Informalising Formal Mathematics:
Searching the Mizar Library with Latent Semantics

Paul Cairns

UCL Interaction Centre
University College London
London WC1E 7DP, UK
p.cairns@ucl.ac.uk

Abstract. Finding required information in a library of mathematics can be problematic, just as in any other library. However, so far, there are no strong search methods based on the semantics of formal mathematics. This paper describes a new approach based on latent semantic indexing (LSI). Using this, the semantics of terms need not be explicitly defined but is indirectly inferred from a body of documents in which the terms occur. The Mizar library is used as it is a substantial resource of formal mathematics. The system described in the paper adapts Mizar articles to produce an appropriate body of documents that can be used by LSI. Preliminary tests suggest that this approach is able to provide a useful mechanism for the search and retrieval of formal mathematics.

1 Searching for Mathematics

Mathematical knowledge management, since its inception, has had two key themes: the organisation of mathematical knowledge; and the successful retrieval of mathematical knowledge. In this paper, we consider the retrieval task when the mathematical knowledge has already been organised and standardised, namely retrieving knowledge from the Mizar Mathematical Library (MML) [20].

Bancerek and Rudnicki [2] have already considered information retrieval in the MML. Their approach rightly criticised the poor quality of keyword and `grep`-based approaches. However, they do have a hypertext method of presentation that allows a person to browse the library and easily find definitions of terms by clicking on the appropriate terms in the presentation. Additionally, they developed a query language that allowed more sophisticated querying based on the structural and semantic properties of Mizar articles. This was recognised as only a first step towards full semantic search and as such may only be useful in this form to expert Mizar authors.

Other search techniques have used type isomorphisms [9], metadata [17] or a combination of metadata and reasoning [5]. With type isomorphisms or reasoning, the search engine is effectively doing some proving, albeit tailored to the particular task. As fully automated proving is still a significant research topic, this suggests that there is a limit to how far these approaches could extend. With metadata, there is of course the possibility to make a search engine very effective but then there is the overhead of who must prepare the metadata. Authors of webpages are already very poor at adding metadata and there is no suggestion that authors of mathematics are likely to be any better.

Latent semantic indexing avoids these issues entirely as the semantics of the documents to which it is applied are never explictly represented. Instead, the semantics of terms are implicitly inferred from contexts in which they occur even if, as in the case of mathematics, the contexts are sometimes the definitions of the terms.

Related to this is the notion that mathematics itself is not a formal language. Mathematics is in principle formalisable but, in practice, formalisation is hardly ever done (which is why the MML represents a valuable contribution to mathematics). I would argue that mathematics, like all human languages, functions at a level of discourse [21] rather than at a level of logic. Search that works at the level of mathematical discourse is more likely to fit better with the needs of working mathemticians.

This paper therefore treats formal mathematics as a representation of mathematical language in general. Mizar is particularly strong in providing a wide range of constructs for mathematical concepts [24] that allow its formal proofs to be more like the less formal proofs found in common mathematical texts. In this sense, the MML is actually a reliable representation of the more usual, informal mathematical language. A well-known, and indeed successful, technique in information retrieval is Latent Semantic Indexing (LSI) [6]. Rather than defining semantics through some explicit ontology, the semantics of terms are defined through their co-occurence in a large document set. This is discussed in detail in the next section but the main point is that the semantics of terms are latent in the substantial body of documents to which LSI is applied. The MML is able to provide exactly such a large set of documents from which semantics can be extracted.

After describing the details of applying LSI to the MML, I give some early results. Despite the counter-intuitive idea of ignoring the formal semantics inherent in formal mathematics, these results actually show some promising indications. The current implementation does not make full use of the MML for reasons discussed elsewhere [8] and so there are natural areas for further work and refinement of these results.

2 Latent Semantic Indexing

Latent semantic indexing is a method developed for doing information retrieval from natural language documents based on the general theory of latent semantic analysis (LSA)[15]. LSA has been used in a variety of text retrieval and analysis tasks including the various Text Retrieval Conferences (TREC) [23] competitions, selecting educational text and the automated assessment of essays [16].

The major task of any information retrieval system of this sort is to define the semantics of the documents and the terms in those documents. The semantics can then be used to reliably answer queries based on the documents. For example, a person seeking information using a query "wars in the Middle East" would probably be satisfied with a document about "conflicts in Iraq." This is because we recognise that "conflict" is a synonym for "war" and "Iraq" is in the "Middle East". Many text retrieval systems rely on some externally defined ontology that would make the semantic relationships between terms explicit. Thus, when given the example query the meaning of the query would be looked up and documents with a similar meaning would be returned.

With LSI, there is no such external ontology. Instead, the meaning of terms is held to be latent in the documents in which those terms appear. That is, a word gains its meaning from the many contexts in which it appears. Conversely, a document gains its meaning from the many words that occur within it. Clearly then for LSI to produce good semantics, it needs a substantial body of documents in which all the relevant terms occur in many and varied contexts. The advantage is that no work needs to be done to define an ontology for querying the documents.

For this reason, LSI seemed an appropriate tool to use and the MML the appropriate context in which to use it. Formal mathematics, in one sense, is an entire ontology of mathematical terms. However, as yet, it has not been extensively used to provide effective information retrieval. Through LSI though, the MML also represents an extensive set of documents that latently define the meanings of a huge number of mathematical terms. In addition, the rich language of Mizar reflects some of the richness of more traditional, human-oriented mathematics. LSI could tap into this to provide an alternative ontology without any extra work.

The other advantage of LSI is that its mechanism relies on some elegant mathematics that has been implemented in the GTP package [10]. GTP is written in Java. The mathematics is briefly described here to give a flavour of how it works. Followed by some details of the implementation specific to GTP.

2.1 The Mathematics of LSI

The occurrence of terms in documents can be simply captured in a rectangular matrix Δ where :

$$\Delta_{ij} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ term appears in the } j^{\text{th}}\text{document} \\ 0 & \text{otherwise} \end{cases}$$

Keyword search can then be implemented by converting a query into row vector $\boldsymbol{t}$ where t_i is 1 if the i^{th} term is in the query and 0 otherwise. Taking $\boldsymbol{d} = \boldsymbol{t}.\Delta$, $\boldsymbol{d}$ is row vector where d_j is the number of query terms occurring in the j^{th} document. In particular, if n is the number of non-zero entries in $\boldsymbol{t}$ (that is, the number of distinct terms in the query) then $d_j = n$ means that the j^{th} document contains all of the query terms.

Using singular value decomposition (SVD), it is possible to find two orthogonal matrices U and V and a diagonal matrix Σ_D such that:

$$\Delta = U \Sigma_D V$$

From this equation, for a term vector $\boldsymbol{t}$, keyword search would give:

$$\boldsymbol{d} = \boldsymbol{t}.U\Sigma_D V$$

$$\boldsymbol{d}.V^{-1} = \boldsymbol{t}.U\Sigma_D$$

However, we require more than keyword search. Instead, for a given query $\boldsymbol{t}$, and any set of documents represented by $\boldsymbol{d}$ we consider the similarity between the two vectors $\boldsymbol{t'} = \boldsymbol{t}.U\Sigma_D$ and $\boldsymbol{d'} = \boldsymbol{d}.V^{-1}$. The similarity, s, between vectors $\boldsymbol{t'}$ and $\boldsymbol{d'}$ is taken to be the cosine of the angle between them, that is:

$$s = \frac{\boldsymbol{t'}.\boldsymbol{d'}}{||\boldsymbol{t'}||.||\boldsymbol{d'}||}$$

One way to view this is that, under the transformations given, terms and documents occupy a common space based on the semantics of the terms. Now to perform a query, the term vector $\boldsymbol{t}$ is generated as before. Each document is represented by the vector $\boldsymbol{d_i}$ where $d_{ij} = 1$ if $i = j$ and 0 otherwise. The query is then compared to all the documents in the term/document space by finding the similarity between $\boldsymbol{t'}$ and each of the vectors $\boldsymbol{d'_i}$. The most similar documents are returned as the results of the query.

2.2 The GTP Package

Given a set of documents as either separate files or documents delimited within a single file, GTP automatically extracts terms from the documents and constructs Δ. By default, a term in GTP is any alphanumeric string of characters. Also, because GTP was constructed with natural language in mind, it consults a file of common words, such as "the" and "of", and does not include a common word as a term. Mizar (like mathematics), however, has a radically different linguistic structure and punctuation from natural language. Instead, this section of GTP was replaced. Common words were simply Mizar keywords, though arguably they could be included as terms, together with ';' and ',' when used purely as separators. This means that terms could be any string of characters excluding whitespace and thus encompasses all of the constructs defined in Mizar articles.

Having defined the terms, the GTP package automatically constructs the appropriate Δ and performs the singular value decompostion. This is the main output of the first stage of GTP.

The querying process is run as a separate stage of GTP using the matrices generated from the first stage. As might be expected, the comparison process can be rather lengthy so there are some approximations that can be made to speed up matters. To facilitate discussion, in the sequel, the diagonal values of Σ_D are referred to as the eigenvalues of Δ.

The first approximation is to omit the scaling by the eigenvalues by taking $\boldsymbol{t'} = \boldsymbol{t}.U$. However, it produced uniformly poor results so this approximation was not used. Secondly, each eigenvalue corresponds to a factor that can be used to compare queries and documents. When the eigenvalues are very small, they can effectively be omitted from the calculation and hence speed it up. In GTP, setting the number of factors to 15 corresponds to calculating similarity based on the fifteen largest eigenvalues. The number of factors used in the test was varied as will be discussed in the results section.

In summary, to a large extent, LSI is treated here as a black box method for querying documents. The GTP implementation is untouched except to replace the code that identifies terms in the MML.

3 Applying LSI to Mizar

As most of the computational effort is done by GTP, the two main issues for applying LSI to Mizar are:

1. What constitutes a document?
2. How should queries be performed?

All coding was done in Java 1.4 and the Java version of the GTP package was used. Version 3.33.722 of the MML was used to generate the documents. More specifically, rather than use the full Mizar articles with proofs, only the abstracts were used in this implementation.

3.1 Making the MML into documents

The decision as to what constitutes a document is crucial because this is how LSI will capture the semantics of Mizar terms. At one level, each Mizar article could be considered to be a document. However, these are substantial pieces of work with many references to a large number of terms. It was felt that with many terms in each article and a reasonable overlap between articles, using articles as documents would not distinguish sufficiently between the meanings of terms. Also, when retrieving a document, a user would probably like something more focused than a whole Mizar article. The natural choice seemed to be to divide each article into smaller parts. These parts should be both meaningful as a stand-alone entity (so not single lines of proofs) and the kind of thing that users would like to retrieve. Accordingly, a document was decided to be one of the following:

1. Theorem
2. Definition
3. Proof

As parsing Mizar articles can be problematic [8], the first stage seemed to be a proof of principle on using only theorems as documents and then only those that were in abstracts.

Simply using a theorem statement directly from a Mizar abstract was also not likely to provide an appropriate document. For example, in the following theorem [3]:

```
theorem :: ORDINAL2:1
  A c= B iff succ A c= succ B;
```

A and B would be identified as terms but without a doubt they are also used as terms in a huge number of other theorems with considerably different meanings. This would greatly confuse the semantics of the library. Also, users are not likely to find theorems about A and B interesting, but rather theorems, like this one, about successors of `Ordinals`.

In this case, A and B have already been defined as `Ordinals` in a reserve statement. A more meaningful document would be:

```
theorem :: ORDINAL2:1
  for A, B being Ordinals holds A c= B iff succ A c= succ B;
```

However, A and B still occur as terms in this document and could add noise to the LSI calculations. Instead, the document is made by replacing each occurrence of a variable by its type:

```
theorem :: ORDINAL2:1
   Ordinal c= Ordinal iff succ Ordinal c= succ Ordinal;
```

Thus Ordinals, successors and subsets all occur in the same context which LSI would use to say that the meaning of `Ordinals` is somehow defined by that of `succ` and `c=`. In addition, this document contains only terms and keywords and these latter are currently ignored.

Thus, each document is a theorem where all of the known variables have been replaced by their types. There is no further reduction of types to simpler types, for example, to say that `Ordinal` is an `epsilon-transitive epsilon-connected set`. This is because the aim is to look at the linguistic structure of the mathematics not the logical.

The process for producing the documents is given in Figure 1 and works as follows. The header of each Mizar abstract is used to generate the vocabulary used in each abstract. This means that the abstract can be reliably parsed into chunks that are either theorems, reserve statements or definitions. The reserve statements are parsed to produce of a list of variables that have a known type. The definitions are also parsed to check exactly which definition of various terms are actually in use at a given point in an article. As a by-product of parsing definitions, a catalogue is produced that states exactly where each item of vocabulary is defined together with any other useful information such as the number of pre- and post-arguments, where applicable.

The catalogue and known variables provide a context for a particular theorem. The theorem can be parsed using the context and transformed into the corresponding document as required.

The process progresses through the chunks, successively updating the context as it goes. So for instance, if a reserve statement redefines a variable as having a new type, this is carried through to the subsequent theorems and hence documents.

On the whole, the process works well. There are still some abstracts that cannot be parsed into chunks because of the use of ';' in the terms. However, this is largely due to the limitation of the current parsing technology being used,

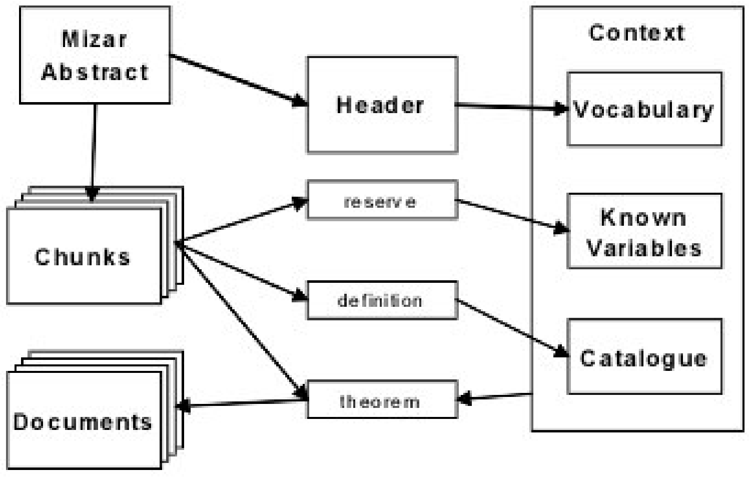

Fig. 1. Generating documents from theorems in an abstract

namely JavaCC [12]. It is hoped to replace this with a more versatile parser in the near future. The final document set is made up of 31251 documents that contain 3181 terms. GTP transforms this data and captures it in 106 factors, that is, 106 eigenvectors with non-zero eigenvalues, though it is not documented to what degree of accuracy GTP distinguishes a nearly zero eigenvalue from actually zero.

3.2 Performing Queries

If queries are to be useful to a user of the MML, the query engine most likely needs to be integrated into some larger system, such as the emacs mode for Mizar. However, at this stage, it is not clear what would constitute a useful query. This can be seen by considering the process that is done to convert a theorem in Mizar into a document for LSI. To what extent should the user do that conversion process to formulate a query? Or conversely, how much of formulating a query can be generated from a user's context of querying?

These are difficult questions in any system where complex work such as writing Mizar articles is being performed [18]. This is compounded by the fact that the full capabilities of LSI-based search are unknown.

Rather than commit at this early stage to a visual interface that may be both inappropriate and lengthy to develop, the system implemented uses a command-line approach to querying and all of the preparation of queries is done by hand.

Thus, a query is hand-written or adapted from a document based on a theorem. For example, a typical query might look like:

```
theorem set c= set implies Cl ( set /\ set ) c= Cl ( set /\ set )
```

The results from the basic GTP are the indexes of documents in the whole collection together with the strength of similarity between the document and the query. For example, GTP might produce results like:

```
4134    0.99987674
8609    0.99812323
8610    0.99809912
25838   0.99731101
```

The first number of each line corresponds to the index of a document. The second number is the strength of match with the query. The resulting documents are looked up in an index prepared at the same time as the documents were produced from the theorems. Thus, the basic results can be expanded to:

```
::theorem set c= set implies Cl ( set /\ set ) c= Cl ( set /\ set )
4134    :: CLASSES1:68  classes1        322     0.99
8609    :: FUNCT_4:6    funct_4 40      0.99
8610    :: FUNCT_4:7    funct_4 43      0.99
25838   :: TOLER_1:57   toler_1 261     0.99
```

The first line is the original query expression. Each subsequent line is a matched document in order of best match. For example, the second line tells us that document 4134 matched best and it corresponds to the theorem with the Mizar label `CLASSES1:6` and it appears in the file `classes1.abs` [4] at line 322. The remaining three lines correspond to theorems in the articles `funct_4.abs` [7] and `toler_1.abs` [11].

These results are then turned into fragments of the corresponding abstracts to save time and effort of looking up each query result individually. So for instance, the result just described would also return the fragment:

```
4134    :: CLASSES1:68  classes1        322     0.99
theorem :: CLASSES1:68
 the_transitive-closure_of (X /\ Y) c=
   the_transitive-closure_of X /\ the_transitive-closure_of Y;
```

The results after being looked up in the index and their corresponding fragments are written to separate files.

4 Some Early Results

At the time of writing, a fully working system has only just been completed. This means that extensive testing has not been done. Also, as only theorems have been used, the full power of the MML has not been exploited so extensive testing would be premature. Instead, these results can be regarded as a demonstration of potential rather than a realisation of it.

The tests so far fall into three categories:

1. Can we retrieve the original theorems?
2. Can we retrieve theorems based on part of the original theorem?
3. Can we retrieve required theorems given a part of proof that needs a theorem?

The first category is there as a sanity check that when throwing documents into the the soup of LSI, they are not lost forever. The second two are moving towards more realistic queries that a user might perform. First, if you are trying to remember a theorem in full but can only remember part, can you retrieve the whole theorem? Secondly, whilst working on a proof, a user might like to find a theorem that would prove the proof step that they are currently working on. Each of these categories of tests is discussed in turn.

4.1 The Identity Query

It is to be hoped that any query with an actual document used in the LSI would return that document as the best matching document. However, as mentioned earlier, queries can be performed with approximations based on the number of factors used in the similarity calculation. The unapproximated SVD of the MML has 106 factors.

All of the theorems in several abstracts were used in these tests. The documents that correspond to the theorems in these abstracts were used as queries but the query was only performed using 15 factors. Nevertheless, in almost every case, the first result returned was always the original theorem. The only exceptions to this were in the situation where two theorems were virtually indistinguishable from each other.

For example, the theorems `TOP_GRP1:5` and `TOP_GRP1:6`, from [14], when used as queries both returned the same two theorems *in that order*. These theorems are in fact:

```
theorem :: TOPGRP_1:5
 P c= Q implies P * h c= Q * h;

theorem :: TOPGRP_1:6
 P c= Q implies h * P c= h * Q;
```

As can be seen, they distinguish only in the order of the symbols that occur in them and as LSI does not take account of the order of terms in queries or documents, it is not surprising that these produced the same results. Also, from a user's perspective, it would not be surprising or even undesirable to have these two documents returned together.

A more surprising result from these tests came from the query based on the following theorem from [13]:

```
theorem :: TOPS_3:10
 A is boundary implies A <> the carrier of X;
```

As expected, the query returned precisely this theorem as the best match. Interestingly though the second best match was the following:

```
theorem :: TOPS_3:23
 A is nowhere_dense implies A <> the carrier of X;
```

These are clearly quite distinct theorems from general topology but the concepts of boundary and nowhere dense are very closely related. Specifically, by `TOPS_1:92` [19], every nowhere dense set is boundary and hence `TOPS_3:10` implies `TOPS_3:23`. LSI has no explicit knowledge of these theorems. The concepts of boundary and nowhere dense are simply related through frequent occurrence in related contexts. However, this vindicates the approach taken in this use of LSI in that it is the linguistic use of the concept and not the logical that in some ways defines the semantics of the concept.

4.2 Retrieving Full Theorems from Partial Theorems

A single theorem can often have multiple quantifiers and, usually, it is crucial to know whether a theorem is an implication, equivalence or contains negations. However, these details may not always be remembered by a human, indeed, they may be the precise reason why a person would look up a theorem.

These few tests were set up to see if, by constructing a query with a target theorem in mind, it is possible to retrieve the target. The queries were constructed using the document version of the theorem.

These preliminary results were somewhat mixed. Leaving out initial quantifiers does manage to retrieve the whole theorem. This works because in removing the variable names to convert a speculated theorem into a query, the types of the variables are replaced into the theorem. Where these types are known correctly, LSI would not distinguish between them being placed in a quantifier clause at the start of the theorem or in place of the variables as in these queries. Thus, these queries should and do work as expected.

Also, the system is reasonably successful at retrieving theorems where the logical direction of the theorem is unknown. This makes sense because in Mizar, this is usually signified with a keyword and keywords like `iff` and so on play no role in the search and retrieval methods as implemented here.

However, the main problem with these tests does seem to be getting the types of variables right. If the type of a variable is only partially entered, say entering only `TopStruct` instead of `carrier of TopStruct`, then the target theorem is not retrieved, at least not in the top 30 results.

Two further tests looked to retrieve a theorem when the relationship between variables in the theorem were unknown. In the first case what should have been an inequality relationship in `TOPS_3:10` was guessed at by entering some likely candidates such as equality and subset as well as inequality. Everything else was as in the document version of the target theorem. In the second case, the inequality was omitted entirely. In both cases, the target theorem was retrieved as the best match.

4.3 Retrieving Theorems to Complete Proof Steps

In writing a Mizar article, proofs need to give full references to theorems that are required to prove proof steps. In practice, this requires an extensive knowledge of what is (and is not) in the MML and more importantly where it is. A natural task for an author would be to find a theorem that applies to their current proof goals.

The following tests looked at proof steps that required a single theorem to complete the proof. The first few were all taken from the proof of `TOPS_3:3` in `tops_3.miz`. Topology was particularly chosen as I have some expertise in this area and hence the domaing knowledge needed to formulate queries.

The first test was trying to retrieve `TOPS_1:44`, [19], for the following proof step:

```
Int A c= A  by TOPS_1:44
```

Accordingly, the following query was generated:

```
theorem Int Subset of carrier of TopStruct
   c= Subset of carrier of TopStruct
```

The target theorem was indeed retrieved first by this query.

The second test was based on:

```
(Int A) /\ B c= A /\ B by XBOOLE_1:26
```

with the corresponding query:

```
theorem  set c= set implies set /\ set c= set /\ set
```

This retrieved `XBOOLE_1:26` as the second result.

Subsequent queries all based on steps in the same proof, however, did not show such promise. In each case, the query formulated did retrieve theorems that were certainly in the right topic but none directly pertinent to the target proof step. These steps all involved the notions of interior and closure of subspaces in topology and its was notable that a lot of the retrieved theorems tended to include one but not both of these concepts. Perhaps these notions were poorly captured by LSI.

Two further queries (avoiding interiors and closures) were attempted, based on line 210 of `tops_3.miz`:

```
G c= A by TOPS_1:20
```

In this context, it was possible to formulate two queries that might have served well:

```
theorem for Subset of TopSpace holds
 ( - Subset of TopSpace ) misses Subset of TopSpace
 implies Subset of TopSapce c= Subset of TopSpace

theorem for Subste of carrier of TopStruct holds
  ( - Subset of carrier of TopStruct ) misses Subset of carrier of TopStruct
  implies Subset of carrier of TopStruct c= Subset of carrier of TopStruct
```

The target theorem was:

```
theorem :: TOPS_1:20
 K c= Q iff K misses -Q;
```

Whilst neither query retrieved this theorem, the second query did actually find the following in the top ten results:

```
theorem :: PRE_TOPC:21
  for T being 1-sorted
  for P,Q being Subset of the carrier of T holds P c= -Q iff P misses Q;
```

Though obviously distinct from the expected theorem, in fact, with a little work the original proof could easily be re-written to use this theorem rather than expected one. This then represents a significant success from a task-oriented perspective.

4.4 Summary of Results

The few tests reported here do indeed show promise though obviously the reliability of these queries is currently limited. The results are summarised in Table 1. All of the tests however did return results that could definitely be described as in the right topic for the queries. Also, as hoped for, LSI through working via the language of mathematics is also highlighting logical and conceptual links between the documents. Improving the reliability of these queries, though, is a clear priority and the current lines of development are discussed in the next section.

5 Conclusion and Future Work

The work done here in applying LSI to a formal mathematics library has shown the following:

– LSI can perform useful retrieval without explicit semantics
– LSI can find conceptual associations between mathematical notions
– Users can formulate query expressions for which LSI returns useful results

However, there is clearly an issue with the reliability of the queries and future work will address this.

Task	**Target type**	**Query**	**Outcome**
Identity	Any theorem	The target itself	Retrieves target or theorem with identical terms
	`TOPS_3:10`	`TOPS_3:10`	`TOPS_3:23` retrieved second because of close conceptual relationship
Partial	Any theorem	Target without quantifiers	Successful retrieval
	Any theorem	Target without correct variable types	Target not in top 30
	Theorem with binary relation	Target with multiple terms to cover target relation	Successful retrieval
	Theorem with binary relation	Target with target relation omitted	Successful retrieval
Proof goals	Theorem to prove step	Guess at target theorem	Limited success
	`TOPS_1:20`	Guess at target theorem	Retrieved `PRE_TOPC:21` which could be applied

Table 1. Summary of Test Results

Definitions and proofs were not included as documents in the LSI calculations. Incorporating them would increase the number of documents and could therefore better elucidate the terms in the MML. Actually, I strongly believe that proofs are the discourses of mathematics – theorems merely represent the headlines. Thus, including proofs as documents would provide a strong language-oriented foundation for the MML that I would expect LSI to exploit well.

Making proofs into documents poses two substantial obstacles. The first and easiest is that so far the Mizar language has been difficult to parse with JavaCC. Theorems and definitions use a relatively constrained subset of the full Mizar grammar and so in this implementation have been parsed using a combination of JavaCC and bespoke finite state machines. A further implementation should really use a more generic and hence robust approach to parsing though it is not clear that this would actually overcome some of the difficulties encountered so far.

The more substantial problem of including proofs as documents is the issue of how to represent a proof as a document. With theorems, this is relatively straightforward as a theorem could be viewed as a stand-alone logical statement. Proofs however represent a transformation between logical statements. Already, in trying to complete proof steps in the tests described above, I encountered issues of what exactly would constitute a sensible query in the context. Simple subsitution of variable names by the variable types, as used to turn theorems into documents, was not enough. Some patterns of how to formulate effective queries emerged and it may be possible to automate some of the query formulation task. In many ways, though, the formulation of queries can only be tackled on an empirical basis of what works well.

A wider issue that addresses the broader aim of this work is to make generating queries and understanding results easier to work with for a Mizar author. Some form of visual interface would seem appropriate but deciding what to represent and how to represent it probably represents a significant design and evaluation effort. Related to this is whether users would want to see all the documents that were retrieved or whether in different contexts they could rely on, say, just retrieving proofs or just theorems.

In addition could further search methods or heuristics improve the reliability and therefore value of the LSI-based results? For instance, would these results be enhanced by using the filtering operations of Bancerek and Rudnicki [2]? Could the logical keywords like `iff` be used to filter suitable theorems?

In conclusion, this work shows that using search based on the implicit semantics of a formal mathematical library does in fact yield some meaningful results. With further work and empirical evaluation, it is hoped that this is a step towards a full semantic search of mathematical libraries that real mathematicians would find useful in their work.

Acknowledgments

Thanks to Dr. Jeremy Gow for his encouragement and his comments on this paper, to Dr. Andrzej Trybulec for a useful correction to an earlier version of this paper and the anonymous referees for some very useful feedback.

References

1. Asperti, A., Buchberger, B., Davenport, J. H.: Mathematical Knowledge Management, Proceedings of MKM 2003. LNCS **2594** Springer Verlag (2003)
2. Bancerek, G., Rudnicki, P.: Information Retrieval in MML. In [1] (2003) 119–132
3. Bancerek, G.: Sequences of Ordinal Numbers. Journal of Formalized Mathematics **1** (1990) 281–290
4. Bancerek, G.: Tarski's Classes and Ranks. Journal of Formalized Mathematics **1** (1990) 263–567
5. Baumgartner, P., Furbach, U.: Automated Deduction Techniques for the Management of Personalized Documents. Annals of Mathematics and Art. Intelligence **38** (2003) 211-288
6. Berry, M. W., Dumais, S.: Latent Semantic Indexing Web Site. Accessed 29th March, 2004 `http://www.cs.utk.edu/ lsi`
7. Byliński, C.: The modification of a function by a function and the iterations of the composition of a function. Journal of Formalized Mathematics **1** (1990) 521–527
8. Cairns, P., Gow, J.: Using and Parsing the Mizar Language. Electronic Notes in Theoretical Computer Science **93** Elsevier (2004) 60–69
9. Delahaye D.: Information Retrieval in Coq Proof Library using Type Isomorphisms. In T. Coquand *et al.* (eds), TYPES, LNCS **1956** (2000) 131–147
10. Giles, J. T., Wo, L., Berry, M. W.: GTP (General Text Parser) Software for Text Mining. In H. Bozdogan (ed.): Statistical Data Mining and Knowledge Discovery. CRC Press, Boca Raton (2001) 457–473

11. Hryniewiecki, K.: Relations of tolerance. Journal of Formalized Mathematics **2** (1991) 105–109
12. Java Compiler Compiler. Accessed 31st March, 2004
`http://javacc.dev.java.net`
13. Karno, Z.: Remarks on special subsets of topological spaces. Journal of Formalized Mathematics **3** (1992) 297–303
14. Korniłowicz, A..: The definition and basic properties of topological groups. Journal of Formalized Mathematics **7** (1998) 217–225
15. Landauer, T. K., Foltz, P. W., Laham, D.: Introduction to Latent Semantic Analysis. Discourse Processes **25** (1998) 259–284
16. Landauer, T. K., LSA @ Colorado University. Accessed 31st March, 2004
`http://lsa.colorado.edu`
17. Miller B. R., Youssef A.: Technical Aspects of the Digital Library of Mathematical Functions. Annals of Mathematics and Art. Intelligence **38** (2003) 121–136
18. Mirel, B.:Interaction Design for Complex Problem Solving. Morgan Kaufmann (2004)
19. Wysocki, M., Darmochwał, A.: Subsets of Topological Spaces. Journal of Formalized Mathematics **1** (1990) 231–237
20. The Mizar Mathematical Library
`http://mizar.org`
21. Philips, L., Jørgensen, M. W.: Discourse Analysis as Theory and Method. Sage Publications (2002)
22. Rudnicki, P.: An overview of the Mizar project. In Proceedings of 1992 Workshop on Types and Proofs for Programs (1992)
23. Text REtreival Conference (TREC). Accessed 31st March, 2004
`http://trec.nist.gov`
24. Wiedijk F.: Comparing Mathematical Provers. In [1] (2003) 188–202