and the resulting set is *exact*, i.e. a set in the classical sense. Unfortunately, in the above mentioned approach, this is not the case. In [Gra03] we did not use this notion in fact, but we have chosen some other solution which describes rough sets more effectively.

Regardless what approach we are claiming, the key notion of RST is the notion of an *approximation.* A lower approximation of a set X consists of objects which are surely (w.r.t. indiscernibility relation of A) in X. Similarly, the upper one extends the lower for the objects which are possibly in X (see [Gra03] for detailed explanations).

```
definition let A be Tolerance_Space;
           let X be Subset of A;
  func LAp X -> Subset of A equals  :: ROUGHS_1:def 4
  { x where x is Element of A : Class (the InternalRel of A, x) c= X };
  func UAp X -> Subset of A equals  :: ROUGHS_1:def 5
  {x where x is Element of A : Class(the InternalRel of A, x) meets X};
end;
```

One of the most powerful Mizar linguistic constructions are *attributes* (which are constructed by adjectives). As we found modes to be useless within RST, we introduced the attribute `rough` in the following way to describe sets X with non-empty approximation boundary `BndAp` X (the set-theoretical difference of the upper and the lower approximations of a given set X).

```
definition let A be Tolerance_Space;
           let X be Subset of A;
  attr X is rough means  :: ROUGHS_1:def 7
    BndAp X <> {};
end;
```

If both the upper and lower approximation of a set X coincide, `BndAp` X equals $\emptyset$ and X is a set in the usual sense. In Mizar script, we introduced an adjective `exact` as an antonym to the above. The apparatus of adjectives has proved its value especially when merging theories together.

2.3 Formal Concept Analysis

Formal context analysis (FCA for short) has been introduced by Wille [Wil82] as a formal tool for the representation and analysis of data. The main idea is to consider not only data objects, but to take into account properties (attributes) of the objects also. This leads to the notion of a *concept* which is a pair of a set of objects and a set of properties. In a concept all objects possess all the properties of the concept and vice versa. Thus the building blocks in FCA are given by both objects and their properties following the idea that we distinguish sets of objects by a common set of properties.

In the framework of FCA the set of all concepts (for given sets of objects and properties) constitute a complete lattice. Thus based on the lattice structure the given data – that is its concepts and concept hierarchies – can be computed, visualized, and analyzed. In the area of software engineering FCA has been successfully used to build intelligent search tools as well as to analyze and reorganize the structure of software modules and software libraries.

In the literature a number of extensions of the original approach can be found. So, for example, multi-valued concept analysis where the value of features is not restricted to two values (true and false). Also more involved models have been proposed taking into account additional aspects of knowledge representation such as different sources of data or the inclusion of rule-based knowledge in the form of ontologies.

Being basically an application of lattice theory FCA is a well-suited topic for machine-oriented formalization. On the one hand it allows to investigate the possibilities of reusing an already formalized lattice theory. On the other hand it can be the starting point for the formalization of the extensions mentioned above. In the following we briefly present the Mizar formalization of the basic FCA notions necessary for the next section.

The starting point is a formal context giving the objects and attributes of concern. Formally such a context consists of two sets of objects O and attributes A, respectively. Objects and attributes are connected by an incidence relation $I \subseteq O \times A$. The intension is that object $o \in O$ has property $a \in A$ if and only if $(o, a) \in I$. In Mizar [Sch98] this has been modelled by the following structure definitions.

```
definition
  struct 2-sorted (# Objects, Attributes -> set #);
end;

definition
  struct (2-sorted) ContextStr
    (# Objects, Attributes -> set,
       Information -> Relation of the Objects,the Attributes #);
end;
```

Now a formal context is a non-empty `ContextStr`. To define formal concepts in a given formal context C two derivation operators `ObjectDerivation(C)` and `AttributeDerivation(C)` are used. For a set O of objects (A of attributes) the derived set consists of all attributes a (objects o) such that $(o, a) \in I$ for all $o \in O$ (for all $a \in A$). The Mizar definition of these operators is straightforward and omitted here.

A formal concept FC is a pair (O, A) where O and A respect the derivation operators: the derivation of O contains exactly the attributes of A, and vice versa. O is called the extent of FC, A the intent of FC. In Mizar this gives rise to a structure introducing the `extent` and the `intent` and an attribute `concept-like`.

```
definition let C be 2-sorted;
  struct ConceptStr over C
    (# Extent -> Subset of the Objects of C,
       Intent -> Subset of the Attributes of C #);
end;
```

```
definition let C be FormalContext;
           let CP be ConceptStr over C;
  attr CP is concept-like means  :: CONLAT_1:def 13
    (ObjectDerivation(C)).(the Extent of CP) = the Intent of CP &
    (AttributeDerivation(C)).(the Intent of CP) = the Extent of CP;
end;
```

```
definition let C be FormalContext;
  mode FormalConcept of C is concept-like non empty ConceptStr over C;
end;
```

Formal concepts over a given formal context can be easily ordered: a formal concept FC_1 is more specialized (and less general) than a formal concept FC_2 iff the extent of FC_1 is included in the extent of FC_2 (or equivalently iff the intent of FC_2 is included in the intent of FC_1). With respect to this order the set of all concepts over a given formal context C forms a complete lattice, the concept lattice of C.

```
theorem
  for C being FormalContext holds ConceptLattice(C) is complete Lattice;
```

This theorem, among others, has been proven in [Sch98]. The formalization of FCA in Mizar went rather smoothly, the main reason being that lattice theory has already been well developed. Given objects, attributes and an incidence relation between them, this data can now be analyzed by inspecting the structure of the (concept) lattice; see [Wil82,GW98] for more details and techniques of formal concept analysis.

3 Rough Concept Analysis

In this section we present issues concerning the merging of concrete theories in the Mizar system. Since we want to keep this presentation possibly self-contained and we assume basic knowledge of Mizar syntax (which is very close to the language used by mathematicians), we will illustrate them by living examples from Rough Concept Analysis and skipping most technical details. For details of the Mizar type system, see [Ban03].

Below we enumerate ten features that proved their usability in merging the theories of RST and FCA. Though they are Mizar specific we claim that any mathematical knowledge should support the general principles of these features in one or another form in order to make theory development more feasible. We like to mention that in the course of FCA formalization the formal apparatus yet existing in the Mizar Mathematical Library also had to be improved and cleaned up.

Data structure. A basic structure for the merged theory should inherit fields from its ancestors, which would be hard to implement if structures were implemented as ordered tuples (multiple copies of the same selector, inadequate ordering of fields in the result). The more feasible realization is by partial functions rather, and that is the way Mizar structures work.

```
definition
  struct (ContextStr, RelStr) RoughContextStr
    (# carrier, carrier2 -> set,
       Information -> Relation of the carrier, the carrier2,
       InternalRel -> Relation of the carrier #);
end;
```

The maximal number of fields in a single structure in the Mizar library is ten (as of a Cartesian category). The choice of having a long structure with many specialized inherited fields is what actually depends on the author. On the one hand an ordering can be defined in a natural way as an external predicate, e.g. on concepts. On the other hand, using the internal field of a structure one can benefit from the notions and theorems about relational structures. It is hard to formulate quantitative criteria in this subject, but it seems that six is a good upper approximation for a number of structure fields.

Homogeneous structure hierarchy. In Mizar the same names of fields are necessary to merge structures (the problems with multiple carriers), some automatizing could be made, though. The same names of operations or attributes will result in serious troubles if one has them together with different meanings assumed originally. As it can be observed from the previously cited example, the first two selectors have the names `carrier` and `carrier2` which is incompatible with the names of ascendant structures (`Objects` and `Attributes` occurring in `ContexStr` vs. `carrier` in `RelStr`). Since there are no synonyms for selectors in Mizar, a revision for `ContextStr` had to be made. This policy however is highly unlikely and Mizar developers should consider a language extension.

Description	Quantity	% of total
descendants of 1-sorted	72	79
prefixed by 1-sorted	23	25
initial	15	16
standalone	12	13
multiprefixed	22	24
structures total	91	100
articles using structures	533	64
articles total	834	100

Table 1. Statistical data about structures in MML

Apart from its visualization (see 'Net of structures in MML' (MML stands for Mizar Mathematical Library) section at [Miz04]), we gather basic data about structures in Table 1. As it can be concluded, the clustering of structures is good. Every fourth structure inherits its fields from at least two others, so the background for theory merging is comparatively well prepared. As a future work for the Library Committee, the number of initial (i.e. without parents) structures should be decreased.

Forgetful functors. The user would have a chance to use an original part of merged object with a type of its parent, i.e. the back-translation should be provided where possible. It is provided in Mizar by the construction

`the` *Structure-symbol* `of` *Variable*

In this manner, if a `RoughContextStr` X is given, `the RelStr of X` will result in the rough part of the structure X.

Strict objects. Sometimes it is useful to deal only with the objects with the type of source theories even if we work within a target theory.

`strict` *Structure-Symbol*

yields an object without any additional fields than those in the input structure, e.g. `strict ContextStr` denotes the structure of a formal context without roughness.

Inheritance of properties. It is natural to have notions formulated as general as possible, especially in a large repository, when unnecessary assumptions are inherited in some sense. The care is advised here, because if the definition of the lower approximation were introduced as follows:

```
definition let A be strict Tolerance_Space;
           let X be Subset of A;
  func LAp X -> Subset of A equals
    {x where x is Element of A : Class(the InternalRel of A,x) c= X};
end;
```

i.e. with tolerance space without any additional fields as a locus, this notion might not be used within RCA. There is software available in the Mizar system detecting some unnecessary assumptions, but not those searching for a more general type which may be used. Of course, the above functor has been defined in [Gra03] without adjective `strict`.

Free extensions. As it often happens, an extension of the theory to another need not be unique. There are at least three different methods of adding roughness to formal concepts [Ken96,SD01]. The question which approach to choose depends on the author. The notion of a free structure in a class of descendant type conservative w.r.t. the original object is very useful.

```
definition let C be ContextStr;
  mode RoughExtension of C -> RoughContextStr means
    the ContextStr of it = the ContextStr of C;
end;
```

Now, if C is a given context, we can introduce roughness in many different ways by adjectives.

Interferences. Up to now, we described only mechanisms of independent inheritance of notions. Within the merged theory it is necessary to define connections between its source ingredients. Here the attributes describing mutual interferences between selectors from originally disjoint theories proved their enormous value. They may determine the set of properties of a free extension.

```
definition let C be RoughFormalContext;
  attr C is naturally_ordered means
    for x, y being Element of C holds
    [x,y] in the InternalRel of C iff
      (ObjectDerivation C).{x} = (ObjectDerivation C).{y};
end;
```

Since the relation from the definiens above is an equivalence relation on the objects of C and hence determines a partition of the set of objects of C into the so-called *elementary sets*, it is a constructor of an approximation space induced by given formal context.

Inherited theorems and clusters. Theory merging makes no sense, if proving the same theorem would be necessary within both source and target theory. Since a new Mizar type `RoughFormalContext` is defined analogously to the notion of `FormalContext`, as `non quasi-empty RoughContextStr` (compare subsection 2.3), the following Fundamental Theorem of RCA is justified only by the Fundamental Theorem of FCA. Even more, clusters providing automatic acceptance of the original theorems do it analogously within target theory. That is also a workplace for clusters `rough` and `exact` mentioned in Subsection 2.2.

```
for C being RoughFormalContext holds
  ConceptLattice(C) is complete Lattice by CONLAT_1:48;
```

Uniform object naming. We should work out a comfortable and uniform naming system for attributes. E.g. it is a rule that the structure prefixed by 1-sorted is named `non empty` provided its carrier is non empty. But how it should be named to describe the 'second carrier' to be non empty? This concerns multi-sorted structures in general. We want to find a reasonable compromise between technical naming and that close to a natural language, but the reasonable length of a name is also an important criterion. So between `non empty2` and `with_non_empty_carrier2` we decided to use `quasi-empty`.

Synonyms for objects. Although uniform name spaces improve especially the searching process, sometimes having the same notation for different notions is a serious drawback. Lattice theory can bring more appropriate examples: on the one hand we have binary suprema and infima operations in a properly defined lattice, on the other hand those in a relational structure induced by the ordering. If we merge both, the problems with identification occur, so it is reasonable to introduce a new notational synonym for one of them, e.g.

```
notation let R be RelStr,
             x, y be Element of R;
  synonym x "|_|" y for x "\/" y;
end;
```

(Here "\/" stands for an original binary supremum in lattices).

4 Mathematical Knowledge Repositories

Mathematical knowledge management aims at providing mathematics with the help of computers. One major point in doing so is to build repositories holding the knowledge. We believe that mathematical knowledge repositories should be more than simple databases, that is the knowledge included should be verified by a prover or checker in order to avoid the inclusion of untrue knowledge. However, even then revisions may become necessary. We have seen this in the course of formalizing rough concept analysis, where renaming of the selectors of structure `ContextStr` was necessary in order to fully benefit from the inheritance mechanisms in Mizar. Thus the organization and maintenance of mathematical knowledge repositories is, and will stay, an ongoing process. In the following we report on the latest developments concerning MML, discuss the evolution of mathematical repositories and briefly review other approaches to theory building from the literature.

4.1 Mizar Mathematical Library

The Mizar Mathematical Library [Miz04] is the (still evolving) result of a long term project that aims at developing both a comprehensive library of mathematical knowledge and a formal language – also called Mizar – for doing so. So far MML just collects articles written in the Mizar language. At the time of writing there are 834 articles with 36847 theorems/lemmas and 7093 definitions included, written by more than 150 authors.

Recently the development of the Encyclopedia of Mathematics in Mizar (EMM for short) has been started. Here definitions and theorems of MML are extracted semi-automatically into new articles with monographical character. This obviously becomes necessary having a larger number of authors who introduce definitions and theorems whenever these are convenient for the author's goals – and not when it seems reasonable to introduce them in order to get a well-developed repository. So far there are five EMM articles (`XBOOLE_0`, `XBOOLE_1`, `XREAL_0`, `XCMPLX_0`, and `XCMPLX_1`) including basic facts on boolean operators, real and complex numbers. Thus, in some sense, theories are built in EMM by collecting knowledge (belonging to the same topic) spread over the whole library.

In parallel, work on the environment directive `requirements` continues which strengthens the Mizar checker in the following sense: requirement files contain statements that become obvious for the Mizar checker if the file is included in the environment. So far properties of real and complex numbers as well as of boolean operators and of subsets has been addressed.

In the course of developing EMM and the requirements directive we propose to also include theories based on structures and attributes such as for example groups, fields, or even RST or FCA. As a consequence these theories could be easier imported into new articles improving both the organization and reusing facilities of MML.

4.2 Evolution of a Repository

Though we cope mainly with the question of computer-supported mathematical theory development in this paper, machine-managed knowledge retrieval from a repository of mathematical facts is also an important issue, especially if one considers KDD-based methods usage.

The larger the database is, the easier linkages and connections can be studied. Even if MML is known as the largest library of formalized mathematics, it has a number of disadvantages. C-CoRN [Wie04], the Constructive Coq Repository at Nijmegen, uses experience gathered when building MML. The number of people involved in this project is rather small (about 15 in total) as their library is centralized as a rule. C-CoRN grew out as a side-effect of the Fundamental Theorem of Algebra project. It is not an objection in any sense, but some obstacles might be anticipated in this way and carefully be discussed. The data structure may also profit from the uniform manner of its design. Searching for analogies, e.g. between fuzzy and rough sets, can be more successful if both are coded in similar manner or if they can be lifted to a common formal platform.

In this paper we address only issues of what Trybulec and Rudnicki [RT01] classify as abstract mathematics. In classical mathematics theories are hardly merged, but rather inherited. It is one of Trybulec's recent suggestions to cut MML into classical (not using the notion of a structure) and abstract mathematics to study interconnections between theories more carefully. There is also the opinion, that this will make the library more coherent, where by a 'coherence' we mean, similarly to [Wie04], that results about a specific theory should be grouped together and theories extending others should be built on the top of them.

What we are going to do with MML is to make it better organized but we have to watch for authors' rights as well. They sign a copyright form when submitting to MML and the Library Committee of the Association of Mizar Users can revise their work. However, the copyright issues should be discussed by the MKM community since the importance of electronical repositories is growing remarkably.

It is hard to imagine a living system without any changes anyway. If the interfaces evolve, repositories can follow their development and vice versa, by exploring knowledge stored in libraries, new techniques can be worked out. In MML CVS service which is publicly available (by a usual Concurrent Version System engine with web-browsing facility) official versions of MML are archived to document revision stages. Since its initial version in May 2002, over 140 versions were archived (as of May 2004). It shows rather dynamic growth of the repository, especially if one takes into account that not all intermediate stages of the library were stored (some of the versions of the system consisted only of the new build of the executables etc.). The policy of the Library Committee of the Association of Mizar Users is that MML is archived in CVS as a whole because of the many interdependencies. Just for the record how tight this structure is, let us note here that there are over 400 thousand cross-citations (for definitions and theorems, internal references are not counted) between Mizar articles.

4.3 Other Repositories

The IMPS system [FGT93] provides theory development in forms of the little theories approach [FGT92]. Here theories are developed with respect to a particular set of axioms which can be compared to using a set of attributes in Mizar. To use a theorem in another context a theory interpretation is constructed. This usually includes a number of obligations ensuring basically that axioms of the old theory are mapped to theorems in the new one. Combining theories therefore can be done by establishing the axioms of the new theory and importing theorems via theory interpretations.

A similar approach can be observed in PVS [OS01], via the mapping mechanism it is easy to specify a general theory and have it stand for a number of instances (the latter correspond to an *aggregate* in Mizar). Theory declarations allow theories to be encapsulated and instantiated copies of the implicitly imported theory are generated. In Isabelle/Isar [NPW02] there are mechanisms of merging both theories and locales. The latter modules correspond to a context rather which is a narrower notion than we dealt in this paper.

In Theorema [Buc01] theories can be constructed by using a special language construct. This allows to explicitly state which definitions, lemmas, and theorems are part of the theory. Thus every user can construct his own theories. These theories are then used as attachments to proof attempts in this way reusing the knowledge stored in a theory. A theory may include theories again so that hierarchies of theories can be built.

Tecton [MS02,LMSS99] is a specification language developed mainly for the description of requirements for generic algorithms. It does include language constructs to formulate lemmas and theorems, but unfortunately no means to prove these correct. However, theories can be combined by gluing together already existing theories using so-called refinement. This basically means that the axioms of the old theories are imported into the new one, in this way ensuring that theorems of used theories still hold. Furthermore, in doing so carriers and operators can be renamed which makes the language more flexible and user-friendly.

5 Conclusion

The aim of mathematical knowledge repositories is twofold: On the one hand a repository has to provide language and proof constructs to represent mathematical objects and proofs in a convenient way. On the other hand, even more important for mathematical knowledge management, the knowledge stored in a repository must be kept manageable in the sense that both retrieving and extending it is supported. Building, refining, and extending theories is an obvious approach to organize knowledge in mathematical repositories and therefore crucial for the development of mathematical knowledge repositories; in particular the flexibility of theories as known from everyday mathematical life has to be addressed. We believe that constructing – and hence refining and extending – theories using structures and attributes as presented in this paper is a step into this direction.

References

[Ban03] G. Bancerek, On the Structure of Mizar Types; in: H. Geuvers and F. Kamareddine (eds.), Proc. of MLC 2003, ENTCS 85(7), 2003.

[Buc01] B. Buchberger, Mathematical Knowledge Management in Theorema; in: B. Buchberger, O. Caprotti (eds.), Proc. of MKM 2001, Linz, Austria, 2001.

[FGT92] W. Farmer, J. Guttman, and F. Thayer, Little Theories; in: D. Kapur (ed.), Automated Deduction – CADE-11, LNCS 607, pp. 567–581, 1992.

[FGT93] W. Farmer, J. Guttman, and F. Thayer, IMPS – An Interactive Mathematical Proof System; J. of Automated Reasoning 11, pp. 213–248, 1993.

[Gra03] A. Grabowski, Basic Properties of Rough Sets and Rough Membership Function; to appear in Formalized Mathematics, 2004, available from [Miz04].

[GW98] B. Ganter and R. Wille, Formal Concept Analysis – Mathematical Foundations; Springer Verlag, 1998.

[Hur03] J. Hurd, Verification of the Miller-Rabin Probabilistic Primality Test; Journal of Logic and Algebraic Programming, 56, pp. 3-21, 2003.

[Jär01] J. Järvinen, Approximations and Rough Sets Based on Tolerances; in: W. Ziarko, Y. Yao (eds.), Proc. of RSCTC 2000, LNAI 2005, pp. 182–189, 2001.

[Ken96] R.E. Kent, Rough Concept Analysis: A Synthesis of Rough Sets and Formal Concept Analysis; Fundamenta Informaticae 27, pp. 169–181, 1996.

[LMSS99] R. Loos, D. Musser, S. Schupp, and C. Schwarzweller, The Tecton Concept Library; Technical Report WSI 99-2, Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, 1999.

[Miz04] The Mizar Home Page, `http://mizar.org`.

[MS02] D. Musser and Z. Shao, The Tecton Concept Description Language (Revised Version); Technical Report 02-2, Rensselaer Polytechnic Institute, 2002.

[NPW02] T. Nipkow, L. Paulson, and M. Wenzel, Isabelle/HOL – A Proof Assistant for Higher-Order Logic; LNCS 2283, 2002.

[OS01] S. Owre and N. Shankar, Theory Interpretations in PVS; Tech. Report, NASA/CR-2001-211024, 2001.

[Paw82] Z. Pawlak, Rough Sets; International Journal of Information and Computer Science, 11, pp. 341–356, 1982.

[RS90] K. Raczkowski and P. Sadowski, Equivalence Relations and Classes of Abstraction; Formalized Mathematics, 1(3), pp. 441–444, 1990, available in JFM from [Miz04].

[RT01] P. Rudnicki and A. Trybulec, Mathematical Knowledge Management in Mizar; in: B. Buchberger, O. Caprotti (eds.), Proc. of MKM 2001, Linz, Austria, 2001.

[RT03] P. Rudnicki and A. Trybulec, On the Integrity of a Repository of Formalized Mathematics; in: A. Asperti, B. Buchberger, and J. Davenport (eds.), Proc. of MKM 20203, LNCS 2594, pp. 162–174, 2003.

[SD01] J. Saquer and J.S. Deogun, Concept Approximations Based on Rough Sets and Similarity Measures; Intl. J. on Appl. of Mathematics in Computer Science, 11(3), pp. 655–674, 2001.

[Sch98] C. Schwarzweller, Introduction to Concept Lattices; Formalized Mathematics, 7(2), pp. 233–242, 1998, available in JFM from [Miz04].

[Wie04] L. Cruz-Filipe, H. Geuvers, and F. Wiedijk, C-CoRN, the Constructive Coq Repository at Nijmegen; `http://www.cs.kun.nl/~freek/notes/`.

[Wil82] R. Wille, Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts; in: I. Rival (ed.), Ordered Sets, Reidel, Dordrecht-Boston, 1982.