Searching the Space of Mathematical Knowledge

Michael Kohlhase, Mihnea Iancu

Computer Science, Jacobs University, Bremen, Germany
initial.last@jacobs-university.de

Abstract. We discuss formula search in highly modular libraries, such as the LATIN atlas. In such libraries, statements can be inherited (and thus need not be explicitly represented) via morphisms that can include translations. This is good for knowledge management as the number of induced (i.e. not explicitly represented) statements can grow exponentially in the explicitly represented ones. But this bad for accessibility, since conventional (computer supported) access methods only work on explicitly represented materials.
In this paper we note that if the representation framework of the modular library provides a systematic naming scheme for induced statements and the library system implements a flattening operation that generates all induced statements, then we can use this for access. We present the FlatSearch extension of MathWebSearch system and show this access method in action.

1 Introduction

In the last two decades the way we access information has been completely transformed by search engines. If we want to know the birthdate of Barak Obama, the first hit of a Google query for "obama birthdate" leads us to a wikipedia page that has the information in the first line of text: August 4. 1961. But not all information can be directly queried: to the best of our knowledge there is no standard search engine that can directly answer the question *who was US president when Barak Obama was born*, because the answer combines two pieces of information, Obamas birthdate and the fact that John F. Kennedy was in office in January 1961 until his assassination in November 1963. In fact, the question breaks one of the basic assumptions of search engines: i.e. that it is enough to return links to relevant web pages, which have the answer (as part of the text). Indeed, there is apparently no page on the wide web that explicitly answers our question (and if there were, we could easily find a question whose answer isn't represented). But of course, the answer is induced by knowledge explicit on the web. If we call the space of induced answers the **knowledge space**, then we would like to (but currently cannot) search the knowledge space induced by the web. Humans naturally build up a knowledge space from the facts they learn, and can answer queries based on it, but are – compared to search engines – severely limited in the breadth of information they can process.

There are search engines that attempt to search the knowledge space. For instance DBPedia extracts facts from Wikipedia, represents them as RDF triples,

and can answer queries that can be expressed by chaining relations that occur as predicates of the triples; see e.g. [AL07] for details. Even though the RDF formalization of Wikipedia facts as well as the expressivity of queries is rather limited, it can answer (a suitable version of) the Obama president question above.

But the space of mathematical knowledge cannot suitably be expressed in RDF triples, since it contains mathematical formulae (see [Lan11] for a discussion), makes non-trivial use of quantifications and introduces concepts and vocabulary dynamically. Here a question that cannot be answered by mathematical search engines is the following situation: we have identified a structure $(\mathbb{S}, \sharp)$ as an associative, unital, idempotent magma, and we are interested whether $\sharp$ is commutative. To find out, we would like to search for $\boxed{x}\sharp\boxed{y} = \boxed{y}\sharp\boxed{x}$ and find this as an instance of the fact that idempotent monoids are Abelian, which we proved in our Algebra 101 course, even though the statement of the theorem does not involve the operator $\sharp$.

In this paper we also present a method for searching the mathematical knowledge space that allows to answer such questions. In comparison with DBPedia, a) we start out from fully formal knowledge, so that the need for (automated) formalization is alleviated, and b) we only allow for a controlled regime of inducing knowledge items in the knowledge space: inheritance of statements via theory morphisms. We will see that in the context of mathematical knowledge, the second restriction is very natural (if the modularity framework is sufficiently strong), and the first restriction can be alleviated by automated semantization methods that are researched independently of the work presented in this paper.

This paper picks up ideas introduced in [Lau07] under the first author's supervision. Bastian Laubner's system was based on a partial formalization of Bourbaki's Algebra [Bou74] in the OMDoc format [Koh06], which supported modular formalization in theory graphs, but did not provide naming conventions for induced statements and did not support flattening. In the time since 2007, we have developed the Mmt format [RK13] and system [MMT], which does, and what is more, we now have a Mmt theory graph of more than 800 modules: the LATIN Logic Atlas [Cod+11]. As the atlas grows, it becomes more and more difficult to read the atlas, since most of the theories are specified by inclusion reference to other theories, and the actual statements are only reached by long and confusing walks through the graph. This problem can be solved by technology: incremental, in-place flattening in the Mmt front-end. At the same time, it becomes more and more difficult to avoid duplication and maintain a maximally shared structure, for this we need a search system that takes induced statements (these are the ones we want to share) into account.

In this paper we specify such a search system for the LATIN atlas as an extension of our MathWebSearch formula search engine [KMP12]. In section 2 we briefly introduce MathWebSearch, Mmt and the LATIN atlas. In section 3 we discuss the problem of generating and accessing induced statements in Mmt. Then, in section 4 we describe how MathWebSearch can be leveraged

to search for induced statements. Finally, in section 5 we discuss future work and conclude.

2 Preliminaries

2.1 MathWebSearch

MathWebSearch (MWS)[KP] is an open-source, open-format, content-oriented search engine for mathematical expressions.

The MWS system consists of three components:

- A *crawler subsystem* collects data from a mathematical knowledge repository and converts the mathematical expressions into MathWebSearch *harvests*. The harvests are files containing mathematical expressions encoded in a MWS-specific XML-based format (see [KP] for details).
- The *core system* builds the search index from MWS harvests and processes search queries.
- The *RESTful interface* provides a public HTTP API for interacting with the core system (i.e. submitting queries and receiving results).

Therefore, with respect to MathWebSearch, our FlatSearch approach involves providing a MWS crawler that generates MWS harvests from flattened Mmt libraries and a MWS frontent for submitting queries that will interact with the MWS RESTful interface.

2.2 MMT

Mmt [RK11] is a generic, formal module system for mathematical knowledge. The Mmt language is designed to be applicable to a large collection of declarative formal base languages and all Mmt notions are fully abstract in the choice of the base language.

We will only give a brief introduction to Mmt here and then discuss the concepts using examples in section 3. We refer to [RK11] for further details.

The central notion is that of a **theory graph** containing *theories* and *views* (morphisms between theories). Note that Mmt libraries actually contain (possibly nested) collections of documents which in turn contain modules but we omit this here for simplicity and focus on theory graphs instead.

Theories S are formed from a set of typed symbols and axioms describing their properties. Views $v : S \to T$ are morphisms between theories which map axioms of the source theory (S) to theorems of the target theory (T). This property ensures that all theorems of the source theory induce theorems of the target theory and induces a homomorphic translation from S-terms to T-terms by replacing every occurrence of an S-axiom with is corresponding T-theorem.

In addition to views, the module level structure in Mmt theory graphs is given by theory inheritance. The most general kind of inheritance in Mmt is represented by *structures* which are (possibly) partial named imports (and defined using theory morphisms). *Includes* are trivial structures which are unnamed

and total while *metas* are distinguished includes representing the meta-theory (each theory can have at most one meta-theory).

Every MMT declaration is identified by a canonical, globally unique URI. Theories and views can be referenced relative to the URI U of the theory graph that contains them by $U?\langle\!\langle$theory-name$\rangle\!\rangle$ and $U?\langle\!\langle$view-name$\rangle\!\rangle$, respectively.

Symbol declarations can be referenced relative to the URI of their containing theory T by $T?\langle\!\langle$symbol-name$\rangle\!\rangle$, where $\langle\!\langle$symbol-name$\rangle\!\rangle$ is of the form $\langle\!\langle m_1\rangle\!\rangle/\ldots\langle\!\langle m_n\rangle\!\rangle/\langle\!\langle$const$\rangle\!\rangle$, where the m_i are the names of morphisms inducing the symbol $\langle\!\langle$const$\rangle\!\rangle$ into T. Similarly, assignment declarations can be referenced relative to the URI of their containing view v by $v?\langle\!\langle$symbol-name$\rangle\!\rangle$.

The MMT system provides an API to the MMT data structures described above and the MMT implementation [Rab08; RK11] provides a Scala-based [OSV07] open source implementation of the MMT API. The MMT system can also generate MWS harvests from MMT libraries so, from the perspective of MathWebSearch, it can be seen as a MWS crawler. Therefore, with respect to MMT our FlatSearch approach involves generating the flattened version on a library and then exporting it as MWS harvests.

2.3 LATIN

The LATIN project (**L**ogic **At**las and **In**tegrator) [KMR09; LATIN] aims at developing tools for interfacing logics and proof systems. It focuses on developing a knowledge representation framework that is foundationally unconstrained and thus allows for the representation of most meta-theoretic foundations of mathematical knowledge in the same format.

The LATIN logic graph is built as an MMT library and already contains work related to first-order logic (TPTP [SS98], CASL [Ast+02; Mos04] and Mizar [TB85; UB06]), higher-order logic (PVS [ORS92], Isabelle/HOL [NPW02] and HasCASL [SM02]), logical frameworks (LF [Pfe91] and Isabelle [Wen+13]) and mathematics (set theory, natural and rational numbers, algebraic structures and others). Furthermore, whenever possible, these representations are structured and related using logic morphisms.

The LATIN library currently contains 449 theories and 382 views between them. The theories themselves contain a total of 2310 symbol declarations and 1072 direct imports (including metas) resulting in an average of 5.14 declarations and 2.39 direct imports per theory. This amounts to a size of 123.9 MB in its native OMDoc format and 25.2 MB when converted to MWS harvests.

3 Induced Statements in MMT Libraries

To understand the situation in modular libraries, consider the theory graph in Figure 1. The right side of the graph introduces the elementary algebraic hierarchy building up algebraic structures step by step up to rings; the left side

contains a construction of the integers. In this graph, the nodes are theories [1], the solid edges are imports and the dashed edges are views.

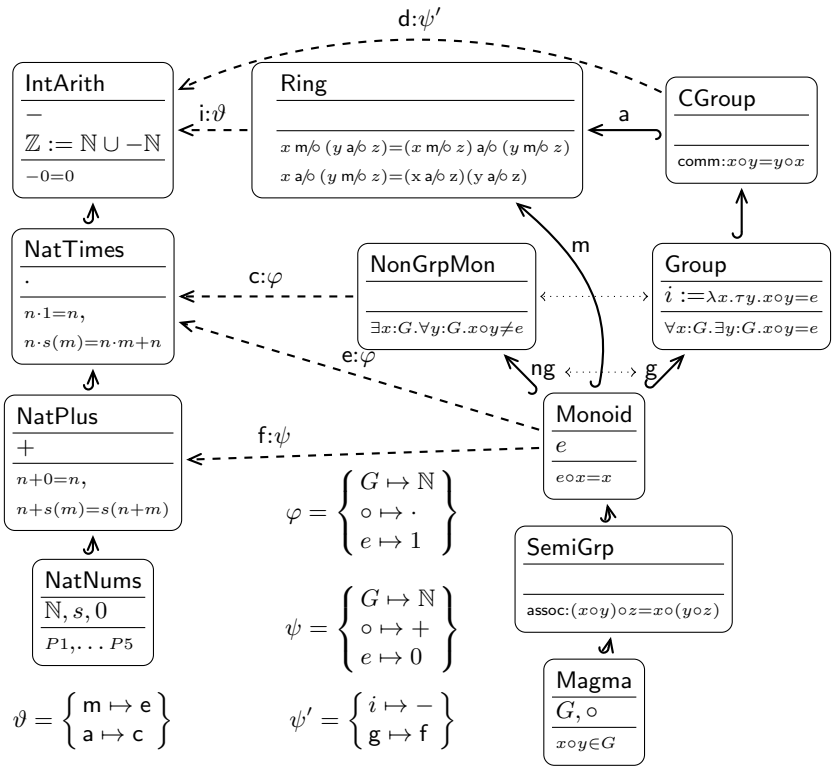

Fig. 1. A MMT Graph for Elementary Algebra

In MMT theory graphs, theory morphisms can carry a name, and inherited constants can be disambiguated via the inclusion that induced them. An application of this is in the definition of the ring theory, which inherits all of its operators (and their axioms) via the two inclusions m (for the multiplicative operations) and a (for the additive operations). To complete the ring we only need to add the two distributivity axioms in the inherited operators m$\diamond$ and a$\diamond$.

Furthermore, morphisms can carry an assignment which maps symbols and axioms from the source theory to terms in the target theory. We see this in the view e from Monoid to NatArith, which assigns $\mathbf{N}$ to the base set G, multiplication $(\cdot)$ to $\circ$ and the number 1 to the unit e. The document theory morphism property, e also contains proofs for all Monoid axioms in NatArith.

[1] We have left out the quantifiers for the variables x, y, and z from the axioms to reduce visual complexity. The always range over the respective base set. Furthermore, all axioms are named; but we only state the names we actually use in the examples.

It is a special feature of MMT that assignments can also map morphisms into the source theory to morphisms into the target theory. We use this to specify the morphism c modularly (in particular, this allows to re-use the proofs from e and c).

Note that already in this small graph, there are a lot of induced statements. For instance, the associativity axiom is inherited in seven times (via inclusions; twice into Ring) and induced four times (via views; twice each into NatArith and IntArith). All in all, we have more than an hundred induced statements from the axioms alone. If we assume just 5 theorems proven per theory (a rather conservative estimation), then we obtain a number of induced statements that is an order of magnitude higher.

Another crucial ingredient of MMT for the endeavor of searching for induced statements is the fact that, as discussed in section 2.2, MMT supplies names (called MMT URIs) for all induced statements. For example, if we take u to be the URI of the theory graph in Figure 1, the statement $\forall x, y, z : \mathbb{Z}.(x+y)+z = x+(y+z)$ induced by the view c in IntArith has the MMT URI u?IntArith?c/g/assoc. Note that given a theory graph, the MMT system can dereference valid MMT URIs and return the induced statements, even though they are not explicitly represented in the graph.

Generating Induced Statements Following the example above we can formally define the process of flattening a theory (or a theory graph).

Definition 1. *Given a theory graph γ the flattening of a theory T in γ is a theory $\overline{T}$ with the same URI as T containing:*

- *all symbol declarations that are in T.*
- *all symbol declarations that are imported into T.*
- *for every view $v : S \to T$ the projection of every S-based declaration over view v. Here, by S-based declaration we refer to the declarations in S and in theories that import S.*

The URIs of the induced declarations are based on the definition of MMT URIs from section 2.2 (see also the example above) and permit recovering the origin of induced declarations. Specifically, symbol declarations from T or imported in T by the meta or an include preserve their name while symbol declarations imported in T by a structure or induced by a view are additionally qualified with the name of that structure or view.

Definition 2. *The flattening of theory graph γ is a theory graph $\overline{\gamma}$ with the same URI as γ containing the following module declarations :*

- *for every theory T in γ the theory $\overline{T}$*
- *every view $v : S \to T$ in γ*

Note that, since theory flattening preserves theory URIs and doesn't add new axioms (only new theorems) any view from S to T is also a view from $\overline{S}$ to $\overline{T}$ so the views in $\overline{\gamma}$ are still valid.

An important observation is that the flattening operation for theory graphs is not idempotent and flattening can be applied repeatedly to a theory graph to generate new induced symbol declarations. Effectively this generates symbol declarations induced not only by one view but by a chain (composition) of views. More precisely, a theory graph flattened n times will all contain declarations induced by view chains of length at most n. Clearly, if there is a view cycle (e.g. $v_1 : S \to T$ and $v_2 : T \to S$) this process can continue indefinitely yielding ever bigger content libraries.

Induced statements in the LATIN *library* We implemented library flattening as described above in MMT and tested it on the LATIN library. The flattening (once) of the LATIN library increases the number of declarations from 2310 to 58847 (a factor of 25.4) and the total size of the library from 123.9 MB to 1.8 GB (a factor of 14.8). As expected, the multiplication factor depends on the level of modularity of the library. For instance, the highly modular math sub-library containing mainly algebraic structures increases from 2.3 MB to 79 MB thus having a multiplication factor of 34.3, more than double the library average. The size of the MWS harvests also increases considerably, from 25.2 MB to 539.0 MB.

4 Searching for Induced Statements

To search for induced statements, we use our MATHWEBSEARCH system [KMP12], which indexes formula-URL pairs and provides a web interface querying the formula index via unification. This can be used for

Instance Search e.g. to find all instance of associativity we can issue the query $\forall x, y, z : \boxed{\text{S}}.(x \boxed{\text{op}} y) \boxed{\text{op}} z = x \boxed{\text{op}} (y \boxed{\text{op}} z)$, where the $\boxed{\text{-}}$ are query variables that can be instantiated in the query. In the library from Figure 1 we would find the commutativity axiom SemiGrp/assoc, its directly inherited versions in Monoid, to Ring and in particular the version u?IntArith?c/g/assoc.

Applicable Theorem Search where universal variables in the index can be instantiated as well; this was introduced for a non-modular formal libraries in [Ian+11]. Here we could search for $3 + 4 = \boxed{\text{R}}$ and find the induced statement u?IntArith?c/comm with the substitution $R \mapsto 4+3$, which allows the user to instantiate the query and obtain the equation $3 + 4 = 4 + 3$ together with the justification u?IntArith?c/comm that can directly be used in a proof.

The implementation of a web service that conducts such searches is very simple: instead of harvesting formulae directly from a formal digital library directly as in [Ian+11], we flatten the library first, and then harvest formulae. Conveniently,MMT flattening gives the included constants and axioms local names that are syntactically identical to the respective symbol paths in MMT URIs, so that the generation of MMT URIs for the formula harvests is trivial.

The only non-trivial part of the implementation is the search human-oriented front-end, i.e. the input of search queries and the presentation of search results.

The LATIN atlas is written in an extension of the TWELF encoding [RS09] of LF [HHP93], so it is natural to use an extension of LF notation with query variables for input. Therefore, we use the MMT notation language and interpretation service described in [IR12] to transform LF-style input into MMT objects and subsequently to MWS queries.

The presentation of the MMT URIs requires some work as well: while the MMT system can directly dereference the MMT URI and thus be used to present the induced statement, humans want a justification that is more understandable than a MMT URI. Fortunately, this can be generated from the MMT URI by a simple template-based algorithm.

Let us consider the search result $u?\mathsf{IntArith}?\mathsf{c}/\mathsf{g}/\mathsf{assoc}$ from the instantiation search above, where we take u to be `http://cds.omdoc.org/cds/elal`. The first step is to localize the result in the theory $u?\mathsf{IntArith}$ with the sentence

$$\begin{array}{l}\mathsf{Induced\ statement}\ \forall x,y,z\ :\ \mathbb{Z}.(x+y)+z\ =\ x+(y+z)\\ \mathsf{found\ in}\ \underline{\texttt{http://cds.omdoc.org/cds/elal?IntArith}}\ (\underline{\mathsf{subst}},\\ \underline{\mathsf{justification}}).\end{array} \tag{1}$$

Here the underlined fragments carries hyperlinks, the second pointing to the justification:

$$\underline{\mathsf{IntArith}}\ \mathsf{is\ a}\ \underline{\mathsf{CGroup}}\ \mathsf{if\ we\ interpret}\ \circ\ \mathsf{as}\ +\ \mathsf{and}\ G\ \mathsf{as}\ \mathbb{Z}. \tag{2}$$

which can be directly from the information associated to the morphism c in the MMT URI. Then we skip over g, since its assignment is trivial and generate the sentence.

$$\underline{\mathsf{CGroups}}\ \mathsf{are}\ \underline{\mathsf{SemiGrps}}\ \underline{\mathsf{by\ construction}} \tag{3}$$

and finally we ground the explanation by the sentence

$$\begin{array}{l}\mathsf{In}\ \underline{\mathsf{SemiGrps}}\ \mathsf{we\ have\ the\ axiom}\\ \underline{\mathsf{assoc}:\forall x,y,z:G.(x\circ y)\circ z=x\circ(y\circ z)}\end{array} \tag{4}$$

The sentences (1) to (4) can be generated from templates, since the MMT system gives access to the necessary information: source and target theory as well as the assignment ψ' for (2), the fact that the path from $\mathsf{SemiGrp}$ to CGroup only consists of inclusion that triggers the template for (3) and the original formulation of the axiom assoc.

Note that we make use of another peculiarity of the MMT system in this explanation: all constants in the theory graph carry notation declarations [KMR08], which can be used to generate human-readable presentations of arbitrary formal objects in the graph.

5 Conclusion and Future Work

We have presented an extension of a formula search search engine to modular formal libraries. In our application case, most of the services that are needed for this extension: *i)* compositional naming of induced statements, *ii)* flattening, *iii)* transformation of formulae to content MathML, and *iv)* presentation of arbitrary objects already came with the MMT system. Any other system (e.g. the

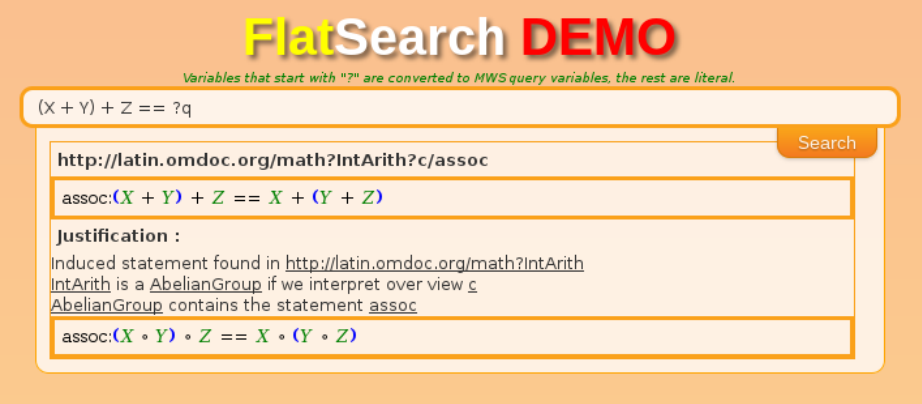

Fig. 2. The FLATSEARCH Web Interface for LATIN

Isabelle system [NPW02] for the Isabelle library [Isa]) would have to implement similar functionality. Our experience with querying shows that the statements induced by views are by far more useful than those induced by inclusions, because (as in our graph in Figure 1) the interest is in inducing statements from the abstract part to concrete structures. As a consequence, FLATSEARCH becomes more and more useful as the graph acquires more and more views.

Currently, our FLATSEARCH is restricted to fully formal modular libraries, since flattening is only understood for those. With a notion of flattening of flexiformal representations (representations of mathematical knowledge at flexible levels of formality; see [KK11]) we could extend FLATSEARCH to modular flexiformalizations of traditional mathematical documents. Indeed Laubner's study of the first 35 pages of Bourbaki's Algebra mentioned in the introduction revealed an underlying theory graph with 51 theory nodes and 107 theory morphisms of which 12 were views, but 63 had non-trivial assignments. Especially the last number shows the value of the FLATSEARCH method. Indeed one of the problems readers face with the Bourbaki books (which are otherwise well-liked for their structured approach) is that particular mathematical structures and objects can only be understood, if one already knows all the material they depend on. One author even said that

> *Bourbaki was a dinosaur, the head too far away from the tail. Explaining: [...] You could say "Dieudonné what is the result about so and so?" and he would go to the shelf and take down the book and open it to the right page. After Dieudonné retired no one was able to do this. So Bourbaki lost awareness of his own body* [Ric]

A flexiformalization of the Bourbaki books together with an extension of MMT that can deal with flattening of informal texts would go a long way to alleviate these problems.

FlatSearch queries are currently restricted to pure unification queries, for a more expressive user experience, we plan to embed it into a more comprehensive mathematical query language, most probably starting with [Rab12].

Finally note that our query method is similar in spirit to "Semantic Web Queries", where queries for RDF triples that are induced by a background ontology from explicitly represented RDF facts are possible. The inferencing involved in this corresponds to the flattening step in FlatSearch. However, DL-reasoners like Racer [HM03] answer queries by performing inferences on the fly, and do not have the efficiency of a search index at their disposal, and can only deal with relatively small A-boxes as a result. Current high-efficiency triple stores that do employ (database) indexing techniques only support a very restricted subset of ontologies (as a typical example, Virtuoso [Olv] supports transitive closures of selected relations, but not a lot more).

References

[AL07] Sören Auer and Jens Lehmann. "What have Innsbruck and Leipzig in common? Extracting Semantics from Wiki Content". In: *The Semantic Web: Research and Applications*. 4th European Semantic Web Conference (ESWC). (Innsbruck, Austria, June 3–7, 2007). Ed. by Enrico Franconi, Michael Kifer, and Wolfgang May. Lecture Notes in Computer Science 4519. Springer Verlag, 2007, pp. 503–517.

[Ast+02] E. Astesiano et al. "CASL – the Common Algebraic Specification Language". In: *Theoretical Computer Science* 286 (2002), pp. 153–196. URL: `http://www.cofi.info`.

[Bou74] Nicolas Bourbaki. *Algebra I*. Elements of Mathematics. Springer Verlag, 1974.

[Cod+11] Mihai Codescu et al. "Project Abstract: Logic Atlas and Integrator (LATIN)". In: *Intelligent Computer Mathematics*. Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 289–291.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. "A framework for defining logics". In: *Journal of the Association for Computing Machinery* 40.1 (1993), pp. 143–184.

[HM03] Volker Haarslev and Ralf Möller. *Racer: A core inference engine for the semantic web*. `http://www.franz.com/products/racer/Racer_whitepaper.pdf`. 2003. URL: `http://www.franz.com/products/racer/Racer_whitepaper.pdf`.

[Ian+11] M. Iancu et al. "The Mizar Mathematical Library in OMDoc: Translation and Applications". 2011.

[IR12] M. Iancu and F. Rabe. "An MMT-Based User-Interface". In: *Workshop On User Interfaces for Theorem Provers*. see `http://kwarc.info/frabe/Research/IR_ui_12.pdf`. 2012.

[Isa] *The Isabelle 2012library*. URL: `http://isabelle.in.tum.de/library/` (visited on 06/20/2012).

[Jeu+12] Johan Jeuring et al., eds. *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM). (Bremen, Germany, July 9–14, 2012). LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012.

[KK11] Andrea Kohlhase and Michael Kohlhase. "Towards a Flexible Notion of Document Context". In: *Proceedings of the 29th annual ACM international conference on Design of communication (SIGDOC)*. (Pisa, Italy). ACM Special Interest Group for Design of Communication. New York, NY, USA: ACM Press, 2011, pp. 181–188. URL: `http://kwarc.info/kohlhase/papers/sigdoc2011-flexiforms.pdf`.

[KMP12] Michael Kohlhase, Bogdan A. Matican, and Corneliu C. Prodescu. "MathWebSearch 0.5 – Scaling an Open Formula Search Engine". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM). (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 342–357. URL: `http://kwarc.info/kohlhase/papers/aisc12-mws.pdf`.

[KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. "Notations for Living Mathematical Documents". In: *Intelligent Computer Mathematics*. 9$^{\text{th}}$ International Conference, AISC, 15$^{\text{th}}$ Symposium, Calculemus, 7$^{\text{th}}$ International Conference MKM. (Birmingham, UK, July 28–Aug. 1, 2008). Ed. by Serge Autexier et al. LNAI 5144. Springer Verlag, 2008, pp. 504–519. URL: `http://omdoc.org/pubs/mkm08-notations.pdf`.

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

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: `http://omdoc.org/pubs/omdoc1.2.pdf`.

[KP] Michael Kohlhase and Corneliu Prodescu. *MathWebSearch Manual*. Web Manual. Jacobs University. URL: `https://svn.mathweb.org/repos/mws/doc/manual/manual.pdf` (visited on 04/07/2012).

[Lan11] Christoph Lange. "Enabling Collaboration on Semiformal Mathematical Knowledge by Semantic Web Integration". Also available as a book [**Lange:PhD:book**]. PhD thesis. Jacobs University Bremen, 2011. URL: `https://svn.kwarc.info/repos/swim/doc/phd/phd.pdf`.

[LATIN] Michael Kohlhase, Till Mossakowski, and Florian Rabe. *LATIN: Logic Atlas and Integrator*. `http://latin.omdoc.org`. Project Homepage. URL: `http://latin.omdoc.org`.

[Lau07] Bastian Laubner. "Using Theory Graphs to Map Mathematics: A Case Study and a Prototype." MA thesis. Bremen: Jacobs University, Aug. 2007. URL: `https://svn.eecs.jacobs-university.de/svn/eecs/archive/msc-2007/blaubner.pdf`.

[MMT] Florian Rabe. *MMT – A Module system for Mathematical Theories*. URL: http://trac.kwarc.info/MMT/ (visited on 08/18/2010).

[Mos04] Peter D. Mosses, ed. *CASL Reference Manual*. LNCS 2960. Springer Verlag, 2004.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. LNCS 2283. Springer, 2002.

[Olv] OpenLink Software. *OpenLink Universal Integration Middleware – Virtuoso Product Family*. URL: http://virtuoso.openlinksw.com (visited on 10/22/2009).

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. "PVS: A Prototype Verification System". In: *Proceedings of the 11th Conference on Automated Deduction*. Ed. by D. Kapur. LNCS 607. Saratoga Springs, NY, USA: Springer Verlag, 1992, pp. 748–752.

[OSV07] M. Odersky, L. Spoon, and B. Venners. *Programming in Scala*. artima, 2007.

[Pfe91] Frank Pfenning. "Logic Programming in the LF Logical Framework". In: *Logical Frameworks*. Ed. by Gérard P. Huet and Gordon D. Plotkin. Cambridge University Press, 1991.

[Rab08] F. Rabe. *The MMT System*. see https://trac.kwarc.info/MMT/. 2008.

[Rab12] Florian Rabe. "A Query Language for Formal Mathematical Libraries". In: *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics (CICM). (Bremen, Germany, July 9–14, 2012). Ed. by Johan Jeuring et al. LNAI 7362. Berlin and Heidelberg: Springer Verlag, 2012, pp. 142–157. arXiv: 1204.4685 [cs.LO].

[Ric] Émilie Richter. *Nicolas Bourbaki*. URL: http://planetmath.org/NicolasBourbaki.html.

[RK11] F. Rabe and M. Kohlhase. "A Scalable Module System". see http://arxiv.org/abs/1105.0548. 2011.

[RK13] Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: http://kwarc.info/frabe/Research/mmt.pdf.

[RS09] F. Rabe and C. Schürmann. "A Practical Module System for LF". In: *Proceedings of the Workshop on Logical Frameworks: Meta-Theory and Practice (LFMTP)*. Ed. by J. Cheney and A. Felty. Vol. LFMTP'09. ACM International Conference Proceeding Series. ACM Press, 2009, pp. 40–48.

[SM02] Lutz Schröder and Till Mossakowski. "HasCASL: towards integrated specification and development of functional programs". In: *Algebraic Methodology and Software Technology — 9th International Conference AMAST 2002*. Ed. by Hélène Kirchner and Christophe Ringeissen. LNCS 2422. Springer Verlag, 2002, pp. 99–116.

[SS98] G. Sutcliffe and C. Suttner. "The TPTP Problem Library: CNF Release v1.2.1". In: *Journal of Automated Reasoning* 21.2 (1998), pp. 177–203.

[TB85] A. Trybulec and H. Blair. "Computer Assisted Reasoning with MIZAR". In: *Proceedings of the 9th International Joint Conference on Artificial Intelligence*. Ed. by A. Joshi. 1985, pp. 26–28.

[UB06] Josef Urban and Grzegorz Bancerek. "Presenting and Explaining Mizar". In: *Proceedings of the International Workshop "User Interfaces for Theorem Provers" 2006 (UITP'06)*. Ed. by Serge Autexier and Christoph Benzmüller. Seattle, USA, 2006, pp. 97–108.

[Wen+13] Makarius Wenzel et al. *The Isabelle/Isar Reference Manual*. Ed. by Makarius Wenzel. Nov. 7, 2013. URL: `http://isabelle.in.tum.de/dist/Isabelle2013-1/doc/isar-ref.pdf`.