Wikis and Collaborative Systems for Large Formal Mathematics

Cezary Kaliszyk[1]* and Josef Urban[2]**

[1] University of Innsbruck, Austria
[2] Czech Technical University in Prague

Abstract In the recent years, there have been significant advances in formalization of mathematics, involving a number of large-scale formalization projects. This naturally poses a number of interesting problems concerning how should humans and machines collaborate on such deeply semantic and computer-assisted projects. In this paper we provide an overview of the wikis and web-based systems for such collaboration involving humans and also AI systems over the large corpora of fully formal mathematical knowledge.

1 Introduction: Formal Mathematics and its Collaborative Aspects

In the last two decades, large corpora of complex mathematical knowledge have been encoded in a form that is fully understandable to computers [7, 13, 14, 17, 30, 38]. This means that the mathematical definitions, theorems, proofs and theories are explained and formally encoded in complete detail, allowing the computers to *fully understand the semantics* of such complicated objects. While in domains that deal with the real world rather than with the abstract world researchers might discuss what *fully formal encoding* exactly means, in computer mathematics there is one undisputed definition. A fully formal encoding is an encoding that ultimately allows computers to verify to the smallest detail the correctness of each step of the proofs written in the given logical formalism. The process of writing such computer-understandable and verifiable theorems, definitions, proofs and theories is called *Formalization of Mathematics* and more generally *Interactive Theorem Proving* (ITP).

The ITP field has a long history dating back to 1960s [19], being officially founded in 1967 by the mathematician N.G. de Bruijn, with his work on the Automath system [11]. The development of a number of approaches (LCF, NQTHM, Mizar) followed in the 1970-s, resulting in a number of today's established ITP systems such as HOL (Light) [18], Isabelle [56], Mizar [15], Coq [9], ACL2 [29] and PVS [42]. This development was often intertwined with the development of its cousin field of *Automated Theorem Proving* [47] (ATP), where

* Supported by the Austrian Science Fund (FWF): P26201.
** Supported by NWO grant nr. 612.001.208 and ERC Consolidator grant nr. 649043 *AI4REASON*.

proofs of conjectures are attempted fully automatically, without any human assistance. But unlike ATP systems, the ITP systems allow human-assisted formal theory encoding and proving of theorems that are often beyond the capabilities of the fully automated systems.

Twenty years ago, the anonymous QED Manifesto [1] proposed to start a unified encyclopedic effort formalizing all of today's mathematics. This led to a lot of discussion about suitable logical foundations for such a unified effort, and to two QED workshops in 1995 and 1996 [39]. While no such unified foundations and a top-down managed effort emerged, formal libraries of impressive size have been built since then with ITPs: the Mizar Mathematical Library (MML) contains today over 50000 theorems and the Isabelle/HOL and the Flyspeck project (with the large HOL Light libraries) have some 20000 theorems each. ITP is also becoming an indispensable technology for verifying complex software-assisted proofs, and other complicated (e.g., hardware and software) designs. Other exact sciences, such as economics [36] and physics [2] have started to be fully formally encoded recently.

Recent examples of the large projects in formal mathematics include the completed formal proofs of the Kepler conjecture (Flyspeck) [17], the Odd Order theorem [14], the Four Color theorem [13], and verification of more than a half of the Compendium of Continuous Lattices textbook [7]. Verification of the seL4 kernel [30] and the CompCert compiler [38] show comparable progress in full-scale verification of complicated software. Such projects are often linked to various advances in verification technology, such as strong automation methods [24, 26, 32] that allow less and less verbose formal proofs, however, ITP still remains very labor-intensive. For example, the Flyspeck project is estimated to take 20 person-years, and the Feit-Thompson project took about twice as much.

This means that such large projects in formal mathematics have also very important collaboration aspects. Sometimes they may be managed in a very tight top-down manner, with the formal terminology, naming conventions and majority of stated lemmas being designed by the project leaders, and only the proofs are written by other team members. This has many advantages in consistency, similar for example to the one-man (or one-committee) design of the "upper ontologies" in common-sense reasoning projects such as SUMO [41] and CYC [45]. Sometimes such total top-down control is however missing, and the projects or libraries grow much more chaotically, similar to the way how Wikipedia grows, i.e., only with a basic set of rules for contributing and a lot of later (possibly collaborative) refactoring of the formal terminology, statements, proofs, and theories.

The combination of all these aspects makes formal mathematics a very interesting domain for several reasons:

1. It allows collaboration of experts from very different mathematical areas, both professionals and students and amateurs. As long as the formal proofs are correct, nobody needs to be interested about who wrote them and how, and if they understood the defined concepts correctly. All this consistency and proof checking is ensured by the underlying ITP systems. This *power of*

objectivity has been noted by John McCarthy who proposed to transfer such mechanisms to other sciences and venues of lives.[3]

2. In a similar way, it allows collaboration with machines, particularly AI and ATP systems. Such systems can use very strong semantic search methods such as automated theorem proving, to assist the humans. Again, the final correctness and consistency of such strong machine assistance is checked by the ITP systems.
3. It has all kinds of relations to less formal domains. A very interesting topic is how to present the formal proofs to humans, and also how to assist humans as much as possible with converting informal mathematics into formal. The latter includes all kinds of specialized editors and authoring environments. Such editors and environments may support various semantic features, can be web-based, allow hyperlinking and inter-linking via Semantic Web techniques to more fuzzy concepts defined in Wikipedia and DBpedia, etc. A related interesting topic is how to connect and re-use the formalizations done in various formal systems, or using various libraries.

The rest of this paper is organized as follows. Section 2 summarizes the main differences between encyclopedic efforts like Wikipedia and the reality of formal mathematics. In Section 3 we introduce formal mathematical wikis and their main components, such as formal verification, suitable rendering, versioning, editing, and semantic assistance. Section 4 then discusses the examples of today's formal wikis and their various features and subsystems.

2 Some Obstacles to Wikis for Formal Mathematics

In the informal world, Wikipedia can today be largely considered as *The World Encyclopedia* written in a collaborative wiki style, covering vast number of topics and serving more semantics projects such as DBpedia and Wikidata, which can in turn be considered to be the hub of the Semantic Web. In formal mathematics, there is unfortunately no such unique resource, despite the early QED proposals and efforts. Whether such fragmentation is necessary is not clear, however, the following factors largely contribute to this state.

Disagreement about the logical and mathematical foundations. Mathematicians and logicians have different opinions about what should be the most convenient and the most correct foundations for mathematics, ranging from philosophical topics such as constructive versus classical logic, to more practical topics such as using higher-order logic (HOL) versus set theory versus type theory. There are logical frameworks (such as Isabelle [56] or Twelf [48]) that try to cater for everybody, however those may be viewed as an additional burden by some, and their universality, practicality and foundations may be further questioned. The most widely used systems today commit to just one foundational framework.

[3] http://www-formal.stanford.edu/jmc/future/objectivity.html

Disagreement about the formal language and its mechanisms. This is very similar to the disagreement about the appropriateness of various programming languages for various projects. Analogously to that, the chance of all formalizers ultimately agreeing on some of the current formal languages is quite small.

Disagreement about how formal concepts should be defined and mathematics should be built. Even when within common foundations and when using a common formal language and an ITP system, there are sometimes different approaches to formalizing mathematics. In the Coq system, there are several definitions of real numbers, algebraic hierarchies (monoids, groups, rings, fields, modules, etc.) have been built in several competing ways, etc. In the Mizar Mathematical Library, there are several definitions of graphs, categories, and other structures, and parallel theorems about these parallel concepts. Again, this is quite analogous to the current state in programming, where several programming libraries/projects may address the same task differently, according to the tastes and opinions of its implementors. Unlike in Wikipedia, one usually does not prefer to present the different implementations side-by-side within one library, because that makes the library less focused and compact, possibly leading the library's users to confusion and multiple efforts in the same direction.

All these issues have considerable impact on the kind of collaborative tools that can be useful for formalization of mathematics. Around 2004, inspired by the success of Wikipedia, various proposals started to appear suggesting to speed up the relatively slow formalization efforts by wiki-like methods.[4] Ten years later, it is however still not completely clear what is the right combination of features suitable for fast collaborative formalization. We discuss the main components in the next section. While a number of interesting wiki systems have been produced for less formal mathematics, knowledge engineering, research support, and related domains [8,37,40,57], so far there has been no successful attempt to port such systems to the fully formal setting.

3 Formal Wikis and Their Main Components

Traditionally, formal mathematics used to be constructed locally by a single author, either in an editor or in a toplevel[5] of an interactive theorem prover. The prover immediately checks the steps, using its proof rules and the previous knowledge available in the formal libraries, which are locally installed. After the formal definitions, theorems and proofs are written in a file, the file is typically checked again in batch mode by the prover and included in the locally installed library. Various methods have evolved for making one's results available to others. For a long time this would be just an e-mail to library managers, who would

[4] http://wiki.mizar.org/twiki/bin/view/Mizar/MizarWishlist

[5] Many proof assistants have been implemented inside programming language interpreters and inherit their toplevels.

then distribute the new library version using FTP or WWW. In the last decade, a major step towards fast collaborative development was done by switching to version control systems (VCS), which have become also one of the building blocks of today's formal wikis.

In general, a formal wiki would typically address the following six components to various extent:

Formal verification on the server. This is the defining element of the formalization process. Each text that is submitted to a formal wiki will first be evaluated with respect to a particular *formal verifier* (i.e., a proof assistant, ITP) implementing the logic rules and checking the statements and proofs with respect to the logic and the available formal library. Various result statuses of the formal verification are possible. For example, the formal text might be correctly parsed (all concepts are known and used in a correct way), but not completely proof-checked (some proofs make logical inferences that the proof checker cannot understand, perhaps requiring to refine some steps). Or changes in one part of the library might invalidate another part of the library, putting the library as a whole into an inconsistent state. Depending on the results of the verification, the wiki might either completely reject the text, or accept it with some status, usually also updating its existing library with the text, and/or possibly performing some other actions such as branching the development automatically.

Formal library or libraries available on the server, used to supply mathematical terminology and theorems needed for verification of more advanced texts. This is one of the main ways in which interactive theorem proving differs from completely automated theorem proving, where all the information needed for proving a particular statement is typically included in one file. Such large formal libraries are typically stored efficiently in some pre-compiled fast-loadable format, and their parts are loaded into the interactive provers using various inclusion mechanisms. One of the major issues that need to be addressed in formal wikis is the updating, refactoring and maintenance of such libraries of formal concepts, theorems and proofs, their versioning and suitable rendering.

Versioning. Often one wants to see older versions of various files in the library, track the changes done by various users, or even try to work with a non-default version of the library. For example MediaWiki, Wikipedia's internal wiki engine, uses sequential numbering of files (independent of other files), and for every change, the id of the user that made the change is stored with the file. Viewing of differences and histories is possible, although more limited than in more advanced versioning systems. This simplicity can also be an advantage: Wikipedia does not include branches, tags, etc., and the casual user does not need to understand such features. On the other hand, more advanced version control systems operate with the concept of change sets (or commits), which seem more appropriate when several files need to be simultaneously modified (for example when renaming some function) in order for the library to be in a consistent state. If such library consistency is the main concern which is strictly enforced,

the Wikipedia-style file-based version control is insufficient, and such advanced version control systems with simultaneous multi-file commits are a necessity.

(Semantic) Rendering. Many interactive provers include documentation generators that process raw prover input files and generate rendered output. The output of a documentation generator is usually HTML or PDF format, with HTML being of particular interest for wikis. Links between files and items are created, different conceptual elements of the prover input are colored in different color, and sometimes mathematical formulas are rendered in a graphical way. Since we are dealing with a very semantic domain, various sorts of information can be added to make the proofs more understandable by the readers, various dependencies can be explicitly shown, etc. While the authors of formal articles typically know how to read the raw formal texts, the HTML presentation (often including MathML) is one of the main aspects of formal wikis that make formal mathematics accessible to newcomers. One of the main reasons is the typical brevity of the formal language which is often optimized for writing rather than reading, and the ubiquitous overloading of mathematical symbols like +, whose particular meaning in a particular context might be very hard to decipher for a casual reader. For example, the large Mizar Mathematical library defines about 200 different uses of the symbol + (for example addition introduced for natural numbers and addition in a group with addition use this symbol). Suitable HTML rendering can link such symbols to their definitions, or even allow their immediate preview by mechanisms such as mouse-over tooltips. In a similar way, aligning with various less formal resources such as textbooks is possible, and also aligning with Semantic Web resources such as Wikipedia or DBpedia.

Editing. While not strictly necessary, it is very useful to have a server part that allows interactive editing of the formal text in a way that resembles local work. Formal proof editors typically offer many advanced semantic features, such as stepping through a proof with the interactive prover and rendering the resulting proof state. Having some of the most used features as a part of the web-based editor is likely to make immediate browser-based updates of the wiki much more attractive for casual readers.

(Semantic) Assisting tools. Searching for suitable previously defined terminology and suitable theorems is one of the main activities when formalizing mathematics. The better this process is assisted, the more efficient are the authors of formal articles. As any user of today's web search engines knows, server-based technology allows much more sophisticated search tools that can make use of much more expensive processing steps than the user's machine, and that can also benefit from processing much more data than is available on the user's machine. A typical example would be suggesting hints for a new proof based on machine learning on the large libraries of previous proofs stored in the wikis, or running dozens of automated theorem provers on a particular interactively entered goal, making use of a large ("cloud-based") parallelization on the server.

4 Examples of Formal Wikis and Related Systems

The existing examples and prototypes of formal mathematical wikis (and related systems) can be broadly divided by whom they serve and what is their primary purpose.

4.1 Targeted Formal Wikis: Mizar Wiki and Others

Wikis for a particular formal library or a formalization project are perhaps the most visible examples of a wiki-like formal-proof technology. The main example is the *Mizar wiki* prototype [3,53]. It implements fast parallelized server verification, library update and versioning on the server, and HTML rendering for the Mizar system and its large formal library. The HTML rendering can produce all kinds of additional semantic information such as linking symbols to their definitions, showing of the proof state after each proof step, linking of theorems and concepts to Wikipedia and DBpedia, etc.

The verification and rendering are implemented just as hooks to an advanced distributed version control system (DVCS, `git` in this case). This allows the authors to work with the wiki locally, collaborate with others using just the DVCS features, and replicate and distribute the wiki very easily, just as if it was a standard software project. The distinguished Mizar wiki server receives updates from authorized users (as `git` *pushes*), using authorization systems such as `gitolite`[6], and verifies the updates prior to including them into the main or other branch.

The verification against a particular branch of the large library is done using a smart copy-on-write filesystem such as ZFS or BTRFS, making it possible to keep many versions (possibly for many users) of the large library and compiled files on the server without significant space overhead and repetition of work and making the cloning and creation of new (possibly experimental) versions very cheap and fast. The fast cloning also serves for the verification process, which necessarily first has to update the existing library before the library gets formally re-verified. If the re-verification fails, resulting in the rejection of the library update, the new ZFS or BTRFS clone is simply deleted, and the library lives in its original state. If the particular update of a particular version of the library satisfies the verification policy (and the library is thus in some kind of a consistent state after the update), the library is also re-rendered and made available for browsing and further updates.

Updates can be done either using a web interface or by power-users via git. The updates from git can be used for multifile updates (such as library-scale symbol renaming) that would break the library consistency if done on a file-by-file basis. Rudimentary search tools exist for the Mizar wiki, however most of the advanced semantic search functions are not yet integrated with it, and instead run as separate tools produced for distinguished versions of the Mizar library. Similarly, only basic web editing is implemented.

[6] http://gitolite.com/

Similar early prototype wiki has been built for Coq [3], and a number of formalization projects and libraries share today some of these features. For example the Coq Users' Contributions[7] are managed inside a VCS, automatically compiled and rendered, however the versioning and user submission are quite restricted. The Isabelle Archive of Formal Proofs[8] (AFP) allows authors to update their entries via a DVCS (mercurial) and also provides automated server-based checking, however the HTML rendering is very limited, and the users typically cannot update other entries. In this sense, the archive resemble an evolving online journal, rather than a massively collaborative wiki or a software project.

4.2 Wikis Focused on Editing Support: ProofWeb

To use a proof assistant, one needs to install some software. In the case of HOL Light one needs OCaml with a particular version of CamlP5 compiled with special flags (different than those used by popular Linux distributions), and HOL Light code itself, possibly with checkpointing software. To use an interface to access the prover, one needs the Emacs mode and one of the supported Emacs versions. The process described above is already complicated, not to mention other operating systems and architectures, or additional desirable patches and libraries, or less commonly used provers. The first web interface to the Coq system, LogiCoq [43], would not provide any support for editing: the whole buffer would be sent with standard HTTP request and refreshes the whole page.

ProofWeb [22] provides a web-interface to various proof assistants, that allows ProofGeneral-style [6] interaction. It implements a client-server architecture with a minimal lightweight client interpreted by the browser, a specialized HTTP server and background HTTP based communication between them. The key element of the architecture is the asynchronous DOM modification technique (sometimes referred to as AJAX - Asynchronous JavaScript and XML or Web application). The client part is stored on the server, and when the user accesses the interface page, it is downloaded by the browser, which is able to interpret it without any installation. The user of the interface, accessing it with the browser, does not need to do anything when a modification is done on the server. Every time the user accesses a prover, the version of the prover that is currently installed on the server is used. The user can access any of the provers installed on the server (ProofWeb supports Coq, Isabelle, Matita, Lego, Plastic, and has a minimal support for HOL Light).

The first wiki for Coq that integrated formal text editing with proof assistant feedback [10] was implemented as an extension of the MediaWiki engine, that used ProofWeb for editing pages and rendered the completed articles with Coqdoc in the viewing mode allowing LaTeX snippets. The convenience of a centralized proof assistant environment made it also appealing for teaching (Fig. 1). By defining special tactics for basic logical rules and proof tree rendering code [28]

[7] http://www.lix.polytechnique.fr/coq/pylons/contribs/index
[8] http://afp.sourceforge.net/

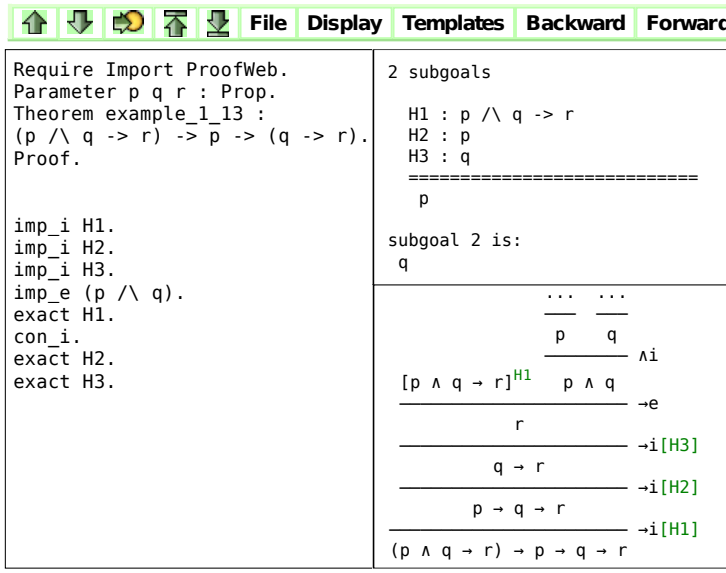

Figure 1: The ProofWeb interface editing a Coq proof script with a Gentzen-style proof tree.

ProofWeb became a convenient tool for various computer courses and has been to date used in 49 courses at 12 universities [20].

A dedicated web interface has also been developed for Matita [5]. Apart from allowing Unicode input and syntax highlighting, it can better exploit the hypertextual document structure offered by the prover, by providing various annotations and active elements. Similarly the Clide [46] interface is a web-reimplementation of the Isabelle interface. It provides a document-oriented interaction with the prover, and allows collaborative editing of an Isabelle script: the edit operations performed by each user are immediately propagated to all.

4.3 Meta Wikis: Agora, Flyspeck Wiki, and Others

While it has so far turned out to be hard to make a one-for-all formalization system and a formal library, the efforts to make a unified interface to different corpora of existing formal mathematics have never stopped.

A recent example is the *Agora wiki* prototype by Tankink [50]. Here, the user combines informal narratives written in the Creole syntax with antiquotations that allow transclusion of formal texts from an arbitrary formal library that has been suitably annotated using the OMDoc ontology developed by Lange [34,35]. This ontology provides a wide supply of types of mathematical knowledge items, as well as types of *relations* between them, e.g. that a proof proves a theorem. It is a reimplementation of the conceptual model of the OMDoc XML markup

language [31] for the purpose of providing semantic Web applications with a vocabulary of structures of mathematical knowledge.

Regardless of the exact details of the formal systems involved, and their output, the annotation process generally yields HTML+RDFa, which uses the OMDoc ontology as a vocabulary. For example, if the formal document contains an HTML rendition of the Binomial Theorem, Agora expects the following result (where the prefix *oo:* has been bound to the URI of the OMDoc ontology[9]):

```
<span typeof="oo:Theorem" about="#BinomialTheorem">...</span>
<span typeof="oo:Proof"><span rel="oo:proves" resource="#BinomialTheorem"/>
...</span>
```

The "..." in this listing represent the original HTML rendition of the formal text, possibly including the information that was used to infer the annotations now captured by the RDFa attributes. `@about` assigns a URI to the annotated resource; here, we use fragment identifiers within the HTML document.

The corresponding RDF annotation has been so far done for the Mizar, Flyspeck and Coq libraries. For instance in Mizar this was done as a part of the XSL transformation that creates HTML from the Mizar's custom semantic XML format [51]. While the OMDoc ontology defines vocabulary that seems suitable also for many Mizar internal proof steps, the current Mizar implementation only annotates the main top-level Mizar items, together with the top-level proofs. Even with this limitation this has already resulted in about 160000 annotations exported from the whole MML. The existing Mizar HTML namespace was re-used for the names of the exported items, such that, for example, the Brouwer Fixed Point Theorem:[10]

```
:: $N Brouwer Fixed Point Theorem
theorem Th14:
  for r being non negative (real number), o being Point of TOP-REAL 2,
      f being continuous Function of Tdisk(o,r), Tdisk(o,r)
  holds f has_a_fixpoint
proof ...
```

gets annotated as[11]

```
<div about="#T14" typeof="oo:Theorem">
  <span rel="owl:sameAs"
        resource="http://dbpedia.org/resource/Brouwer_Fixed_Point_Theorem"/> ...
  <div about="#PF23" typeof="oo:Proof"><span rel="oo:proves" resource="#T14"/> ... </div>
</div>
```

Apart from the appropriate annotations of the theorem and its proof, an additional *owl:sameAs* link is produced to the DBpedia (Wikipedia) "Brouwer_Fixed_Point_Theorem" resource. Such links are produced for all Mizar theorems and concepts for which the author defined a long (typically well-known) name using the Mizar `::$N` pragma. Such pragmas provide a way for the users to link the formalizations to Wikipedia (DBpedia, ProofWiki, PlanetMath, etc.),

[9] http://omdoc.org/ontology#

[10] http://mizar.cs.ualberta.ca/~mptp/7.12.02_4.178.1142/html/brouwer.html#T14

[11] `T14` is a *unique internal* Mizar identifier denoting the theorem. `Th14` is a (possibly non-unique) *user-level* identifier (e.g., `Brouwer` or `SK300` would result in `T14` too).

and the links allow data consumers (like Agora) to automatically mesh together different (Mizar, Coq, etc.) formalizations using DBpedia as the common namespace.

A particular instantiation of Agora is the *Flyspeck wiki* prototype [49] used for aligning, cross-linking, and potential further joint refactoring of Hales's informal Flyspeck book [16] about the proof of the Kepler Conjecture, and the corresponding formal Flyspeck development. This alignment takes advantage of the following annotated LaTeX form (developed by Hales in his book), which already cross-links informal objects to some of the formal counterparts (formally defined symbols and theorem names living in the formalization):

```
\begin{definition}[polyhedron]\guid{QSRHLXB}
A \newterm{polyhedron} is the
intersection of a finite number of closed half-spaces in
$\ring{R}^n$.
\end{definition}

\begin{lemma}[Krein--Milman]\guid{MUGGQUF}
Every compact convex set $P\subset\ring{R}^n$ is the convex hull
of its set of extreme points.
\end{lemma}
```

where QSRHLXB and MUGGQUF are the identifiers of the formal definition and theorem. The text contains many further mappings between informal and formal concepts, e.g.:

```
\formaldef{$\op{azim}(x)$}{azim\_fan}
\formaldef{M\"obius contour}{is\_Moebius\_contour}
\formaldef{half space}{closed\_half\_space, open\_half\_space}
```

The wiki relies on MathJaX for rendering the rich informal mathematics, and on custom transformations from LaTeX to the Creole wiki syntax. The formal text is automatically included as formal Agora islands, and aligned with the corresponding informal snippets, allowing their simultaneous view.

A related project is the MMT [44] formal logical framework and theory browser, where different formal libraries together with their foundations are *deeply embedded* into the logic of the framework, allowing translations between them and their (at least theoretical) combined use within the framework. The associated HTML theory browser already includes versions of the Mizar [21] and HOL Light libraries [23], however, so far this technology rather serves interested readers than as an user-updatable collaborative wiki.

Another interesting project that combines statistical and semantic methods for aligning the terminologies and theorems in different formal libraries has been started recently [12]. This project shows that quite a lot of such alignment can be achieved fully automatically based on the structure of the library statements. Such tools could complement the manual alignment and annotations used by systems like Agora.

4.4 Server-based Assisting Tools: HOL(y)Hammer and MizAR

In general, collaboration with machines is a very interesting aspect of the fully formal domain. The large formal libraries can be subjected both to various data-mining and machine learning methods that are being developed for less semantic

domains such as web search. On the other hand, one can also use very strong semantic search methods such as automated theorem proving (ATP), to assist the humans with finding proofs. Since the final correctness and consistency of such strong machine assistance is checked by the ITP systems (which are today very secure), one can use very efficiently implemented AI/ATP tools as parts of such toolchains, without a risk of introducing an incorrect proof due to implementational errors. The two main examples of such server-based assistance systems are the HOL(y)Hammer system serving the HOL Light users, and the MizAR system, serving the Mizar users. Since the two systems are otherwise quite similar, below we only explain their workings for the case of HOL(y)Hammer. The details of MizAR can be found in [26,54,55].

HOL(y)Hammer is an online AI/ATP service for formal (computer-understandable) mathematics encoded in the HOL Light system. The service allows its users to upload or modify and automatically process an arbitrary formal development (project) based on HOL Light, and to attack with automated theorem provers arbitrary conjectures that use the concepts defined in some of the uploaded projects. For that, the service uses several automated reasoning systems combined with several premise selection methods [33] trained on all the project proofs. The projects that are readily available on the server for such query answering include the recent versions of the Flyspeck, Multivariate Analysis and Complex Analysis libraries. The wiki-like features include upload and modification of existing projects, git-based versioning, HTML rendering of the libraries and their optional linking to Wikipedia and DBpedia and production of the OMDoc-based annotations for systems like Agora. The service runs on a 48-CPU server, currently employing in parallel for each task 7 AI/ATP combinations and 4 decision procedures that contribute to its overall performance. The current version of the system can prove about 40% of all Flyspeck toplevel lemmas fully automatically, thus significantly speeding up the formalization efforts.

The overall problem solving architecture without the updating functions is shown in Figure 2. The service receives a query (a HOL conjecture to prove, possibly with local assumptions) generated by one of the clients/frontends (Emacs, web interface, HOL session, etc.). The parsed query is processed in parallel by the (time-limited) AI/ATP combinations and the native HOL Light decision procedures (each managed by its forked HOL Light process, and terminated/killed by the master process if not finished within its global time limit). Each of the AI/ATP processes computes a specific feature representation of the query (used for knowledge selection), and sends such features to a specific instance of a premise advisor trained (using the particular feature representation) on previous proofs. Each of the advisors replies with a specific number of premises, which are then translated to a suitable ATP format, and written to a temporary file on which a specific ATP is run. The successful ATP result is then (pseudo-)minimized, and handed over to the combination of HOL Light proof-reconstruction procedures. These procedures again run in parallel, and if any of them is successful, the result is sent as a particular tactic application to the

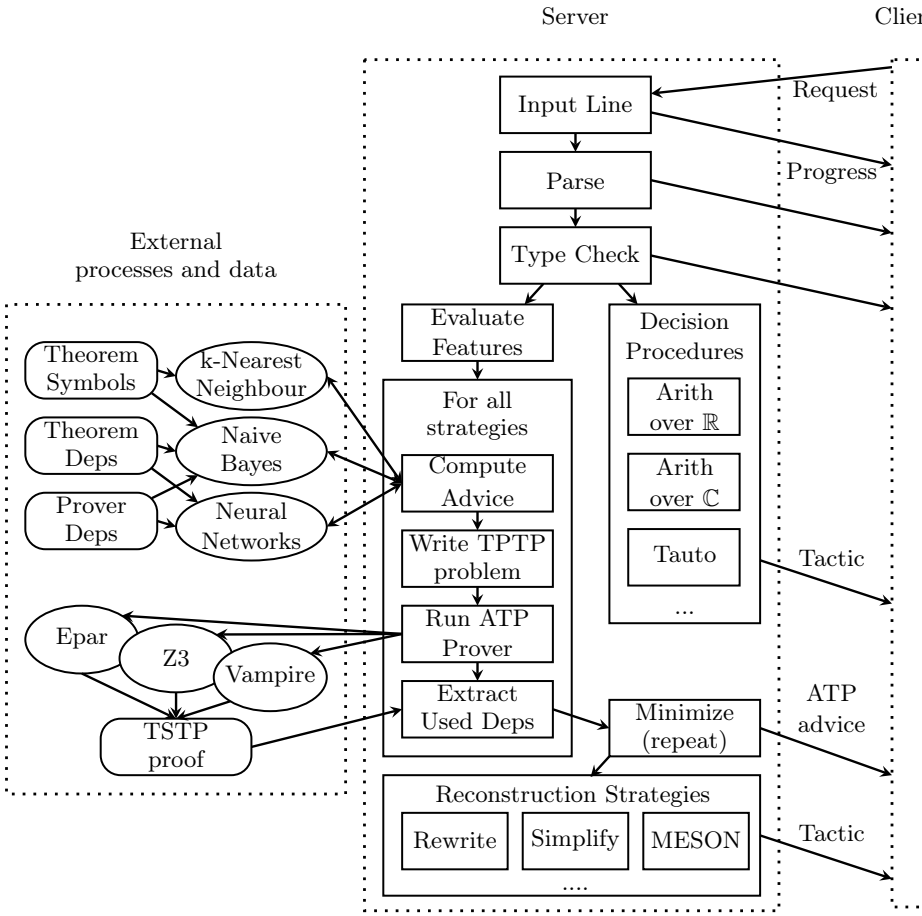

Figure 2: Overview of the HOL(y)Hammer problem solving functions

frontend. In case a native HOL Light decision procedure finds a proof, the result (again a particular tactic application) can be immediately sent to the frontend.

Since formal projects frequently modify their terminology and theorem names, and updating all the AI/ATP data on the server is expensive, HOL(y)Hammer has introduced *recursive content-based encoding* of the formal theorem and symbol names [52] for all projects and for re-using information across different projects and their versions. This is implemented as follows:

1. The name of every defined symbol is replaced by the content hash (we use MD5) of its variable-normalized definition containing the full types of the variables. This definition already uses content hashes instead of the previously defined symbols. This means that symbol names are no longer relevant in the whole project, neither white space and variable names.

2. The name of each theorem is also replaced by the content hash of its (analogously normalized) statement.
3. The proof-dependency data extracted in the content encoding from all projects are copied to a special "common" directory.
4. Whenever a project P is started or modified, we find the intersection of the content-encoded names of the project's theorems with such names that already exist in other projects/versions.
5. For each of such "already known" theorems T in P, we re-use for the AI/ATP systems its "already known" proofs D that are *compatible* with P's proof graph. This means, that the names of the proof dependencies of T in D must also exist in P (i.e., these theorems have been also proved in P, modulo the content-naming), and that these theorems precede T in P in its chronological order of library development (otherwise we might get cyclic training data for P).

This procedure very significantly improves the live updates of the projects, because typically over 90% of the library's new version remains the same after these normalization steps [25].

5 Conclusions

The world of formal mathematical wikis is still very young, and it is even hard to say that there is a single most advanced system or that a particular wiki-like system has been largely adopted by the formal community. In this article we have therefore tried to summarize the topics that have emerged recently, when the venerable fifty-year old field of formalization of mathematics, with its venerable history of efforts such as QED, got interested in the collaborative wiki-like technologies.

Some of the pieces of the technology have already made it to the formal mainstream. Examples are advanced version control systems that have additional verification and rendering features. Other wiki-related techniques, such as web-based editing supported by verification in ProofWeb, or strong semantic advisors such as HOL(y)Hammer and Miz$\mathbb{AR}$ have also already found some adopters. The various meta-wiki ideas and ideas linking the formal libraries with semantic web and informal mathematics are however still waiting to be fully developed and utilized.

There are many interesting topics that this brief overview has not dealt with. For example, change propagation is an important research topic and fine-grained dependency tracking can very significantly improve the speed of consistency checking of library updates [4]. Strong semantic assistance can include semi-automated translations between the informal and formal mathematics, with the outlook of gradually training smarter and smarter natural-language-understanding tools for mathematics on such aligned corpora [27]. And very interesting topics arise with future – hopefully wiki-supported – formalizations of other exact sciences such as physics. There, the many possible models of the (never completely known) real world need to somehow formally co-exist, while still being linked to

accessible informal explanations about their relations to the underlying reality that they try to capture.

Perhaps the biggest challenge of formal mathematical wikis thus today seem to be the varied and evolving treatments of formality. While fully formal and computer-verified mathematics has made great steps forward in the last decade, there is still no agreement on the ultimate formalism. This includes the particular steps and stages that should link informal and formal knowledge and smoothly proceed in a wiki-like way from one to the other. On the other hand, formal mathematics and the strong semantic tools that have become available for its assistance are already clearly showing how disruptive for our way of doing science could be frameworks that achieve very smooth transition from informal (human-understandable) to formal (machine-understandable) scientific knowledge.

6 Acknowledgments

The following colleagues have collaborated with us on various aspects of formal wikis and on a number of formal wiki-related systems mentioned in this paper: Mark Adams, Jesse Alama, Grzegorz Bancerek, Kasper Brink, Johan Commelin, Pierre Corbineau, Thibault Gauthier, Herman Geuvers, Mihnea Iancu, James McKinna, Michael Kohlhase, Christoph Lange, Lionel Mamane, Florian Rabe, Piotr Rudnicki, Geoff Sutcliffe, Carst Tankink, Jiri Vyskocil and Freek Wiedijk. Thanks to the anonymous SWCS referees for their valuable comments.

References

1. The QED Manifesto. In Alan Bundy, editor, *CADE*, volume 814 of *LNCS*, pages 238–251. Springer, 1994.
2. Sanaz Khan Afshar, Umair Siddique, Mohamed Yousri Mahmoud, Vincent Aravantinos, Ons Seddiki, Osman Hasan, and Sofiène Tahar. Formal analysis of optical systems. *Mathematics in Computer Science*, 8(1):39–70, 2014.
3. Jesse Alama, Kasper Brink, Lionel Mamane, and Josef Urban. Large formal wikis: Issues and solutions. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Symposium on Intelligent Computer Mathematics (CICM 2011)*, volume 6824 of *LNCS*, pages 133–148. Springer, 2011.
4. Jesse Alama, Lionel Mamane, and Josef Urban. Dependencies in formal mathematics: Applications and extraction for Coq and Mizar. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics (CICM 2012)*, volume 7362 of *LNCS*, pages 1–16. Springer, 2012.
5. Andrea Asperti and Wilmer Ricciotti. A web interface for Matita. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics (CICM 2012)*, volume 7362 of *LNCS*, pages 417–421. Springer, 2012.
6. David Aspinall. Proof General: A generic tool for proof development. In Susanne Graf and Michael I. Schwartzbach, editors, *6th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2000)*, volume 1785 of *LNCS*, pages 38–42. Springer, 2000.

7. Grzegorz Bancerek and Piotr Rudnicki. A Compendium of Continuous Lattices in MIZAR. *J. Autom. Reasoning*, 29(3-4):189–224, 2002.
8. Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe. KnowWE: a Semantic Wiki for knowledge engineering. *Appl. Intell.*, 35(3):323–344, 2011.
9. The Coq Proof Assistant. http://coq.inria.fr.
10. Pierre Corbineau and Cezary Kaliszyk. Cooperative repositories for formal proofs. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Proc. of the 6th International Conference on Mathematical Knowledge Management (MKM'07)*, volume 4573 of *LNCS*, pages 221–234. Springer Verlag, 2007.
11. N.G. de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In M. Laudet, editor, *Proceedings of the Symposium on Automatic Demonstration*, pages 29–61, Versailles, France, December 1968. Springer-Verlag LNM 125.
12. Thibault Gauthier and Cezary Kaliszyk. Matching concepts across HOL libraries. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics (CICM 2014)*, volume 8543 of *LNCS*, pages 267–281. Springer, 2014.
13. Georges Gonthier. The four colour theorem: Engineering of a formal proof. In Deepak Kapur, editor, *Computer Mathematics, 8th Asian Symposium, ASCM 2007, Singapore, December 15-17, 2007. Revised and Invited Papers*, volume 5081 of *LNCS*, page 333. Springer, 2007.
14. Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP*, volume 7998 of *LNCS*, pages 163–179. Springer, 2013.
15. Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Mizar in a nutshell. *J. Formalized Reasoning*, 3(2):153–245, 2010.
16. Thomas Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 2012.
17. Thomas C. Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Hoai Thi Ta, Trung Nam Tran, Diep Thi Trieu, Josef Urban, Ky Khac Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *CoRR*, abs/1501.02155, 2015.
18. John Harrison. HOL Light: A tutorial introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *FMCAD*, volume 1166 of *LNCS*, pages 265–269. Springer, 1996.
19. John Harrison, Josef Urban, and Freek Wiedijk. History of interactive theorem proving. In Jörg H. Siekmann, editor, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 135 – 214. North-Holland, 2014.
20. Maxim Hendriks, Cezary Kaliszyk, Femke van Raamsdonk, and Freek Wiedijk. Teaching logic using a state-of-the-art proof assistant. *Acta Didactica Napocensia*, 3(2):35–48, June 2010.
21. Mihnea Iancu, Michael Kohlhase, Florian Rabe, and Josef Urban. The Mizar Mathematical Library in OMDoc: Translation and applications. *J. Autom. Reasoning*, 50(2):191–202, 2013.

22. Cezary Kaliszyk. Web interfaces for proof assistants. In S. Autexier and C. Benzmüller, editors, *Proc. of the Workshop on User Interfaces for Theorem Provers (UITP'06)*, volume 174[2] of *ENTCS*, pages 49–61, 2007.
23. Cezary Kaliszyk and Florian Rabe. Towards knowledge management for HOL Light. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics (CICM 2014)*, volume 8543 of *LNCS*, pages 357–372. Springer, 2014.
24. Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014.
25. Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.
26. Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reasoning*, 55(3):245–256, 2015.
27. Cezary Kaliszyk, Josef Urban, Jirí Vyskocil, and Herman Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics (CICM 2014)*, volume 8543 of *LNCS*, pages 435–439. Springer, 2014.
28. Cezary Kaliszyk and Freek Wiedijk. Merging procedural and declarative proof. In Stefano Berardi, Ferruccio Damiani, and Ugo de'Liguoro, editors, *Proc. of the Types for Proofs and Programs International Conference (TYPES'08)*, volume 5497 of *LNCS*, pages 203–219. Springer Verlag, 2008.
29. Matt Kaufmann and J. Strother Moore. An ACL2 tutorial. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *21st International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2008*, volume 5170 of *LNCS*, pages 17–21. Springer, 2008.
30. Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: formal verification of an operating-system kernel. *Commun. ACM*, 53(6):107–115, 2010.
31. Michael Kohlhase. *OMDoc - An Open Markup Format for Mathematical Documents [version 1.2]*, volume 4180 of *LNCS*. Springer, 2006.
32. Daniel Kühlwein, Jasmin Christian Blanchette, Cezary Kaliszyk, and Josef Urban. MaSh: Machine learning for Sledgehammer. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP 2013*, volume 7998 of *LNCS*, pages 35–50. Springer, 2013.
33. Daniel Kühlwein, Twan van Laarhoven, Evgeni Tsivtsivadze, Josef Urban, and Tom Heskes. Overview and evaluation of premise selection techniques for large theory mathematics. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 378–392. Springer, 2012.
34. Christoph Lange. OMDoc ontology. http://kwarc.info/projects/docOnto/omdoc.html, 2011.
35. Christoph Lange. Ontologies and languages for representing mathematical knowledge on the semantic web. *Semantic Web*, 4(2):119–158, 2013.
36. Christoph Lange, Colin Rowat, and Manfred Kerber. The ForMaRE project - formal mathematical reasoning in economics. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Intelligent Computer Mathematics (CICM 2013)*, volume 7961 of *LNCS*, pages 330–334. Springer, 2013.

37. Christoph Lange and Josef Urban, editors. *Proceedings of the ITP 2011 Workshop on Mathematical Wikis (MathWikis)*, number 767 in CEUR Workshop Proceedings, Aachen, 2011.
38. Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, 2009.
39. Roman Matuszewski, editor. *The QED Workshop II*, 1995. Warsaw University Technical Report No. L/1/95.
40. Grzegorz J. Nalepa. Collective knowledge engineering with semantic wikis. *J. UCS*, 16(7):1006–1023, 2010.
41. Ian Niles and Adam Pease. Towards a standard upper ontology. In *FOIS*, pages 2–9, 2001.
42. Sam Owre and Natarajan Shankar. A brief overview of PVS. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *21st International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2008*, volume 5170 of *LNCS*, pages 22–27. Springer, 2008.
43. Loïc Pottier. LogiCoq, 1999.
URL: http://wims.unice.fr/wims/wims.cgi?module=U3/logic/logicoq.
44. 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 (CICM 2013)*, volume 7961 of *LNCS*, pages 339–343. Springer, 2013.
45. D. Ramachandran, Reagan P., and K. Goolsbey. First-orderized ResearchCyc: Expressiveness and Efficiency in a Common Sense Knowledge Base. In Shvaiko P., editor, *Proceedings of the Workshop on Contexts and Ontologies: Theory, Practice and Applications*, 2005.
46. Martin Ring and Christoph Lüth. Collaborative interactive theorem proving with clide. In Gerwin Klein and Ruben Gamboa, editors, *5th International Conference on Interactive Theorem Proving, ITP 2014*, volume 8558 of *LNCS*, pages 467–482. Springer, 2014.
47. John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001.
48. Carsten Schürmann. The Twelf proof assistant. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics, TPHOLs 2009*, volume 5674 of *LNCS*, pages 79–83. Springer, 2009.
49. Carst Tankink, Cezary Kaliszyk, Josef Urban, and Herman Geuvers. Formal mathematics on display: A wiki for Flyspeck. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Intelligent Computer Mathematics (CICM 2013)*, volume 7961 of *LNCS*, pages 152–167. Springer, 2013.
50. Carst Tankink, Christoph Lange, and Josef Urban. Point-and-write - documenting formal mathematics by reference. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics (CICM 2012)*, volume 7362 of *LNCS*, pages 169–185. Springer, 2012.
51. Josef Urban. XML-izing Mizar: Making semantic processing and presentation of MML easy. In Michael Kohlhase, editor, *MKM*, volume 3863 of *LNCS*, pages 346–360. Springer, 2005.
52. Josef Urban. Content-based encoding of mathematical and code libraries. In Christoph Lange and Josef Urban, editors, *Proceedings of the ITP 2011 Workshop on Mathematical Wikis (MathWikis)*, number 767 in CEUR Workshop Proceedings, pages 49–53, Aachen, 2011.

53. Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. A wiki for Mizar: Motivation, considerations, and initial prototype. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *Intelligent Computer Mathematics (CICM 2010)*, volume 6167 of *LNCS*, pages 455–469. Springer, 2010.
54. Josef Urban, Piotr Rudnicki, and Geoff Sutcliffe. ATP and presentation service for Mizar formalizations. *J. Autom. Reasoning*, 50:229–241, 2013.
55. Josef Urban and Geoff Sutcliffe. Automated reasoning and presentation support for formalizing mathematics in Mizar. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *Intelligent Computer Mathematics (CICM 2010)*, volume 6167 of *LNCS*, pages 132–146. Springer, 2010.
56. Makarius Wenzel, Lawrence C. Paulson, and Tobias Nipkow. The Isabelle framework. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *TPHOLs*, volume 5170 of *LNCS*, pages 33–38. Springer, 2008.
57. Lee Worden. WorkingWiki: a MediaWiki-based platform for collaborative research. In Lange and Urban [37], pages 63–73.