Integrating Searching and Authoring in Mizar

Paul Cairns and Jeremy Gow
UCL Interaction Centre, University College London, 31-32 Alfred Place, London WC1E 7DP, UK

Abstract. The vision of a computerised assistant to mathematicians has existed since the inception of theorem proving systems. The Alcor system has been designed to investigate and explore how a mathematician might interact with such an assistant by providing an interface to Mizar and the Mizar Mathematical Library. Our current research focuses on the integration of searching and authoring while proving. In this paper we use a scenario to elaborate the nature of the interaction. We abstract from the scenario two distinct styles of searching and describe how the Alcor interface implements these with keyword and LSI-based search. Though Alcor is still in its early stages of development, there are clear implications for the general problem of integrating searching and authoring, as well as technical issues with Mizar.

Keywords: LSI, Alcor, Mizar, information retrieval, mathematics

1. User interfaces for mathematicians

Since the very early days of theorem proving systems, it has been envisioned that one day these systems would be able to support mathematicians at work (MacKenzie, 2001). Even now, however, very few mathematicians are actively using theorem provers. In part, this may be because of the difficulty of using theorem provers but also it is increasingly recognised that mathematics is more than simply proving statements (Kerber, Kohlhase and Sorge, 1998). Mathematicians also need to do algebraic manipulations and to draw on extensive knowledge of existing mathematics. Computer algebra systems have proven to be extremely successful and their integration with theorem provers is making significant advances (Farmer and Mohrenschildt, 2003). To integrate mathematical knowledge into such systems has become the focus of mathematical knowledge management (MKM)(Buchberger, Gonnet and Hazewinkel, 2003). As this field has developed, it has become clear that it is not just mathematicians that could use these tools but anybody working with mathematics such as economists, physicists, biologists and so on. MKM aims to develop software, protocols and representations that support the storage and retrieval of substantial bodies of mathematical knowledge (Adams, 2003).

There will not be a single user interface that will satisfy all prospective users of MKM technology. Instead, in this paper, we focus on the original vision of the mathematical assistant. The question motivating our work is: how might a mathematician develop a proof with the aid of MKM tools? In particular, we consider the situation of a mathematician trying to write a proof with the support of a proof checker and a searchable repository of mathematical knowledge. To address this question, we have developed the Alcor user interface for the Mizar proof system and Mizar Mathematical Library (MML)(Rudnicki, 1992). As will be discussed in more detail, the MML provides an excellent resource for experimenting with different MKM user interfaces. Specifically for this paper, we consider how to integrate within Alcor tools for authoring a proof and for searching the MML.

We motivate the design of Alcor using a scenario, a common approach in the user-centred design of interfaces (Rosson and Carroll, 2002; Cooper, 1999). This, in part, expands on the basic idea of a mathematical assistant but more usefully specifies how we envision a mathematical assistant could work. This scenario is not intended to be definitive and, like all scenarios, is not a full specification of the assistant's functionality. Rather, the scenario allows us to motivate and evaluate our design decisions explicitly.

Though an earlier version of Alcor has been described (Cairns, 2005), the version here has been extended with search over the Mizar library using latent semantic indexing (LSI) (Cairns, 2004) and thus has two distinct search styles. We present an initial evaluation of the LSI-based search, which has been significantly re-implemented to exploit Urban's XML-ization of Mizar (Urban, 2006). The addition of this search method to the existing keyword search reveals some issues in the different styles of search that mathematicians need and how they might best be integrated into an authoring environment.

2. Scenario: working with a mathematical assistant

Carl is a mathematician in the second year of his PhD. He is working in the area of topological groups and he has just completed his first big theorem that he would like to write up for submission to the Journal of Topological Groups. The journal, in accordance with many other maths journals, now encourages submissions that have been written with the support of a proof checking system. This ensures the accuracy of the published paper whilst leaving the referees to comment on the value and relevance of the contribution of the paper.

Carl begins writing up his paper using an integrated proving environment. He has used the system before to prepare his end of year report last year and so is basically confident with using the system. As he is entering his proof, the system occasionally asks for clarification of the meaning of the symbols used, for example, whether they are constants or functions. Also, it may ask Carl to disambiguate possible different interpretations of statements particularly where functions are being used with varying numbers of arguments.

As he is entering the proof, Carl realises that there is a technical definition used in his proof and he is not sure that he has met all of the criteria of the definition. He searches on the defined term which gives him several hits from a selection of journals and textbooks. He recognises one of the textbooks and knows that this hit will give him the definition he is looking for. When the hit is retrieved, he is able to check his proof with the definition itself. As he knows that the proof checker would benefit from knowing the definition, he makes the link between his proof and the definition.

Eventually, Carl feels he has entered the basic structure of the proof and so runs the proof checker. The proof checker, as expected, highlights some places where it cannot fully complete the proof to its satisfaction. However, this does not stop the checker from moving on to later proof steps. The final result is Carl's proof with indications of where further additions and refinements need to be made.

One step in the proof in particular surprises Carl as he thought that it was relatively easy but the checker has indicated that it cannot prove it. Carl knows he is implicitly using a lemma but he cannot remember which lemma or where it might be found. He therefore searches the electronic resources using the proof step itself. There are many returned hits that are like his in using the lemma implicitly so he refines his search just to look at the statements of propositions and theorems. This is much better as fewer hits are found and, a short way down the list, he spots a statement of the lemma. Unfortunately, the lemma is not quite in the form that is needed in his proof. Carl therefore adapts his proof to use the lemma that he found. He then re-runs the proof checker on this new step. He is quite pleased when the proof checker indicates that it has successfully proved the step.

2.1. Some issues raised by the scenario

Before addressing issues raised directly by the scenario, it is worth noting that scenarios are usually developed through gathering requirements from users. This has not been done here partly because mathematical assistants do not yet exist and it is not easy to ask users to talk meaningfully about something that does not exist (Cooper, 1999). Moreover, some mathematicians are quite resistant to the idea of this sort of system — one has even told us that they find such systems quite threatening. Thus, to develop appropriate functionality in a mathematical assistant, it is necessary to engage in this sort of futurological daydreaming in order to compensate for the lack of explicit requirements. This scenario reflects one view of how a mathematical assistant should work. It is not, therefore, definitive but is nonetheless useful in explicitly motivating our design decisions.

The key elements of the scenario are how the searching integrates with the authoring. Two distinct styles of search are identified and we consider each of these in more detail.

First, Carl wants to be able to look up the precise definitions of terms that he already knows. This may also correspond to looking up the statement of theorems or equations by their name, for example, Brouwer's Fixed Point Theorem or Bessel functions. This is the sort of checking of details that are essential for theorem proving systems but also for mathematicians to ensure that they have correctly applied their knowledge.

The second style of search is more about helping the mathematician to produce a correct proof. It is not the case that there is a specific lemma or proposition that is needed but rather that something is needed to allow the proof to advance. This is quite distinct from the first sort of search because it is possible to re-write a proof in order to make use of whatever propositions or theorems were found. In some cases, it may even be that the theorem being proven can be restated in order to accommodate a particular proposition used in the proof (Lakatos, 1976). This search is therefore asking for something "like this" or "to help here". The results returned do not need to be precisely what was expected but similar enough in meaning to be useful.

Like any search of a significant body of work, it is expected that many hits can be returned but that these can also be refined to include particular sorts of documents. For instance, if Carl is looking for a definition, then he can filter out things that are not definitions. Also, some search results will be directly relevant to a proof, either allowing a step to be proven or filling in a gap for the proof checker. Thus, having found useful results, the system should make it easy to link the search results into the authoring of the proof.

Another feature worth noting is that Carl's searches are over many sources of information. Some results may be from a formal library like the MML, some may be from a more usual mathematical journal or textbook and some may even be in languages other than Carl's. These differences in source could also belie differences in logic. Some of these can be minor, but some can be mutually inconsistent and could cause real problems. Translation between the sources both in terms of logic and language would be useful but then this requires appropriate services to do the translation. These present a further set of MKM tools necessary for good mathematical assistance (Kohlhase & Anghelache, 2003).

Note that the scenario deliberately does not specify the method of input that Carl uses nor how the various different outputs from searches and the proof checker are fed back to Carl. This avoids biasing our design whilst also suggesting new approaches. For instance, whilst not the focus of this paper, Carl might be entering his proof using a pen-based input (Thimbleby, 2004). Not only would this more closely correspond to his pencil and paper working but it would also suggest that the system is doing "on the fly" graph parsing (Rekers and Schürr, 1995) in order to clarify what Carl means.

Scenarios should also give some motivation for using the system that is going to be developed. The current low uptake of theorem proving systems amongst mathematicians suggests that they are, if not actively resistant to, not interested in using such systems — certainly they are currently perceived with some caution by mainstream mathematicians (Bundy et al., 2005). The scenario therefore externalises Carl's motivation to one of being able to publish if he uses the system. In addition, there is the implicit benefit that proof checked journals would be better able to integrate into MKM tools more generally. Of course, this does not address why the community of mathematicians might choose to run their journals in such a way, and indeed, this may be a chicken-and-egg situation. Though this is the broader concern of MKM, it is not addressed with the work described here.

3. Alcor

The purpose of the Alcor system is to explore what an integrated proving environment may be like. In particular, the current focus of Alcor's development is on integrating authoring and searching in Mizar, using a single user interface with the two styles of searching described above.

MKM is still some way from providing the tools that would allow search across a wide-range of heterogeneous sources. However, this full functionality is not necessary to study the interaction of searching and authoring. The Mizar system and the MML form a homogeneous, semantically explicit, machine-checked and very substantial body of mathematical knowledge (Rudnicki, 1992; MML, 2006). This removes the need for advanced (currently non-existent) MKM tools whilst retaining the goals of a mathematician wanting to work with a proof checker, Mizar, and to use the MML as an integrated resource in authoring proofs.

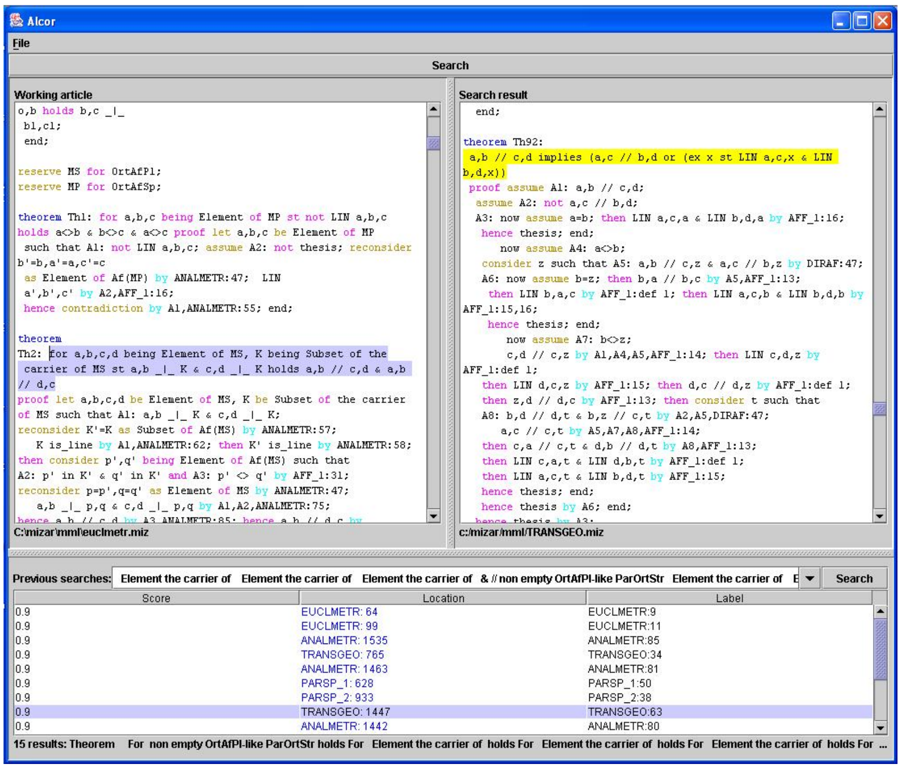
As a result of using Mizar as an MKM repository, Alcor is effectively a user interface for the Mizar system — but the goal of Alcor is not primarily to serve as a front end for Mizar. Instead, Alcor is a system that demonstrates what might be possible with a mathematical assistant and how the functionality of an assistant may be made available to a mathematician. As such, Alcor is intended to help elicit and specify better what mathematicians might require from such a tool.

The name Alcor was chosen as the Mizar system is named after a star in the constellation of Ursa Major. This star has a companion star called Alcor and they are distinguishable by the naked eye to those with good acuity. The two stars together are colloquially called the horse and rider and it seemed an appropriate name to reflect that Alcor is a user interface that sits on top of Mizar.

3.1. Searching in Alcor

The inspiration for the basic layout of Alcor is the Phrasier system (Jones and Stavely, 1999) which has a split screen for word processing and searching. In an analogous way, Alcor has one area for authoring a Mizar article and another area for displaying found articles as shown in Figure 1. This way authoring and searching are given equal status in the system. There is also a section across the bottom of the interface that displays a search bar and a list of search results, as found in many applications and web-sites.

In order to conduct a search, the author enters a search term either by typing it in the search bar or by selecting text from either of the article windows. Results from the search are listed in the lower section with the three columns being relevance to the search term, the MML article and the line number where the result can be found, and a label for the search result. Clicking on one of these items in the list causes the search result to be displayed in the right hand window and the line where the result can be found is highlighted in yellow. For example, Figure 1 shows a theorem highlighted in blue in the left-hand, authoring window. The results from searching on this theorem are displayed. Notice that the labels for the results for this particular search are the theorem numbers as they appear in Mizar abstracts. The highlighted item in the results list has been clicked to show the result, highlighted in yellow, in the right-hand, searching window. The author need not use the search bar for entering terms, simply typing and highlighting text in the authoring window has the same outcome. The search bar also has a drop-down menu to allow the author to select previous searches and so acts as search history.



```
o,b holds b,c _|_
 b1,c1;
 end;

reserve MS for OrtAfPl;
reserve MP for OrtAfSp;

theorem Th1: for a,b,c being Element of MP st not LIN a,b,c
holds a<>b & b<>c & a<>c proof let a,b,c be Element of MP
 such that A1: not LIN a,b,c; assume A2: not thesis; reconsider
b'=b,a'=a,c'=c
 as Element of Af(MP) by ANALMETR:47;  LIN
 a',b',c' by A2,AFF_1:16;
 hence contradiction by A1,ANALMETR:55; end;

theorem
Th2: for a,b,c,d being Element of MS, K being Subset of the
 carrier of MS st a,b _|_ K & c,d _|_ K holds a,b // c,d & a,b
// d,c
proof let a,b,c,d be Element of MS, K be Subset of the carrier
of MS such that A1: a,b _|_ K & c,d _|_ K;
reconsider K'=K as Subset of Af(MS) by ANALMETR:57;
  K is_line by A1,ANALMETR:62; then K' is_line by ANALMETR:58;
then consider p',q' being Element of Af(MS) such that
A2: p' in K' & q' in K' and A3: p' <> q' by AFF_1:31;
reconsider p=p',q=q' as Element of MS by ANALMETR:47;
  a,b _|_ p,q & c,d _|_ p,q by A1,A2,ANALMETR:75;
hence a,b // c,d by A3,ANALMETR:85; hence a,b // d,c by
```

C:\mizar\mml\euclmetr.miz

```
  end;

theorem Th92:
 a,b // c,d implies (a,c // b,d or (ex x st LIN a,c,x & LIN
b,d,x))
 proof assume A1: a,b // c,d;
  assume A2: not a,c // b,d;
  A3: now assume a=b; then LIN a,c,a & LIN b,d,a by AFF_1:16;
   hence thesis; end;
      now assume A4: a<>b;
   consider z such that A5: a,b // c,z & a,c // b,z by DIRAF:47;
   A6: now assume b=z; then b,a // b,c by A5,AFF_1:13;
    then LIN b,a,c by AFF_1:def 1; then LIN a,c,b & LIN b,d,b by
AFF_1:15,16;
    hence thesis; end;
       now assume A7: b<>z;
       c,d // c,z by A1,A4,A5,AFF_1:14; then LIN c,d,z by
AFF_1:def 1;
   then LIN d,c,z by AFF_1:15; then d,c // d,z by AFF_1:def 1;
   then z,d // d,c by AFF_1:13; then consider t such that
   A8: b,d // d,t & b,z // c,t by A2,A5,DIRAF:47;
      a,c // c,t by A5,A7,A8,AFF_1:14;
   then c,a // c,t & d,b // d,t by A8,AFF_1:13;
   then LIN c,a,t & LIN d,b,t by AFF_1:def 1;
   then LIN a,c,t & LIN b,d,t by AFF_1:15;
   hence thesis; end;
   hence thesis by A6; end;
  hence thesis by A3;
```

c:/mizar/mml/TRANSGEO.miz

Previous searches: Element the carrier of   Element the carrier of   Element the carrier of   & // non empty OrtAfPl-like ParOrtStr   Element the carrier of   E

Score	Location	Label
0.9	EUCLMETR: 64	EUCLMETR:9
0.9	EUCLMETR: 99	EUCLMETR:11
0.9	ANALMETR: 1535	ANALMETR:85
0.9	TRANSGEO: 765	TRANSGEO:34
0.9	ANALMETR: 1463	ANALMETR:81
0.9	PARSP_1: 628	PARSP_1:50
0.9	PARSP_2: 933	PARSP_2:38
0.9	TRANSGEO: 1447	TRANSGEO:63
0.9	ANALMETR: 1442	ANALMETR:80

15 results: Theorem   For non empty OrtAfPl-like ParOrtStr holds For   Element the carrier of holds For   Element the carrier of holds For   Element the carrier of holds For ...

Figure 1. The Alcor system

There are two distinct search algorithms implemented in Alcor, corresponding to the two styles of search. The first is a simple keyword search that looks up the definition of terms, while the second is based on latent semantic indexing (LSI). At the moment, a simple heuristic is used to distinguish the two types of search: if the query consists of a single term then it is a keyword search, otherwise it is an LSI search. We briefly describe the keyword search here before moving on to give more detailed description and evaluation of LSI-based search.

The keyword search allows a user to enter a single term from Mizar and its definition is looked up. To aid the user when the exact term required is not known or simply just for speed, the search uses the entered term as a prefix and finds the definition of all terms that begin with the entered term. So for instance, entering the term "even" returns hits for the term itself as well as "Event", "eventually-directed" and so on.

In this way, the keyword search reflects the need for searches where the terms are known and it is details of those particular terms that is required. We have introduced a second search method to address the need to find results like the search term but that do not necessarily precisely match it. LSI was chosen because it has proven to be very successful in capturing the informal semantic sense of queries in natural language information retrieval (Landauer, Foltz and Laham, 1998).

3.2. LSI in Alcor

In formal mathematics, the precise semantic sense of every term is explicit. Thus it could be argued that some sort of automated reasoning system could just as well be used to find which statements are semantically close to a given query. However, the sophisticated reasoning across a large number of domains that this would require are beyond the capabilities of current automated theorem proving systems. Moreover, it is not only the logical relationships between concepts that are required, but also some form of search that captures the informal relationships between concepts. More concretely, mathematicians name and thereby isolate concepts in order to communicate more effectively amongst each other. Their choice of words, the definitions they make and the theorems that they formulate function in as rich a manner as any natural language. It is through such discourse that relationships are defined between concepts in a way that is not necessarily explicit in the logical formulation of the definitions (Zinn, 2004).

For example, compact and connected are terms commonly used in topology and there are several theorems and definitions that link these terms together. However, aside from being properties of topological spaces, there is no logical link between them. It may be that, in proving a theorem about connectedness, an author would be happy to restrict attention to only compact spaces if that would allow the proof to proceed. However, if search results are only related to connectedness, this possibility might not occur to the mathematician.

LSI is a well known technique that is able to provide the informal relationships between concepts that are required in our second type of search. LSI works by taking a corpus of documents and the terms from those documents and processing them in such a way that it is possible to retrieve documents from the corpus even if they do not contain any of the query terms. It does this by using a matrix, Δ, representing the occurrence of terms in documents. More specifically, the terms and documents are listed in some fixed order and if the i^{th} terms occurs in the j^{th} document then the matrix element $\Delta_{ij} = 1$ otherwise $\Delta_{ij} = 0$.

Singular value decomposition of Δ produces a reduction of the matrix to a diagonal matrix, Σ_D where the diagonal entries effectively represent the eigenvalues of the original matrix. Symbolically,

$$\Delta = U \Sigma_D V$$

where U and V are orthogonal matrices. The non-zero eigenvalues of Σ_D correspond to a set of eigenvectors of Δ that capture the underlying term-document space of Δ. It is the structure of this space that is taken to be the 'latent meaning' of the terms and the documents as defined by the corpus. In other words, LSI provides a 'surrogate semantics' based on term co-occurrence.

A query can be converted into a row vector, t, where $t_i = 1$ if the i^{th} term is in the query and otherwise $t_i = 0$. Ordinary keyword search is then implemented by matrix multiplication where

$$d = t.\Delta = t.U\Sigma_D V$$

so that d is a column vector and d_j is equal to the number of terms in the query that occur in the j^{th} document.

However, terms and documents can be used to retrieve similar terms and documents depending on how close they are in the underlying term-document space and not through the occurrence of specific terms in specific documents. By transforming any document vector d to be $d' = d.V^{-1}$ it is possible to find the similarity between that document and the transformed query vector $t' = t.U\Sigma_D$. Thus, it is possible to compare any document with the query and so produce a similarity rating based on the underlying term-document space rather than on the precise terms occurring in the document. The rating is actually the cosine of the angle between the document vector and term vector in the term-document spaces. A perfect match has rating 1 and an orthogonal document has rating 0, which can be taken to mean that the document has nothing in common semantically with the query vector.

Furthermore, the underlying space can be reduced in dimension by neglecting dimensions corresponding to small eigenvalues in Σ_D. This has the effect of removing from the retrieval process areas of the space that are not well represented in the original corpus and hence improving the ability of the search to generalise "meaning" across the body of documents as a whole. It is therefore recommended that LSI uses fewer dimensions (known as factors). More details of the mathematics behind LSI are given in (Cairns, 2004).

The MML represents a suitably large repository of documents and mathematical terms for LSI to work with. The current implementation uses version 4.35.912 of the MML. At the moment, Alcor is only using theorems in the MML as documents for LSI. Terms are any of the Mizar keywords. This results in Δ being a 41,072 by 29823 matrix.

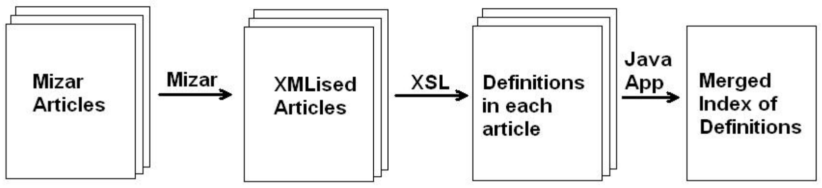


Figure 2. Generating the keyword index from MML

3.3. Implementation details

The implementations of both search methods currently used in Alcor are significantly different from previous versions thanks to the XML-ization of the MML introduced by Urban (Urban, 2006). Previous versions of Alcor (Cairns, 2005) used either AntLR (Parr, 2006) or JavaCC (JavaCC, 2006) and despite both being standard, fully functional parsers, were not up to the job of parsing the Mizar language (Cairns and Gow, 2004). Urban has adapted Mizar so that it is able to produce an XML version of any articles that it is proof checking. This means that if Mizar is run on an article from the MML, one of the outputs is a semantically explicit XML version of the article. Currently, due to the complexities of the Mizar language, this seems to be the only practical way to produce a complete, parsable form of the MML outside of Mizar itself.

Both keyword and LSI search are based on information extracted from the MML. Most of this is done as pre-processing and the results of which are used by Alcor.

For keyword search, once each article has been obtained in XML form, it is simply a matter of applying a stylesheet to extract from each article a list of the definitions occurring in the article. A small Java application then merges the lists for each article into a single large index of where each term is defined in the MML. This process is summarised in Figure 2. Within Alcor, the keyword index is stored as a trie which makes it easy to treat the query term as a prefix.

The pre-processing required for the LSI-based search is somewhat more complicated. The basic process is illustrated in Figure 3 and consists of turning the MML into a set of documents. Currently, these documents are theorems (without their proofs), that is, each theorem is used to make one document. The theorem itself is not used though because, like any good mathematics, Mizar makes extensive

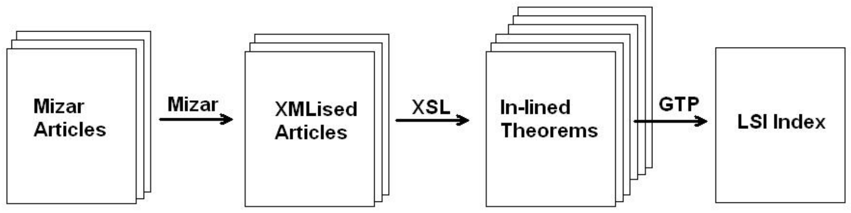


Figure 3. Applying LSI to the MML

use of variables to stand in for actual objects. It is pointless to use, for example, "X" as a term in LSI because in most cases there is no particular consistent meaning of this term across the corpus. Also, it is possible that if LSI were to treat variable names as terms there could be accidental association between concepts as a result of using the same names for variables. This would reduce the quality of the LSI search results. Instead, all such variables are in-lined with their type as defined in Mizar, once again using XML stylesheets on the XML version of the articles. For example, in:

```
theorem :: SIN_COS6:2
for f being Function, X, Y being set st f|X is one-to-one &
Y c= X holds   f|Y is one-to-one;
```

the variable f,X and Y are replaced by their types to give, rather verbosely:

```
For  Relation-like Function-like  set holds For set holds
For set  holds not ( | Relation-like Function-like set
set is one-to-one &
c= set set & not | Relation-like Function-like set
set is one-to-one )
```

Note also from this that the terms do not need to be in any specific order as LSI does not use the order of terms in its calculations.

These in-lined theorems constitute the basic documents over which LSI is applied. Mike Berry kindly provided us with the GTP package which is a Java implementation of LSI and this is used for all the LSI processing in Alcor.

Querying using LSI requires taking the search terms and converting them into a query suitable for LSI to use. For the search to be effective, the terms in the query should also be terms used in the original documents. This requires in-lining variables just as for the original theorems.

However, given the difficulties of parsing Mizar, Mizar itself has to be used to parse search terms in Alcor. That is, to formulate a query for LSI, the search terms are extracted into a small Mizar article and this is sent to the proof checker (knowing that the proof checker is not going to be able to check it properly) to be converted to XML. The resulting XML is then converted into a query by applying the same stylesheet used to convert theorems into documents. This somewhat circuitous process does work but it is very unsatisfactory as Mizar might fail to parse the search terms because of a problem in trying to prove the search terms rather than because the search terms are particularly difficult to parse.

To apply stylesheets, in all cases, the stand-alone application msxsl is used. This is freely available from Microsoft. In the pre-processing steps, it is applied to each XML-ized article using a Visual Basic script (development was done in Windows XP). This is clearly not the cleanest method for applying stylesheets to XML but it would be relatively straightforward to adapt this process into a single Java application by using a Java XML/XSL system such as Xalan (Kay, 2001). This would also avoid using the platform specific VB script for the batch process. However, at this early stage in development, we have simply used whatever tools were to hand to prove that in principle it could be done.

3.4. Evaluating LSI search

Even though only theorems are currently being used as documents for LSI, there are good indications that LSI-based search is working as intended, though is not without some unexpected outcomes. This section describes some initial tests designed to assess the effectiveness of the approach.

3.4.1. *Retrieving existing theorems*

Given that LSI loses information when using a smaller number of factors than the dimension of the full term-document space, it is possible that finding a known theorem in the library may not always be successful. This test therefore ran a set of queries which were based on actual theorems in the MML, using 1422 theorems from the first two editions of Formalized Mathematics (MML, 2006) in 2005 — that is, volume 13, issues 1 and 2. LSI was asked to return only the first 20 best hits, as measured by the LSI similarity rating. Additionally, because of compatability issues between the different packages used in the implementation of the testing environment, not all theorems were used but only those that had a label as given by the XMLiser such

Table I. Number of theorems of SIN_COS6 retrieved in top 50

Number of factors	15	30	50	100
Theorems retrieved (out of 68)	3	4	2	4

as SIN_COS6:88. However, despite this restriction, it was felt that the remaining 1422 theorems did give a good representative cross section of the MML that would indicate the reliablity of LSI in retrieving theorems that are known to exist in the MML.

To test the effect of a significant reduction in information, only 15 factors were used in the LSI retrieval. This is a rather small number of factors to use but it does allow for reasonable search times whereas with 100 factors, say, search times can be 10 to 20 seconds. Out of the 1422 labelled theorems used as queries, 284 theorems were not found at the top of the retrieval list and 67 of those were not found in the top 20 at all.

Of the 217 theorems that were found but not at the top of the list, 198 were found in the next top three positions. Further examination of the theorems that were not found at all showed that they all belonged to the same article labelled in the MML SIN_COS6 (Kornilowicz and Shidama, 2005). Indeed, only one searched for theorem in that article was found, namely SIN_COS6:88.

A straightforward reason why a theorem was not found in the top twenty could be that the strength of the match between a theorem and itself was swamped by many similar strength matches. However, this cannot be the case for all because in the top 20 retrievals for SIN_COS6:77, the least similar match had rating 0.891 and best match of 1.000 with an average rating of 0.939. Thus, it must be the case that the rating of the theorem for itself is significantly below the ideal rating of 1. Of course, it could also be that the theorem is retrieved close to the top twenty.

To test whether the number of factors used was truly too few to do accurate retrieval, the theorems from SIN_COS6 were run again with different numbers of factors as summarised in Table 3.4.1 and with the top 50 hits.

This suggests that there is something unusual to do with the use of terms in this article. A closer look at the article shows that many of the theorems are in fact very similar. It could be that this similarity between them is forming a very dense area in the term-document space. If this area is also relatively distinct from the space defined by other articles then it may be that the LSI is unable to distinguish between these articles using the same factors as are effective with other articles. This is necessarily pure speculation as the LSI semantics are not explicit but this may be a plausible way to understand this unusual feature.

With the exception of this article though, it would seem that LSI does in fact provide a robust way to retrieve known theorems from the MML. 1354 (95%) of the theorems are retrieved in the top twenty of 1422 theorems searched for even when using only 15 factors. This gives comfort that although LSI is inexplicit and information is lost when doing retrieval, it can still find something that is known to exist in the library.

3.4.2. *Overlap between queries and results*

It is expected that if LSI is returning semantically close results to a query, there ought to be some overlap between the terms in the results and the query. To assess this, a measure of commonality was taken to be the number of terms common to both theorems taken as a percentage of the smaller theorem. Thus a 100% match means either that all of the terms of the query were also in the result or vice versa depending on which was the smaller.

Of the 1422 labelled theorems that were tested above, this results in 19129 retrieved labelled theorems. The average percentage overlap between a query and a result is 71.3%. 17,233 retrieved theorems had more than 50% overlap with their queries. This hard figure should be interpreted with a pinch of salt though because terms include the equals signs and numerals which may occur in quite widely varying contexts. Nevertheless, this gives a good indication that theorems retrieved by LSI do generally have some appreciable overlap in terms and therefore probably in meaning as well.

Curiously, 129 of the results had no common terms with the corresponding query. Moreover, some of these had a similarity rating very close to 1. The ten with the highest similarity ratings were examined further and did indeed seem to have very little semantic relationship between them. For instance, `SIN_COS6:98` and `SIN_COS6:2` (Kornilowicz and Shidama, 2005) had a perfect match of 1 but the two theorems are:

```
theorem :: SIN_COS6:98
  -1 <= r & r <= 1 & arccos r = 0 implies r = 1;

theorem :: SIN_COS6:2
for f being Function, X, Y being set st f|X is one-to-one &
Y c= X holds   f|Y is one-to-one;
```

Table II. Number of mismatched results arising from the sample of 10 problem queries

Number of factors	15	30	50	100
Mismatched results	10	8	6	5

These really do have little in the way of semantic similarity though the latter theorem could conceivably be useful in proving the former. Again, this could be a feature of using so few factors for the purposes of retrieval. The ten worst queries were then re-run with varying numbers of factors and the results are summarised in Table 3.4.2. This suggests that it is the number of factors that is responsible in part for these peculiar matches.

In many ways though, this is actually a valuable feature of LSI in that it can retrieve apparently unrelated theorems on the basis that they occupy a similar part of the term-document space. Unlike keyword based searches, LSI is not constrained to only find theorems containing given keywords and thus could provide results that whilst unexpected are not less useful. Too much diversity though would not be desirable but on average, over the 19,129 results, 31% of terms in the result theorems did not occur in the original query. This suggests that there is in fact a good degree of diversity and overlap between the query terms and the results terms.

Even if very low overlap does tend to indicate irrelevant results, it could be that such matches could be tolerated by the user. In addition, the common overlap indicator used here could also be given to the user to allow them to assess the possible value of the retrieved result in a different way.

3.4.3. *Qualitative evaluation of LSI-based search*

Though the figures obtained above give some sort of confidence in the LSI approach to searching formal mathematics, it is really only in the value of the retrieved results that LSI will prove its worth. Of course, it is impossible to quantify this as it depends enormously on the context in which the search is conducted. Thus, we attempt here to give some flavour that suggests that LSI is also an effective tool from a qualitative perspective. Because assessing the value of a result requires knowledge of the domain, the examples given here are all taken from general topology as this is the mathematical background of Cairns.

An exhaustive evaluation of a lot of theorems, whilst conclusive, may not be particularly illuminating. Instead, the examples here are chosen to illustrate particular sorts of search results that demonstrate the strengths and weaknesses of LSI. All queries were run using 100 factors and with 20 results returned.

The following theorem illustrates that LSI is actually associating logically related results. The search query was from `TOPS_3` (Karno, 1992):

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

As is usual, the first returned theorem was the very same theorem. More interestingly, the second returned theorem was (Karno, 1992):

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

These are quite distinct theorems since the concepts of boundary and nowhere dense are distinct. However, the concepts are also quite closely related, every nowhere dense set is boundary (TOPS_1:92) (Wysocki and Darmochwał, 1990) and hence the first theorem implies the second. LSI of course cannot have used this logical relationship but instead had identified the relationship on the basis that these two concepts occur frequently in related contexts.

Generally, such clear logical relations do not appear. This examples is taken from `TOPGEN_1` (Grabowski, 2005):

```
theorem :: TOPGEN_1:10
 Fr A = Cl A \ Int A;
```

which is about the relationship between the frontier or boundary of a set and its closure and interior. Most of the results for this query came from `TOPS_1` (Wysocki and Darmochwał, 1990) and concerned other relationships between the three concepts involved, for instance that the closure of set is the union of the set and its frontier. Thus, LSI is indeed returning theorems that are closely related to this one. More unusally, three results were taken from `TSEP_1` (Karno, 1991) and concerned the notions of separated and weakly separated sets. For example:

```
theorem :: TSEP_1:43
 for C being Subset of X holds
  A1,A2 are_separated implies
  A1 /\ C,A2 /\ C are_separated;
```

Whilst the attribute “are_separated” is not directly related to frontiers of sets, the notion is about the symmetrical overlap between one set and the closure of another. Thus, despite superficial differences in terms, these theorems from TSEP_1 are of possible relevance if the topic of work is frontiers of sets.

Other queries from TOPGEN_1 produced similar outcomes where the first few results were quite obviously relevant and later results in the top 20 were not immediately obviously relevant but it was reasonable to see why they could be considered so.

When it comes to more specialised topics though, LSI cannot necessarily be expected to perform well as the space of documents in a specialised area is likely to be under-represented in the overall corpus. One such theorem that was felt to be reasonably specialised was (Bancerek, 2005):

```
theorem :: TOPGEN_2:6
  for T being non empty TopSpace holds
    T is first-countable iff Chi(T) <=‘ alef 0;
```

This is making the relationship between a local property of the space, namely being first countable, and what are called cardinal functions on the space. Cardinal functions are an important tool for the study of topological spaces though they are not extensively formalised in the MML. Indeed, this is reflected in the search results from this query where there are very few theorems returned that clearly relate to this result. However, LSI does seem to be finding that the notion of a local property as relevant to this because some of the returned theorems refer to local connectedness, local compactness and convergence of a sequence to a point.

A more specialised result still is one taken from TOPGEN_3, (Bancerek, 2006) about the Sorgenfrey line which is a variant topology on the real line and is a useful example in topology:

```
theorem :: TOPGEN_3:33
  weight Sorgenfrey-line = continuum;
```

Needless to say that with such a specialised query, LSI did not return very much of relevance to this theorem. There were two results related to cardinality which makes sense as this is a theorem about cardinality. Other than that though, it was hard to see how most of the results had any relevance to this theorem.

Overall then, it seems that when a concept is well used in the MML, LSI is able to extract a wide range of relevant results, even ones where it is not immediately obvious why they are relevant. In some cases, as shown here, it can even find logically related theorems which could be very useful when looking for a theorem with which to construct a proof step.

Naturally, when a concept is less well represented in the corpus, LSI still is able to find some relevant results but their relevance is much less clear and therefore much less likely to be useful. In the extreme case, LSI cannot provide any useful results when concepts are extremely specialised and therefore poorly represented in the MML. This is not to say that the approach is invalid but it may be useful if somehow the user were able to get a sense of when a concept is poorly represented as opposed to LSI performing poorly. This may require another approach to mapping out the concepts formalised in the MML.

4. Implementation issues with XML-ization of Mizar

Urban's addition of XML to Mizar has made an enormous difference to the development of Alcor. Developing a suitable parser for Mizar has been the biggest obstacle to producing effective search, even just keyword search let alone more complex search such as LSI. There are still many things that can be done to improve the indexing and hence the effectiveness of both search methods, now that the XML-ization of Mizar has made such improvements technically feasible. Nonetheless, this implementation highlights some limitations of the current XML-ization of Mizar and it is worth pointing out where it could be improved.

The simplest problem with the XML-ization is that the original text is lost. Relating a definition, a theorem or just a line of proof back to the original text is not easy as the XML is significantly transformed from the original even when transformed back to a Mizar-like form by Urban's stylesheet miz.xsl. Whilst this is not a problem for machine tools building on the XML, it is a significant drawback for LSI when the idea is to capture the original language in order to exploit it as discourse. There are two significant areas where this has been an issue already.

Mizar clearly replaces some definitions with logically equivalent forms based on previously defined terms. For example, the definition of "odd" in (Rudnicki and Trybulec, 1992) is as an adjective which means "not even". However, in converting from Mizar to XML, no final XML article has the term "odd" in it. It has been consistently and completely removed in preference for "not even". This is, of course, logically correct but it loses the essential discourse element of defining the term odd, which clearly has connotations with how we usually talk about numbers. This means that an author working on odd numbers would not even be able to find the term using a keyword search! Furthermore, any informal relationships the term odd may have to other mathematical objects are inaccessible to LSI.

The other issue is that the logical structure of all statements is reduced using the logically complete subset of connectives “for all”, “not” and “and”. Again, this is perfectly acceptable logically but still the original discourse structure is lost. Consider, for instance, of the simple statement:

If A holds then either B or C holds.

In the reduced logic, this would be equivalent to

It is not the case that A holds and not B holds and not C holds.

This has a very different air to it not to mention clumsiness. Arguably, this formulation is more general and therefore more useful but the principle of using LSI is that the details of the discourse structure are important cues to help a mathematician. Whilst LSI cannot use the discourse structure, it would possible to refine LSI search results along these lines and so enhance the search results in terms of discourse not just logic. This would not be currently possible with the XML output of Mizar.

The simplest solution to this problem might be that the XML be given an attribute containing the original text that gave rise to a particular section of XML. Thus, at least in principle, it is possible to correctly reconstruct the original text. At the very least, it should be possible to give attributes to the XML that indicate the original logical structure and terms used in the Mizar statements.

Another problem of using the XML to parse Mizar articles is the linking with queries as implemented in Alcor. The generation of queries is always going to be inelegant whilst it is not possible to parse Mizar expressions independently of Mizar itself. Ideally, it should be possible to produce a correct parse of any Mizar statement without doing a proof check. Moreover, if terms in the query are not known, these should be easily replaced with the actual terms used by the user. We understand that Mizar does not work this way for a number of technical and historical reasons, but the ideal is still worth holding in mind for future developments.

5. Implications for Integrated Proving Environments

The purpose of developing Alcor is to explore integrating search and authoring when writing proofs. As such, even though Alcor is still in the early stages of development, and certainly has not been used for user testing, it is still possible to identify issues for the integration of searching and authoring.

The two styles of search are clearly necessary, reflecting the need for retrieval of specific objects versus the less constrained search for “something like this” during proof development. Also, it may be possible to use more of the formal structure, such as the logical structure of statements and logical relationships between concepts to improve and refine search results found with a more informal method. The decision of which search methods to use is currently on the basis of number of search terms involved. This is clearly a very crude heuristic and it is not clear how best to make different styles of search available to the user.

Using LSI to reflect the more informal sorts of search that mathematicians might do is not without its problems. The most obvious problem is defining what constitute appropriate documents and terms. We anticipate that it will be relatively straightforward to convert more of the XML-ized articles into documents, specifically the full proofs and definitions. These are crucial elements in the discourse of mathematics and as such reflect a lot of the informal relationships between terms and concepts in mathematics. Even so, LSI is known not to work well if documents are too short. It is possible that some definitions would not make good documents on this basis. However, it is to formal mathematics advantage that it is somewhat more verbose than ordinary mathematics so that proofs in normal mathematical language end up being four or even ten times longer in formal mathematics (Wiedijk, 2000). In this way, formal proofs may actually be an appropriate size for effective documents in LSI.

Like many search methods, there are parameters to set. Possibly the most crucial one in LSI is specifying the dimensionality of the underlying singular value decomposition. Without some form of reduced dimensionality, LSI is not generalising the meaning of terms to other terms yet with too much reduction, it may not sufficiently distinguish concepts. The testing suggests that fewer than 100 factors in the current implementation does increase the chances of retrieving irrelevant theorems. However, this comes at a cost as the retrieval of results using 100 factors is appreciably slower than with fewer factors. Thus, the testing suggests a range of factors that could be tested with users. 100 factors would produce more reliable results but at a time cost and 15 factors produces worse results but more quickly. Depending on what users want from the search it could be possible to settle on a good compromise between these two values. In addition, the number of overlapping terms could be used as a second indicator of the relevance of a result and users could use this with the LSI similarity measure together.

There is also an interaction between LSI and the language of mathematics. For instance, the symbol '=' is abundant in mathematics but if considered fully logically has many different meanings. Equals therefore causes many incidental links between terms when LSI is applied to mathematical documents. If you like, equals acts to bring lots of concepts together that are not genuinely related. Leaving equals out of LSI analysis is the natural decision but which other symbols or terms ought to be omitted on the same grounds? Given this issue, the current work has proceeded with including all but Mizar keywords as terms. This should be re-examined though if it is felt that the quality of LSI results needs to be improved.

In many ways, though, it is the implicit nature of LSI that is its biggest drawback. It is effectively impossible to make analytical judgments on what would improve the search results. Using trial and error to set parameters and choose appropriate sets of documents and terms, whilst acceptable for one system such as Alcor, becomes problematic and essentially atheoretical across many systems.

In due course, it may become clear that LSI is not the most appropriate approach to addressing informal "things like this" queries. This is not a problem for Alcor as it is intended to bring together different tools to try them out. However, we do strongly believe that some sort of informal analysis, orthogonal to the formal structure of the MML is going to be essential to support mathematicians. Urban is currently looking at alternative approaches to analysing the MML (Urban, 2004) and this may provide an interesting alternative or complement to LSI in Alcor.

In any case, the next stage in the development of Alcor is to make both the searching and authoring aspects more robust and to at least match what is currently available to users of Mizar. In this way, it will be possible to make a fair comparison on the advantages or disadvantages of this sort of environment. We believe this would be a valuable contribution to the understanding of user interfaces that would support mathematicians in their work.

6. Related Work

The Mizar community has developed a number search tools to support authoring of Mizar articles. All of this work is integrated into MizarMode[1] for the Emacs text editor, included of the Mizar distribution. The mode provides standard Emacs functionality that allows users to search for a theorem or definition given a unique identifier.

[1] `http://ktilinux.ms.mff.cuni.cz/~urban/MizarModeDoc/`

MML Query is a database-like query system in MizarMode that allows more advanced browsing and search of the MML (Bancerek & Rudnicki, 2003; Bancerek & Urban, 2004). Users can use MML Query to retrieve items based on Mizar syntax and the the structure of the MML. For example, one can search for an ordering over functions by querying for definitions (constructors) based on concept of functions (`FUNCT_1:attr 1`) that use `<=` or `>=` in their notation. Although the language allows precise and complex queries to be constructed, there are potential disadvantages from a user perspective: they may lack background knowledge about the MML or find it difficult to express their needs in the query syntax. However, its widespread use suggests that it provides an effective tool for expert users.

The MoMM system (Urban, 2006) performs fast, non-redundant retrieval of theorems from the MML that match a given formulae. The system is based on efficient term-indexing techniques from automated theorem proving, and is designed to return the strongest available version of a search result. Retrieval of multiple redundant results is a potential problem in large libraries where several theories may have independant;y developed slight variations on a single result. MoMM has the advantage for users that unnecessary search results are filtered out, making it more likely that relevant results will be identified.

A third search system that integrates into MizarMode is the Mizar Proof Advisor (Urban, 2004), which uses machine learning over the MML to retrieve results that *may* be useful in proving a given goal. That is, it suggests results based on the results that have already been used to prove similar goals. Evaluation using the MML showed that on average about 50% of the items actually used appear in the first 20–30 search results, and 70% are in the first 100. From a user perspective this is already a very large set of results, so in practical terms there will be a significant minority of items that could be used in the proof but that will not ranked highly by the Advisor. However, it is clear that the system can relieve the user of some of the burden of search while authoring articles.

The Whelp system provides search over the Coq library (Asperti et al., 2006). Four kinds of search are currently supported: *Locate* finds a library item given its identifier; *Elim* retrieves induction principles for a given type; *Match* retrieves results that match a given goal or goal pattern; *Hint* find results that may be applied to the given goal, in that they have a matching antecedant. Whelp returns results which are syntactically related based on an abstract model of terms. Previous work on information retrieval in Coq had used type isomorphisms to locate matching lemmas (Delahaye, 2000).

Comparing Alcor to the state-of-the-art search in formal mathematics, we see that all these systems can play the role of 'advancing the proof', just as our LSI search has been designed to support. MoMM and Whelp's Match operation find equivalent results to a given goal. They can tell the user if a similar result has already been established. The Proof Advisor and Whelp's Hint retrieve results that can be applied to a given goal. They allow the proof to be advanced, but not immediately completed. In contrast, both MML Query and Alcor's LSI search do not base relevance on known *applicability* — they don't require a result to match syntactically or have been used in a known proof. Nevertheless, such a result may be still be useful in suggesting ways the proof can progress. The motivating result for a proof idea need not be directly applicable nor have been used in this context before.

Another important factor is that MML Query, MoMM and Whelp are based on purely syntactial notions of relevance. This limits results to items that are syntactically 'similar' in some way. In contrast, the Proof Advisor and Alcor's LSI search both take account of the semantics implicit in the MML when defining relevance. These methods can return results that are not directly related by syntax, but nonetheless useful in moving the proof forward (as discussed above). In summary, Alcor has potential advantages over current state-of-the-art because relevance is not defined in terms on applicability or pure syntax. However, relative performance and utility and the potential for integration of these techniques are still open questions.

7. Conclusions

The goal of a full mathematician's assistant is still some way off but such systems seem to promise a great deal particularly when considered as tools for mathematical knowledge management. The Alcor system, therefore, is a test bed for exploring how the user interface and interaction of such an assistant could work. In particular, we have developed Alcor to investigate how to integrate the activities of authoring and searching whilst proving. The Mizar system and MML provide an excellent resource and foundation for these activities. We have identified at least two sorts of search that could be useful to mathematicians and implemented them in Alcor, namely the search for specific, known objects and the more informal search for "something like this". Both go some way to fulfilling a scenario of how a mathematician works but also Alcor highlights the difficulties of making such search effective and useful to the user. In particular, Alcor's LSI search has a unique notion of relevance compared to other search systems for formal mathematics, and we have argued that this has potential advantages for authors. However, it is clear that more work needs to be done to compare and perhaps integrate this approach with other approaches also currently being developed.

Acknowledgements

Many thanks to Mike Berry for providing the GTP code. Also, thanks to Eduardo Calvillo and the referees for their very useful comments on this paper.

References

Adams, A.A. Digitisation, Representation and Formalisation: Digital Libraries of Mathematics. In (Asperti et al., 2003), pp. 1–16.

Asperti, A., B. Buchberger & J.H. Davenport, editors. *Mathematical Knowledge Management, 2nd Int. Conf., MKM 2003*, LNCS 2594, Springer, 2003.

Asperti, A., G. Bancerek and A. Trybulec, editors. *Mathematical Knowledge Management, 3rd Int. Conf., MKM 2003*, LNCS 3119, Springer, 2004.

Asperti, A., F. Guidi, C. S. Coen, E. Tassi & S. Zacchiroli. A Content Based Mathematical Search Engine: Whelp. In J. Filliâtre, C. Paulin-Mohring & B. Werner (eds), *Types for Proofs and Programs, International Workshop, TYPES 2004*, pp. 17–32, LNCS 3839, Springer 2006.

Bancerek, G. and P. Rudnicki. Information Retrieval in MML. In (Asperti et al., 2003), pp. 119–132, 2003.

Bancerek, G. and J. Urban. Integrated Semantic Browsing of the Mizar Mathematical Library for Authoring Mizar Articles. In (Asperti et al., 2004), pp. 44–57, 2004.

Bancerek, G. On the characteristic and weight of a topological space. *Formalized Mathematics*, 13:163–169, 2005.

Bancerek, G. On constructing topological spaces and Sorgenfrey line. *Formalized Mathematics*, 13:171–179, 2005.

Bothner, P. JEmacs — the Java/Scheme-based Emacs Text Editor. `http://jemacs.sourceforge.net/`, 2006

Buchberger B., G. Gonnet and M. Hazewinkel. Preface. *Annals of Mathematics and Artificial Intelligence*, 38:1–2, 2003.

Bundy, A., M. Atiyah, A. Macintyre and D. MacKenzie (eds). The nature of mathematical proof. *Phil. Trans. R. Soc. A*, 363(1835): 2329–2461, 2005.

Cairns, P. Alcor: A user interface for Mizar. *Mechanized Mathematics and its Applications*, 4(1):83–88, 2005.

Cairns, P. Informalising Formal Mathematics. In (Asperti et al., 2004), pp. 58–72, 2004.

Cairns, P. and J. Gow. Using and Parsing the Mizar Language. *Electronic Notes in Theoretical Computer Science*, 93:60–69, 2004.

Cooper, A. *The Inmates are Running the Asylum.* SAMS, Macmillan Computer Publishing, 1999.

Delahaye, D. Information Retrieval in a Coq Proof Library Using Type Isomorphisms. In T. Coquand, P. Dybjer, B. Nordström & J. M. Smith (eds), *Types for Proofs and Programs, International Workshop, TYPES'99*, pp. 131–147, LNCS 1956, Springer 2000.

Farmer, F. and M. Mohrenschildt. An Overview of a Formal Framework for Managing Mathematics. *Annals of Mathematics and Artificial Intelligence*, 38:165–191, 2003.

Grabowski, A. On the boundary and derivative of a set. *Formalized Mathematics*, 13(1):139–146, 2005.

Java Compiler Compiler. `https://javacc.dev.java.net/`, 2006.

Jones, S. and M.S. Stavely. Phrasier: a system for interactive document retrieval using keyphrases. In *Proc. of 22nd ACM SIGIR Conf. on Research and Development in Information Retrieval*, 160–167, 1999.

John, B. and D. Kieras. The GOMS Family of User Interfaces Analysis Techniques: Comparison and Contrast. *Transactions on Computer-Human Interaction*, 3(4):320–351, 1996.

Journal of Formalized Mathematics. `http://mizar.org/JFM/`, 2006

Karno, Z. Separated and Weakly Separated Subspaces of Topological Spaces. *Formalized Mathematics*, 2:665–674, 1991.

Karno, Z. Remarks on special subsets of topological spaces. *Formalized Mathematics*, 3:297–303, 1992.

Kay, M. *XSLT Programmer's reference, 2nd edition.* WROX Press, 2001.

Kerber, M., M. Kohlhase, V. Sorge. Integrating Computer Algebra into Proof Planning. *Journal of Automated Reasoning*, 21(3):327–355, 1998.

Kohlhase, M. and R. Anghelache. *Towards Collaborative Content Management and Version Control for Structured Mathematical Knowledge.* In (Asperti et al., 2003), pp. 147–161, 2003.

Kornilowicz, A. and Y. Shidama. Inverse trigonometric functions arcsin and arccos. *Formailized Mathematics*, 13(1):73–79, 2005.

Lakatos, I. *Proofs and Refutations.* Cambridge University Press, 1976

Landauer, T.K., P.W. Foltz and D. Laham. Introduction to Latent Semantic Analysis. *Discourse*, 25:259–284, 1998.

MacKenzie, D. *Mechanizing Proof: Computing, Risk and Trust.* MIT Press, 2001.

Newman, W. and M. Lamming. *Interactive System Design.* Addison-Wesley, 1995.

Parr, T. AntLR. `http://www.antlr.org/`, 2006.

Rosson, M.B. and J.M. Carroll. *Usability Engineering: Scenario-based development of Human-Computer Interaction.* Academic Press, 2002.

Rekers, J. and A. Schürr. A Parsing Algortithm for Context-Sensitive Graph Grammars. Technical Report 95-05, Leiden University, 1995

Rudnicki, P. An overview of the Mizar project. In *Proceedings of 1992 Workshop on Types and Proofs for Programs*, 1992.

Rudnicki, P. and A. Trybulec. Abian's Fixed Point Theorem. *Formalized Mathematics*, 6(3):335–338, 1992.

Thimbleby, W. A novel pen-based calculator and its evaluation. In *Proc. of 3rd Nordic Conf. on Human-Computer Interaction*, ACM Press, 445–448, 2004.

Urban, J. MPTP - motivation, implementation, first experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.

Urban, J. XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. In M. Kohlhase, editor, *Mathematical Knowledge Management, 4th Int. Conf.*, Springer Verlag LNCS 3863:346–360, 2006.

Urban, J. MoMM — Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics. *Int. J. on AI Tools*, 15(1):109–130, 2006.

Wiedijk, F. The De Bruijn Factor. Poster at *TPHOL 2000.*

Wysocki, M. and A. Darmochwał. Subsets of Topological Spaces. *Formalized Mathematics*, 1:231–237, 1990.

Zinn, C. *Understanding Informal Mathematical Discourse*. PhD Thesis, Arbeitsberichter des Instituts für Informatik, Friedrich-Alexander-Universität, 37(4), 2004.