

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*

31st January, 2006

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 this two distinct types 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 already been described (Cairns, 2005), the version described here has the new functionality of search over the Mizar library based on latent semantic indexing (LSI)(Cairns, 2004) and thus has two distinct search styles. The LSI-based search method has also 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, the 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 and 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 extensive (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 an mathematical assistant

and how the functionality of a mathematical 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 however also has a drop-down menu to allow the author to select previous searches and so acts as search history.

There are two distinct algorithms implemented in Alcor for conducting a search corresponding to the two styles of search. The first algorithm is simply a keyword search that looks up the definition of terms, the second is based on latent semantic indexing (LSI). At the moment, a simply heuristic is used to distinguish the two types of

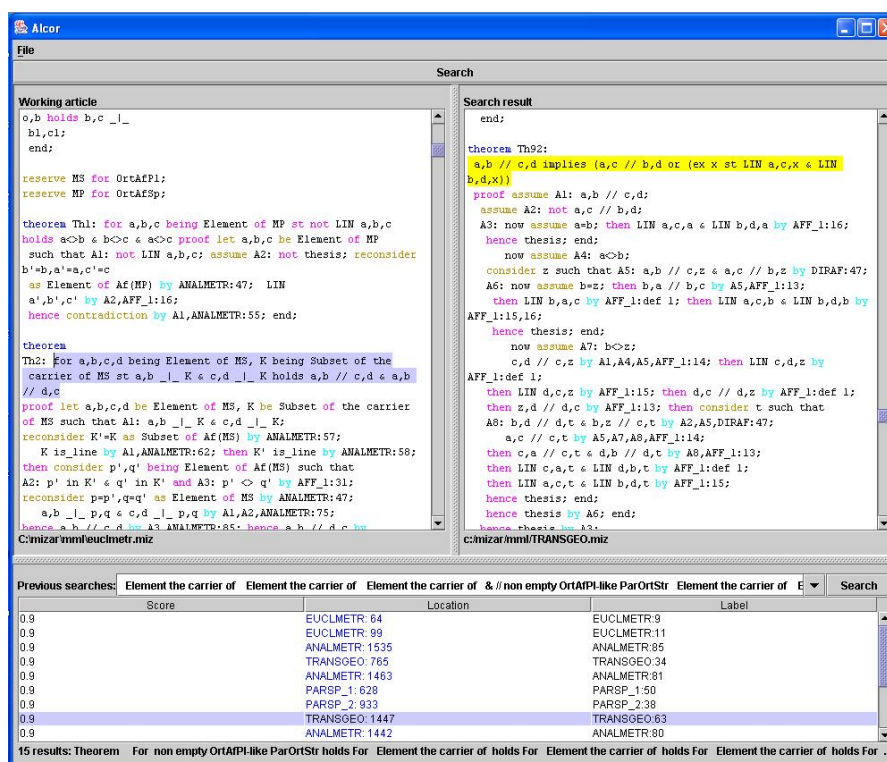


Figure 1. The Alcor system

search: if the query consists of a single term then it is a keyword search otherwise it is an LSI search. Each type of search is considered in turn.

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. The second search method is trying 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).

Of course, 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 of humans. 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 currently available technique that is able to provide the informal relationships between concepts that is required in the second sort 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 through relating terms to the documents in which they occur thus forming a co-occurrence matrix of terms and documents. Singular value decomposition of this matrix produces a reduction of the matrix to a diagonal matrix where the diagonal entries effectively represent the eigenvalues of the original matrix. The non-zero entries therefore define a term-document space underlying the original matrix and it is the structure of this space that is the latent meaning of the terms and the documents as defined by the corpus.

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. Moreover, the underlying space can be reduced in dimensions by neglecting dimensions corresponding to small eigenvalues in the singular value decomposition. 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. 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. At the moment, Alcor is working at the level of each theorem in the MML being a document for LSI. Thus, when the user enters a search query in Alcor, the query is processed by LSI to retrieve theorems from the MML that ideally are semantically related in some way to the query.

Even though only theorems are currently being used as documents for LSI, there are good indications that LSI-based search is working as hoped. Any query based on an existing theorem does in fact return that theorem, usually as the first hit. This indicates that despite the implicit nature of meaning in LSI, the original meaning of documents is not lost. One particular search of this sort was on the theorem (Karno, 1992):

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

As expected, 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. This vindicates the choice of LSI as a basis for searching across the informal relationships between mathematical concepts.

3.2. 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

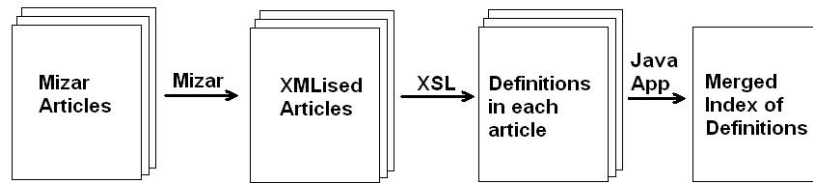


Figure 2. Generating the keyword index from MML

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 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

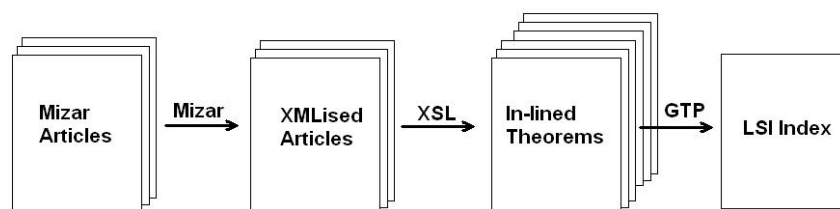


Figure 3. Applying LSI to the MML

articles. 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.¹ 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.

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 is has a very different air to it not to mention clumsiness. Though this detail of logical structures is not currently used in the LSI

implementation, some sort of refinement of search could be done on the logical structure. 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 but, thanks to the verbosity of formal proofs (Wiedijk, 2000), proofs are not likely to have this problem.

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.

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? And, as such decisions are really a matter of degree rather than absolutes, how will these choices be made?

For the purposes of Alcor, setting parameters, documents and terms can largely be done by trial and error. This however is not satisfactory for if Alcor is to inform more general systems, how are these decisions to be made for other bodies of mathematical knowledge? In this case, the implicit nature of meaning in LSI is potentially its biggest drawback.

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. 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. Through further developing Alcor into a fully functional searching and authoring environment, we hope to significantly inform the design of future mathematical assistants.

Acknowledgements

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

Notes

¹ As a technical aside, the Mizar verifier is called from Alcor using the Java Runtime object's `exec()` method. This is nowhere near as simple as the API documentation suggests and Daconta's excellent on-line article (Daconta, 2006) is extremely helpful in getting it to work properly.

References

- Adams, A.A. Digitisation, Representation and Formalisation: digital libraries of mathematics. In A. Asperti, B. Buchberger and J.H. Davenport, editors, *Mathematical Knowledge Management*, LNCS 2594, 1–16, Springer Verlag, 2003.
- Bothner, P. JEmacs - the Java/Scheme-based Emacs Text Editor. <http://jemacs.sourceforge.net/>

- 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 A. Asperti, G. Bancerek and A. Trybulec, editors, *Proc. of the Third Int. Conf. on Mathematical Knowledge Management*, LNCS 3119, 58–72, Springer Verlag, 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.
- Daconta, M. When Runtime.exec() won't.
<http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html>
- Farmer, F. and M. Mohrenschildt. An Overview of a Formal Framework for Managing Mathematics. *Annals of Mathematics and Artificial Intelligence*, 38:165–191, 2003.
- Java Compiler Compiler, 2006 <https://javacc.dev.java.net/>
- Jones, S. and M.S. Staveland. 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.
- 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 A. Asperti, B. Buchberger and J.H. Davenport, editors, *Mathematical Knowledge Management*, LNCS 2594, 147–161, Springer Verlag, 2003.
- 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. *Journal of Formalized Mathematics*. <http://mizar.org/JFM>, 2006
- Newman, W. and M. Lamming. *Interactive System Design*. Addison-Wesley, 1995.
- Parr, T. AntLR, 2006 <http://wwwantlr.org/>
- 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 Algorithm 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. XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. In M. Kohlhase, editor, *Mathematical Knowledge Management, 4th Int. Conf.*, LNCS 3863, 346–360, Springer Verlag, 2006.
- Urban, J. MPTP - motivation, implementation, first experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.
- 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, Arbeitsbericht des Instituts für Informatik, Friedrich-Alexander-Universität, **37**(4), 2004.

