Towards Integrating Mathematical Libraries via Alignments

Bachelor Thesis in Applied and Computational Mathematics

Author:
Roxana Nadrag

Supervisors:
Dr. Florian Rabe
Prof. Dr. Michael Kohlhase
Dr. Keivan Mallahi-Karai

May 26, 2015

Declaration

With my signature, I certify that this paper has been written by me using only the indicated resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Roxana Nadrag
May 26, 2015

Abstract

Formalization of mathematical knowledge is becoming very important as more and more theorem provers and reasoning assistants have been developed over the last decades. As a result of this growth, we have systems with incompatible foundations and libraries with many overlapping concepts. The challenge with these systems is they have grown to be so large that there is no straightforward way to allow them to share knowledge.

As a first step, we envision a single framework for different logical systems united under a common interface. Within this framework we use alignments as a yet imperfect but lightweight and scalable approach.

1 Introduction

Mathematical libraries have been increasingly developing over the last years, as they represent the core of formalizing theorems and algorithms. These libraries have been extended continuously and as a result, reusing parts of one library into another library comes natural once the systems grow. However, such libraries are usually mutually incompatible and rather complex, each having been developed from a different foundation and by different communities. One step towards making cross-library integration is to export the knowledge available in one library to the others, however, this step could result in a loss of the high-level structures which are a core part of the system. A better solution is to have each library exported to a logical framework and later imported by other tools as it is envisioned in the OAF project [KR14b] for various proof assistant libraries like Mizar [IKR11] and HOLLight [KR14a]. These two libraries also serve as a good example of two popular methods to formalize proofs: procedural (HOLLight) and declarative (Mizar) [Wie01]. While the former is a commonly used paradigm for proof assistants, the latter is closer to mathematical vernacular, hence, easier to understand by humans but harder to understand from a machine's perspective.

HOLLight [Har96a] implements the HOL logic, a variation of Church's simple type theory extended by shallow polymorphism, being the second of the four HOL systems. HOLLight started as an experiment, a lightweight version of the HOL foundation, but grew into a large HOL logic variant retaining the minimalist foundation principle it was originally built for [Har09]. HOLLight was used together with Isabelle to assist with the formal proof of the Kepler conjecture in the Flyspeck project [Hal05] which finished successfully in August 2014 [1].

Mizar [TB85] implements first order set theory (based on Tarski Gröthendieck set theory) and it is a representation format for mathematics which stays close to usual mathematical language The Mizar project includes the Mizar Mathematical Library (MML)[2] which is a big coherent body of strictly formalized mathematics with over 1000 articles written by more than 200 authors and more than 10,000 formal definitions with 52,000 theorems [Wie99].

Integrating different proof assistants would have the advantage of exposing the knowledge in each system and allows an end user to work with a uniform interface without having to understand each logic. However, the problem with proof assistants which are built on

[1] https://code.google.com/p/flyspeck/wiki/AnnouncingCompletion
[2] http://www.mizar.org/library

different foundations is they are usually mutually incompatible mathematical bodies despite having lots of overlapping concepts in their libraries. Since they are so different, it is hard to capture the overlaps formally and, as a result, it is very difficult to create a common knowledge base across such libraries.

Contribution We introduce alignments which are tuples of concepts (with similar semantics) from different libraries. We seek for an alignment between different logical systems. We define alignments and further use this concept to align Mizar and HOLLight for a selected subset of symbols (booleans, arithmetics, sets). We also use alignments as part of a navigation interface which enables an end user of the MMT Web Server Interface to navigate between aligned symbols. Alignments will be performed based on **MMT URIs**[RK13] (unique identifiers of symbols) with the goal of having all symbols with similar semantics grouped together.

The purpose of this thesis is to propose an approach that will raise above a specific library and focus on a general and scalable solution. The main goal is to prove that this idea will extend the knowledge base for each concept, thus, making concepts in each library more discoverable. To this end, we do not focus on gathering a large database of alignments, but rather, proving that with this approach one can increase the discoverability of concepts across libraries.

Outline This thesis is organized as follows: this section has introduced the research problem and the solution we propose, in section 2 we present the related work, section 3 introduces several important concepts used in the upcoming sections, in section 4 we introduce and formally define alignments. Section 5 will focus on alignments between Mizar and HOLLight, section 6 presents an application of alignments and finally, in section 7 we discuss our results and future work.

2 Related work

Formal alignments To understand why it is difficult to integrate formal libraries, one needs to understand that these deductive systems are usually built on a fixed foundation (fixed logic) [KR14b] and so, integrating different formal systems is a complex project.

The HOL-based logics are good candidates for obtaining alignments since they share the same foundation. This advantage has been explored in research projects in many forms. Importing proofs from HOL 4 and HOLLight into Isabelle/HOL in [OS06] enables mapping concepts across these libraries which leads to a good integration of imported theorems with the ones already available in Isabelle/HOL. There is also an interpretation of a significant part of Isabelle/HOL into HOLLight discussed in [McL06] which is interesting because Isabelle/HOL supports features which HOL does not support like constant overloading or axiomatic type classes.

The most recent methods employ machine learning for automatic discovery of isomorphic structures between HOL-based systems[GK14]. Making use of patterns and properties of the concepts by using scoring functions for measuring similarity between concepts the authors are able to obtain valuable results.

Integrating libraries with different foundations requires a greater effort and there are many attempts to solve this problem. The HOL Light library has been translated to Coq [KW10] and the Coq library has been imported into Matita facilitated by the fact that both used very similar foundations at the time.

Reconstruction of the complex Mizar type system in HOLLight and the partial implementation are outlined in [Kun10]. Another approach is to layer a declarative interface on top of the procedural system as presented in [Har96b].

An attempt to prove that different foundations can be integrated seamlessly is the implementation of the Mizar proof language on top of HOL discussed in [Wie01] where the key idea is to implement Mizar 'steps' as HOL 'tactics'.

3 Preliminaries

MMT is a foundation-independent knowledge management system which treats specific foundations (ZFC and LF) as theories and the associated semantics as theory morphisms [RK13]. MMT is intentionally not bound to certain type systems or logics and focuses on providing an open and extensible interface.

The architecture of the MMT system [Ian13] (see Figure 5) has several elements among which, the MMT API constitutes the core component. MMT API is a scala-based implementation of the MMT language [RK13]. The API offers a generic plugin interface and implements several backends and frontends [Rab13].

MMT Web is a web-based MMT application which uses the MMT HTTP API and enables interactive browsing using HTML + MathML and JOBAD [GLR09]. MMT is designed to be web-scalable and it uses URIs also known as **MMT URIs** for uniquely identifying knowledge items. The URIs are then mapped into URLs, indicating the physical location of the pointed elements. In MMT, all constants that are available in a theory have canonical MMT URIs. These are structured as URIs doc?mod?sym formed from a document URI doc, a module name mod, and a symbol name sym [RK13].

Considering all of the above, the MMT system is a suitable platform for integrating formal libraries.

OAF project [KR14b] aims to provide a universal archiving solution for formal mathematical libraries by having a generic framework suitable for all logics but also be aware of the semantics of the formalized content. The project focuses on archiving large-scale logics and both Mizar and HOLLight are part of the current archives available in the MMT system. Both libraries have been formally imported into MMT as described in [IKR11]and respectively in [KR14a] so they constitute perfect candidates for our research.

The goal of this thesis is to find similar concepts between Mizar and HOLLight and unify them under a foundation-neutral system (MMT), using MMT URIs.

4 Alignments

The **key idea of alignments** is to use tuples of URIs from different libraries to represent the information in which MMT declarations formalize a concept. We introduce the **interface library** as an additional library which is a reference point for all mathematical concepts. For our purposes, we see the interface library as a set of MMT URIs. An existing example of a library which we can consider an interface library is the set of Open Math Standard Content Dictionaries [Ope09]. Another interface library is the LATIN library [KMR09] which provides interfaces for logics.

Definition 4.1 (Realization)**.** We say a concept d declared in L (data library) realizes a concept c declared in I (interface library) if the intended semantics of d is c. We call the pair (c, d) a **realization pair** as illustrated in Figure 1.

Figure 1: d realizes c

Definition 4.2 (Alignments of n libraries)**.** Let us assume an interface library I and a set of data libraries L_1, L_2, ... ,L_n and further, let A be a set of realization pairs, then we say $s_i \in L_i$ **is aligned with** $s_j \in L_j$ under $s \in I$ iff $(s,\, s_i) \in A$ and $(s,\, s_j) \in A$ $(1 \leq i, j \leq n)$ as seen in Figure 2.

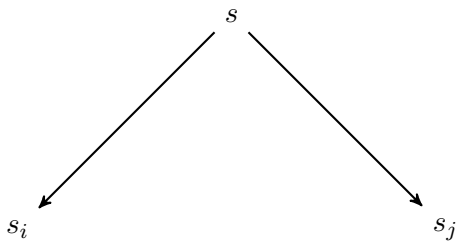

Figure 2: both s_i and s_j realize s so they are aligned

Alignments relate n libraries to the interface library and as an immediate gain from this definition, instead of having n^2 connections we have only n connections as it can be observed in Figure 3.

Alignments between library L and library I can be seen as a precursor of theory morphism from I to L since the interface library is just a library with concept names.

It is worth noting that the alignment relation is reflexive, symmetric and transitive so, it is an equivalence relation.

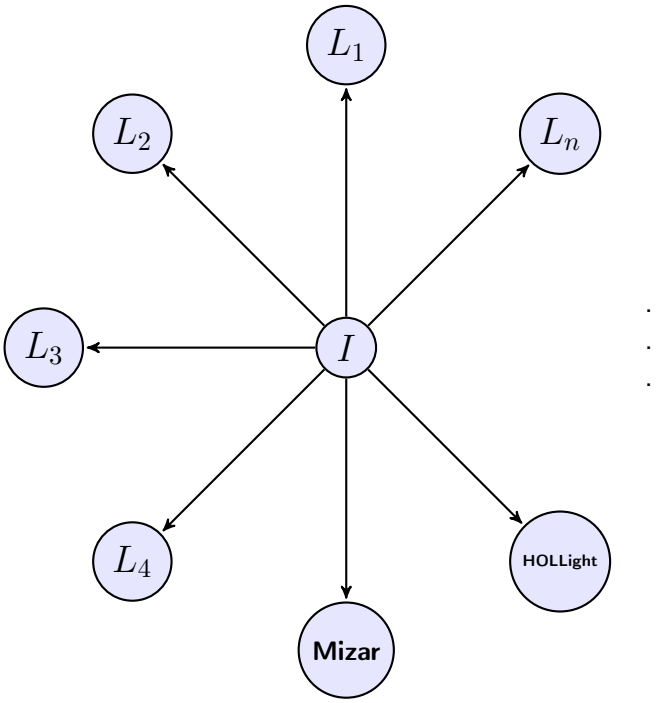

Figure 3: n libraries aligned under the interface library

5 Towards Aligning the HOLLight and Mizar Libraries

As an initial case study, we apply our approach to HOLLight and Mizar which are already available as MMT archives within the OAF project. We are collecting the realization pairs manually and for this reason we have selected the following subset of concepts to work with: **arithmetics**, **booleans** and **sets**. These concepts are broad enough to give us a better understanding of how aligned the two libraries might be but are still simple enough so that we can find alignments manually to work with and later on use in our application.

As we will work with MMT URIs which can be quite verbose we will simplify the notations by introducing namespaces representing each library.

hol	http://code.google.com/p/hol-light/source/browse/trunk
mizar	http://oaff.mathweb.org/MML/5.22
latin-mizar	http://latin.omdoc.org/foundations/mizar?Mizar-Curry
latin-hol	http://latin.omdoc.org/foundations/hollight?Kernel

Table 1: namespaces

So, for example, by writing $hol?file?sym$ where $file$ is the file name where the concept is found and sym is the concept itself we will understand it refers to the following MMT URI (the same holds for the mizar namespace):

http://code.google.com/p/hol-light/source/browse/trunk?file?sym

Latin contains the logic definitions of Mizar and HOLLight using LF [HHP93]. Latin/Mizar is the foundation of Mizar represented in LF on which the Mizar library import is

done (the same holds for Latin/HOLLight). So, the latin-mizar and latin-hol namespaces are different from the first two and will give us further URIs for concepts in Mizar and respectively HOLLight of the form latin-mizar?sym and latin-hol?sym.

For the purpose of this thesis, we fix the set of concepts and assign to each such concept a M-MMT URI (meta MMT URI) which we think of, as a pointer to MMT URIs[3] as we will see in table 2 which presents the interface library we are working with.

mmt?BOOL	mmt?T	mmt?F	mmt?AND	mmt?OR	mmt?NOT	
mmt?NAT	mmt?INT	mmt?ZERO	mmt?ADD	mmt?SUB	mmt?MULT	mmt?DIV
mmt?IN	mmt?INTER	mmt?UNION	mmt?EMPTY			

Table 2: concepts in the interface library

The M-MMT URI mmt?NOT for example, expresses a meta MMT URI which will point to actual MMT URIs with semantics of boolean negation.

The naive approach to perform alignments would be to search for the same symbol name in both libraries and align what we find under that concept name. This idea, however, will not result in a valuable alignment. To understand why this is not what we want, let us look at how this approach would go in practice: we search for zero in HOLLight and we find one result (hol?bool?_0), by checking the source definition we conclude is the right meaning of zero (neutral element with respect to addition for the real or natural numbers). Now, doing the same for Mizar we find over 100 files where the name 'zero' exists. By looking at the sources we realize that only a few of those files have the definition of what we are looking for, while the rest carry a different meaning (for example, zero in the context of categories of groups which is not what we want).

Having established that we need human intelligence to find alignments, we search both libraries and we present the results for the chosen concepts. Many alignments are not intuitive by just reading off the URIs because most of the file names in Mizar do not give information about the content or because the symbols have names which do not reveal their semantics. To solve this issue, we have explained some of the alignments in this document and we provide a separate file on github [4] which contains additional information with description of each file name used and definitions for the selected symbols from HOLLight and Mizar.

Booleans

mmt?BOOL	latin-hol?bool
mmt?T	hol?bool?T
mmt?F	hol?bool?F
mmt?AND	hol?bool?/\
mmt?OR	hol?bool?\/
mmt?NOT	hol?bool?~

Table 3: Realization pairs for booleans in HOLLight

[3] using M-MMT URIs we preserve consistency of working only with URIs

[4] https://gist.github.com/rnadrag/23d7e2dcdd14d2768a57

mmt?BOOL	mizar?MARGREL1?K5
mmt?BOOL	mizar?XBOOLEAN?V1
mmt?T	latin-mizar?true
mmt?T	mizar?XBOOLEAN?K2
mmt?T	mizar?MARGREL1?K7
mmt?F	latin-mizar?false
mmt?F	mizar?XBOOLEAN?K1
mmt?F	mizar?MARGREL1?K6
mmt?AND	latin-mizar?and
mmt?AND	mizar?MMLQUERY?NK17
mmt?AND	mizar?XBOOLE_0?K3
mmt?OR	latin-mizar?or
mmt?OR	mizar?MMLQUERY?NK18
mmt?OR	mizar?XBOOLE_0?K2
mmt?NOT	latin-mizar?not
mmt?NOT	mizar?XBOOLEAN?K3

Table 4: Realization pairs for booleans in Mizar

We can see that alignments are not as simple as they might appear from the definition 4.2 since some concepts which are part of one language's library are built into the another language and so, aligning these libraries is not trivial. For example, in Mizar, all logical operators are built-in and the library defines additional types for boolean while in HOL-Light only the type of booleans is built-in and we can see this reflected in the alignments we have gathered for the concept of boolean with M-MMT URI mmt?BOOL in tables 3 and 4.

We present a detailed table for obtaining the realizations defined in 4.1 for the concept of truth (with M-MMT URI mmt?T) in Mizar which are afterwards aligned with the correspondent concept from HOLLight.

MMT URI and file description	**Source definition**
mizar?XBOOLEAN?K2 (xboolean: arithmetic of boolean values)	`func TRUE -> set equals :: XBOOLEAN:def 2 1;`
mizar?MARGREL1?K7 (margrel1: many-argument relations)	`:: original: TRUE redefine func TRUE -> Element of BOOLEAN ; coherence by TARSKI:def 2;`
latin-mizar?true (foundations/mizar gives the encoding of Mizar and its semantics in terms of ZFC)	`%% proposition constructors true : prop.`

Table 5: 'true' in Mizar

As we can see from table 3 in HOLLight there is only one concept which realizes truth since the logical operators are defined in the library. However, in Mizar there are three such concepts (see table 4) since on one hand there is the Latin/Mizar atlas declaring Mizar in LF (for type checking Mizar in MMT) which allows for the Mizar Library import. On the other hand, in the Mizar Library itself the concept appears twice: defined and afterwards redefined as 'Element of BOOLEAN' which is a set $= \{0, 1\}$. We also present

the corresponding realizations aligned together for the selected example in Fig.4.

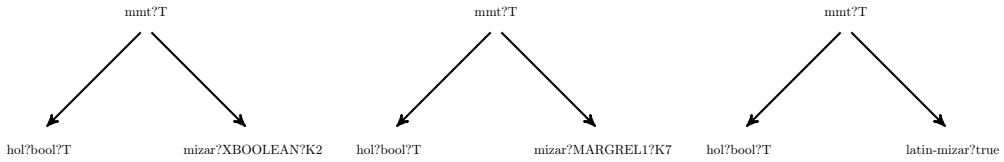

Figure 4: Mizar and HOLLight aligned for T

Arithmetics

mmt?INT	mizar?INT_1?NM1
mmt?INT	mizar?NUMBERS?K4
mmt?INT	mizar?INT_1?V1
mmt?NAT	mizar?NUMBERS?NK1
mmt?NAT	mizar?ORDINAL1?V7
mmt?NAT	mizar?NAT_1?NM1
mmt?ZERO	mizar?ORDINAL1?K5
mmt?ADD	mizar?NAT_1?K1
mmt?ADD	mizar?ARYTM_0?K1
mmt?SUB	mizar?ARYTM_1?K2
mmt?SUB	mizar?REAL_1?K5
mmt?MULT	mizar?NAT_1?K2
mmt?MULT	mizar?ARYTM_0?K2
mmt?MULT	mizar?REAL_1?K4
mmt?DIV	mizar?INT_1?K4
mmt?DIV	mizar?NAT_D?K1
mmt?DIV	mizar?NAT_D?K3

Table 6: Realization pairs for arithmetics in Mizar

mmt?INT	hol?int?integer
mmt?NAT	hol?nums?num
mmt?ZERO	hol?nums?_0
mmt?ADD	hol?arith?ADD
mmt?ADD	hol?arith?+
mmt?ADD	hol?int?int_add
mmt?SUB	hol?arith?SUB
mmt?SUB	hol?arith?-
mmt?SUB	hol?int?int_sub
mmt?SUB	hol?realax?real_sub
mmt?MULT	hol?arith?MULT
mmt?MULT	hol?arith?*
mmt?MULT	hol?int?int_mul
mmt?MULT	hol?realax?real_mul
mmt?DIV	hol?int?div
mmt?DIV	hol?realax?real_div

Table 7: Realization pairs for arithmetics in HOLLight

Let us now look at addition (with the corresponding M-MMT URI mmt?ADD).

As we can see in table 8 for addition we have a different problem, we have found that there are several symbol in each of the libraries. This can be explained if we look at the definitions of each symbol and the respective file description. In HOLLight, addition is defined once as part of the natural numbers arithmetics as a recursive definition, as the symbol '+' which is an infix notation and also in the theory of integers. For Mizar, addition is defined once as an operation between a NAT and an Element of NAT returning an Element of NAT and a second time as an infix notation returning an element of real.

MMT URI and file description	**Source definition**

hol?arith?add (arith: natural number arithmetic)	`let ADD = new_recursive_definition num_RECURSION`
	`` `(!n. 0 + n = n) /\ ``
	`  (!m n. (SUC m) + n = SUC(m + n))`;;`

hol?arith?+ (arith: natural number arithmetic)	`parse_as_infix("+",(16,"right"));;`
hol?int?int_add (int: theory of integers.)	`["+",`int_add:int->int->int`;` `let int_add = new_definition` `` `x + y = int_of_real((real_of_int x) + (real_of_int y))`;; ``

mizar?NAT_1?K1 (nat_1: fundamental properties of natural numbers)	`definition`
	`  let n be Nat;`
	`  let k be Element of NAT ;`
	`  :: original: +`
	`  redefine func n + k -> Element of NAT ;`
	`  coherence by ORDINAL1:def 12;`
	`end;`

mizar?ARYTM_0?K1 (arytm_0: introduction to arithmetics)	`definition`
	`  let x, y be Element of REAL ;`
	`  func x + (x,y) -> Element of REAL means :Def1: :: ARYTM_0:def 1`
	`  ex x9, y9 being Element of REAL+ st`
	`  ( x = x9 & y = y9 & it = x9 + y9 ) if ( x in REAL+ & y in REAL+ )`
	`  ex x9, y9 being Element of REAL+ st`
	`  ( x = x9 & y = [0,y9] & it = x9 - y9 ) if ( x in REAL+ & y in [:{0},REAL+:] )`
	`  ex x9, y9 being Element of REAL+ st`
	`  ( x = [0,x9] & y = y9 & it = y9 - x9 ) if ( y in REAL+ & x in [:{0},REAL+:] )`
	`  otherwise ex x9, y9 being Element of REAL+ st`
	`  ( x = [0,x9] & y = [0,y9] & it = [0,(x9 + y9)] );`
	`  existence`
	`  proof end;`

Table 8: 'addition' in both libraries

Sets

Finally, we will see the alignments found for the empty set, inclusion, union, intersection and we will also present the detailed table with definitions for inclusion (with M-MMT URI mmt?IN).

mmt?EMPTY	hol?sets?EMPTY
mmt?IN	hol?sets?IN
mmt?UNION	hol?sets?UNION
mmt?INTER	hol?sets?INTER

Table 9: Realization pairs for sets in HOLLight

mmt?EMPTY	mizar?XBOOLE_0?V1
mmt?IN	mizar?HIDDEN?R2
mmt?IN	mizar?TARSKI?R2
mmt?UNION	mizar?TARSKI?K3
mmt?UNION	mizar?SETFAM_1?K2
mmt?INTER	mizar?SETFAM_1?K3

Table 10: Realization pairs for sets in Mizar

In table 11 we can see that inclusion has one realization in HOLLight and two realizations in Mizar. This can be explained by the fact that this concept appears once as built-in notion in hidden file which shows how the primitives of set theory are introduced in the MML. Additionally, inclusion appears in tarski which contains the first part of the axiomatics of the Mizar system(including the axioms of the Tarski Grothendieck set theory).

MMT URI and file description	**Source definition**
hol?sets?IN (sets: basic set theory)	`let IN = new_definition` `  `!P:A->bool. !x. x IN P <=> P x`;;`

mizar?HIDDEN?R2 (hidden: built-in concepts - primitives of set theory)	`definition`
	`  let x be object ;`
	`  let X be set ;`
	`  pred x in X;`
	`end;`

mizar?TARSKI?R2 (tarski: TG set theory)	`definition`
	`  let x, X be set ;`
	`  :: original: in`
	`  redefine pred x in X;`

Table 11: 'inclusion' in both libraries

With this, we conclude the data gathering stage and go further to using this data for our goal of making these concepts more discoverable in MMT.

6 Navigation via alignments

The MMT system is designed to be modular and easily extensible via plugins as we already mentioned in section 3. Figure 5 presents an overview of the entire MMT architecture and in particular, the components which we work with are the Archives for storing the information, the Frontend for handling the requests and the MMT Web as the client.

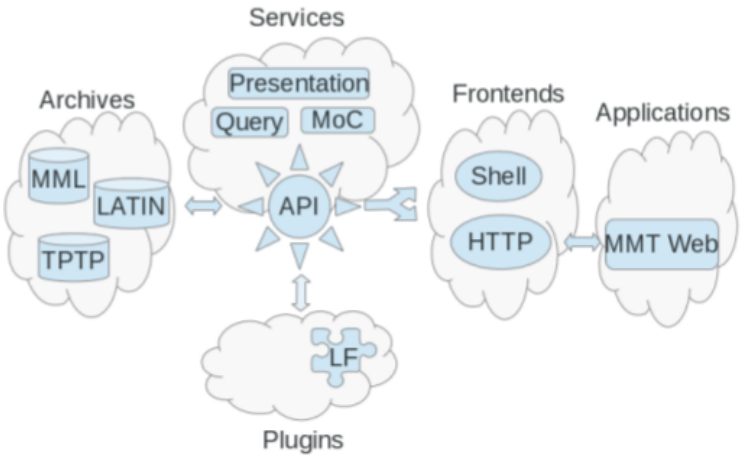

Figure 5: MMT System architecture

6.1 Integrating with MMT and Implementation

MMT Archives are folders with a special subfolder structure intended to produce a project representation. The subfolders include: content (primary content of the archive), narration (index of all documents produced by the build tool), relational (the relational index with respect to the MMT ontology) and others. We will use the relational folders from Mizar and HOLLight archives to store the alignments in text files with one realization per line preceded by the name of the relation (IsAlignedWith). We extend the current ontology infrastructure with a binary relation 'IsAlignedWith' and add our alignments to the relational index. This relation implements the equivalence relation we defined in 4.2.

The MMT API provides a number of extension interfaces which allow enhancing the functionality. We will create the navigation interface as a frontend extension, in particular the extension interface which we use is the web.ServerExtension which adds functionality to the HTTP server. The navigation interface is deployed as part of the overall MMT Web Server Interface in the form of a ServerPlugin and it belongs the info.kwarc.mmt.api.web package.

Technologies used:

- Scala (server-side)
- JavaScript, jQuery, D3 JS Library[5], CSS, Bootstrap (client-side)

We incorporate our client-side code within the current MMT Web platform which can produce XHTML and MathML content that is interactively browsable by integration with

[5]http://d3js.org/

the MMT-aware JOBAD. JOBAD is a framework for interactive mathematical documents JavaScript Library[GLR09]. When a user left-clicks an item in the web interface, a context menu entry with several options appears. We add one more option for items with non-null MMT URIs - 'show alignments'. When a user selects this option the client sends a GET request to the server via an AJAX, the server finds the alignments corresponding to the element which generated the context menu entry and sends back the response as a JSON object. The JavaScript client-side code parses the response and generates an interactive display of the data as a graph of alignments. The repository for the MMT project can be found at this link [6].

6.2 Visualization

The design of the navigation interface is meant to provide an intuitive representation of the aligned concepts and to blend in with the overall design of the MMT Web Server. That is why a graph display was the most suitable representation out of several which I tested as it stays close to the idea of theory graphs which governs MMT. Additionally, the colors (two shades of blue) are chosen such that they fit together with the current color scheme of the MMT Web Server Interface .

Interaction with the system A typical interaction with the interface will be: a user browsing one of the two libraries (HOLLight or Mizar) from the MMT WebServer as seen in Figure 6. For an item which has a context menu, the user will have among others, the option to see the concept as part of the 'bigger picture', that is, to see all the alignments for that concept (Figure 7) and to further navigate to them as seen in Figures 8 and 9.

An extension to the previous use case would be when a user is interested in comparing concepts from these libraries, he/she can do so by having the alignments for the respective concepts as presented in Figure 11.

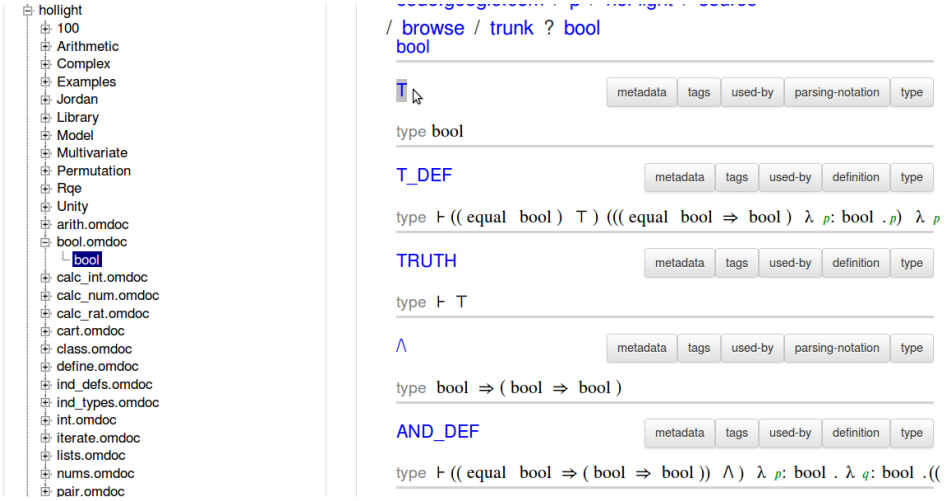

Figure 6: Browsing HOLLight booleans

[6] https://svn.kwarc.info/repos/MMT

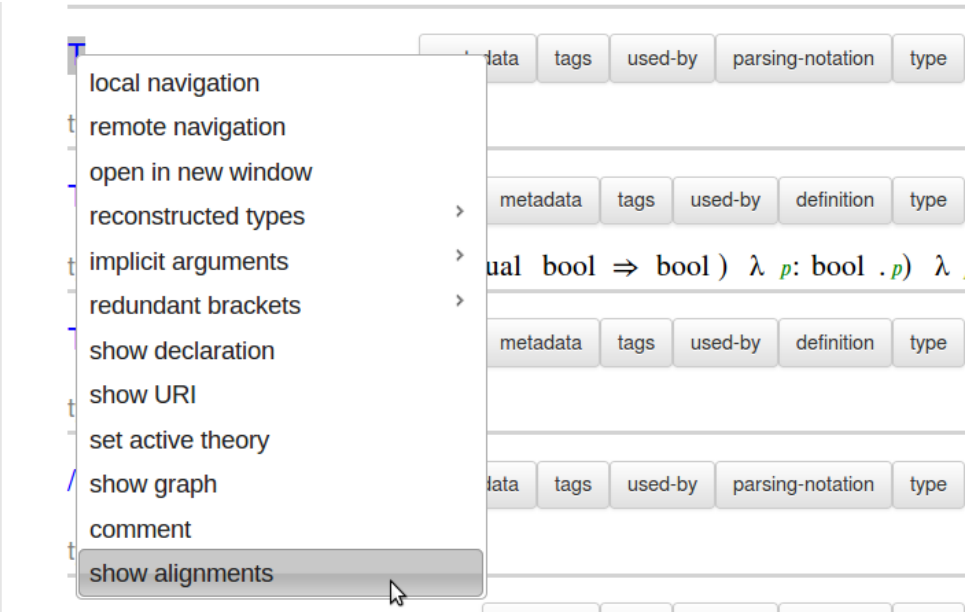

Figure 7: Selecting a concept (T) to investigate

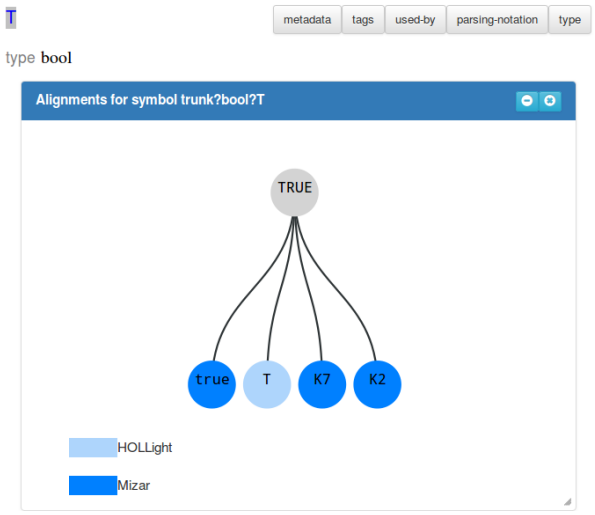

Figure 8: Alignments for that concept (T) are shown

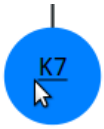

Figure 9: Nodes are themselves knowledge items

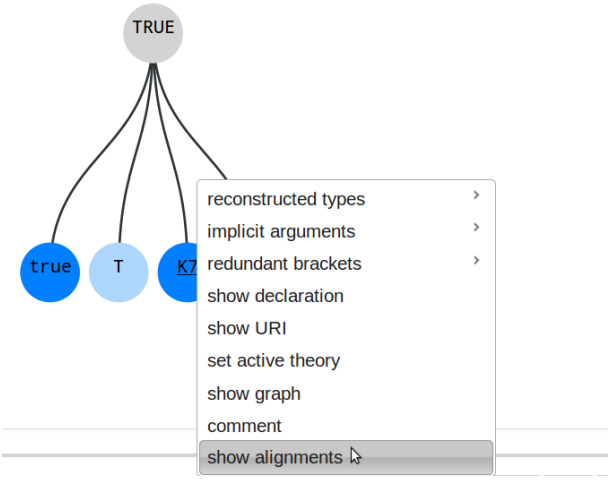

Figure 10: Investigate nodes

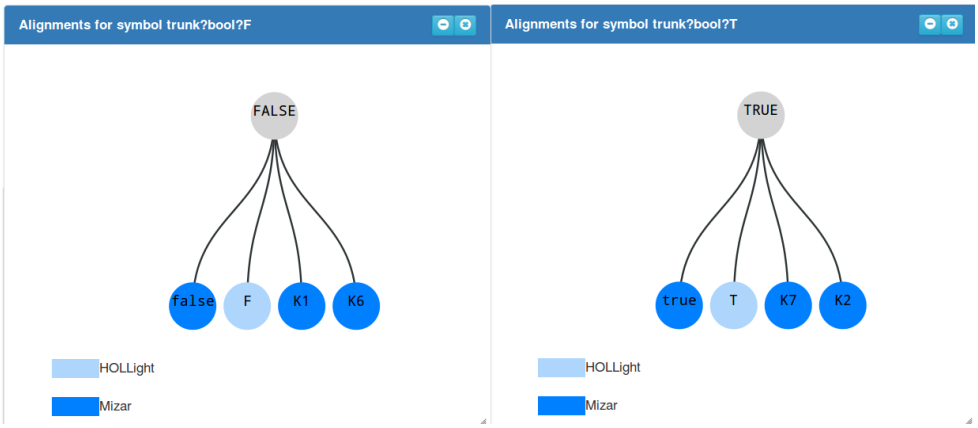

Figure 11: Compare concepts T and F by using alignments

7 Conclusion and Future Work

We introduced **alignments** as a heuristic approach for library integration. With alignments, there is no formal reasoning needed and for this reason we have an easy to generate and low-cost solution compared to a formal reasoning based solution that systematically finds matches. Our expectation is that a high percentage of problems related to library integration can be solved in this way and since some of the integration problems are inexact to begin with, alignments are the right solution in these cases.

The **results** we have obtained are summarized in the following:

- We have introduced alignments as the very first step towards proper library integration.
- We have successfully gathered the knowledge base for the chosen concepts from both libraries.
- Moreover, we have provided an intuitive graph-display of the alignments in the MMT Web Interface.

- Overall, we have reached our goal to make the libraries more discoverable by extending the knowledge base with the alignments for the selected concepts.

Future Work Alignments enable a wide range of applications such as search for theorems or development of interfaces for knowledge discovery. The results which can be obtained in this way will allow for cross-library knowledge discovery.

More specifically, alignments could be used in future projects such as:

- Efficiently searching theorems and proofs in different libraries. As these systems are large formalized libraries, by using alignments to reduce the search space one would retrieve more meaningful results in less time.

- Requesting or choosing hints from different libraries would be a great addition to the overall process of working with proof assistants. This would allow for proofs to be generated using many sources of knowledge which is a core idea of formal library integration. Additionally, an end user would not need to know that the hints presented come from different formal libraries but would have exposure to the knowledge available in all of the aligned systems.

- Documentation discovery can greatly benefit from using alignments by directly importing the documentation available in each library into the interface library resulting in a better documented library, hence, easier to use, learn and navigate.

  As we can see there are many ideas which become possible once a sufficiently generous knowledge base of alignments is available so, we highlight one more time that alignments are a powerful stepping stone towards formal library integration.

Acknowledgments I would like to thank Florian Rabe and Mihnea Iancu for the guidance and support they provided during the development of this guided research.

References

[GK14] Thibault Gauthier and Cezary Kaliszyk. Matching Concepts across HOL Libraries. In StephenM. Watt, JamesH. Davenport, AlanP. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics*, volume 8543 of *Lecture Notes in Computer Science*, pages 267–281. Springer International Publishing, 2014.

[GLR09] Jana Giceva, Christoph Lange, and Florian Rabe. Integrating Web Services into Active Mathematical Documents. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *MKM/Calculemus Proceedings*, number 5625 in LNAI, pages 279–293. Springer Verlag, July 2009.

[Hal05] Thomas C. Hales. Introduction to the Flyspeck project, 2005.

[Har96a] J. Harrison. HOL Light: A Tutorial Introduction. In *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design*, pages 265–269. Springer, 1996.

[Har96b] John Harrison. A Mizar Mode for HOL. In *Proceedings of the 9th International Conference on Theorem Proving in Higher Order Logics*, TPHOLs '96, pages 203–220, London, UK, UK, 1996. Springer-Verlag.

[Har09] John Harrison. HOL Light: An Overview. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 60–66. Springer Berlin Heidelberg, 2009.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.

[Ian13] Mihnea Iancu. Towards Flexiformal Mathematics, PhD Research Proposal, 2013.

[IKR11] M. Iancu, M. Kohlhase, and F. Rabe. Translating the Mizar Mathematical Library into OMDoc format. Technical Report KWARC Report-01/11, Jacobs University Bremen, 2011.

[KMR09] M. Kohlhase, T. Mossakowski, and F. Rabe. The LATIN Project, 2009. see `https://trac.omdoc.org/LATIN/`.

[KR14a] Cezary Kaliszyk and Florian Rabe. Towards knowledge management for HOL Light. In Stephan Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics 2014*, Lecture Notes in Computer Science, pages 357–372. Springer, 2014.

[KR14b] M. Kohlhase and F. Rabe. The OAF Project, 2014. `https://svn.kwarc.info/repos/frabe/www/OAF/index.html`.

[Kun10] Ondřej Kunčar. Reconstruction of the Mizar Type System in the HOL Light System. In Jiri Pavlu and Jana Safrankova, editors, *WDS Proceedings of Contributed Papers: Part I – Mathematics and Computer Sciences*, pages 7–12. Matfyzpress, 2010.

[KW10] Chantal Keller and Benjamin Werner. Importing HOL Light into Coq. In Matt Kaufmann and LawrenceC. Paulson, editors, *Interactive Theorem Proving*, volume 6172 of *Lecture Notes in Computer Science*, pages 307–322. Springer Berlin Heidelberg, 2010.

[McL06] Sean McLaughlin. An Interpretation of Isabelle/HOL in HOL Light. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 192–204. Springer Berlin Heidelberg, 2006.

[Ope09] OpenMath Content Dictionaries, 2009. `http://www.openmath.org/cdnames.html`.

[OS06] Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 298–302. Springer Berlin Heidelberg, 2006.

[Rab13] Florian Rabe. The MMT API: A Generic MKM System. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Intelligent Computer Mathematics*, number 7961 in Lecture Notes in Computer Science, pages 339–343. Springer, 2013.

[RK13] Florian Rabe and Michael Kohlhase. A scalable module system. *Information & Computation*, 0(230):1–54, 2013.

[TB85] A. Trybulec and H. Blair. Computer Assisted Reasoning with MIZAR. In A. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 26–28. Morgan Kaufmann, 1985.

[Wie99] Freek Wiedijk. Mizar: An Impression, 1999.

[Wie01] Freek Wiedijk. Mizar Light for HOL Light. In RichardJ. Boulton and PaulB. Jackson, editors, *Theorem Proving in Higher Order Logics*, volume 2152 of *Lecture Notes in Computer Science*, pages 378–393. Springer Berlin Heidelberg, 2001.