Ad-Hoc Knowledge Engineering with Semantic Knowledge Wikis

Jochen Reutelshoefer, Joachim Baumeister and Frank Puppe

Institute for Computer Science, University of Würzburg, Germany
email: {reutelshoefer,baumeister,puppe}@informatik.uni-wuerzburg.de

Abstract. A couple of semantic wikis have been proposed to serve as collaborative knowledge engineering environments – however, the knowledge of almost all systems is currently based on the expressiveness of OWL (Lite/DL). In this paper we present the concept of a *semantic knowledge wiki* that extends the capabilities of semantic wikis by strong problem-solving methods. We show how problem-solving knowledge is connected with standard ontological knowledge using an upper ontology. Furthermore, we discuss different possibilities to formalize problem-solving knowledge, for example by semantic knowledge annotation, structured text and explicit markups.

1 Introduction

Recently, different approaches of semantic wikis have been presented as applications especially designed for the distributed engineering of ontological knowledge, for example see IkeWiki [1] and OntoWiki [2]. Such systems are used to build ontologies of a specific domain in a collaborative manner and use the well-known metaphor of wikis as the primary user interface. New concepts are usually defined by creating a new wiki page with the name of the concept. Properties of the new concept are described by semantically annotating text phrases of the particular wiki page. The semantic extension of wikis allows for a richer set of possible applications when compared to standard wikis: due to the semantic annotation of content the user is able to semantically search for ontological concepts and/or related concepts. Furthermore, a semantic wiki can be browsed in a semantic way; for example, users can click on semantically relevant (and probably personalized) links that are placed appropriately.

Based on the expressiveness of OWL (Lite/DL) the defined ontologies are able to capture a wide range of general knowledge but lack in the possibility to represent active problem-solving knowledge that is necessary to generate and drive a (semi-)automated problem-solving session with a user. Such knowledge typically relates the class of *findings* – provided as user inputs and describing the current problem – with the class of *solutions* that are derived for the given problem description. In previous work we presented the concept of *knowledge wikis* as an extension of standard wikis adding the possibility to capture, maintain, and share explicit problem-solving knowledge [3, 4]. The presented concept

provided strong support to represent explicit knowledge like rules and models, but was not able to capture ontological knowledge beyond subclass hierarchies of solutions and part-of hierarchies of input groups/inputs.

In this paper we describe the (refined) concept of *semantic knowledge wikis* that are interpreted as an extension of semantic wikis. Besides basic ontological knowledge – such as the definition of classes, taxonomic and user-defined properties – a semantic knowledge wiki is able to represent problem-solving knowledge that is applied on selected classes of the ontology. The problem-solving knowledge can be seen as an external knowledge source for changing the values of concept instances; for example in most of the cases the state of a solution instance is set to the value "established". Beyond that it is not intended that the problem-solving knowledge interacts with other knowledge defined in the ontology. In future work, we will consider the exchanging semantics of problem-solving knowledge with the knowledge defined in the ontology, as for example it is currently done in the context of the RIF working group[1]. In summary, a semantic knowledge wiki represents a distributed knowledge engineering environment not only representing semantic relations between the concepts of an ontology but also explicit derivation knowledge.

In contrast to trained knowledge engineers – well educated in knowledge representation and reasoning – we adress experienced users to act as "ad-hoc knowledge engineers". For this reason, the provided interfaces and markups of the wiki need to be as simple as possible in order to lower the barriers of knowledge acquisition. Furthermore, in running projects we experienced the requirement to handle a mixed granularity of the represented knowledge. An interesting research question is to find a collection of suitable markups that can cope with the different types of knowledge and its granularity, respectively.

In this paper, we first describe the basic concepts of a semantic knowledge wiki by introducing an upper ontology for problem-solving. This ontology represents the basic classes and properties that are used in a typical problem-solving process. Moreover, this ontology is used as the basis in every new wiki project, since newly defined concepts are implicitly or explicitly aligned to classes of the upper ontology. We also introduce different types of markups for the definition of problem-solving knowledge. The markups take into account that knowledge can be "formalized" in many different ways, ranging from explicit models and rule bases to semantically annotated text or structured text phrases.

2 An Overview of KnowWE

As discussed above every (semantic) wiki page describes a distinct concept together with formalized properties linking the entity with other classes. In a knowledge wiki we also capture the problem-solving knowledge that is necessary for deriving the particular concept. Then, every wiki page embeds not only semantically annotated text and multimedia but also explicit problem-solving knowledge.

[1] RIF WG wiki: http://www.w3.org/2005/rules/wiki/RIF_Working_Group

As a typical use case, the user of a semantic knowledge wiki can browse the contents of the wiki in a classic web-style –possibly using semantic navigation and/or semantic search features. Moreover, he/she is also able to start an interactive interview where giving a problem description. Based on these inputs an appropriate list of solutions is presented that are in turn linked to the wiki pages representing these particular concepts. Thus, every solution represented in the wiki is considered during a problem-solving process. Instead of using an interactive interview mode the user can enter findings inline by clicking on *inline answers* embedded in the normal wiki page. These inline answers are generated based on the semantic annotations made in the article.

In the following we briefly describe the processes of capturing and sharing knowledge in the semantic knowledge wiki KnowWE [3].

2.1 Knowledge Capture

Semantic annotations and knowledge is edited in the mandatory edit pane of the wiki together with standard wiki content like text and pictures. At the moment

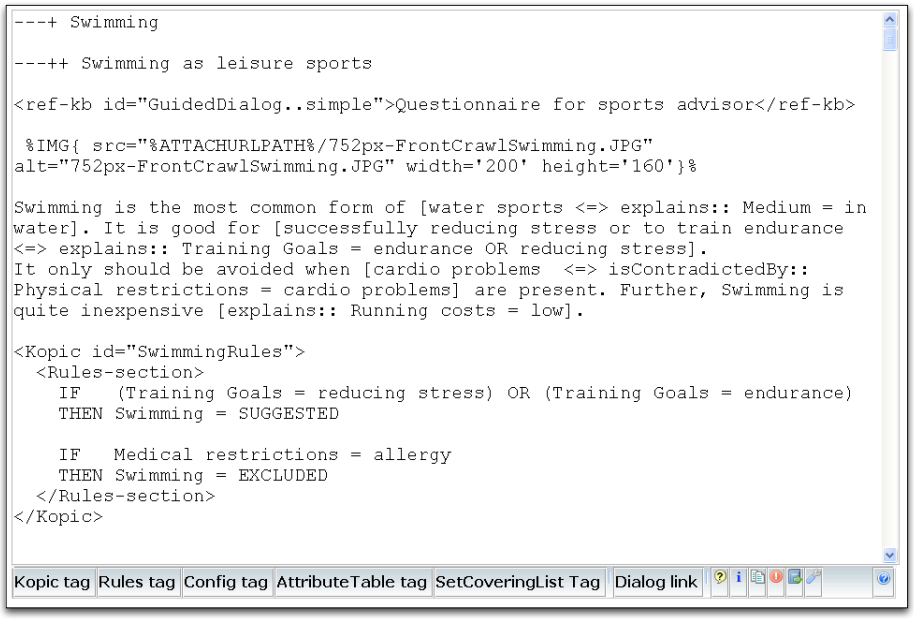

Fig. 1. Editing a wiki article of the knowledge wiki KnowWE: A rule base is embedded into the standard wiki text (bottom of the edit pane) and semantic annotations are made in the first paragraph of the text.

KnowWE proposes the textual acquisition of annotations and knowledge in the edit pane, thus a collection of textual markups for annotations and knowledge

is required. For a new solution a corresponding wiki page with the solution's name is created. The wiki page includes describing text in natural language and the explicit knowledge for deriving the solution. In this paper, we introduce the concept of distributed knowledge engineering with knowledge wikis, and we demonstrate the methods and techniques using the toy application of a *sports advisor wiki*. The running example considers a wiki providing knowledge about different forms of sports, both in textual and in explicit manner. Explicit knowledge can be used to derive an appropriate form of sport for interactively entered (user) preferences. Besides such a simple recommendation application the wiki can be used for a variety of tasks briefly sketched in the case study.

For example, in Figure 1 we see the edit pane of an article describing the form of sports "Swimming": Standard text is semantically annotated by the particular properties `explains` and `isContradictedBy` for which their meaning is described in the following. Additionally, the first part of a formalized rule base is shown at the bottom of the edit pane. Here, knowledge for deriving and excluding the solution "Swimming" is defined. Besides rules derivation knowledge can be formalized in different manners, for example, explicit set-covering models, structured texts, semantic knowledge annotations. We discuss the different markups in the rest of the paper.

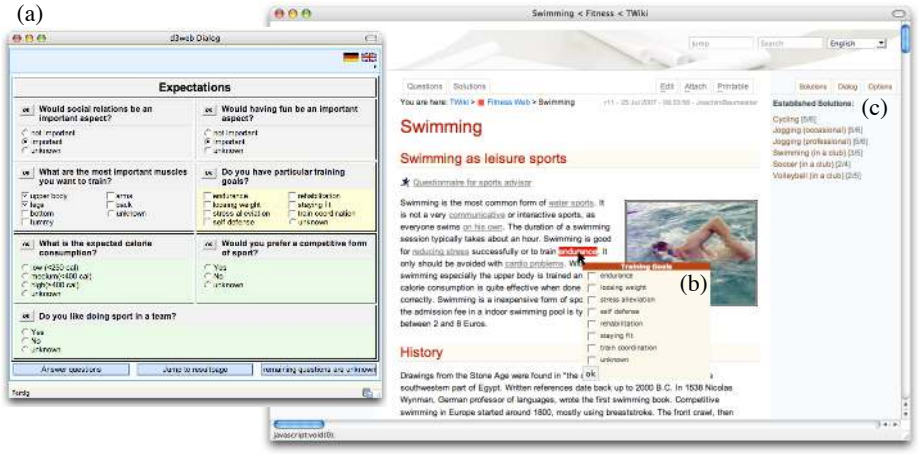

Fig. 2. Possible interfaces for a problem-solving session: interview mode (a) and in-place answers (b). Derived solutions are presented immediately in the wiki (c).

When saving a wiki article the included knowledge is extracted and compiled to an executable knowledge base. In consequence, we arrive at one separate knowledge base for each wiki article capturing the derivation knowledge for the corresponding concept of the article. With the increasing number of wiki articles the number of knowledge bases will also increase. As we see in the next section the concepts created in the wiki are naturally aligned due to the upper ontology

of the knowledge wiki. Furthermore, the developers are encouraged to reuse a pre-defined application ontology which is build on the fixed upper ontology. However, ad–hoc defined findings not corresponding to the application ontology can be easily aligned by expressing alignment rules, that match these concepts with concepts of the application ontology.

2.2 Knowledge Sharing

Besides standard ways of knowledge sharing in (semantic) wikis like (semantic) searching and browsing we provide two ways for a more interactive knowledge sharing in knowledge wikis: first, every wiki page can generate an interactive interview from the included knowledge base by asking questions represented by the findings used in the knowledge base, as for example depicted in Figure 2 a. Second, semantic annotations in text are used to offer inline answers, i.e., clickable text in the article asking for meaningful facts corresponding with the highlighted text, cf. Figure 2 b. In both ways a new finding instance is entered into the knowledge wiki corresponding to the clicked finding object. The instance is propagated to the *knowledge wiki broker* that in turn derives solutions based on the entered findings. The propagation paths of the broker are depicted in Figure 3. The entered finding instances are propagated to the broker which aligns

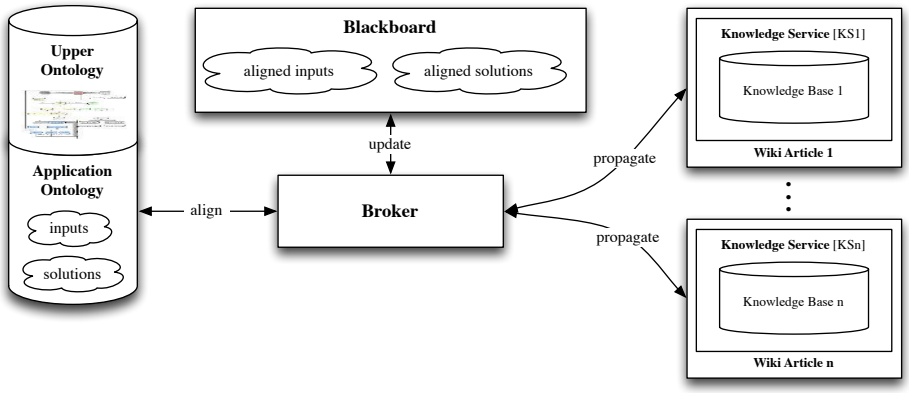

Fig. 3. Blackboard architecture for the distributed problem–solving of the knowledge wiki KnowWE.

the findings to a global application ontology (building on an upper ontology) and then files the aligned instances to a central blackboard. Also, the broker notifies all knowledge bases contained in the wiki for the new fact added to the blackboard and gives the possibility to derive solutions based on the currently available facts. Derived solutions are also propagated by the broker as new facts. With the use of this simple broker/blackboard architecture we are able to allow for a distributed problem-solving incorporating the multiple knowledge bases of

the wiki. Therefore, all solutions represented in the knowledge wiki can be derived at any page; already derived solutions are presented at the right pane of the wiki as for example shown in Figure 2 c. Here, the solutions "Cycling" and "Jogging" were derived as the most appropriate solutions, even though the findings were entered on the page describing the solution "Swimming". The presented example can be seen as a specialized case of *semantic navigation*.

In comparison to our previous work [5], we focus on the knowledge acquisition issues of a semantic knowledge wiki: we discuss an upper ontology as an enabling technology for problem-solving and semantic annotation, and we introduce alternative ways to enter problem-solving knowledge into a wiki, for example by using semantic annotations and structured texts using NLP techniques.

3 An Upper Ontology for Classification Tasks

Studer et al. [6] introduced in detail how the input data and output data of problem-solving methods is structured by specific ontologies. Similarly, we introduce an upper ontology for the classification problem class used in semantic wiki context. The upper ontology is the foundation of every new wiki project. The upper ontology includes the general definitions of findings and solutions that are the basic elements of a problem-solving task. A new wiki project maintains an application ontology by creating specific findings and solutions that are subclassing the concepts of the upper ontology.

3.1 Concepts and Properties of the Upper Ontology

In the following we describe an upper ontology for problem-solving that is used in the semantic knowledge wiki KnowWE. An excerpt of the upper ontology is shown in Figure 4 a; we omitted less important concepts like textual inputs and values for clarity. All unlabeled associations denote *subClassOf* relations.

The concept *Input* plays a key role and allows to describe the world state as a set of variables. Inputs are grouped by the concept *Questionnaire* to structure inputs into meaningful clusters. The two main subclasses of Input are *InputChoice* and *InputNum* to define variables with discrete (named) values and numerical value ranges, respectively. Accordingly, a corresponding value subclassing *Value* is assigned to each Input. The concept *Solution* denotes a special type of a one-choice input that is not entered by the user but derived by a knowledge base, thus representing the final output of a problem-solving session. The value range of a solution is restricted to the possible values *Established*, *Suggested*, *Undefined*, and *Excluded* for expressing the current derivation state of the particular solution.

A concrete problem-solving session is represented by an instance of the concept *PSSession* where a knowledge consumer describes his/her current problem by entering the values of the corresponding observed inputs. The reasoning processes of different users are completely independent from each other as each user is describing his own specific problem instance. The assignment of a value to a

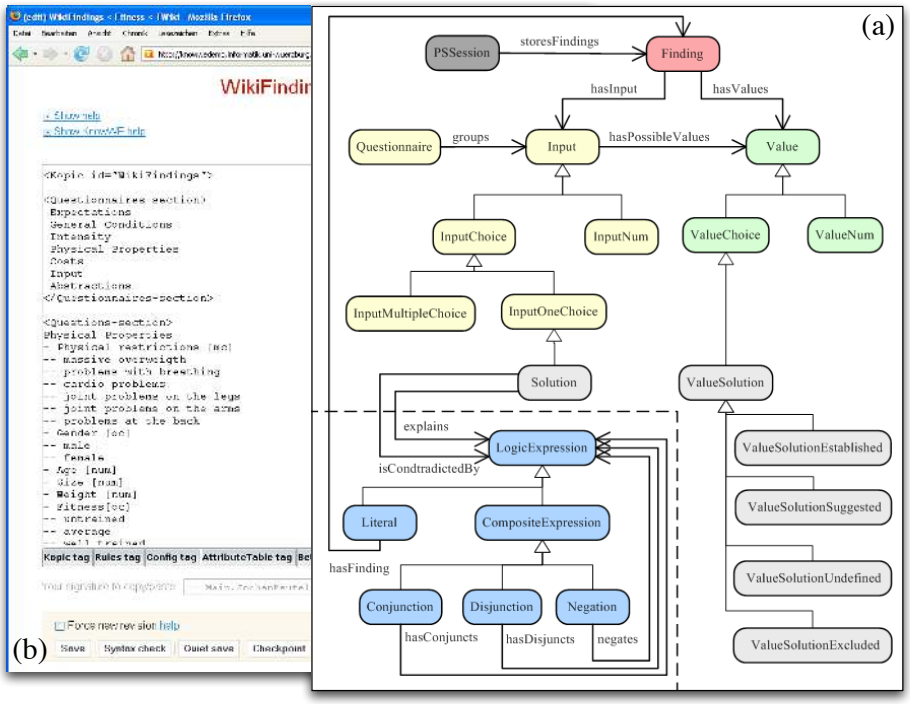

Fig. 4. a) The upper ontology of the semantic knowledge wiki KnowWE. b) a part of the input definitions (WikiFindings page) of a sports advisor demo.

corresponding input is captured by the concept *Finding*, depicted at the top of Figure 4 a.

The proposed knowledge wiki allows for a free and general use of various, alternative knowledge representations to actually derive the concrete solutions defined in the application ontology. For this reason, derivation knowledge is represented in the upper ontology only in a very general manner, as depicted in the left lower corner of Figure 4 a. For the specification of problem-solving relations we introduce the general object properties *explains* and *isContradictedBy* with *Solution* as the domain and *LogicExpression* as its range: The abstract concept *LogicExpression* is subclassed by *CompositeExpression*, which allows fo the composition of logical expressions over findings by the subclasses *Conjunction*, *Disjunction*, and *Negation*. The semantics of the properties *explains* and *isContradictedBy* are described more detailed in the context of the XCL knowledge representation (eXtensible Covering List) in Section 4.

3.2 Creation and Maintenance of the Application Ontology

The concrete inputs and solutions of a new wiki application are defined in the *application ontology*. With the two special wiki pages WikiFindings and Wiki-

Solutions the structure of the application ontology is maintained: The (user) inputs together with their values are defined in the article WikiFindings using the special textual markup `Kopic`. Within this tag we textually define new inputs and their corresponding values together with questionnaires grouping the particular inputs. Defined solutions and inputs of the application ontology are automatically subclassing the corresponding concepts of the upper ontology. Figure 4 b shows a part of the input hierarchy of the sports advisor demo already mentioned before. With one dash we denote a new input followed by its type definition, for example *[oc]* for one-choice inputs and *[num]* for specifying numeric inputs. For choice inputs the possible values are listed in the following lines with an additional preceding dash. Analogously, the solutions of the application ontology are organized in the article WikiSolutions. In summary, the application ontology is created and modified in a wiki-like way, i.e., by editing the wiki pages WikiFindings and WikiSolutions.

For the knowledge engineering task we propose an evolutionary process model as introduced by Fischer [7]: at the beginning of a project an initial effort – called *seeding phase* – has to be made to create a simple but usable basic application ontology. In the working progress the ontology is extended and restructured in cyclic *evolutionary growth* and *reseeding* phases.

4 Simple Knowledge Representations for Ad-Hoc Knowledge Engineers

In the previous section we introduced an upper ontology for problem-solving representing the basis of an application ontology. This application ontology defines the specific inputs and solutions of the particular application domain. As mentioned earlier, every wiki page is able to capture problem-solving knowledge relating defined inputs with the corresponding solution of the particular wiki page. Since we aim to motivate experienced user to act as ad-hoc knowledge engineers the problem-solving knowledge needs to meet the following requirements:

1. easy to understand and formalize,
2. a compact and intuitive textual representation,
3. yields a transparent and comprehensible inference process.

This will help to break down the initial barriers when

1. making personal knowledge explicit,
2. inserting it into a wiki page text,
3. and finally evaluating the created knowledge.

As complexity in knowledge representation and inference constitutes a major barrier for contribution we face the trade-off between simplicity and expressiveness. Beneath simplicity, we have to make an open world assumption concerning the derivation knowledge for a solution concept. During the development process only a part of the total imaginable/retrievable amount of knowledge is present in the knowledge wiki. Thus, it is desirable that any subset of the (fictional)

complete knowledge can be used to derive the best possible results with respect to the given subset. Further, this subsets of course need to be extensible easily. A knowledge representation meeting this requirements needs to be based on small knowledge units which are to a great extend independent of each other. On the one hand this characteristics enables to build very small but already working knowledge bases which then can be extended subsequently step by step. On the other hand the knowledge bases show some robustness with respect to the deletion of small parts and redundant definitions by accident or ignorance which is important within the scope of "ad-hoc knowledge engineering".

Considering the concrete problem-solving process of a knowledge consumer we need to regard that it is unpredictable which exact subset of inputs is actually observable, as real world situations are often diverse and wicked. Thus, it is desirable to design the knowledge and the inference process in a way, that each subset of entered findings will yield appropriate solution ratings. Of course, in this case a confidence value based on the size of the entered finding set needs to be presented along with the resulting solution states. To cope with these challenges we provide the wiki user a simplified but easily extensible version of set-covering models [8], called eXtensible Coverings Lists (XCL). In our approach the extensible-covering model of a solution basically consists of a set of n findings. The weighting of the findings set to $1/n$ as default and we use the individual similarity function. Apriori, the resulting function for a solution rating is therefore restricted to m/n, where m is the number of correct findings and n the number of total findings defined by the model. The possible result spans a real range from $[0, 1]$ which is partitioned into four disjoint intervals representing the corresponding values of a solution *ValueSolutionEstablished*, *ValueSolutionSuggested*, *ValueSolutionUnclear*, and *ValueSolutionExcluded*. Obviously, there is a crucial lack of sensitivity concerning single findings, which is bounded by $1/n$. To improve the limited expressiveness we introduced several extensions to this simple finding list, for example combined findings and exclusion knowledge. In the following section XCL and its textual markup is explained in more detail.

4.1 Embedding Simple Knowledge in Wiki Texts

We present three different textual markups to integrate simple knowledge as described above in standard wiki texts. To demonstrate the similarities and differences of the markups we define knowledge in each way for the solution *Swimming* corresponding to the sports advisor demo.

eXtensible Covering Lists (XCL) The most compact representation of the covering knowledge is its formalization as an *eXtensible Covering List* that is wrapped in a `Kopic` tag. As noted earlier the `Kopic` tag can be placed anywhere in the wiki article and is also used to define other classes of knowledge like the solutions and findings of the application ontology. The names of the inputs and the corresponding values are matched against the definitions made in the application ontology found in the WikiFindings article (cf., Figure 4b). In the

following example an XCL model for the solution *Swimming* is shown. Each line represents a positive coverage of the finding by the solution and is called *explains* relation. The order of the listed findings is not relevant for the inference process and thus is arbitrary. If the domain knowledge is already available as informal text, then it denotes a simple task to transfer the key findings described in the text into basic findings contained in an XCL.

```
1 <Kopic id=‘‘Swimming scmodel’’>
2   <XCL-section>
3     Swimming  {
4       medium = water,
5       Type of sport = individual,
6       Training goals ALL {endurance, stress reduction},
7       Running costs = medium,
8       Trained muscles = upper part,
9       Trained muscles = back
10    }
11  </XCL-section>
12 </Kopic>
```

During the inference process the best rated solution is chosen. A solution is rated by comparing the findings defined in the XCL against the findings entered by the user. The rating of a solution is expressed by its *covering score*. As mentioned before, this numeric score is mapped to four predefined solution states *Unclear* (default), *Suggested*, *Establised* and *Excluded*.

The basic XCL representation can be extended in multiple ways: Besides the simple listing of findings shown above the XCL representation offers further elements to extend/refine the expressiveness and selectivity of the covering model that are briefly discussed in the following (the textual markup is given in paratheses):

Exclusion knowledge `[--]`**:** This marks a relation such that the derivation of the solution becomes impossible to be positively derived when the relation is fulfilled. This type of relation is called *isContradictedBy* and it sets the state of the solution to *Excluded* when fulfilled. Such a constraining relation is defined by two minus signs in brackets (`[--]`) at the end of the relation line, as for example shown in line 7 of the markup shown below.

Required relations `[!]`**:** Relations can be marked as *required* by using a bracket containing an exclamation mark (`[!]`). Then a solution can only be established as a possible output when all required relations are fulfilled, i.e., the corresponding findings were positively entered by the user. An example is shown in line 2 in the following markup.

Sufficient relations `[++]`**:** By adding a bracket with two plus signs (`[++]`) to a relation we define this solution to be a sufficient relation. Then, the solution is always established if the corresponding finding is fulfilled. We call this

relation *isSufficientlyDerivedBy*. It is important to know that contradicting relations are dominating sufficient relations.

Adding weights `[ num]`: In the initial version every explains relation is equally important when compared to the other relations of the XCL, thus having the default value 1. The default value can be overridden for relations in order to express their particular importance. In the textual notation the weight is then entered in brackets at the end of the relation definition, for example `[2]` to double weight a relation. See line 8 in the following markup for a further example.

Logical operators (`AND`, `OR`): A complex relation can be created by combining relations by logical operators. For example in line 2 of the model shown below the findings *medium = water* and *Type of sport = individual* are connected by the logical or-operator (`OR`). The resulting knowledge demands that either *medium = water* or *Type of sport = individual* needs to be observed to fullfil the relation. The three basic operators of propositional logic *or*, *and* and *not* can be used.

Threshold values: When rating a solution the numeric covering score is mapped to a solution state. The mapping function is defined by the threshold values (`establishedThreshold` and `suggestedThreshold`). In most cases the internal default threshold values are adequate, but for distinct solutions they can be overriden as shown in line 12-13 of the following example model. In the example, 70% of the expected and observed findings need to be correctly observed to set the solution tot the state *Established*, and 50% to set the solution to the state *Suggested* (higher states overwrite lower states). Further, with `minSupport` we specify how many percent of the findings defined in the XCL model need to be entered by the user in order to activate the solutions rating process.

The following markup shows a refined version of the previous model of the solution *Swimming* using the described elements. Essentially, the already defined relations were mostly refined by relational extensions like sufficient, contradicting and necessary properties.

```
<Kopic id=‘‘Swimming scmodel’’>
  <XCL-section>
    Swimming  {
      medium = water OR Type of sport = individual [!],
      My favorite sports form = swimming [++],
      Training goals ALL {endurance, stress reduction},
      Favorite color IN {red, green, blue},
      Running costs = medium,
      Running costs = nothing [--],
      Trained muscles = upper part [2],
      Trained muscles = back [2],
      Physical problems = skin allergy [--],
      Type of sport = group [--],
    }[ establishedThreshold = 0.7,
       suggestedThreshold   = 0.5,
       minSupport           = 0.5
      ]
  </XCL-section>
</Kopic>
```

Although the presented representation is experienced to be compact and intuitive for most of the users, it is clearly separated from the remaining text of the wiki article, which usually describes the same concept and its knowledge in natural language. This separation increases the risk of “update anomalies”, for example if users are modifying or extending the wiki article but not the corresponding part of formalized knowledge. Therefore, a tighter integration of formal knowledge and informal text of a wiki article is desirable, and for this aim we introduce two possible approaches in the following.

Inline Annotation Many semantic wikis use inline annotation techniques to describe semantic properties between concepts of the ontology, for example Semantic MediaWiki [9]. Using special properties of the upper ontologies like `explains` and `isContradictedBy` we are able to capture set-covering knowledge by evaluating semantic annotations. The following example shows sentences describing the solution “Swimming”, where text phrases are annotated by the property `explains` for defining positive set-covering relations and the property `isContradictedBy` for the definition of exclusion rules. For example, the first two lines state a relation between the solution `Swimming` (the concept of the page) and the finding `Medium = in water`: the first expression after the opening brace and before the `<=>` is the textual part of the sentence, that will be

rendered in the view mode of the article, shown in Figure 2 b. The phrase before `<=>` can be omitted; then the preceding word before the annotation is highlighted and related to the corresponding relation. The following part of the annotation states the name of the property (e.g., `explains`, `isContradictedBy`) followed by two colons. After that, the actual finding related to the solution concept is specified, i.e., the range of the given property. In the topic view the annotated text can be used as an interview method with inline answers. In this case the input of a set-covering relation is posed as question to the knowledge consumer as shown in Figure 2 b.

```
1 Swimming is the most common form of [water sports <=> explains::
2 Medium = in water]. Swimming is good for successfully [reducing
3 stress or to train endurance <=> explains:: Training Goals =
4 stress alleviation OR Training Goals = endurance]. It only
5 should be avoided when [cardio problems <=> isContradictedBy::
6 Physical restrictions = cardio problems] are present. Further,
7 Swimming is quite inexpensive [explains:: Running Costs = low].
```

The semantic annotation of existing sentences means less knowledge acquisition workload compared to the explicit markups introduced before. Although, standard wiki text is tightly integrated with formal knowledge the readability of the text suffers from annotations as shown above.

Structured Text A more radical approach is to omit semantic annotations when possible and to use NLP techniques for annotating distinct parts of the wiki text. In the context of our work, we are able to use "structured texts" since 1) the available text needs to be mapped to a rather simple knowledge representation, and 2) we can employ the given application ontology as background knowledge for the concept extraction task. This is very similar to the approach of the DBpedia project[2] described by Auer et al. [10] which uses the article structure of wiki pages to formalize knowledge about the described topic. Instead of creating RDF-triple we want to generate problem-solving knowledge for the classification task. The key problem here is the matching of the natural language expressions to the concepts of the application ontology.

Using this technique we assume, that a distinct block of a wiki article is tagged as a "structured text". Then, this block is parsed in order to identify findings for which set-covering relations are created. In a first step we are working with semi-structured texts (e.g. bullet lists, tables). In the following example in Figure 5 a we show a bullet list in standard wiki syntax, where each line contains one (combined) finding explaining the solution `Swimming`. While the inline annotations expect exact matches we applied some lightweight linguistic methods for matching findings in structured texts, for example simple string matching combined with stemming and synonym lists already lead to fairly good

[2] DBpedia: http://www.dbpedia.org

results. In the simplest case we can identify an input name that is defined in the application ontology together with a corresponding value name, for example as found in line 5 of Figure 5 a: the text phrase "when low risk of injuries is desired" yields the finding "risk of injury = low". The finding defined by this input-value-pair tuple can be added as a set-covering relation of the solution *Swimming*. Sometimes only an input is listed and humans implicitly refer to a default value of this input, especially for inputs only having "yes" and "no" as possible values (e.g. "practiced outdoor"). For this type of input we assume "yes" as the default value when creating set-covering relations. For other inputs the default answer needs to be defined in the application ontology, otherwise the finding cannot be completely identified. Another popular case is appearance of the value name in the line as the only indicator of a finding, for example in line 2 `in water`. If the corresponding input can be clearly identified due to the unique

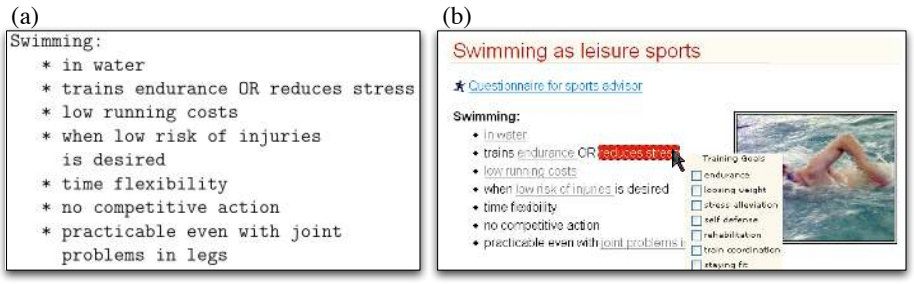

Fig. 5. a) Wiki syntax of a bullet list in structured text, b) automatically annotated text phrases in a semi-structured text.

name of the found value, then we automatically generate a relation accordingly. We often can disambiguate the occurrence of such a finding, since most of the times humans only reduce the text to the value name if this would not result in an ambiguity. All identified findings are implicitly annotated and can be used to provide inline answers as shown for example in Figure 5 b.

4.2 Knowledge Representations in Explicit Markup

Although in various forms extensible the XCL representation has limited expressiveness for some type of domain knowledge.

```
<Rules-section>
  // Abstraction rule r1 for body mass index calculation
  IF   (Height > 0) AND (Weight > 0)
  THEN  BMI = (Weight / (Height * Height))

  // Derivation rule r2  for solution Running
  IF ("Training goals" = endurance)
  AND ("BMI"<30) AND NOT("Physical Problems" = knees)
  THEN Running = SUGGESTED
</Rules-section>
```

In order to allow for the creation of complex knowledge relations we provide further knowledge representations to be used by more experienced users. The textual markup of alternative representations was introduced in [5]. In the context of this paper we briefly show the definition of rules for the derivation of abstractions – for example to be used for concept mapping – and rules for the derivation of solutions. The rules section shown above contains two rules taken from the sports demo. The first calculates the body mass index (BMI) and the second rule sets the solution *Running* to the value *Suggested*. In addition, we provide decision trees and several table-based representations for rules and set-covering relations.

5 Case Studies

We have implemented the presented approach with the system KnowWE [3], a semantic knowledge wiki, that is still under lively development. For an extensive evaluation of the applicability of our system we made a student based case study considering the creation of knowledge wikis with a group of 45 students. Within about three weeks 11 recommendation systems with a total amount of about 700 knowledge bases containing rules an set-covering relations were created. Beyond further student projects, the system is currently used in the context of the BIOLOG Europe project (http://www.biolog-europe.org). Its purpose is the integration of socioeconomic and landscape ecological research results in order to produce a common understanding of the effects of environmental change on managed ecosystems. Inter- and trans-disciplinary research projects with economists yielded socioeconomic knowledge on how the biodiversity can be supported in managed agro-ecosystems. The research results are present in the form of large amounts of (unstructured) knowledge on landscape diversity of life with respect to the given landscape structures, management decisions and their progression [11], for example described in papers, data sheets, and further multimedia content.

The project wiki LaDy (for "Landscape Diversity") aims to support domain specialists and interested people to collect and share knowledge in the context of the BIOLOG project. The knowledge appears at different levels of detail ranging from textual descriptions and multimedia to formal knowledge covering the effects on landscape diversity. Typically user inputs consider the description of the investigated landscape, whereas solutions are defined with respect to the biodiversity of various taxa, different ecosystem services and management decisions. At the moment, the knowledge wiki is under development incorporating ecological domain specialists distributed all over germany.

The participating domain specialists have neither background in knowledge representation nor in ontology engineering, and therefore the interfaces need to be as simple and intuitive as possible. In the various kick-of meetings we learned that a simple set-covering list representation was experienced to be intuitive and suitable for the first steps. After some simple examples the requirements of the users concerning the expressiveness usually grew, and in many cases these requirements could be covered by extended covering lists as shown for example in the following. In Figure 6 a, the solution `High plant diversity` is defined by a set-covering list of constrained findings, where the second item (line 3–5) is a complex finding combining a list of atomic findings by a disjunction and a conjunction (simplified example shown).

In other cases we offered to transform the existing knowledge to a rule base, since the largest degree of expressiveness can be provided by a rule-based representation. An excerpt for a rule base is shown in Figure 6 b where the value of management productivity is defined in correspondence of inputs such as genetic diversity and optimized soil retention. Providing a platform for both, exchanging textual knowledge and implementing explicit rules on ecosystem behaviour, LaDy provides a service to condense and to communicate knowledge needed for an efficient management of ecosystem services.

6 Conclusions

We have introduced the concept of a semantic knowledge wiki with the implementation KnowWE that extends the known OWL-based expressiveness of other semantic wikis by active problem-solving capabilities. Whereas related approaches provide strong support to capture ontological knowledge – for example see [1, 2] – our main goal is to make the engineering of executable problem-solving knowledge as simple as possible thus supporting the formation of ad-hoc knowledge engineers. For this reason, we presented an upper ontology connecting ontological knowledge with strong problem-solving knowledge, and we introduced different possible ways to formalize problem-solving knowledge, for example semantic knowledge annotations, (semi-)structured texts, and explicit knowledge markups.

In the future we are planning to improve the power of natural language to be used as direct input for knowledge acquisition, incorporating more linguistic methods and controlled languages. Related work is reported by the Attempto

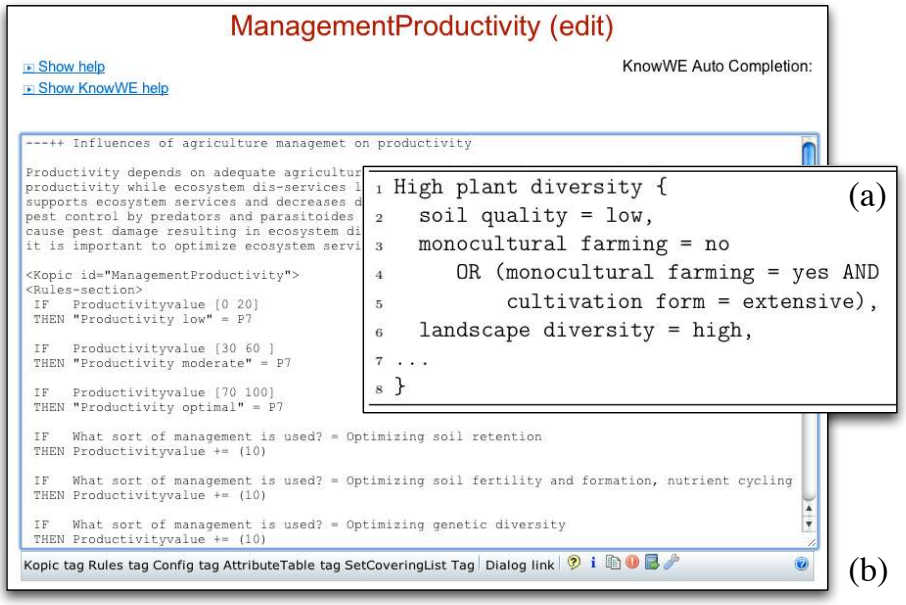

Fig. 6. a) Excerpt of a simplified version of a set-covering model for deriving "High plant diversity" b) rules describing the value of management productivity depending on inputs such as genetic diversity and optimized soil retention.

project [12], where *Attempto Controlled English* (ACE) uses a knowledge representation that is equivalent to first order logic and is also being combined with a wiki technology in the system AceWiki. Besides more sophisticated methods to formalize knowledge we have further research questions that need to be adressed in the future: In a distributed setting existing methods and tools for the evaluation and the refactoring need to be reconsidered and refined in order to facilitate the maintenance and quality of an evolving semantic knowledge wiki.

References

1. Schaffert, S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In: STICA'06: 1st International Workshop on Semantic Technologies in Collaborative Applications, Manchester, UK (2006)
2. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, Berlin, Springer (2006) 736–749
3. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE – Community–based Knowledge Capture with Knowledge Wikis. In: K-CAP '07: Proceedings of the 4th International Conference on Knowledge Capture, New York, NY, USA, ACM (2007) 189–190

4. Baumeister, J., Puppe, F.: Web-based Knowledge Engineering using Knowledge Wikis. In: Proceedings of Symbiotic Relationships between Semantic Web and Knowledge Engineering (AAAI 2008 Spring Symposium). (2008)
5. 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
6. Studer, R., Eriksson, H., Gennari, J., Tu, S., Fensel, D., Musen, M.: Ontologies and the Configuration of Problem-Solving Methods. In: Proc. of the 10th Knowledge Acquisition for Knowledge-based Systems Workshop, Banff. (1996)
7. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain–Oriented Design Environments. Automated Software Engineering **5**(4) (1998) 447–464
8. Reggia, J.A., Nau, D.S., Wang, P.Y.: Diagnostic Expert Systems Based on a Set Covering Model. Journal of Man-Machine Studies **19**(5) (1983) 437–460
9. 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
10. Auer, S., Lehmann, J.: What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In: The Semantic Web: Research and Applications. (2007) 503–517
11. Otte, A., Simmering, D., Wolters, V.: Biodiversity at the Landscape Level: Recent Concepts and Perspectives for Multifunctional Use. Landscape Ecology **22** (2007) 639–642
12. Kuhn, T.: AceRules: Executing Rules in Controlled Natural Language. In: Proceedings of First International Conference on Web Reasoning and Rule Systems. Volume 4524 of LNCS. (2007) 299–308