Engineering Expressive Knowledge with Semantic Wikis. ⋆

Joachim Baumeister[1], Grzegorz J. Nalepa[2]

[1] Institute for Computer Science
University of Würzburg, Am Hubland, 97074 Würzburg, Germany
joba@uni-wuerzburg.de
[2] Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
gjn@agh.edu.pl

Abstract. Semantic wikis are successfully used in various application domains. Such systems combine the flexible and agile authoring process with strong semantics of ontologies. The current state–of–the–art of systems, however, is diverse in the sense of having a common ground. Especially, the expressiveness of the knowledge representation of semantic wikis undergoes continuous improvement. In the paper, two semantic wiki implementations are discussed, that are both extending semantic wiki implementations by strong problem-solving knowledge. We compare their approaches and we aim to condense the fundamental characteristics of a strong problem-solving wiki.

1 Introduction

Recently, the most important development of the Internet concerned not the lower network network layers, but the higher application or service layers related to the Web technology. This is mainly due to the fact, that while the speed and storage capacities of the Web increased by orders of magnitude, its search and processing capabilities remained almost unchanged on the conceptual level.

This phenomena led, almost a decade ago, to the proposal of the *Semantic Web*. In this architecture a number of higher level semantic facilities built on top of the Web would allow not just to *search data* but to *reason with knowledge.* In fact, this was the point where the focus of the Web development moved from *content* (data) to *knowledge* (in a broad sense). A decade later, a number of semantic technologies is available and widely used, starting from the data structuring XML, to meta-data annotations with RDF and ontologies with RDFS and OWL. While these technologies provided knowledge encoding and representation solutions, the challenge remains to provide an efficient knowledge processing and reasoning with rules on the Web. This is in fact the point, where most of the current Semantic Web research focuses. Recent rule standards from W3C include RIF and SWRL.

⋆ The paper is partially carried out within the AGH UST Project No. 10.10.120.105

Besides knowledge representation and reasoning, a sensible knowledge engineering solution for the Web is another important challenge. While the Semantic Web initiative targets mainly representation aspects, it does not directly address the specific problems stemming from the massively parallel and collaborative nature of the Web. Social networks, that provide specific services on top of the Web and the Semantic Web, try to cope with these problems. Recently the wiki technology has gained importance with respect to the collaborative knowledge acquisition and engineering. The development of *semantic wikis,* such as IkeWiki [1], Semantic MediaWiki [2], and SweetWiki [3], allowed to use the Semantic Web methods and tools on top of the existing content-centered wiki solutions.

Existing semantic wikis allow for an introduction of semantic information (e.g. meta-data, ontologies) into a wiki. In fact, they often allow to build a wiki around an ontology, which improves their conceptual coherence. Most of the semantic wikis reached a stage where the reasoning capabilities have to be added. This is where some limitations of existing solutions become exposed.

In this paper, we introduce a categorization of semantic wiki functions in order to simplify the comparison of system approaches and implementations. The categorization will concentrate on the features required during knowledge engineering activities and will omit other important aspects such as data storage and scalability of the implementations. We compare the two semantic wikis *PlWiki* and *KnowWE* according to the introduced scheme.

2 Categorization of Expressive Semantic Wikis

The semantic wiki community tracks the recent developments of semantic wiki features within a matrix[3]. For example, the matrix lists features such as *editing paradigm*, *annotation mechanism*, *programming language* and *license.* For systems —extended by expressive knowledge formalizations— we see that this categorization is too broad to capture the their special capabilities. For this reason, we propose a supplementary set of feature categories in the following, which we use to describe the wiki implementations in the latter of the paper.

We see that semantic wikis are not used in a unified application context, but are often designed to match to a more specific application area. Thus, the particular systems cannot be compared in a linear order, but show advantages and disadvantages for specific applications. A new matrix can help to simplify the selection of a system for a given work task.

A. Targeted applications

Often, the systems are designed with a specific application context in mind. Possible values are:

1. Community-based ontology engineering
2. Encyclopedia-like application, e.g., Wikipedia enhancement

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

3. E-learning
4. Special-purpose system for knowledge engineering tasks (closed communities)
5. Other (please describe your context)

B. Underlying knowledge representation

The semantic wiki represents the developed knowledge in a particular format. This sub-area specifies the type of structure, how the atomic concepts of the knowledge base are represented, it names the used knowledge representation language, and discusses if/how additional sources of knowledge are connected.

1. **Subject granularity:** What is the level of detail of the atomic concepts represented in the knowledge base. This also corresponds to the main structuring paradigm of the system; i.e., most wikis use the single wiki pages as the underlying structuring paradigm, where the particular concepts are represented by single wiki pages. Possible values are:
   (a) One concept/property for each wiki page, (b) Multiple concepts/properties for one wiki page, (c) Other—please specify
2. **Knowledge representation language:** This sub-area names the expressivity of the underlying knowledge representation. Possible values are:
   (a) RDF(S), (b) OWL, (c) Other/combination—please specify
3. **Additional knowledge sources:** In addition to standard ontological knowledge, often a special type of additional knowledge is attached, for example, rules and domain models. Possible values are:
   (a) SWRL rules, (b) Prolog rules, (c) Model-based knowledge, (d) Text in controlled language, (e) Other—please specify

C. UI for knowledge capture and sharing

In practical applications the user interface for the development and use of the represented knowledge is of prime importance. Here, most of the current semantic wiki implementations strongly differ and show emphasis on their chosen application context.

1. **Knowledge editing paradigm:** How is the knowledge acquired and maintained within the wiki system? Possible values ares:
   (a) Inline text markup: The knowledge entered and maintained in combination with the usual textual edit pane of the wiki. The wiki offers special purpose markup to define the knowledge entities.
   (b) Semantic forms/visual editors: Ontological concepts and more expressive knowledge is defined using customized forms and graphical editors.
   (c) Multiple (combination of above approaches)
   (d) Other—please specify
2. **Semantic search/knowledge retrieval:** One prominent application of knowledge use is the integration of semantic search facilities. Of interest is the applied approach of integrating the search into the wiki as well as the language to formulate the queries.

   (a) Used query language: SPARQL, Datalog-like queries, other—please specify.
   (b) Query integration: Special-purpose forms, queries embedded into the wiki article, interview questionnaires, other—please specify.
   (c) Further capabilities of knowledge use: Generated knowledge-based interviews, Prolog queries, other—please specify.
3. **Semantic navigation:** The semantic annotations of the wiki articles allows for an improved navigation through the contents of the system. Some systems provide a simple extension of semantic navigation by extending the standard link structure, whereas other implementations (additionally) facilitate semantic search by generated fact sheets interlinking connected articles.
   (a) Extended links within wiki article
   (b) Generation of fact sheets
   (c) Other—please specify

D. Connectivity

This sub-area specifies the capabilities of the wiki with respect to the import and export of knowledge from and to external sources.

1. **Import facilities:** Is it possible to import external knowledge sources, and if yes then what type of sources are supported? Possible values are:
   (a) RFS(S)/OWL, (b) SWRL, (c) RIF, (d) Proprietary knowledge sources—please specify, (e) none
2. **Export facilities:** Is it possible to export the knowledge base of the wiki to an external storage?
   (a) RDF(S)/OWL, (b) SWRL, (c) RIF, (d) Proprietary knowledge sources—please specify, (e) none

E. Extensibility

Due to the fast development of new technologies and application areas, the extensibility of existing wiki systems by new features is very important. This sub-area captures the type of extensibility and lists a number of already existing extensions/modifications of the wiki.

1. Extension mechanism:
   (a) Plug-in mechanism, (b) Code-in mechanism, (c) none, (d) other—please specify
2. Existing extensions/modifications: Please list running extensions of the wiki system.

With the proposed categorization we try to capture the conceptual properties of semantic wikis. Please note that we tried to not include technical details of the implementation of the wiki systems, such as data storage (databases vs. text) and the programming language.

3 The Semantic Wiki KnowWE

We first summarize the features of KnowWE according to the introduced feature matrix and then we introduce the system in more detail.

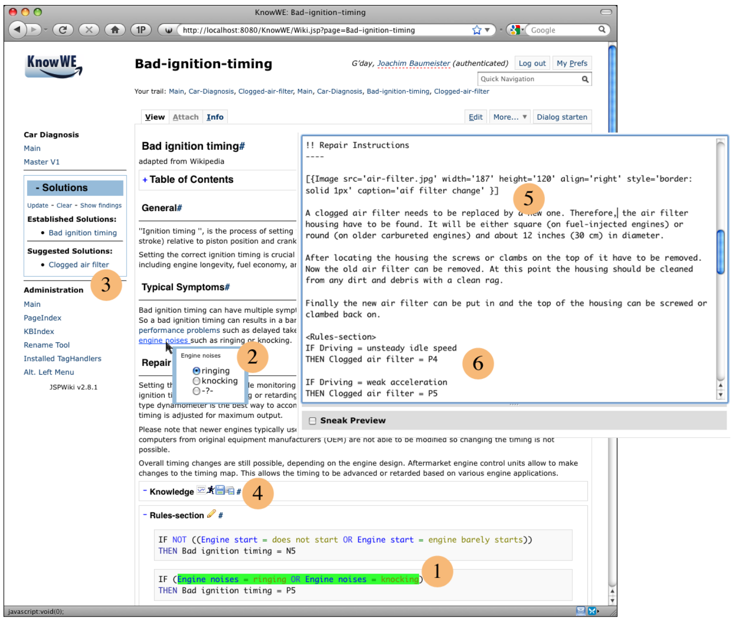
3.1 KnowWE in the Feature Matrix

We summarize the KnowWE features according to the introduced feature matrix:

A. Targeted applications	
4) Special-purpose system for knowledge engineering tasks	
B. Underlying knowledge representation	
1) Subject granularity	a) One concept/property for each wiki page
2) Knowledge repr. lang.	c) Combination (OWL and Special-purpose problem-solving knowledge)
3) Add. knowledge sources	e) Other: heuristic rules, decision trees, fault models
C. UI for knowledge capture and sharing	
1) Editing paradigm:	a) Inline text markup, b) visual editors (for d3web extension)
2) Search/Retrieval	a) SPARQL, b) Queries embedded into the wiki article, interview questionnaires, c) Further capabilities: Generated knowledge-based interviews
3) Semantic navigation	a) Extended links within wiki article, b) Generation of fact sheets, c) Other: Knowledge-based interview
D. Connectivity	
1) Import facilities	a) OWL, d) Proprietary: text (KnOffice)
2) Export facilities	a) OWL, d) Proprietary: text (KnOffice), xml (d3web)
E. Extensibility	
1) Extension mechanism	(a) Plug-in mechanism
2) Extensions mods	d3web plugin, semantic tagging plugin Hermes plugin

3.2 KnowWE in a Nutshell

In this section, we introduce the general features of KnowWE by using a simple example application for diagnosing car faults. The basic idea is, that possible causes for a car fault —the solutions of the problem— are represented by corresponding wiki articles. The wiki contains, for instance, articles about *empty battery*, *clogged air filter*, and *bad ignition timing.*



"Ignition timing ", is the process of setting
stroke) relative to piston position and crank

Setting the correct ignition timing is crucial
including engine longevity, fuel economy, ar

Bad ignition timing can have multiple sympt
So a bad ignition timing can results in a bar
performance problems such as delayed take
engine noises such as ringing or knocking.

Setting th le monitoring
ignition ti g or retarding
type dynamometer is the best way to accon
timing is adjusted for maximum output.

Please note that newer engines typically us
computers from original equipment manufacturers (OEM) are not able to be modified so changing the timing is not
possible.

Overall timing changes are still possible, depending on the engine design. Aftermarket engine control units allow to make
changes to the timing map. This allows the timing to be advanced or retarded based on various engine applications.

```
IF NOT ((Engine start = does not start OR Engine start = engine barely starts))
THEN Bad ignition timing = N5

IF (Engine noises = ringing OR Engine noises = knocking)
THEN Bad ignition timing = P5
```

```
!! Repair Instructions
----

[{Image src='air-filter.jpg' width='187' height='120' align='right' style='border:
solid 1px' caption='aif filter change' }]

A clogged air filter needs to be replaced by a new one. Therefore, the air filter
housing have to be found. It will be either square (on fuel-injected engines) or
round (on older carbureted engines) and about 12 inches (30 cm) in diameter.

After locating the housing the screws or clambs on the top of it have to be removed.
Now the old air filter can be removed. At this point the housing should be cleaned
from any dirt and debris with a clean rag.

Finally the new air filter can be put in and the top of the housing can be screwed or
clambed back on.

<Rules-section>
IF Driving = unsteady idle speed
THEN Clogged air filter = P4

IF Driving = weak acceleration
THEN Clogged air filter = P5
```

Fig. 1. A wiki article describing the solution *bad ignition timing* in the context of a car diagnosis application (1–4), and the edit panel of another wiki page describing the solution *Clogged air filter* (5–6).

Inline Editing of Text and Knowledge In Figure 1, a page of the wiki is shown, describing the solution *bad ignition timing*. Besides standard text describing the problem in more detail, also explicit problem-solving knowledge is included on the page. In Figure 1-(1) two heuristic rules of the rule base are displayed, that describe some derivation knowledge of the solution. The first rule states, that the solution *Bad ignition timing* will receive a negative score, if the user enters for the symptom *engine start* that it neither *does not start* nor *barely starts*. The second rule states a positive score, if the *engine noise* was observed by the user as *ringing* or *knocking*. Besides the representation of rules, we also allow for the inclusion of model-based knowledge and decision trees, see [4] for more details on the implemented knowledge connectors.

We see, that the derivation knowledge for a solution is locally defined and maintained together with the corresponding article of the solution; see the right top of Figure 1 for an example edit panel of the wiki, where a rule base (Figure 1-6) is edited inline. This allows for a simplified update of informal (e.g., text) *and* explicit knowledge (e.g., rules) about one entity.

Although, the wiki is mainly used as a tool for knowledge engineering, it also provides interfaces for interactive problem-solving. We give an example of the problem-solving process in the following: Some parts of the text are related to concepts of the knowledge base, and thus have a meaning for the problem-solving process. Specific semantic annotations relate these text parts with the concepts. In the view mode of the wiki the user is able to click on the annotated text and can enter findings based on the corresponding concept. We call this approach *inline answers* for problem-solving in wikis. The text phrase "engine noises" within the text was annotated by the corresponding concept *Engine noises* available in the knowledge base, see Figure 1-(2). In the given example, the value *knocking* for the concept *Engine noises* was entered by the user. In the solutions pane of the wiki — Figure 1-(3) — we see that the solution *Bad ignition timing* is derived with a high score, whereas the alternative solution *Clogged air filter* was also derived and is considered as a possible solution. Both solutions were derived on the basis of this finding and previously entered findings. By clicking on the solution *Clogged air filter* in the solutions pane, we quickly can navigate to the wiki article describing the corresponding article.

Alternatively, the user is able to download an executable version of the knowledge base by clicking the download button, see Figure 1-(4). This way, the knowledge bases can be developed using the wiki and later are exported to an external application when required.

Semantic Annotations In KnowWE, semantic annotations are defined inline with the wiki text. The markup for those annotations was inspired by the syntax of Semantic MediaWiki [2], and ontological concepts can be simply linked by the definition of ontological properties. The general syntax of the markup connects a text phrase of the wiki text with a concept using an ontological property.

```
[Bad ignition timing is a technical problem
 <=> subClassOf:: TechnicalProblem] that can be solved ...
```

In the example shown above, the text phrase "Bad ignition ... problem" is annotated, stating, that the concept represented by this article is a `subClassOf` the concept `TechnicalProblem`. The annotation itself states, that the annotated text phrase documents/justifies the given relation.

By this type of annotations many useful ontological relations can be defined inline the wiki text. All annotations are represented in the application ontology and can be queried using a SPARQL. Queries are embedded into the wiki text, and the results of the queries are shown in the view mode of the article.

3.3 Applications of KnowWE

KnowWE is typically used together with the d3web plugin for building knowledge-based applications. The system is currently used in a number of (industrial and academic) projects, ranging from simple recommender systems to complex decision-support systems for technical and medical devices. For example, KnowWE provides a technical platform to support a scientific community in the biological domain in the context of the *BIOLOG Wissen*[4] project. BIOLOG Wissen serves as a web-based application for the collaborative construction and use of a decision-support system for landscape diversity. It aims to integrate knowledge on causal dependencies of stakeholders, relevant statistical data, and multimedia content. We refer the interested reader to [5,6] for more details. In another recent project, KnowWE is extended by diagnostic workflow knowledge in the context of the CliWE project[5]. By this extension, the wiki will be used to collaboratively develop clinical guidelines, that are integrated as compiled knowledge bases into next-generation medical devices. A first prototype of this extension is reported in Hatko et al. [7].

4 The Semantic Wiki PlWiki

The PlWiki [8] features are briefly introduced first. Then, the basic architectural assumptions are given, and specific knowledge representation aspects presented.

4.1 PlWiki in the Feature Matrix

The features of PlWiki according to the introduced feature matrix are:

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

[4] BIOLOG is funded by the German Federal Ministry of Education and Research from 2007-2009 (final funding phase).

[5] CliWE (Clinical Wiki Environment) is funded by Drägerwerk, Germany and runs from 2009-2012.

4.2 PlWiki in a Nutshell

System Architecture The main objective of the PlWiki design is to deliver a generic and flexible knowledge engineering solution [8]. Instead of modifying an existing wiki engine or implementing a new one, a development of an extension of the DokuWiki system was chosen. 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. 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 semantical information encoded with the use of RDF and possibly OWL representation. This layer uses the Semantic Web library provided by SWI-Prolog. An optional decision rule layer is also considered with the use of the HeaRT runtime for the XTT^2 framework [9].

DokuWiki provides a flexible plugin system, providing five kinds of plugins (see `www.dokuwiki.org/devel:plugins`). The current version of PlWiki implements both the *Syntax* and *Renderer* plugin 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 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.

Knowledge Representation Features Below basic use examples of the generic Prolog representation are given.

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

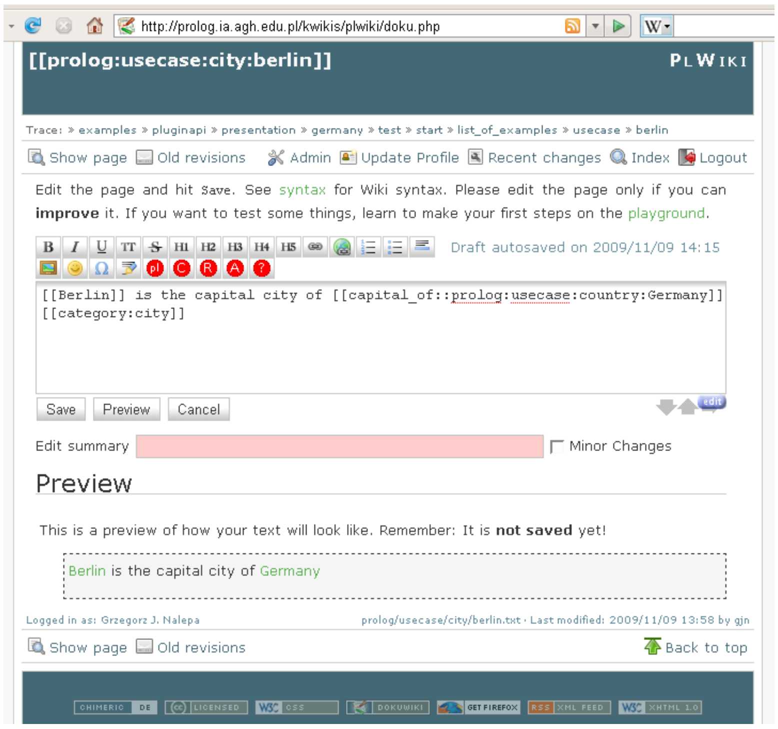
It is possible to specify a given *scope* of the query (in terms of namespaces):

```
<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>
```

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, semantic enhancements are possible. These can be easily mapped to Prolog clauses. An example of editing session with PlWiki can be observed in Fig. 2.



http://prolog.ia.agh.edu.pl/kwikis/plwiki/doku.php

[[prolog:usecase:city:berlin]] PlWiki

Trace: » examples » pluginapi » presentation » germany » test » start » list_of_examples » usecase » berlin

Show page Old revisions Admin Update Profile Recent changes Index Logout

Edit the page and hit Save. See syntax for Wiki syntax. Please edit the page only if you can **improve** it. If you want to test some things, learn to make your first steps on the playground.

Draft autosaved on 2009/11/09 14:15

```
[[Berlin]] is the capital city of [[capital_of::prolog:usecase:country:Germany]]
[[category:city]]
```

Save Preview Cancel

Edit summary ☐ Minor Changes

Preview

This is a preview of how your text will look like. Remember: It is **not saved** yet!

Berlin is the capital city of Germany

Logged in as: Grzegorz J. Nalepa prolog/usecase/city/berlin.txt · Last modified: 2009/11/09 13:58 by gjn

Show page Old revisions Back to top

Fig. 2. PlWiki editing session

Semantic Representation Layer Besides the generic Prolog-based knowledge representation features based on pure Prolog clauses, typical semantic wiki features are supported. Semantic Media Wiki (SMW) [10], a standard semantic wiki solution, provides a simple yet flexible mechanism for annotating categories, and properties. In the first version of PlWiki three main features are considered: categories definitions as in SMW, simple queries from SMW (with SPARQL queries in the future), and generic RDF annotations.

To provide a better compatibility with existing solutions parsing of SMW wikitext is provided, with a corresponding Prolog representation available. The wiki user can use the SMW syntax directly in DokuWiki to enter wikitext. The PlWiki plugins transforms the wikitext to Prolog clauses, asserted to the internal knowledge base. In fact these clauses could also be introduced by using the PlWiki `<pl></pl>` tags. Examples are shown below, with the SMW syntax given first, and the corresponding Prolog representation after it.

```
Berlin is the capital city of [[capital_of::Germany]] [[category:city]]
  wiki_category('City','Berlin').
  wiki_property(capital_of,subject_page_name,'Germany').
Germany is a country in central Europe.
[[category:country]] [[location:=Central Europe]]
  wiki_category('Country,'Germany').
  wiki_attribut(page_uri,location,'Central_Europe').
```

The Prolog clauses are asserted to the PlWiki knowledge base by the syntax plugin analyzing the wiki text. In a similar fashion simple queries are handled. A query for a category or property is simply mapped to a Prolog goal:

```
{{#ask: [[category:city]] [[capital of::Germany}}
  wiki_category('Cities',Page),
  wiki_property(capital_of,Page,'Germany'), wiki_out(Page).
```

Plain RDF annotations are also supported. Currently, these are separated from the explicit annotations mentioned above. For compatibility reasons an RDF annotation can be embedded directly in XML serialization, then it is parsed by the corresponding Prolog library, and turned to the internal representation, that can also be used directly. SWI-Prolog's represents RDF triples simply as: `rdf(?Subject, ?Predicate, ?Object)`. So mapping the above example would result in: `rdf('Berlin',capital_of,'Germany')`.

Using wiki knowledge it is possible to define rules, e.g.: "Nordic country is a country with `location` set to `Northern Europe`" is in Prolog:

```
<pl cache="true"> nordic_country(X) :-
                    wiki_category(X,'country'),
                    wiki_attribute(X,'location','Northern Europe'). </pl>
```

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

4.3 Applications of PlWiki

PlWiki is in 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. Future applications are planned, including dedicated knowledge intensive closed community portals. System development will focus on flexible user interfaces supporting complex knowledge representation features.

5 Conclusions

Semantic wikis are successful examples of semantic applications and are widely used in academia and industry. From the technological viewpoint, however, we currently see no common baseline for the expressiveness and functionality of semantic wikis. In contrast, the available and active semantic wiki implementations strongly differ in their characteristics. In this paper, we introduced a catalog of functions with which systems can be compared and evaluated. We emphasized the functionality of wikis with respect to knowledge engineering activities, thus excluding certainly important aspects such as editing paradigm, data storage, and scalability. The semantic wiki implementations KnowWE and PlWiki were presented according to the introduced catalog.

This work is an initial attempt in this research direction. In the future, we are planning to undertake an extensive study not only incorporating two systems, but comparing all available and active semantic wiki implementations within a feature matrix. The studies would we also aimed at providing benchmark case studies implemented using different wiki implementations which would allow for a synthetic comparison and feature analysis.

References

1. Schaffert, S., Gruber, A., Westenthaler, R.: A semantic wiki for collaborative knowledge formation. In: Proceedings of SEMANTICS 2005 Conference, Trauner Verlag (2006)
2. Krötzsch, M., Vrandecić, D., Völkel, M.: Semantic MediaWiki. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, LNAI 4273, Berlin, Springer (2006) 935–942
3. Buffa, M., Gandon, F., Ereteo, G., Sander, P., Faron, C.: SweetWiki: A semantic wiki. Web Semantics **8**(1) (2008) 84–97
4. Baumeister, J., Reutelshoefer, J., Puppe, F.: Markups for knowledge wikis. In: SAAKM'07: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop, Whistler, Canada (2007) 7–14
5. Nadrowski, K., Baumeister, J., Wolters, V.: LaDy: Knowledge Wiki zur kollaborativen und wissensbasierten Entscheidungshilfe zu Umweltveränderung und Biodiversität. Naturschutz und Biologische Vielfalt **60** (2008) 171–176
6. Baumeister, J., Reutelshoefer, J., Haupt, F., Nadrowski, K.: Capture and refactoring in knowledge wikis – coping with the knowledge soup. In: SCOOP'08: Proceedings of 2nd Workshop on Scientific Communities of Practice, Bremen, Germany (2008)
7. Hatko, R., Baumeister, J., Puppe, F.: Diaflux: Diagnostic flows in wikis. In: FGWM'09: Proceedings of German Workshop of Knowledge and Experience Management (at LWA'09). (2009)
8. Nalepa, G.J.: Plwiki – a generic semantic wiki architecture. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, S.M.C., ed.: 1st International Conference on Computational Collective Intelligence - Semantic Web, Social Networks & Multiagent Systems. Volume LNAI 5796., Springer (2009) 345–356
9. Nalepa, G.J., Ligęza, A.: Hekate methodology, hybrid engineering of intelligent systems. International Journal of Applied Mathematics and Computer Science (2009) accepted for publication.
10. Krötzsch, M., Vrandecić, D., Völkel, M., Haller, H., Studer, R.: Semantic Wikipedia. Web Semantics **5**(4) (2007) 251–261