Loki – Semantic Wiki with Logical Knowledge Representation*

Grzegorz J. Nalepa

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
gjn@agh.edu.pl

Abstract. To fulfill its ambitious promises, Semantic Web needs practical and accessible tools for collective knowledge engineering. Recent developments in the area of semantic wikis show how such tools can be built. However, existing semantic wikis implementations have both conceptual and technological limitations. These limitations are in the areas of knowledge representation, strong reasoning as well as appropriate user interfaces. In this paper a proposal of a new semantic wiki is presented. *Loki* uses a coherent logic-based representation for semantic annotations of the content. The same representation is used for implementing reasoning procedures. The representation uses the logic programming paradigm and the Prolog programming language. The proposed architecture allows for rule-based reasoning in the wiki. It also provides a compatibility layer with the popular Semantic MediaWiki (SMW) platform, directly parsing its annotations. In the paper a prototype implementation called PlWiki is described, and a practical use case is given. [1]

1 Introduction

The Internet has been evolving from a network that stores and massively links human readable information to an information system where machine readable content can be processed by intelligent agents supporting humans in their information interpretation task. The vision of the Semantic Web [2,3] promises interoperable web services that are able to automatically process structured data and perform simple inference tasks. To implement such an architecture the Semantic Web initiative proposed the Semantic Web stack. This layered architecture provided number of data structurization languages such as XML [4], as well as knowledge representation languages of different expressiveness e.g. RDF [5], RDFS [6], and OWL [7], as well as query languages including SPARQL [8]. The new version of OWL [9] supports different perspectives on ontology development with OWL 2 profiles [10]. Current efforts are focused on providing a rule layer

* The paper is supported by the *BIMLOQ* Project funded from 2010–2012 resources for science as a research project.

[1] This paper is partially based on the results presented in the paper [1] presented at the ICCCI conference in Wrocław in 2010.

combined with ontologies. While some different solutions exist, e.g. SWRL [11], or DLP [12], so far they have not been able to overcome all of the challenges [13] of the rules and description logics [14] integration. These methods and technologies, combined with the collective nature of the Web, aim for providing solutions for massively distributed content and knowledge engineering.

To fulfil these promises one needs not just methods and languages, but engineering solutions, practically applicable for a wide number of users, not only experts. So there is a growing pressure for the Semantic Web development to focus on usable tools, not only on conceptual inventions. Recently the so-called semantic wikis proved to be an effective Semantic Web application meeting some of the most important requirements for collective knowledge engineering and querying on the Web. In fact the technology is young and evolving, with number of new solutions being developed and old ones disappearing. Apparently some of the main aspects of the semantic wikis evolution and development are effective and intuitive user interfaces, that help in using the semantic content, expressive knowledge representation [15], as well as reasoning and inference [16]. Even though there has been a rapid development in these areas in recent years, the technology still needs to address its current limitations in these areas to become a successful collective knowledge engineering solution.

The main contribution of this paper is a proposal of a new semantic wiki architecture. It uses a strong, logic-based representation for knowledge processing in the wiki. Such an approach allows for expressive knowledge engineering including decision rules. As the low-level knowledge representation language Prolog is used. However, a dedicated rule engine supporting decision tables and trees is also provided. In the paper a proof of concept implementation of this idea is discussed using an example.

In the rest of the paper the semantic wikis technology is described in Sect. 2. Then, the drawbacks of the current solutions, as well as development trends give motivation for the research as presented in Sect. 3. The proposal of *Loki*, a wiki based on a logical knowledge representation, is introduced in Sect. 4. The system has a prototype implementation called PlWiki presented in Sect. 5. The evaluation of the approach is given in Sect. 6. Directions for the future development and the summary of the paper are contained in the final section.

2 Semantic Wikis Technology

2.1 Wiki systems

A wiki can be simply described as a collection of linked webpages Wikis appeared in the mid 90s to provide a conceptually simple tool for massively collaborative knowledge sharing and social communication. Some important features of wiki include: remote editing using a basic web browser, simplified text-based syntax for describing context (wikitext), rollback mechanism, thanks to built in versioning, diversified linking mechanism (including internal, interwiki, and external links), access control mechanisms (from simple ones, to regular ACL

(Access Control Lists) solutions.) A comprehensive comparison of different wiki systems can be found on http://www.wikimatrix.org.

While wiki systems provide an abstract representation of the content they store, as well as standard searching capabilities, they lack facilities helping in expressing the semantics of the stored content.[2] This is especially important in the case of collaborative systems, where number of users work together on the content. This is why wikis became one of the main applications and testing areas for the Semantic Web technologies.

2.2 Semantic Wikis

The so-called *semantic wikis* enrich standard wikis with the semantic information expressed by a number of mechanisms. Some basic questions every semantic wiki needs to address are [17]: 1) how to annotate content?, 2) how to formally represent content?, 3) how to navigate content?. In last several years multiple implementations of semantic wikis have been developed, including IkeWiki [18], OntoWiki [19], SemanticMediaWiki [20], SemperWiki [21], SweetWiki [22], and AceWiki [23]

In general, in these systems the standard wikitext is extended with semantic annotations. These include relations (represented as RDF triples) and categories (here RDFS is needed). It is possible to query the semantic knowledge, thus providing dynamic wiki pages (e.g. with the use of SPARQL). Some of the systems also allow for building an OWL ontology of the domain to which the content of the wiki is related. This extension introduces not just new content engineering possibilities, but also semantic search and analysis of the content.

The summary of semantic wiki systems is available[3], some of them are in the development stage, while others have been discontinued.Selected implementations are briefly characterized. IkeWiki [18] was one of the first semantic wikis with powerful features. The Java-based systems offered semantic annotations with RDF and OWL support, with OWL-RDFS reasoning. It had introduced simple ontology editing in the wiki with certain visualization techniques. OntoWiki [19] provided improved visualization of the ontological content and more advanced collaborative editing features. SemperWiki [21] used advanced semantic annotations with explicit differentiation of documents and concepts. SweetWiki [22] was based on the CORESE engine (an RDF engine based on Conceptual Graphs (CG)) and used ontology-based model for wiki organization.

When it comes to active implementations, one of the most popular is Semantic MediaWiki [20] (SMW). It is built on top of the MediaWiki engine, and extends it with lightweight semantic annotations and simple querying facilities. AceWiki [23] uses Attempto Controlled English (ACE) for natural language processing in the wiki. A recent FP7 project Kiwi (http://www.kiwi-project.eu) aims at providing a collaborative knowledge management based on semantic

[2] Besides simple tagging mechanisms, that can later be used to create the so-called *folksonomies*.

[3] See http://semanticweb.org/wiki/Semantic_Wiki_State_Of_The_Art.

wikis (it is the continuation of IkeWiki effort) [24]. It can be observed that from the knowledge engineering point of view, expressing semantics (i.e. representing knowledge) is not enough. In fact a knowledge-based system should provide both effective knowledge representation and processing methods. In order to extend semantic wikis to knowledge-based systems, ideas to use rule-based reasoning and problem-solving knowledge have been introduced. An example of such a system is the *KnowWE* semantic wiki [25,26]. The system allows for introducing knowledge expressed with decision rules and trees related to the domain ontology. A research aiming at the analysis of social collaborations in such systems is also active [27,28]. Number of new solutions use the so-called Web 2.0 tools and methods to enrich knowledge acquisition and management, e. g. see [29,30].

2.3 Wiki Feature Matrix

A new matrix can help to simplify the selection of a semantic wiki system for a given work task has been introduced by Baumeister in [31]. Main motivation for it was the extension and specialization of the matrix[4] used by the semantic wiki community to track the recent developments of semantic wiki features. That matrix lists number of features such as *editing paradigm*, *annotation mechanism*, *programming language* and *license*. However, this categorization seems too general for systems using expressive knowledge formalizations. Therefore, in [31] a supplementary set of feature categories has been proposed. These include:

- Targeted applications, e.g. Community-based ontology engineering, E-learning.
- Underlying knowledge representation, w.r.t.
  - Subject granularity (One/multiple concepts/properties for each wiki page),
  - Knowledge representation language (RDF(S), OWL, other),
  - Additional knowledge sources (SWRL rules, Prolog rules, Model-based knowledge, Text in controlled language),
  - UI for knowledge capture and sharing,
    - Knowledge editing paradigm: e.g. Inline text markup, Semantic forms/ visual editors,
    - Semantic search/knowledge retrieval: Used query language (SPARQL, Datalog-like queries), Query integration (Special-purpose forms, queries embedded into the wiki article), Further capabilities of knowledge use (Generated knowledge-based interviews, Prolog queries).
    - Semantic navigation (Extended links within wiki article, Generation of fact sheets)
  - Connectivity, Import/export facilities (e.g. RFS(S)/OWL, SWRL, RIF),
- Extensibility, including Extension mechanism (e.g. Plug-in mechanism, Code-in mechanism), and Existing extensions/modifications.

This categorization aims at capturing the conceptual properties of semantic wikis. Hopefully it will be adapted (possibly in a modified form) by the community. The features of KnowWe have been summarized in the matrix in [31]. In this work the matrix will be used to summarize the solution proposed herein.

[4] `http://semanticweb.org/wiki/Semantic_Wiki_State_Of_The_Art`

3 Motivation

The semantic wikis technology has been rapidly developing. The growing number of practical applications and regular users helps developers in getting feedback useful in development. New applications also reveal limitations of the existing technology and expose persistent challenges facing it. Some of the main areas of possible improvement include:

1. *user interface*, that allows for effective visualization of the semantically enriched content and supports its customization. It should also help in navigating in the wiki.
2. *knowledge representation*, which should be rich enough to be able to express explicit knowledge of different classes, including rules, but also decision trees and tables.
3. *reasoning* should be stronger than simple classification tasks, possibly rule-based, able to infer new facts and provide answers for complex queries.

Research presented in this paper addresses two of the above areas, namely knowledge representation and reasoning.

The principal idea consists in representing the semantic annotations in a formalized way, and simultaneously enriching them with an expressive rule-based knowledge representation. Semantic annotations should use standard solutions, such as RDF, and RDFS and possibly OWL for ontologies of concepts. Such a knowledge base should be homogeneous from the logical point of view, in order to allow strong reasoning support. The knowledge base is coupled with the contents of the wiki. Moreover, the basic wiki engine is integrated with an inference engine. Practical implementation of these concepts should provide backwards compatibility with important existing semantic wiki implementations. In the next section the design of a wiki system following these principles is given. Then a prototype implementation is described.

4 Loki Proposal

Logic-based Wiki, or *Loki* for short, uses the logic programming paradigm to represent knowledge in the wiki, including semantic annotations and rules.[5] The main design principles are as follows:

- provide an expressive underlying logical representation for semantic annotations and rules,
- allow for strong reasoning support in the wiki,
- preserve backward compatibility with existing wikis, namely SMW.

[5] While the solution presented here allows for processing both ontological information, as well as rules, it does not address directly the well-known problems related to possible integration scenarios of ontologies and rules, for example see [13,32,33].

The design has been based on the use of the Prolog programming language.

Prolog [34,35] is an expressive logic programming language. The Prolog system uses a knowledge base to answer queries. From a logical point of view, the knowledge base is composed of Horn clauses and a goal (so in fact it is a subset of First Order Predicate Logic). In Prolog two basic forms of statements are possible: facts and rules. While the default inference strategy is backward-chaining it is trivial to built custom meta-interpreters implementing any inference strategy. While Prolog has been traditionally related to Artificial Intelligence [36], it is a mature and universal programming language. Number of powerful implementations are currently available, e.g. SWI-Prolog [6]. It is a mature implementation widely used in research and education as well as for commercial applications. It provides a fast and scalable development environment, including graphics, libraries and interface packages, portable to many platforms, including Unix/Linux platforms, Windows, and MacOS X. SWI-Prolog provides a rich set of libraries, including the *semweb* library for dealing with standards from the W3C standard for the Semantic Web (RDF, RDFS and OWL). This infrastructure is modular, consisting of Prolog packages for reading, querying and storing Semantic Web documents. Additional modules also support TCP/IP networking and XML parsing. The SWI-Prolog environment is licensed under the LGPL.

To build Loki using Prolog several problems has been solved, namely:

- Prolog representation for the Semantic annotations of SMW,
- RDF and OWL support in Prolog,
- integration of the Prolog engine and a wiki engine,
- support for an expressive knowledge representation with decision tables and decision trees.

The solutions addressing these issues are described in the following subsections.

4.1 SMW Support in Loki

There are three main methods of semantic annotations in SMW [37]:

- categories – a simple form of annotation that allows users to classify pages. To state that article (Wiki page) belongs to the Category `Category1` one has to write `[[Category:Category1]]` within the article.
- relations – there is a possibility to describe relationships between two Wiki pages by assigning annotations to existing links. For example there is a relation `capital_of` between Warsaw and Poland. To express this one has edit the page Warsaw and add `[[capital_of::Poland]]` within the page content.
- attributes – allow to specify relationships of Wiki pages to things that are not Wiki pages. For example, one can state that Wiki page `Page1` was created at April 22 2009 by writing `[[created:=April 22 2009]]`.

[6] See `http://www.swi-prolog.org`

Annotations are usually not shown at the place where they are inserted. Category links appear only at the bottom of a page, relations are displayed like normal links, and attributes just show the given value. A factbox at the bottom of each page enables users to view all extracted annotations, but the main text remains undisturbed (see Fig. 1). After making annotations a factbox is displayed in each article, and this factbox also features quicklinks for browsing and searching. In addition to this users can search for articles using a simple query language that was developed based on the known syntax of the Wiki. For example to display city which is capital of Poland following code may be used:

```
{{#ask: [[Category:city]] [[capital of::Poland]]}}
```

The query functionality of Semantic MediaWiki can be used to embed dynamic content into pages, which is a major advantage over traditional wikis. Several other forms of queries can be found in the online documentation.

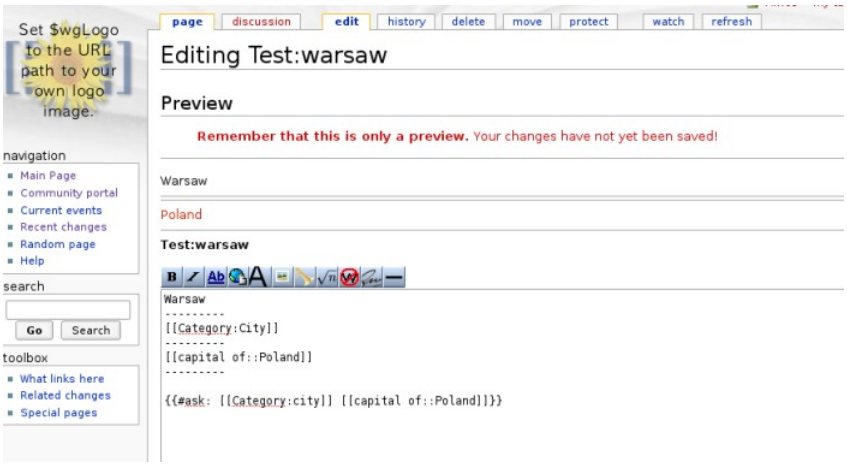

Fig. 1. Annotations in Semantic MediaWiki

Loki allows to describe categories, relations and attributes the same way as in SMW. They are represented by appropriate Prolog terms. Examples are as follows, with the SMW syntax gives first, and the corresponding Prolog representation below.

```
[[category:cities]] Warsaw is in Poland.
    wiki_category('cities','warsaw').

Warsaw is [[capital_of::Poland]].
    wiki_relation('subject_page_uri','capital_of','poland').
```

```
[[created:=April 22 2009]]
    wiki_attribute('subject_page_uri','created','pril 22 2009').
```

Loki also provides a direct support for RDF and OWL.

4.2 RDF and OWL Support

Plain RDF annotations are supported and separated from the explicit annotations mentioned above. For compatibility reasons, an RDF annotation can be embedded directly in the XML serialization, then it is parsed by the corresponding Prolog library, and turned to the internal representation, that can also be used directly. Using the *semweb/rdf_db* library SWI-Prolog's represents RDF triples simply as:

```
rdf(?Subject, ?Predicate, ?Object).
```

So mapping the above example would result in:

```
rdf('Warsaw',capital_of,'Poland').
```

RDFS is also supported by the *semweb/rdfs* library, e.g.:

```
rdfs_individual_of('Warsaw',cities).
```

SPARQL queries are handled by the *semweb/sparql_client* The SWI-Prolog RDF storage in highly optimized. It can be integrated with the provided RDFS and OWL layers, as well as with the *ClioPatria* platform[7] that provides SPARQL queries support. SWI-Prolog supports OWL using the Thea [38] library. It is an experimental feature, currently being evaluated.

4.3 Prolog Reasoning

Thanks to the full Prolog engine available in the wiki, the inference options are almost unlimited. Prolog uses backwards chaining with program clauses. However, it is very easy to implement meta-interpreters for forward chaining.

A simple clause finding recently created pages might be as follows:

```
recent_pages(Today,Page) :-
    wiki_attribut(Page,created,date(D,'May',2010)),
    I is Today - D,
    I < 7.
```

Compound queries can also be created easily and executed as Prolog predicates.

One should keep in mind, that the Prolog-based representation is quite close to the natural language. Not only on the semantical level, but to a degree also on the syntactic level. It is possible thanks to the operator redefinition. A simple example might be as follows:

[7] See `http://e-culture.multimedian.nl/software/ClioPatria.shtml`.

```
:- op(100,xfy,is_capital_of).

'Warsaw' is_capital_of 'Poland'.
```

The first line defines a new infix operator, that is use to construct a fact in the second line. Such an approach is close to controlled languages such as Attempto [39] used in Ace Wiki mentioned in Sect. 2. However, it is not used in the current implementation of Loki.

4.4 Rule-based Reasoning

An optional decision rule layer is also considered with the use of the HeaRT [40] runtime for the XTT2 framework [41,42,43]. XTT2 (*eXtended Tabular Trees v2*) knowledge representation incorporates attributive table format. Similar rules are grouped within separated tables, and the system is split into a network of such tables representing the inference flow – this compromises the visual representation of XTT2. Efficient inference is assured thanks to firing only rules necessary for achieving the goal. It is achieved by selecting the desired output tables and identifying the tables necessary to be fired first. The links representing the partial order assure that when passing from one table to another one, the latter can be fired since the former one prepares an appropriate context knowledge [43,40].

The visual representation is automatically transformed into HMR (HeKatE Meta Representation), a corresponding textual algebraic notation, suitable for direct execution by the rule engine. An example excerpt of HMR is:

```
<hmr>
xschm th: [today,hour] ==> [operation].
xrule th/1:
  [today eq workday, hour gt 17] ==>
  [operation set not_bizhours].
xrule th/4:
  [today eq workday, hour  in [9 to 17]] ==>
  [operation set bizhours].
</hmr>
```

The first line defines an XTT2 table scheme, or header, defining all of the attributes used in the table. Its semantics is as follows: "the XTT table *th* has two conditional attributes: *today* and *hour* and one decision attribute: *operation*". Then two examples of rules are given. The second rule can be read as: "Rule with ID *4* in the XTT table called *th*: if value of the attribute *today* equals (=) value *workday* and the value of the attribute *hour* belongs to the range ($\in$) $< 9, 17 >$ then set the value of the attribute *operation* to the value *bizhours*". The code is embedded into the wikitext using the `hmr` tags.

HeKatE RunTime (HeaRT) is a dedicated inference engine for the XTT2 rule bases [40]. The engine is highly modularized. It is composed of the main *inference module* based on ALSV(FD) logic [42,43]. A proposal of integrating ALSV(FD)

with Description logics has been given in [44]. It supports four types of inference process, Data and Goal Driven, Fixed Order, and Token Driven [43]. The engine is implemented in Prolog, using the SWI-Prolog stack. The main HMR parser is heavily based on the Prolog operator redefinition [34]. A dedicated forward and backward chaining meta interpreter is provided, implementing custom rule inference modes.

Following scenarios for rules embedding are considered including: 1) single table for single wiki page, and 2) rules working in the same content present in multiple pages in the same namespace. Rules are extracted by Loki parser and concatenated into a single HMR file corresponding to a wikipage or namespace.

4.5 Loki Architecture

Considering featured mentioned above, the following Loki architecture is given (observe Fig. 2). The wikitext from the regular wiki contains basic semantic annotations as in SMW. Additionally it contains Prolog code and HMR rules. These are separated by the Loki engine and combined into a Loki knowledge base. The main engine is integrated with the HeaRT interpreter. It also supports querying the knowledge base using both generic Prolog queries, as well as SPARQL. This architecture has been partially implemented with a proof-of-concept prototype called PlWiki described in the next section.

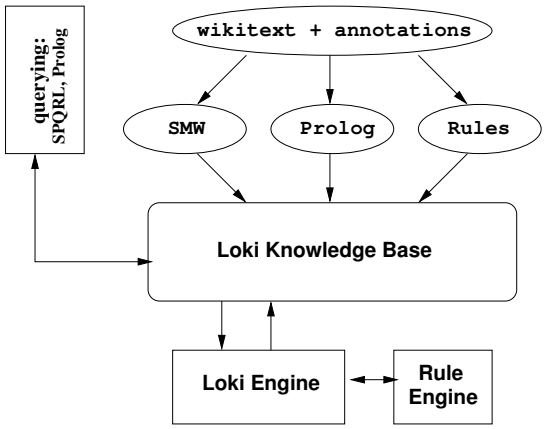

Fig. 2. Loki architecture

5 Prototype Implementation

A prototype implementation of the Loki architecture has been developed. The system is called PlWiki [1,45]. The main goal of the system design is to deliver a generic and flexible solution.

There are tens of wiki engines available. Most of them are similar w.r.t to main concepts and features. However, there are number of differences when it comes to the wikitext syntax, implementation and runtime environment, as well as extra features. This is why, instead of building yet another wiki engine, or modify an existing one, another solution is proposed. The idea is to use a ready, flexible and extensible wiki engine, that could be optionally extended with knowledge representation and processing capabilities. So instead of modifying an existing wiki engine or implementing a new one, a development of an extension of the DokuWiki [8] system was chosen.

DokuWiki is designed to be portable, easy to use and set up. Like number of other solutions DokuWiki is based on PHP. However, it does not require any relational database back-end. It allows for image embedding, and file upload and download. Pages can be arranged into namespaces which act as a tree-like hierarchy similar to directory structure. Its modularized architecture allows the user to extend DokuWiki with plugins which provide additional syntax and functionality. A large number of plugins for DokuWiki is available. The templates mechanism provides an easy way to change the presentation layer of the wiki.

To provide a rich knowledge representation and reasoning for the Semantic Web, the SWI-Prolog environment was selected. The basic idea is to build a layered knowledge wiki architecture, where the expressive Prolog representation is used on the lowest knowledge level. This representation is embedded within the wiki text as an optional extension. On top of it number of layers are provided. These include standard meta-data descriptions with RDF and ontologies specification solutions with RDFS and OWL.

5.1 PlWiki Architecture

The PlWiki architecture can be observed in Fig. 3. The stack is based on a simple runtime including the Unix environment with the Unix filesystem, the Apache web server and the PHP stack. Using this runtime the standard DokuWiki installation is run. The PlWiki functionality is implemented with the use of an optional plugin allowing to enrich the wikitext with Prolog clauses, as well run the SWI-Prolog interpreter. It is also possible to extend the wikitext with explicit semantic information encoded with the use of RDF and possibly OWL representation. This layer uses the Semantic Web library provided by SWI-Prolog.

DokuWiki provides a flexible plugin system, providing several kinds of plugins. These include: *Syntax Plugins*, extending the wikitext syntax, *Action Plugins*, redefining selected core wiki operations, (e.g. saving wikipages), and *Renderer Plugins*, allowing to create new export modes (possibly replacing the standard XHTML renderer) (Admin and Action and Helper Plugins are not mentioned here.) The current version of PlWiki implements both the *Syntax* and *Renderer* functionality. Text-based wikipages are fed to a lexical analyzer (Lexer) which identifies the special wiki markup. The standard DokuWiki markup is extended by a special `<pl>...</pl>` markup that contains Prolog clauses. The

[8] See `www.dokuwiki.org`.

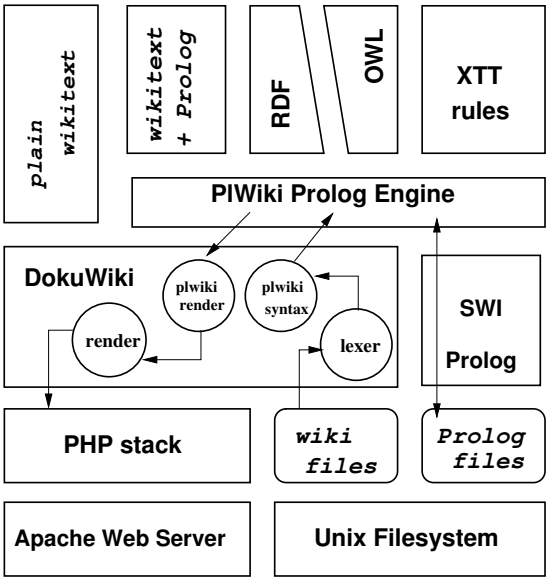

Fig. 3. PlWiki Architecture

stream of tokens is then passed to the Helper that transforms it to special renderer instructions that are parsed by the Parser. The final stage is the Renderer, responsible for creating a client-visible output (e.g. XHTML). In this stage the second part of the plugin is used for running the Prolog interpreter.

The detailed functionality of the PlWiki Syntax Plugin includes parsing the Prolog code embedded in the wikitext, and generating the knowledge base composed of files containing the Prolog code, where each wikipage has a corresponding file in the knowledge base. The PlWiki Renderer plugin is responsible for executing the Prolog interpreter with a given goal, and rendering the results via the standard DokuWiki mechanism.

There are several options how to analyze the wiki knowledge base (that is Prolog files built and extracted from wiki pages). A basic approach is to combine all clauses. More advanced uses allow to select pages (e.g. given namespace) that are to be analyzed.

On top of the basic Prolog syntax, SMW semantic enhancements are also used. They are directly mapped to Prolog clauses. [9]

5.2 PlWiki in Use

As mentioned previously, PlWiki can directly interpret the SMW syntax. Moreover, it allows for embedding any Prolog code, providing more expressive knowledge. To do so, a special markup is used, as shown below:

[9] Currently the PlWiki system is under development. See `http://prolog.ia.agh.edu.pl/kwikis/plwiki` user: iccci, password: PlWiki

```
<pl>
   capital(poland,warsaw).
   country(poland).
   country(germany).
</pl>
```

This simple statement adds two facts to the knowledge base. The plugin invocation is performed using the predefined syntax. To actually specify the goal (query) for the interpreter the following syntax is used:

```
<pl goal="coutry(X),write(X),nl,fail"></pl>
```

It is possible to combine these two, as follows:

```
<pl goal="country(X),write(X),nl,fail">
    country(france).
    country(spain).
</pl>
```

It is possible to specify a given *scope* of the query (in terms of wiki namespaces):

```
<pl goal="country(X),write(X),nl,fail"
    scope="prolog:examples">
</pl>
```

A bidirectional interface, allowing to query the wiki contents from the Prolog code is also available, e.g.:

```
<pl goal="consult('lib/plugins/prolog/plwiki.pl'),
    wikiconsult('plwiki/pluginapi'),list.">
</pl>
```

These features are presented on an extended example in the next section.

5.3 PlWiki Example

Consider an enhanced bookstore system based on PlWiki, including recommendation mechanism, which informs customer about suggested books [46]. There are four page types:

- `genre` in `bookstore:genre` namespace,
- `publisher` in `bookstore:publisher` namespace,
- `author` in `bookstore:author` namespace,
- `book` in `bookstore:book` namespace.

The most important namespace `bookstore:book` contains information about particular books, and for example `bookstore:book:it` page source may look like following (rendered page with additional cover photo is presented in Fig. 4):

```
 ====== Book details: ======
[[category:book]]
**Title**: It [[title:=It]]
**Author**: [[author::bookstore:author:stephen_king]]
**Publisher**: [[publisher::bookstore:publisher:signet]]
**Date**: 1987 [[date:=1987]]
**Language**: English [[language:=english]]
**Genre**: [[genre::bookstore:genre:horror]]
**Pages**: 560
**Keywords**: horror [[keyword:=horror]],
        thriller [[thriller:=horror]],
        bestseller [[keyword:=bestseller]]

====== Recomentadions: ======
**Books by this author**:  {{#ask: [[category:book]]
        [[author::bookstore:author:stephen_king]] }}
**Books by this publisher**: {{#ask: [[category:book]]
        [[publisher::bookstore:publisher:signet]] }}
**Books in this genre**: {{#ask: [[category:book]]
        [[genre::bookstore:genre:horror]] }}
```

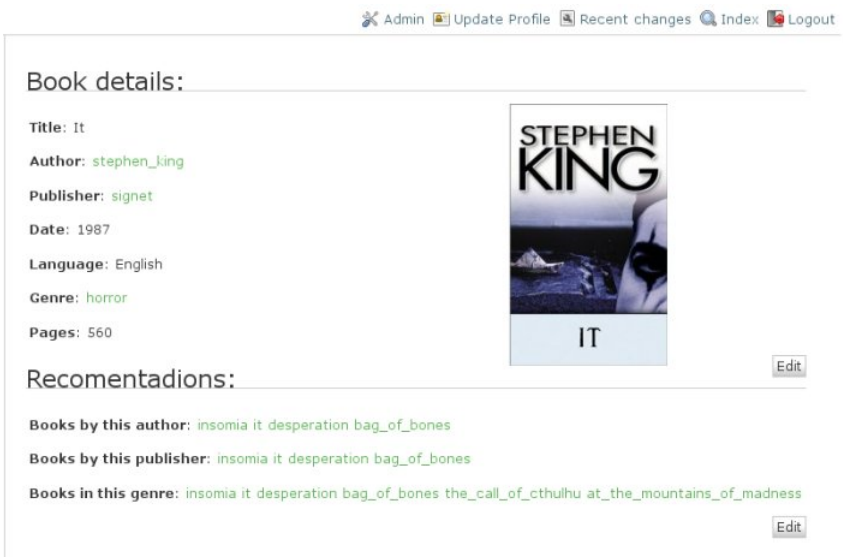

Fig. 4. Bookstore example in PlWiki

Prolog code associated with this page is:

```
wiki_category('bookstore:book:it','book').
wiki_attribute('bookstore:book:it','title','It').
wiki_relation('bookstore:book:it','author',
        ':bookstore:author:stephen_king').
wiki_relation('bookstore:book:it','publisher'
        ,'bookstore:publisher:signet').
wiki_attribute('bookstore:book:it','date','1987').
wiki_attribute('bookstore:book:it','language',
        'english').
wiki_relation('bookstore:book:it','genre',
        'bookstore:genre:horror').
```

This page contains not only basic information about selected book but it also suggests related books. This mechanism may increase profits and it is very flexible. If a new book is added to the system it will be automatically captured by recommendation mechanism. The only important thing is to define author, publisher and genre of this new book.

Because the fact that PlWiki allows to combine SMW markup with Prolog code more complex recommendations could be defined. For example in December Christmas recommendations are adequate:

```
<pl cache="true">
    custom_recommendations(X) :-
        wiki_attribute(X,'keyword','christmas').
</pl>
```

And on the page with book details:

```
<pl goal="custom_recommendations(X),write(X),nl,fail" scope="*"></pl>
```

Custom recommendations can be easily modified, for example:

```
<pl cache="true">
    custom_recommendations(X) :-
        wiki_attribute(X,'keyword','easter').
</pl>
```

The most important is the fact that only one Wiki page, containing definition of `custom_recommendations/1` has to be updated. SMW markup is not powerful enough to support this functionality. The possibility of combining SMW markup with Prolog code is one of the main advantages of PlWiki.

6 Evaluation

The approach presented in this paper offers a unified expressive knowledge representation for semantic annotations, as well as rules and additional procedures. The features of the Loki prototype – PlWiki – according to the introduced feature matrix (see Sect. 2.3) are presented below (see also [31]).

A. Targeted applications	
4) Special-purpose system for knowledge engineering tasks	
B. Underlying knowledge representation	
1) Subject granularity	b) Multiple concepts/properties for one wiki page
2) Knowledge repr. lang.	c) Other/combination (Prolog low-level, OWL higher-level)
3) Add. knowledge sources	b) Prolog rules
C. UI for knowledge capture and sharing	
1) Editing paradigm:	a) Inline text markup, d) Other: Prolog editing support
2) Search/Retrieval	a) Prolog queries, b) Queries embedded into the wiki article, c) Further capabilities: Prolog predicates for knowledge processing
3) Semantic navigation	a) Extended links within wiki article, b) Generation of fact sheets
D. Connectivity	
1) Import facilities	a) OWL (planned) d) Proprietary: SMW knowledge format, Prolog, XTT
2) Export facilities	a) OWL (planned) d) Proprietary: Prolog
E. Extensibility	
1) Extension mechanism	(a) Plug-in mechanism, (b) Code-in mechanism
2) Extensions mods	Custom Prolog extensions, SWI Semantic Layer

PlWiki is an experimental development phase. Current applications include special knowledge engineering tasks, including basic rule-based reasoning tasks in the wiki, and teaching knowledge engineering classes to students. Future applications are planned, including dedicated knowledge intensive closed community portals. System development will focus on flexible user interfaces supporting complex knowledge representation features.

7 Summary and Outlook

In the paper a proposal of a new semantic wiki system has been presented. Based on a review of important semantic wiki systems, important development directions have been identified. An essential area is a strong rule-based reasoning. The proposed system – Loki – offers such a solutions, thanks to a coherent knowledge representation method used. In Loki standard semantic annotations are mapped to Prolog knowledge base, in which also rule-based reasoning is specified. A custom rule-based engine using decision tables and decision trees is also provided.

Future work includes the full integration of the rule engine, as well as GUIs for ontology editing in the wiki. Compatibility with KnowWE wikimarkup is also planned. A prototype pure-Prolog wikiengine is also considered. As of knowledge

processing it is more powerful than the plugin-based PlWiki solution. On top of wiki a knowledge evaluation layer is also considered.

Acknowledgements The author wish to thank Dr. Joachim Baumeister for his valuable comments on semantic wikis, and the proposal of the name Loki, Dr. Jason Jung for his remarks concerning the PlWiki description, as well as Michał Kotra MSc. for his work on the preliminary PlWiki implementation [46] and Krzysztof Kaczor MSc. for his support with the PlWiki development.

References

1. Nalepa, G.J.: Plwiki - a generic semantic wiki architecture. In Nguyen, N.T., Kowalczyk, R., Chen, S.M., eds.: Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, First International Conference, ICCCI 2009, Wroclaw, Poland, October 5-7, 2009. Proceedings. Volume 5796 of Lecture Notes in Computer Science., Springer (2009) 345–356
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web: Scientific american. Scientific American (May 2001)
3. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
4. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Eds.: Extensible markup language (XML) 1.0 (second edition). Technical report, World Wide Web Consortium (2000) `http://www.w3.org/TR/REC-xml`, W3C Recommendation.
5. Lassila, O., Swick, R.R.: Resource description framework (RDF) model and syntax specification. Technical report, World Wide Web Consortium (1999) `http://www.w3.org/TR/REC-rdf-syntax`, W3C Recommendation.
6. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF schema. W3C recommendation, W3C (February 2004) http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
7. Patel-Schneider, P.F., Horrocks, I.: OWL 1.1 Web Ontology Language Overview. W3C member submission, W3C (December 2006) http://www.w3.org/Submission/owl11-overview/.
8. Seaborne, A., Prud'hommeaux, E.: SPARQL query language for RDF. W3C recommendation, W3C (January 2008) http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.
9. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 web ontology language — primer. W3C recommendation, W3C (October 2009)
10. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 web ontology language:profiles. W3C recommendation, W3C (October 2009)
11. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML, w3c member submission 21 may 2004. Technical report, W3C (2004)
12. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of the Twelfth International World Wide Web Conference, WWW2003. (2003) 48–57
13. Horrocks, I., Parsia, B., Patel-Schneider, P., Hendler, J.: Semantic web architecture: Stack or two towers? In Fages, F., Soliman, S., eds.: Principles and Practice of Semantic Web Reasoning. Number 3703 in LNCS, Springer (2005) 37–41

14. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
15. van Harmelen, F., Lifschitz, V., Porter, B., eds.: Handbook of Knowledge Representation. Elsevier Science (2007)
16. Brachman, R., Levesque, H.: Knowledge Representation and Reasoning. 1st edn. Morgan Kaufmann (2004)
17. Oren, E., Delbru, R., Möller, K., Völkel, M., Handschuh, S.: Annotation and navigation in semantic wikis. In: SemWiki. (2006)
18. Schaffert, S.: Ikewiki: A semantic wiki for collaborative knowledge management. In: WETICE '06: Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Washington, DC, USA, IEEE Computer Society (2006) 388–396
19. Auer, S., Dietzold, S., Riechert, T.: Ontowiki - a tool for social, semantic collaboration. In Cruz, I.F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L., eds.: International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science., Springer (2006) 736–749
20. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Web Semantics **5** (2007) 251–261
21. Oren, E.: Semperwiki: a semantic personal wiki. In: Proc. of 1st Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure, Galway, Ireland. (NOV 2005)
22. Buffa, M., Gandon, F., Ereteo, G., Sander, P., Faron, C.: SweetWiki: A semantic wiki. Web Semantics: Science, Services and Agents on the World Wide Web (2008) in press.
23. Kuhn, T.: AceWiki: A Natural and Expressive Semantic Wiki. In: Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges, CEUR Workshop Proceedings (2008)
24. Schaffert, S., Eder, J., Grünwald, S., Kurz, T., Radulescu, M.: Kiwi - a platform for semantic social software (demonstration). In: ESWC. (2009) 888–892
25. Baumeister, J.; Puppe, F.: Web-based knowledge engineering using knowledge wikis. In: Proc. of the AAAI 2008 Spring Symposium on "Symbiotic Relationships between Semantic Web and Knowledge Engineering", Stanford University, USA (2008) 1–13
26. Baumeister, J., Reutelshoefer, J., Puppe, F.: Knowwe: A semantic wiki for knowledge engineering. Applied Intelligence (2010) to appear
27. Jung, J.J., Nguyen, N.T.: Collective intelligence for semantic and knowledge grid. J. UCS **14**(7) (2008) 1016–1019
28. Jung, J.J., Nguyen, N.T.: Consensus choice for reconciling social collaborations on semantic wikis. In Nguyen, N.T., Kowalczyk, R., Chen, S.M., eds.: ICCCI. Volume 5796 of Lecture Notes in Computer Science., Springer (2009) 472–480
29. Jung, J.J.: Knowledge distribution via shared context between blog-based knowledge management systems: A case study of collaborative tagging. Expert Syst. Appl. **36**(7) (2009) 10627–10633
30. Razmerita, L., Kirchner, K., F, S.: Personal knowledge management. the role of web 2.0 tools for managing knowledge at individual and organisational levels. ONLINE INFORMATION REVIEW **33**(6) (2009) 1021–1039
31. Baumeister, J., Nalepa, G.J.: Engineering expressive knowledge with semantic wikis. In Ligęza, A., Nalepa, G.J., eds.: International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2009): November 28, 2009, Kraków, Poland, Kraków, Poland (2009) 13–24

32. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006). (2006) 68–78
33. Horrocks, I.: OWL Rules, OK? In: W3C Workshop on Rule Languages for Interoperability. (April 27-28 2005)
34. Bratko, I.: Prolog Programming for Artificial Intelligence. 3rd edn. Addison Wesley (2000)
35. Covington, M.A., Nute, D., Vellino, A.: Prolog programming in depth. Prentice-Hall (1996)
36. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. 2nd edn. Prentice-Hall (2003)
37. Noga, M., Kaczor, K., Nalepa, G.J.: Lightweight reasoning methods in selected semantic wikis. Gdansk University of Technology Faculty of ETI Annals **18**(8) (2010) 103–108
38. Vassiliadis, V., Wielemaker, J., Mungall, C.: Processing owl2 ontologies using thea: An application of logic programming. In: OWLED. (2009)
39. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto controlled english - not just another logic specification language. In Flener, P., ed.: Logic-Based Program Synthesis and Transformation. Number 1559 in Lecture Notes in Computer Science, Manchester, UK, Eighth International Workshop LOPSTR'98, Springer (June 1999)
40. Nalepa, G.J.: Architecture of the heart hybrid rule engine. In [et al.], L.R., ed.: Artificial Intelligence and Soft Computing : 10th International Conference, ICAISC 2010 : Zakopane, Poland, June 13–17, 2010, Pt. II. Volume 6114 of LNAI., Springer (2010) 598–605
41. Nalepa, G.J., Ligęza, A.: A graphical tabular model for rule-based logic programming and verification. Systems Science **31**(2) (2005) 89–95
42. Nalepa, G.J., Ligęza, A.: XTT+ rule design using the ALSV(FD). In Giurca, A., Analyti, A., Wagner, G., eds.: ECAI 2008: 18th European Conference on Artificial Intelligence: 2nd East European Workshop on Rule-based applications, RuleApps2008: Patras, 22 July 2008, Patras, University of Patras (2008) 11–15
43. Nalepa, G.J., Ligęza, A.: HeKatE methodology, hybrid engineering of intelligent systems. International Journal of Applied Mathematics and Computer Science **20**(1) (March 2010) 35–53
44. Nalepa, G.J., Furmańska, W.T.: Proposal of a New Rule-based Inference Scheme for the Semantic Web Applications. Studies in Computational Intelligence. In: New Challenges in Computational Collective Intelligence. Springer Berlin / Heidelberg (2009) 15–26
45. Nalepa, G.J.: Collective knowledge engineering with semantic wikis. Journal of Universal Computer Science **16**(7) (2010) 1006–1023 `http://www.jucs.org/jucs_16_7/collective_knowledge_engineering_with`.
46. Kotra, M.: Design of a prototype knowledge wiki system based on prolog. Master's thesis, AGH University of Science and Technology in Kraków (2009)