

Knowledge Representation and Query in Semantic MediaWiki: A Formal Study

Jie Bao, Li Ding, and James Hendler

Tetherless World Constellation
Rensselaer Polytechnic Institute
Troy, NY, 12180-3590, USA
{baojie,dingl, hendler}@cs.rpi.edu

Revision: August 12, 2009 13:52
<http://tw.rpi.edu/wiki/TW-2008-42>

Abstract. Semantic wikis extend conventional wikis, which allow users to collaboratively generate data, through semantic web technologies. This allows wiki-based information to be annotated with metadata for facilitating automated search, querying, propagation and inference. Semantic wikis have recently attained wide adoption and, with this, formal research on their functionality (e.g., inference and concept modeling) is starting to be conducted. In this paper, we present a formal analysis of the semantic technologies in an emerging semantic wiki framework called Semantic MediaWiki (SMW). In particular, we define a RDF-like model theoretical semantics for the SMW modeling language, and a closed world, minimal Herbrand model semantics for the SMW query language by a reduction to positive logic programs. We discover that the entailment problem in SMW is NL-complete thus is highly tractable, and query answering in SMW is P-complete in general and is in L if subqueries are disallowed. We show that such a formalization allows us to discover some rather powerful usages of SMW, such as entailment inference, and to extend SMW with some frequently needed features, like negation as failure.

1 Introduction

Semantic wikis, such as Semantic MediaWiki (SMW) [24] and IkeWiki [21], are extensions to conventional wikis that provide content enrichment using Semantic Web based technologies. Wikis, such as MediaWiki¹, allow for content generation through community-based efforts mirroring those of the world wide web. Likewise, Semantic Web technologies (in the form of various languages and tools) help both define and organize data semantics. This enables the automation of many data processing tasks (such as inference), and helps improve services such as content-based search. Thus, by extending wikis with Semantic Web technologies, significant data processing functionality is gained.

¹ <http://www.mediawiki.org>

For this reason, semantic wikis have been widely adopted to handle collaborative content generation in various domains, such as healthcare and the natural sciences. In addition, semantic wikis have been incorporated into various light-weight Semantic Web applications, such as semantic blogging, social recommendation systems, semantic portals and interactive map systems².

However, despite the popularity and usefulness of semantic wikis, there is relatively limited work on the formal aspects of semantic wikis. A formal understanding of the precise meaning of contents in semantic wikis would bring us a couple of advantages that include:

- A better understanding of both their potential usage and limitations. Relevant under this banner include “is it safe to add property domain and property range as natively supported features in a particular wiki implementation?”, as well as “what kind of semantic queries can *not* be run on your semantic wiki?”.

- The ability to formally evaluate a semantic wiki implementation. Some example questions could be: (i) Will the query engine used by the semantic wiki always fetch the correct information?; (ii) Will this information contain all the data meeting the selection condition?; and (iii) Could different users get different answers with the same query?

- The improved ability to interact with external systems, including situations like (i) to enable an external system to query a semantic wiki with guaranteed soundness and completeness; (ii) to support different semantic wiki systems to share a common modeling or query language with the same shared semantic meaning; and (iii) to extend a semantic wiki query language to get data from the outside world, e.g., by translating into SPARQL³.

In this paper, we choose SMW as a representative of semantic wikis for study because of its popularity and maturity. In particular, SMW has both a modeling script language and a specially defined query language that have been proven with high usefulness as well as learnability (due to their simplicity), while some other systems lack one or both. On the other hand, SMW presents some generic features that are commonly shared with many other semantic wiki systems, e.g., the use of wiki pages as basic entities of knowledge modeling and the connection to a triple store for querying. Therefore, we believe many of our results on SMW can be extended to other semantic wiki systems as well.

Some previous attempts have been made to precisely define the syntax and semantics of SMW. Krötzsch, Vrandečić, et al. [14], the creators of SMW, enabled semantics in both its querying and modeling languages based on a variant of OWL DL [26]. Unfortunately, adopting this Description Logic (DL) based semantics for SMW leads to some data expressivity restrictions. In particular, its semantics requires disjointness between concepts, individuals and properties, which may be too restrictive for some applications. For the SMW querying language specifically, a DL based semantics imposes by default an open world assumption. This is not always appropriate for the information on a SMW site, and could make some functional extensions (e.g. negation as failure) difficult.

² Some examples are listed on <http://tw.rpi.edu/wiki/Wiki@RPI>

³ We thank Yaron Koren for suggesting some of these advantages.

Against this background, this paper introduces a formal analysis of semantic wikis based on the SMW framework with an alternative semantics that overcomes many limitations of the DL-based semantics. We analyze the two main components of SMW: its modeling language and corresponding query language. The contributions of the paper include:

- A precise formalization of the complete SMW modeling and query language syntax, thus complement the previous work which has presented partial formalization of SMW [26, 14].
- The analysis of the entailment rules of SMW, which may guide the design of a sound and complete reasoner for SMW, either as an external reasoner or a reasoner natively enabled in an SMW site. We also discover the NL-complete complexity for the entailment problem in SMW.
- The translation of the SMW query language into logic programs, resulting in a closed world, minimal Herbrand model semantics. This will allow for both evaluation of completeness of a SMW query engine and the extension of the SMW query language with features available in logic programs. We show that query answering in SMW is P-complete in general and is in L if subqueries are disallowed.
- The investigation of several use cases in SMW, including wiki view, rule inference and closed/open world reasoning.

2 SMW Modeling Language

SMW lets users create semantic annotations in wiki pages using an extension to the scripting language of MediaWiki. In this section, we review the basic syntax of the modeling language provided by SMW⁴, denoted as SMW-ML, and then clarify its corresponding semantics.

2.1 Syntax

The basic unit in a MediaWiki (MW) site is a *wiki page*. The full name of a wiki page is defined in the form “NAMESPACE:NAME”, where some predefined namespaces exist such as “Category” and “User”. MW has a scripting language to describe contents of wiki pages, which is extended by SMW with the following three basic sets of *semantic annotations*:

Class and Property Definitions: SMW reuses the “Category” namespace to define classes⁵. For example, a wiki page with the name “Category:Person” is intended to represent the class of all persons. Likewise, a new namespace called “Property” is introduced to define concept properties. Through this approach, SMW supports both binary and n-ary properties.

Axioms: SMW allows for the declaration of subclass and subproperty relations. For example, on the page “Category:Student”, the annotation “[[Category:Person]]” would indicate that Student is a subclass of Person. SMW also

⁴ This paper covers SMW features up to the most recent release 1.4.2.

⁵ For this reason, in the following we use “Category” and “Class” interchangeably.

supports equality relations between two wiki pages, or between two classes, via the redirection mechanism of MediaWiki ^{6 7}.

Instance Assertions: SMW allows a page to be declared as an instance of a class or the subject of an RDF triple. For example, if the wiki page “John” contains the annotation “[[Category:Person]] [[Has email:john@foo.edu]]”, this means that John is a Person and he has the email address “john@foo.edu”. It is also possible to create an instance of an n-ary property. For instance, on the wiki page “John” the annotation “[[Has education::RPI;2008]]” would mean that he went to the school RPI and was in the class of 2008.

For convenience, we denote the recursive closure of subclasses (or subproperties) of a class (or property) x as $\text{Sub}(x)$. We use the term *SMW system* to denote a set of wiki pages with semantic annotations in SMW-ML, which could also contain non-semantic text. Likewise, we use the term *vocabulary* to denote the names of all pages mentioned (including those that are not yet created) in a SMW.

In Table 1, we summarize SMW-ML with its correspondence to the RDF Schema (RDFS) language. This complements the previous work on SMW-ML [26] with newly introduced features, e.g., equality axioms. Here we leave out the details about datatype annotation and some special properties in SMW. Since they do not affect the semantics discussion; the detailed description for the language element can be found in the SMW documentation⁸.

It must be noted that the comparison to RDFS in Table 1 is informal, and the detailed formal semantic difference between SMW-ML and RDFS will be discussed in the next subsection.

2.2 Semantics

A formal understanding of SMW-ML semantics is useful to guide the design and evaluation of the query and reasoning systems for SMW. In addition, this will also enable us to better understand the expressivity limitations and complexity bounds on typical query and inference tasks with SMW. Here, we will provide a formulation of the semantics of the modeling language given in the previous subsection.

Comparison with OWL DL and RDF Semantics Before we present our definition of the SMW-ML semantics, we will show why the DL-based Seman-

⁶ Currently, SMW does not support the transitive inference on the equality relation. This restriction, however, may only affect the transitive inference of instance equality because class equality and property equality can be reduced to subclass (subproperty) relations.

⁷ In SMW, users can also define mutual, cyclic and chained redirections. Without losing generality, we assume that for an equality class of pages, there is a hub page such that all other pages in the class are redirected to the hub page, therefore avoid the existence of mutual, cyclic and chain redirection. In practice, a hub page can be easily chosen using a wiki bot.

⁸ <http://www.semantic-mediawiki.org>

	SMW-ML Syntax	RDF Statements	Context
	Class and Property Definitions		
Class	Category:x	x rdf:type rdfs:Class	x as a page name
Binary property	Property:x with [[has type::t]] (optional)	x rdf:type rdf:Property	x as a page name, t a type
N-ary property	[[has type::t ₁ ;...;t _n]]	N/A	x as a page name, t _i a type, n ≥ 2
	Axioms		
Subclass	[[Category:x]]	y rdfs:subClassOf x	on a page Category:y
Subproperty	[[Subproperty of::Property:x]]	y rdfs:subPropertyOf x	on a page Property:y
Instance equality	#REDIRECT [[x]]	y owl:sameAs x	on a page y
Class equality	#REDIRECT [[Category:x]]	y owl:equivalentClass x	on a page Category:y
Property equality	#REDIRECT [[Property:x]]	y owl:equivalentProperty x	on a page Property:y
	Assertions		
Class instance	[[Category:x]]	y rdf:type x	on a non-category page y
Binary prop. instance	[[x::y]]	z Property:x y	on a page z
N-ary prop. instance	[[x::y ₁ ;...;y _n]]	N/A	on a page z,x a (n+1)-ary property,n ≥ 2

Note 1: x,y,z in the “RDF Statements” column should be understood as the URLs of the corresponding wiki pages.

Note 2: a type can be one of the datatypes supported by SMW, including **Type:Page**(default) **Type:String**, **Type:Number**, and many others

Table 1. Correspondence between SMW-ML (version 1.4.2) and RDFS/OWL

tics [26] or a direct RDF-based semantics do not satisfy or maybe unnecessarily restrictive.

The adoption of the DL-based Semantics [26] would require the strict disjointness between categories, properties and individuals. However, by requiring that we may preclude many useful usage scenarios of SMW-ML that are not OWL DL-compliant, as given in the following examples:

- Using a category page as a property value. In building a technical report system⁹, we have used categories (e.g., `Category:Wiki`) to build a topic hierarchy, and assign topics to papers, using SMW-ML scripts like `[[has topic::Category:Wiki]]`. This will also allow us to use SMW-QL to find all papers that have topics of `Category:Wiki` or any of its subtopics.

- Making a property as an instance of a category or a subject of a property instance. For managing a large set of properties, we often need to describe its classifications and attributes. For example, we may add to a property page the annotations `[[Category:Release 1.0]]` and `[[defined by::John]]`.

One could also argue that the semantics of SMW-ML could be obtained by mapping the language to RDFS and using the RDF Semantics [12]. However, such an approach would overlook several important differences between SMW-ML and RDFS:

- Resources (classes, instances, properties) in SMW-ML can only be local wiki pages¹⁰, while in RDF a resource can be identified by any URI.

- SMW-ML supports n-ary properties and equality between instances, which are not supported in RDFS.

- SMW-ML does not support several features in RDFS, including property domains and ranges, blank nodes, collections and lists.

- In SMW-ML, classes and properties are disjoint, while in RDFS this is not necessarily the case.

- There is no round trip translation between SMW-ML and RDFS. For example, in RDFS, one class can be declared as an instance of another class. For instance:

```
Comedy rdfs:type rdfs:Class, and
Comedy rdfs:type Movie;
```

However, in SMW-ML there is no correspondence since the identification of a class is by a syntactic approach (namespace) and the `[[Category:x]]` annotation only indicates the subclass relation in the context of a class (category) page (`Comedy rdfs:subClassOf Movie`).

- The usage of SMW-ML sometimes has different semantic explanations from its RDFS correspondence, more details are in Example 1.

Example 1 Assume the page “John” has the annotation `[[Has hobby::Category:Comedy]]`, and the page “Category:Comedy” has the annotation `[[Category:Movie]]`. One

⁹ <http://tw.rpi.edu/wiki/Literature>

¹⁰ There are some technical solutions, beyond SMW-ML, to reuse remote resources in SMW. One example is to link a local wiki with an existing resource on the web by “importing”. Another example is to allow remote querying using, for example, the “AskMany” extension (<http://tw.rpi.edu/wiki/AskMany>).

would expect the answer to the query “who lists movies as a hobby?” to contain “John”. However, if we map the SMW-ML annotations to their corresponding RDFS statements:

```
John Has_hobby Comedy
Comedy rdfs:subClassOf Movie
```

We will not be able to infer that

```
John Has_hobby Movie
```

from the RDFS semantics [12].

A Model-theoretical SMW-ML Semantics To characterize the semantic traits of SMW-ML, we present a model-theoretical semantics in the following statements. For purpose of conciseness, we do not address datatypes in the semantics. However, our basic framework can be easily extended for the case with datatypes.

Let W be a SMW system and the *vocabulary* V be the non-empty set of wiki page names in W . Let m be the maximal arity of all property pages in V . An interpretation I of W is defined as the tuple $\langle IR, IS, IC, IP, IEXT, ICEXT \rangle$, where:

1. IR is a non-empty set of resources, called the domain or universe of I ;
2. IS is a surjection from V to IR ;
3. $IC = \{IS(x) \mid x \text{ is in the namespace "Category", } \forall x \in V\}$ (Thus $IC \subseteq IR$);
4. $IP = \{IS(x) \mid x \text{ is in the namespace "Property", } \forall x \in V\}$ (Thus $IP \subseteq IR$);
5. $IEXT$ is a mapping from IP into $P(\bigcup_{2 \leq n \leq m} IR^n)$, where P means power set;
6. $ICEXT$ is a mapping from IC to $P(IR)$.

I is a *model* of W if the following conditions are satisfied:

1. $IR = IS(V)$ ¹¹;
2. IC and IP are disjoint;
3. If x is an n -ary property, then $IEXT(IS(x)) \subseteq IR^n$;
4. If a non-category page x has the annotation “[[y]]” where $IS(y) \in IC$, then $IS(x) \in ICEXT(IS(y))$;
5. If a page x has the annotation “[[y]]” where x, y are category pages, then $ICEXT(IS(x)) \subseteq ICEXT(IS(y))$;
6. If a page x has the annotation “[[Subproperty of:: y]]” where x, y are property pages, then $IEXT(IS(x)) \subseteq IEXT(IS(y))$;
7. If a page x has the annotation “[[$p::y_1; \dots; y_n$]]” ($n \geq 1$), where p is a $(n+1)$ -ary property, then $\langle IS(x), IS(y_1), \dots, IS(y_n) \rangle \in IEXT(IS(p))$;
8. If a page x has the annotation “#REDIRECT [[y]]”, then $IS(x) = IS(y)$.

I is a *strong model* of W if it is a model of W and meets the following additional condition:

9. If $\langle IS(y_1), \dots, IS(y_i) \dots, IS(y_n) \rangle \in IEXT(IS(p))$ and $y_i \in \text{Sub}(x)$ for some $1 \leq i \leq n$ where y_i and x are category pages, then $\langle IS(y_1), \dots, IS(x) \dots, IS(y_n) \rangle \in IEXT(IS(p))$

¹¹ Note that when datatypes are allowed, $IS(V) \subseteq IR$ would be true.

The additional condition allows for a page to be interpreted as both an individual (i.e., a “normal” wiki page) and a class (i.e., a category wiki page), so that the inference as needed in Example 1 can be supported.

Please note that the semantics given above is substantially different from the RDF semantics [12] on the aspects we just discussed.

2.3 Entailment

An immediate advantage of having a formal semantics of SMW-ML is that it provides a precise foundation for inference over knowledge on an SMW site. For example, one may wish to infer whether “Undergraduate” a subclass of “Person”, given “Undergraduate” is a subclass of “Student”, and “Student” is a subclass of “Person”. Another example goes as follows: a query asks whether “PaperX” have the topic “Wiki”, where “PaperX” is defined to have the topic “Semantic Wiki” and “Semantic Wiki” is defined as a subclass of “Wiki”.

Currently, to enable such inference in SMW, one may resort to some external support of inference, such as:

1. Dumping the knowledge in a SMW site into an RDF file and using an external tool (e.g., Jena) to do inference;
2. Querying the SMW knowledge base from a SPARQL endpoint;
3. Use server-side wiki bots (otherwise called “gardening scripts”) to add additional annotations to wiki pages.

Approaches 1 and 2 are not fully satisfactory due to the semantic difference between SMW-ML and RDF. In addition, dumping the contents of an SMW site may not be practical when considering scalability issues (e.g., handling very large SMW sites). Approach 3 could only be preformed on the server side, and would be limited only to selected, pre-defined types of inference. None of the three approaches would allow for inference by common wiki users without an external tool.

In this subsection, we present an entailment system based on the semantics formulated in the previous subsection. We will also show in Section 4.2 that such an entailment system can be natively implemented in the wiki system without accessing external tools, e.g., reasoners. This is highly desirable as it lowers the learning curve for SMW users, as a large portion these users will not necessarily understand RDF or SPARQL.

We first define the entailment of SMW systems:

Definition 1 : *Let W, W' be two SMW systems, W weakly entails W' , if and only if every model of W is a model of W' ; W strongly entails W' , if and only if every strong model of W is a strong model of W' .*

Example 2 (Entailment of Wiki Systems): Let W be a wiki system with the following pages and their contents:

- PaperX: “[[Has topic::Category:Semantic Wiki]]”

- Category:Semantic Wiki: “[[Category:Wiki]]”
- Category:Wiki: (empty)

W' is obtained from W by adding to PaperX additional text “[[Has topic::Category:Wiki]]”. W strongly entails W' , but does not weakly entail W' .

In practice, it is usually more useful to preform the entailment inference by a set of *entailment rules*. For SMW-ML we define the following entailment rules:

1) If x contains “[[Subproperty of::y]]” and y contains “[[Subproperty of::z]]”, where x,y,z are all in the Property namespace, then add the annotation “[[Subproperty of::z]]” to x ;

2) If x has the annotation “[[p::y₁; ... ;y_n]]” ($n \geq 1$), where p is an $(n+1)$ -ary property and p contains the annotation “[[Subproperty of::q]]”, then add the annotation “[[q::y₁; ... ;y_n]]” to x ;

3) If a non-category page x has the annotation “[[y]]”, y is in the Category namespace that contains the annotation “[[z]]”, z is also in the Category namespace, then add the annotation “[[z]]” to x .

4) If x contains “[[y]]” and y contains “[[z]]”, where x,y,z are all in the Category namespace, then add the annotation “[[z]]” to x ;

5) If x is in the Property namespace, then add the annotation “[[Subproperty of::x]]” to x ;

6) If x is in the Category namespace, then add the annotation “[[x]]” to x ;

7) (Strong entailment rule) If x has the annotation “[[p:: y₁; ... ; y_i; ... ; y_n]]” ($n \geq 1$), where p is an $(n+1)$ -ary property, y_i is in the Category namespace with the annotation “[[z]]” where z is in the Category namespace, then add the annotation “[[p::y₁; ... ;z; ... ; y_n]]” to x .

Here, rule 7 is only needed for strong entailment inference. Rules 5 and 6 are needed only when completeness of inference is required. Also note that an entailment rule is not required to add a redirection annotation between two “equal” pages, due to the assumption that there is a “hub” page for their equality class.

We argue that the above set of rules is sound (i.e., all annotations added by it conform to the semantics) and complete (i.e., all possible annotations that follow from the semantics can be generated from the set of rules), with the following theorem:

Theorem 1 : *Let W, W' be two SMW systems of the same vocabulary, W weakly (strongly) entails W' if and only if W' can be obtained from W by finite applications of the entailment rules 1-6 (1-7).*

Proof Sketch: *Soundness*, the “if” direction. We need to prove that for every (strong) model m of W , m is also a (strong) model of W' . Without loss of generality, we assume that W' is obtained from W by the application of a single rule, since multiple rule applications can be reduced to a series of single rule applications. We only do a detailed proof for rule 1, where all other rules can be handled similarly. In m , we must have that $\text{IEXT}(x) \subseteq \text{IEXT}(y)$ and

$\text{IEXT}(y) \subseteq \text{IEXT}(z)$, thus, $\text{IEXT}(x) \subseteq \text{IEXT}(z)$; therefore, the added annotation "[[Subproperty of::z]]" on x is satisfied.

Completeness, the "only if" direction. We show by proving that if for *every* (strong) model m of W , m is also a (strong) model of W' , then we can construct a SMW system D by recursively extending W with additional annotations obtained using the entailment rules such that $W' \subseteq D$.

- Conditions 1-3 are always satisfied.
- Condition 4: If $\text{IS}(x) \in \text{ICEXT}(\text{IS}(y))$ is always true in every m , it must be one of the cases 1) "[[y]]" is on x in D ; or 2) $\text{IS}(x) \in \text{ICEXT}(\text{IS}(z_1)), \text{ICEXT}(\text{IS}(z_1)) \subseteq \text{ICEXT}(\text{IS}(z_2)), \dots, \text{ICEXT}(\text{IS}(z_n)) \subseteq \text{ICEXT}(\text{IS}(y))$ are always true in every m , where z_i is a category page, "[[z₁]]" is on x in D ; then we can extend D by applying the rules 4 and 6 until no more annotations can be added.
- Condition 5-9 can be handled similarly.

Recursively adding annotations to D by the entailment rules until a closure of annotation is reached, it must be the case that D contains every possible semantic annotation that can be inferred from W , and that D contains all annotations in W' (i.e., $W' \subseteq D$), otherwise W cannot entail W' . \square

Another desirable characteristic of the entailment system is that it is highly tractable, as shown with the following theorem on its complexity:

Theorem 2 : *Entailment and strong entailment problems of SMW-ML are NL-complete with respect to the number of annotations.*

Proof sketch: We recall that NL (Nondeterministic Logarithmic-space) is a complexity class that can be solved by a nondeterministic Turing machine using a logarithmic amount of memory space. We prove by a mutual reduction between the (strong) entailment problem of SMW-ML and the reachability problem in directed graph, the later is known to be NL-complete [27].

It is easy to prove that the entailment problem of SMW-ML is NL-hard, since we can encode a directed graph as a SMW system with each node in the graph being mapped to a category page, and each edge being mapped to a category inclusion. Thus, the reachability problem of the directed graph can be reduced to the entailment problem of the corresponding wiki. Thus, the strong entailment problem is also NL-hard.

To show that entailment and strong entailment problems of SMW-ML are in NL, we reduce them into the reachability problem of directed graph. For simplicity of discussion, we only discuss binary properties, the case for n-ary properties can be easily extended from the following discussion. We first prove the case for SMW-ML entailment. We construct two graphs $G1$ and $G2$ as follows:

1. Nodes of $G1$ contains all category pages and pages that are instances of some categories. If a page x has the annotation "[[y]]", where y is a category page, create an edge from x to y .
2. Nodes of $G2$ contains i) all property pages; ii) the pairs $\langle x,y \rangle$ if the page x contains "[[P::y]]" for some property y . If a property page x has the annota-

tion “[Subproperty of:y]”, where y is a property page, create an edge from x to y ; if the page x contains “[P::y]”, create an edge from $\langle x,y \rangle$ to P .

It is easy to see that entailment in SMW-ML can be answered by reachability queries in the constructed graph. For example, to find whether an individual is an instance of a category, it is sufficient to query that if the node of the category can be reachable from the node of the individual in G_1 . Since the number of annotations in the SMW-ML system is the same as the number of edges in the union of G_1 and G_2 , the NL complexity upper bound can be obtained.

For strong entailment, we add the following nodes and edges to G_2 : 1) if node $n = \langle x,y \rangle$ is in G_2 , then add edges from $F(x)$ to n and from $F(y)$ to n to G_2 , where F is a function such that $F(x)=x$ if x is not a property page, otherwise $F(x)=x'$ where x' is an alias name for x ; 2) if G_1 has an edge from x to y , both x and y are category pages, and y is in G_2 , then add an edge from x to y in G_2 . It is easy to verify that a SMW system entails “[P::y]” on page x iff P is reachable from x and from y in G_2 . Combine with the result from the discussion for entailment, the strong entailment problem is also solvable in NL.

Since the entailment and strong entailment problems of SMW-ML are both NL-hard and in NL, they are NL-complete. \square

It should be noted the RDFS entailment is NP-complete in general and is P-complete if the graph to be entailed is a ground graph [7]. Since NL is contained in P, the entailment problem of SMW-ML is potentially easier to handle than inference in RDFS. This shows a clear advantage of modeling using SMW.

The set of entailment rules can be implemented without using external tools. Here we list two such possible solutions:

- **Rule extensions and querying.** Rule extensions allow us to add annotations based on existing annotations on a page; querying allows us to fetch annotations from more than one page. By combining their power, we can easily implement the above entailment system as a set of rules. More details will be given in section 4.2.

- **Client-side bots.** MediaWiki allows certain users to use automatic tools (bots) to perform some batch editing work from on the client side. As the entailment rules we proposed only require text comparing and text editing, a client-side bot could be used for their implementation.

While the entailment rules we present are sound and complete, an implementation of these rules will not necessarily be complete (i.e., the annotations generated do not need to be a deductive closure of the wiki). Since a SMW system is usually highly dynamic, the computing of its deductive closure may cause high burden for the server. Users may selectively enable entailment inference on a subset of wiki pages for the best balance of performance and inference gain.

3 SMW Query Language

In this section, we present the syntax and semantics of the the query language of SMW 1.4.2, which will henceforth be denoted as SMW-QL.

3.1 Syntax

SMW provides the “ask” query language that allows users to query annotations in a SMW system.

Example 3 One may want to find all Graduate Students affiliated with RPI, and list their hobbies. This can be formulated as an “ask” query, such as:

```
{#{ask:
  [[Category:Graduate Student]]
  [[Has affiliation::RPI]]
  |?Hobby
}}
```

In the query, `[[Category:Graduate Student]]` and `[[Has affiliation::RPI]]` describe the selecting conditions for the query, and `|?Hobby` describes information that should be fetched. By default, the result set will be returned as a table, with the first column being the page names that match the selecting condition, e.g.,

<u>Page</u>	<u>Hobby</u>
John	Comedy
Jane	Reading, Movie
James	Travel

A partial specification of the SMW-QL syntax has been given in [14]. In this subsection, we complement [14] with a complete SMW-QL syntax in Backus-Naur Form (BNF)¹², as shown in Table 2. As the focus of this paper is on the semantics of the query language, some non-formal features (“Parameters”) of SMW-QL are not covered in the syntax given in Table 2. Some examples include sorting, limiting, formatting of query results (e.g., labels used, table headers, and so on). Complete details of the parameters and supported datatypes can be found in the SMW online manual¹³.

There are several notable features of SMW-QL that deserve further explanation:

¹² We use single quoted character sequence, e.g., ‘abc’, to represent a string; “?” means zero or one occurrence of the preceding element (possibly be enclosed with “(” and “)”, same in the following); “*” means zero or more occurrence of the preceding element; “+” means one or more occurrence of the preceding element; “[” means choice; strings in the form “**text**” are intentionally informal description; text after ‘//’ is comment.

¹³ http://semantic-mediawiki.org/wiki/Help:User_manual

Query	::= <code>{{#ask: Selectors Output? }}</code>
Selectors	::= Selector ('OR' Selector)*
Selector	::= (<code>[</code> AtomicSelector <code>]</code>) +
AtomicSelector	::= NonClassPage ClassPage Property <code>::</code> Conditions
NonClassPage	::= NCNameSpace PageName
NCNameSpace	::= <code>' '</code> <code>' '</code> (a wiki namespace string that is not 'Category' or 'Concept') <code>' '</code>
PageName	::= a wiki page name string without name space
ClassPage	::= ('Category' 'Concept') <code>' '</code> PageName
Property	::= PropertyPage (<code>' '</code> PropertyPage)* // PropertyPage is binary
PropertyPage	::= PageName // where 'Property':PageName is a property page name
Conditions	::= Condition (<code>[</code> Condition)*
Condition	::= PageValue SubQuery
PageValue	::= (<code>' '</code> <code><</code> <code>' '</code> <code>></code> <code>' '</code> <code>~</code>)? Atom
SubQuery	::= <code>'<q></code> Selectors <code>'</q></code>
Atom	::= PageName Literal Variable
Literal	::= datatype values as defined by SMW (may be n-ary) // see footnote ??
Variable	::= a MediaWiki variable, e.g., <code>{{PAGENAME}}</code>
Output	::= (<code>[</code> OutputItem)*
OutputItem	::= <code>'?</code> PropertyPage Parameter
Parameter	::= result control in the form of key=value, details omitted

Table 2. SMW-QL Syntax

Selection. “OR” between selectors have a disjunctive meaning, while the coexistence of atomic selectors have the conjunctive reading. Thus, the selectors of a query are in a Conjunctive Normal Form (CNF), e.g., $(A \wedge B) \vee (C \wedge D)$, where A, B, C, D are atomic selectors. However, as conditions in the property atomic selectors may include another subquery or a disjunction of subqueries, the complete selection conditions could be rather complex.

Subquery. Please note that subquerying in SMW-QL can be used to distinguish the weak and the strong model semantics. For example, the query condition:

```
[[Has topic::<q>[[Category:Wiki]]</q>]]
```

will return all pages that have the topic in one of the subclasses of “Category:Wiki”, e.g., “Category:Semantic Wiki”. This conforms to the strong model semantics. On the other hand, the query condition:

```
[[Has topic::Category:Wiki]]
```

will only return all pages that has the topic “Category:Wiki” but not its subclasses. This conforms to the weak model semantics.

Concept. A concept page (with the namespace “Concept”) is one that is defined using the syntax `{{#concept: Selectors }}`, using the same syntax of selectors as given in Table 2. A concept page is in fact a stored view of the wiki, and proper utilization of concepts can lead to very powerful, yet also more resource demanding, applications. More details will be discussed in Section 4.3.

3.2 Semantics

Comparison to DL Queries, SQL and SPARQL

DL Queries. The authors of [14] have proposed a semantics of SMW-QL by translating it into DL queries, thus adopting a DL semantics. However, the DL semantics requires an open world assumption (OWQ) which may not be suitable for a wiki-based knowledge base. In fact, query answering in SMW is restricted to the facts that are “known” in the wiki. For example, if we query about individuals who have *no* affiliation to RPI using the query:

```
{{#ask:
  [[Has affiliation::!RPI]]
}}
```

Under OWA we will have an empty result set since it is *unknown* that any individual in the wiki must have *no* affiliation to RPI stated elsewhere. It should also be noted that, since the DL semantics requires non-unique name assumption, even if we have stated in the wiki that John has an affiliation to MIT, we can not preclude the possibility that RPI and MIT are the same thing, thus John still can’t be included in the result set.

The DL semantics also mismatches the current implementation of SMW in which the SMW-QL query engine in fact behaves with a closed world assumptions. It uses and only uses knowledge present in a SMW site and by default assumes all wiki pages are different (unique name assumption). Thus, we believe

a more suitable semantics for SMW-QL is a closed world semantics, as the one we will introduce based on logic programs.

SQL and SPARQL. One may also be tempted to explain an SMW-QL query by translating it into corresponding queries for relational databases (using SQL) or RDF data (SPARQL).

The SQL explanation may seem reasonable since the current implementation of query execution of SMW-QL is based on a reduction to relational queries. However, such an approach may not be able to address several important design rationale of SMW-QL:

- SMW-QL is intended to be a highly-efficient and easy-to-use language. It achieves these goals at the cost of some features in SQL, such as join operations and integrity checking¹⁴.

- SMW-QL is a query language for SMW knowledge bases. Thus, a query engine of SMW-QL should do inference as specified by the SMW-ML semantics. Likewise, SQL is a query language for relational databases, thus a SQL query engine does not need to perform inference.

- The basic data unit processed in SMW-QL is a list of wiki pages. These wiki page lists can be generated and consumed by SMW-QL operations; on the other hand, the basic data unit processed by SQL is a relational table.

The other possibility, involving SPARQL-based explanation of SMW-QL, is also not satisfactory. As we have shown in the last section, the semantics of RDF and SMW-ML do not always coincide. It should also be noted that by default SPARQL only performs graph pattern matching, but not inference, why in SMW-QL query answering inference is always required.

Thus, a formal processing of SMW-QL would require a semantics of its own. In particular, this semantics would need to: (i) precisely capture the design rationale of SMW-QL, (ii) be expressive enough to cover features of SMW-QL, and (iii) embody the correct semantic assumptions made by SMW-QL (e.g., the handling of incomplete knowledge).

A LP Semantics for SMW-QL We now give a semantics that satisfies the above requirements by translating SMW-QL into function-free positive logic programs - obtaining a minimal Herbrand model semantics in the process. Here, we omit the discussion about literal values and their comparators '<' (smaller or equal), '>' (larger or equal), and '~' (string pattern matching).

Recall that a positive *logic program* (LP) [20] contains a set of rules in the form:

$$H \text{ :- } B_1, \dots, B_n;$$

where H is the head of the rule, the formula after :- is the body of the rule, B_i is a positive atom, which can be either an atom expression of the form $P(x_1, \dots, x_n)$ (where P is a n -ary predicate), an equality of the form $x=y$, or an inequality of the form $x!=y$

¹⁴ For some cases it may still be possible to do integrity checking in SMW using extensions beyond SMW-QL.

Condition C	a	$\Rightarrow C^T(x) :- x=a;$ where a is a page
	!a	$\Rightarrow C^T(x) :- x!=a;$ where a is a page
	$\langle q \rangle Q \langle /q \rangle$	$\Rightarrow C^T(x) :- Q^T(x);$ where Q is a Selectors
Conditions N	$C_1 \dots C_n$	$\Rightarrow N^T(x) :- C_1^T(x);$ where C_i is a Condition
	\dots	\dots
	$N^T(x) :- C_n^T(x);$	
Property P	P_1, \dots, P_n	$\Rightarrow P^T(x, y) :- P_1(x, z_1), \dots, P_n(z_n, y);$
		where P_i is a property page (no namespace)
AtomicSelector A	$[[a]]$	$\Rightarrow A^T(x) :- x=a;$ where a is a non-class page
	$[[D]]$	$\Rightarrow A^T(x) :- D(x);$ where D is a class page
	$[[P::N]]$	$\Rightarrow A^T(x) :- P^T(x, y), N^T(y);$
		where P is a property, N is a unary Conditions
	$[[P::b_1^1; \dots; b_n^1]]$	$\Rightarrow A^T(x) :- P(x, b_1^1, \dots, b_n^1);$
	$\dots $	\dots where P is a (n+1)-ary property, $n \geq 2$, b_j^i is a page
	$b_1^m; \dots; b_n^m]]$	$A^T(x) :- P(x, b_1^m, \dots, b_n^m);$
Selector S	$A_1 \dots A_n$	$\Rightarrow S^T(x) :- A_1^T(x), \dots, A_n^T(x);$ where A_i is an AtomicSelector
Selectors L	S_1 OR \dots OR S_n	$\Rightarrow L^T(x) :- S_1^T(x);$ where A_i is a Selector
	\dots	\dots
	$L^T(x) :- S_n^T(x);$	
Query Q	$\{\{ \#ask: L$	$\Rightarrow Q^T(x, y_1; \dots; y_n) :- L^T(x), P_1(x, y_1), \dots, P_n(x, y_n);$
	$\{?P_1$	where L is a Selectors , P_i is a property page (no namespace)
	\dots	
	$\{?P_n \}$	

Table 3. Translation from SMW-QL to Logic Program

The translation T from a SMW-QL query to a function-free, positive LP is given in the Table 3. The translation will be done recursively by mapping component blocks of a SMW-QL query into a set of LP rules. The superscript T denotes the mapping from a query block to a corresponding rule head. We also assume that all new predicates generated during the translation (including the final query head) have not been used as wiki page names. For example, consider the query in Example 3, a translation will be:

```

Q(x,y) :- L(x), Hobby(x,y);
L(x) :- A1(x), A2(x);
A1(x) :- Graduate_Student(x);
A2(x) :- Has_affiliation(x,RPI);

```

Special attention should be paid to the mapping of subqueries, disjunctive selectors and negations, as shown in the following example.

Example 4 Consider the query

```

{#{ask:
  [[Category:Paper]]
  [[Has topic::<q>[[Category:Wiki]]</q>
    || Category:Wiki]]
  [[Has author::!John]]
}}
```

(Find all papers that have the topic about “Wiki” or its subclasses, and that its authors contain someone other than John)

Its LP translation is:

```

Q(x) :- L(x);
L(x) :- A1(x), A2(x), A3(x);
A1(x) :- Paper(x);
A2(x) :- Has_topic(x,y), N1(y);
A3(x) :- Has_author(x,y), N2(y);
N1(x) :- Wiki(x);
N1(x) :- x=Category:Wiki;
N2(x) :- x!=John;

```

It should be noted that a disjunction will be mapped to multiple rules. Also note that “Category:Wiki” is used with two different roles: the one in the subquery will be semantically explained and mapped to a predicate “Wiki(x)”, while the one not in the subquery will only be explained as a constant (a wiki page’s name).

“[[Has author::!John]]” in the query is notable for the following reasons:

- A paper has to have an author to be returned by the query;
- It adopts the closed world semantics, that is, if a paper does not say that it has an author “John”, then we assume that John is not its author.

Thus, if there are four pages:

- Paper1: “[[Category:Paper]][[Has topic::Category:Wiki]]”

- Paper2: “[Category:Paper][[Has topic::Category:Wiki][[Has author::John]]”
- Paper3: “[[Category:Paper][[Has topic::Category:Semantic Wiki][[Has author::Jane]]”
- Category:Semantic Wiki: “[[Category:Wiki]]”

Only “Paper3” will be in the query result set. □

To evaluate a translated query in LP, we also need to translate the knowledge base in the SMW system (in SMW-ML) into its LP representation. We assume that all occurrences of redirected pages are globally replaced by the corresponding “hub” pages they are redirected to, thus we do not need to map equality annotations¹⁵. The mapping has the following properties¹⁶:

- If a page b contains “[[C]]” where C is a category page, add a ground rule “C(b);”;
- If a page b contains “[[P::a₁; ... ;a_n]]” (n≥1), where P is a (n+1)-ary property, then add a ground rule “P(b, a₁, ... ,a_n);”;
- If a category page C contains “[[D]]” where D is another category page, add a rule “D(x) :- C(x);”;
- If an n-ary property page P contains “[[Subproperty of::Q]]” where Q is another n-ary property page, then add a rule “Q(x₁, ... ,x_n) :- P(x₁, ... ,x_n);”;
- The translation of a page in the Concept namespace is done as if the page is a query.

It should be noted that this translation conforms to the model theoretical semantics for SMW-ML as we described in the previous section.

Finally, we give a formal semantics of SMW-QL based on the *minimal Herbrand model semantics* [9] of LP. Here we remind readers the definition of minimal Herbrand model semantics:

Definition 2 *Let L be a function-free positive logic program. The grounding of L, gr(L), is the union of all possible ground instantiations of P, obtained by replacing each variable in a rule with a constant occurring in L.*

The Herbrand Base HB(L) of L is the set of all ground atoms of the form p(a₁, ... ,a_n) where p is an n-ary predicate and a_i a constant. A subset of HB(L) is a Herbrand Interpretation of L.

A Herbrand Model M of L is a Herbrand interpretation such that for every rule in gr(L)

$$H :- B_1, \dots, B_n$$

if $B_1, \dots, B_n \in M$, then $H \in M$. M is minimal if no proper subset of it is a Herbrand model of L.

Informally, one may regard a minimal Herbrand model of a logic program (LP) as a minimal subset of its atomic formulas (e.g., C(b) where b is a constant

¹⁵ Note that equality in LP is only syntactical, thus does not capture the right semantics of equality in SMW-ML

¹⁶ Note that a category page b may be mapped to both a constant b and a predicate b(x), their roles are different which will be clear from their contexts. Thus the generated rules are not higher-order.

and C is a predicate), that satisfies constraints in all rules of the logic program. Please refer to [9] for more details.

Now we can define the result set of a SMW-QL query using the minimal Herbrand model semantics:

Definition 3 *Let W be a SMW system in SMW-ML, Q a query in SMW-QL, W' and Q' the LP translation of W and Q , respectively, the head of Q' is $q(x_1, \dots, x_n)$, and L be the union of W' and Q' . A result set of Q with respect to W is a set $\{(x_1, \dots, x_n) | q(x_1, \dots, x_n) \in M\}$, where M is a minimal Herbrand model of L .*

In fact, as a positive logic program always has a unique minimal Herbrand model [9], a SMW-QL query will always have a unique result set.

The current implementation of SMW contains a query engine for SMW-QL, on top of a relational database. This implementation conforms to the semantics given in Definition 3. However, it is incomplete in several aspects - for instance, its treatment of page equality is limited (currently not transitive). Having a precisely defined semantics of SMW-QL, future work could involve investigating other implementations of SMW-QL that are, ideally, sound and complete. Possible approaches could include using other databases systems, or a highly optimized logic program solver (such as Cmodels¹⁷ or DLV¹⁸).

Implementation: We have implemented the presented translation from SMW-QL to logic programs in php. The source code is available from ¹⁹.

3.3 Complexity

A result of the formalization of SMW-QL is that its data complexity, i.e., the complexity of query answering when the query and the terminological annotations (e.g., subclass and subproperty relations) are fixed but the set of facts is variable, is P-complete, therefore it is more difficult than the SMW-ML entailment problem.

Theorem 3 : *Query answering in SMW-QL is P-complete.*

Proof sketch: P membership is because the translated programs from a SMW can be easily transformed into a datalog and datalog in general is P-complete for data complexity [6]. Note that logic programs translated from a SMW-QL query is always a datalog, and that the head of all rules are part of the intensional database; for the rules translated from SMW-ML, for each category C , we introduce a new category C' , rename every ground fact $C(x)$ to be $C'(x)$, and add a rule $C(x) :- C'(x)$; similar renaming will be performed for every properties

¹⁷ <http://www.cs.utexas.edu/tag/cmodels/>

¹⁸ <http://www.dbai.tuwien.ac.at/proj/dlv/>.

¹⁹ <http://tw.rpi.edu/wiki/SMW-QL>

too. Such a renaming ensures that every extensional predicate does not appear in the head of a rule, as required by datalog.

The P-hardness part of the proof is obtained by a reduction from the circuit value problem (CVP), which is known to be P-complete [16]. Recall that the CVP problem is that given an encoding a of a boolean circuit a , inputs x_1, \dots, x_n and a designated output y , to decide if y of a is true. To encode CVP in SMW-QL, we first create two pages “1” and “0” and a property “not” such that “1” contains “[not::0]” and “0” contains “[not::1]”; if an input $x_i = 1$, we add “[Category:x_i]” to “1”, otherwise to “0”. Now we will simulate a circuit using SMW-QL queries.

For each boolean gate, its input-output relation can be simulated by a concept page. For a “not” gate $G := \text{not } P$, we create a concept page for G with content `{#{concept: [not::<q>[[P]]</q>}}}`.

For an “and” and “or” gate with inputs x and y , we use 6 basic queries to construct a query representing the logical relationship. Each of the basic type returns a value on a single combination of inputs and blank for other cases:

- Q1: `<q>[[x]] [[y]]</q><q>[[1]]</q>` - is “1” for $x=y=1$
- Q3: `<q>[[x]] [[not::y]]</q><q>[[1]]</q>` - is “1” for $x=1, y=0$
- Q4: `<q>[[not::x]] [[y]]</q><q>[[0]]</q>` - is “0” for $x=1, y=0$
- Q5: `<q>[[not::x]] [[y]]</q><q>[[1]]</q>` - is “1” for $x=0, y=1$
- Q6: `<q>[[x]] [[not::y]]</q><q>[[0]]</q>` - is “0” for $x=0, y=1$
- Q8: `<q>[[x]] [[y]]</q><q>[[0]]</q>` - is “0” for $x=y=0$

An “and” gate is simulated by a concept page with content

`{#{concept: Q1 OR Q4 OR Q6 OR Q8}}`

and an “or” gate is simulated by a concept page with content

`{#{concept: Q1 OR Q3 OR Q5 OR Q8}}`.

Note that we cannot directly use `<q>[[x]] [[y]]</q>` or `<q>[[x]] OR [[y]]</q>` because they may generate blank or multiple values, thus break value requirement for boolean gates.

By recursively encoding all gates using concept pages, the output value y can be obtained by the result value of a query `{#{ask: [[Concept:y]]}}`, where “Concept:y” is the concept page for y . The size of the query is constant, and the number of wiki pages is the size of the circuit plus two (for “1” and “0”). Thus CVP can be reduced to query answering in SMW-QL. Hence query answering in SMW-QL is P-hard.

Because query answering in SMW-QL is both in P and P-hard, it is P-Complete. \square

The Theorem 3 is a surprising one as query answering in SMW is even more difficult than entailment (assuming $L \neq P$). It is also more difficult than answering FOL queries, hence also than answering queries expressed in relational algebra. Both FOL queries and relational queries can be evaluated in L with respect to data complexity [23]. Thus, Theorem 3 reveals that SMW-QL is a very expressive query language.

In fact, SMW-QL introduces several features that are not expressible in SMW-ML, such as concept, qualified existential restrictions and subqueries. These features may contributed to the increased complexity. If we disallow the use of subqueries, we can obtain a logarithmic space query language.

Theorem 4 : *Query answering in SMW-QL is in L under the restriction that for every atomic selector of the form $[[P::N]]$, N is either “a” or “!a” where a is a page.*

Proof sketch: We prove by a reduction to FOL queries. Note that each subquery-free query in SMW-QL is in the form of

```
{#{ask:
  A11...A1n OR ... OR Am1 ... Amk
}}
```

where each A_{ij} is an atomic selector of the form $[[a]]$, $[[C]]$, $[[P::a]]$ or $[[P::!a]]$, where a is a page, C is a category and P is a property. Their FOL translations are $x=a$, $C(x)$, $P(x,a)$ and $P(x,y)\wedge y\neq a$, respectively. The entire query thus can be translated into a disjunctive normal form in FOL.

Subclass and subproperties relations in the SMW system can be encoded in the FOL query following a similar approach as adopted by DL-Lite [4]. Informally, if $C(x)$ appears in an atomic selector, it will be replaced by $C(x)\vee D_1(x),\dots,\vee D_n(x)$, for every $D_i \in \text{sub}(C)$.

Since FOL queries have the data complexity L -complete [23], query answering in the restricted SMW-QL is in L. □

Theorem 4 shows that performance of query answering in SMW-QL may be improved if subqueries are not used.

4 Discussion

In this section we discuss several critical problems that can be studied given the semantics of SMW-ML and SMW-QL we described. We further show that the formal study of semantic wiki languages benefits us by providing either initial solutions or promising future directions for many of these problems.

4.1 Limitations of SMW-ML

While SMW-ML has been proven a successful language, evidenced by the broad adoption of SMW, it is not without its limitations. Having a precisely defined syntax and semantics can help us overcome many of these limitations.

Expressivity. An apparent limitation is about its expressivity. SMW-ML only corresponds to a small subset of RDF and OWL. For example, several frequently needed features, like transitive properties, inverse properties and class

negations, are currently not supported in SMW-ML. We believe that our analysis on its semantics will help us identify a more expressive subset of OWL that is both relatively simple for wiki-based applications, and highly scalable for handling a large set of data. One potential candidate is a subset or variant of the OWL 2 RL profile [11], which has an RDF-based semantics²⁰ and is suitable for rule-based implementations, thus it is close to SMW-ML in the semantics we just present.

Page-centric modeling. Annotations in SMW-ML describe properties of *wiki pages*. Therefore, to make some knowledge statement about a subject, one is required to first create a wiki page for that subject. For example, to create a technical report system using SMW-ML, one has to create a page for each paper in the system. This is often a tedious process and may discourage users. Using SMW-ML, it is also hard to make knowledge statements about an external URL, or about an RDF blank node, which is too restrictive for some applications. This restriction may be overcome by extending the SMW-ML syntax to allow external URLs or local variables be directed used in a triple.

Context-dependence. SMW-ML is a context-dependent language, that is, a knowledge statement can not be freely asserted everywhere, but has to be located in a specific place. For example, a class inclusion relationship can only be asserted at the subclass's page but not the superclass' page. Similarly, a triple must be asserted at its subject's page. This often requires users, when editing a SMW site, to visit many pages and make frequent jumps between pages. One possible reason for such a restriction may be originated from the need of tracking the provenance of a triple when the wiki is updated. This may be improved with an extension to the SMW-ML syntax and a new provenance tracking mechanism such that a knowledge statement can be asserted context-independent. This is left as a future work.

4.2 Entailment Inference via Rule+Query

Using parser functions²¹ and semantic querying, SMW also supports simple rule representation. For example, we can define an "Uncle Rule", such that if page X contains "[[sonOf::Y]]" and page Y contains "[[brotherOf::Z]]" then we add to Z the annotation "[[uncleOf::X]]"²².

We can implement SMW-ML entailment rules by using the native tools in SMW: conditional parser functions (if...then...else), semantic querying, and variable definitions. Here we show an example of the encoding of the entailment rule 3 (simplified for demonstration purposes), defined itself as a page Rule:Entailment in the wiki system:

²⁰ It should be noted that OWL 2 RL is not a subset of OWL 2 DL and has no direct DL-based semantics.

²¹ <http://meta.wikimedia.org/wiki/ParserFunctions>

²² http://tw.rpi.edu/wiki/Family_Rule. This corresponds to the well-known role composing inference that is beyond OWL-DL.

```

{{#vardefine:a|{{#ask:
  [[:{{FULLPAGENAME}}]]
  |?Category=
  |link=none}}
}}
{{#if:{{#var:a}}|
  {{#vardefine:b|{{#ask: [[:{{#var:a}}]]
    |?Category=
    |link=none }}
  }}
  {{#if: {{#var:b}}|[[{{#var:b}}]]}}
}}

```

Complete details of the encoding are online²³. To trigger the rule, a page only needs to include it using the syntax “`{{Rule:Entailment}}`”. Thus, it is possible to switch on or off the entailment rules on individual pages, or a group of pages using a template-based strategy. In general, this approach will render sound, but not necessarily complete, inference results.

Our approach is also capable of supporting other typical rule inferences. We mention here a few important ones:

- Extensional reasoning, e.g., to check if `Category:Under-graduate` is a subclass of `Category:Person`
- Intensional reasoning, e.g., to check if all instances of `Category:Student` are instances of `Category:Fans`
- Domain inference, e.g., to infer that if a page X has an annotation `[[Has author::Y]]`, then X has the annotation “`[[Category:Paper]]`”.

4.3 “Concept” Pages as View

As we have mentioned earlier, a “Concept” page is a query-based view of the wiki knowledge. For example, one may define a `ComedyFan` to be a person with the hobby of comedy, by making a page `Concept:ComedyFan` with the content:

```

{{#concept:
  [[Category:Person]]
  [[Has hobby::<q>[[Category:Comedy]]</q>]]
}}

```

This definition can then be translated into LP rules (with some simplification):

```

ComedyFan(x) :- Person(x), P(x);
P(x) :- Has_hobby (x,y), Comedy(y) ;

```

Having a LP-based semantics, we can investigate several interesting usage scenarios for concept pages. It should be noted that none of these have been

²³ <http://tw.rpi.edu/portal/rule:Entailment>

formally studied yet, due to the lack of a formal semantics of concept pages.

Cyclic View. As a concept page can be used in a query, it is possible to define two concept pages where each is dependent on the other. From the LP-based semantics, it is safe to have cyclic definitions in rules, such as in the example below:

```
P(x) :- Q(x),R(x);
Q(x) :- P(x),S(x);
```

Thus, cyclic view, from the semantic point of view, would be safe on SMW.

Concept-Category Inclusion. It is syntactically legal to declare a concept to be a subclass of a category. For example, we may declare on the concept `Concept:ComedyFan` the annotation “`[[Category:Fans]]`” (every comedy fan is a fan). In LP-based semantics, this becomes

```
Fans(x) :- ComedyFan(x);
```

However, the current SMW-QL implementation is incomplete with regard to this usage. It may be remedied by either extending the current RDB implementation, or by resorting to a LP solver.

Concept Containment Suppose we have two concept pages:

```
Concept:Baby: {{#concept: [[has age::<3]]}}
Concept:Child: {{#concept: [[has age::<18]]}}
```

Can we infer the fact that every instance of `Concept:Baby` is an instance of `Concept:Child`? This type of inference is the wiki counterpart of the *query containment* problem in relational databases [13]. As with relational databases, the main application of concept containment is in query optimization. We leave its detailed investigation as future work.

4.4 Closed World vs. Open World Querying

The minimal Herbrand model semantics of SMW-QL implies that queries in SMW inherit the closed world assumption (CWA) from LP. Thus wiki queries, instead of being against the “open world” of the whole web, are only run against the information in the wiki itself. However, there are many usage scenarios that call for open world assumption (OWA) in semantic wiki. Some scenarios also require features of CWA that are currently not directly supported by SMW-QL. Below are some motivating examples.

Example 5 (Strong Negation, OWA) In a group wiki, John and Jane mention on their profile page “Movie” as a hobby, while James does not. However, when answering the query “find people whose hobbies don’t include Movie” under the open world assumption, James may not be in the answer list, since he may simply have forgotten to list hobbies on his profile page.

Example 6 (Class Negation, OWA) One may want to query about papers which have an author that is *not* an instance of “Student”.

Example 7 (Scoped Negation, CWA) Suppose a group meeting page lists its attendees including John and Jane. We also know that the group has exactly three members: John, Jane, and James. For the query “who in the group did *not* attend the meeting”, the expected answer could be “James”, with the implicit closed world assumption that the group meeting’s attendee list and the group membership list are complete.

Example 8 (Negation as Failure, CWA) One may want to query about papers where John is not an author. It could be the case that a paper has some known authors but John is not one of them, or that the paper has no known authors. In both cases, we assume the missing information means that there is no such an authorship relation. Note that “[Has author::!John]” does not capture this requirement.

Those scenarios above call for extensions to SMW-QL that can be easily addressed in our semantic framework . In particular, they require the use of both classical negation (“¬”) and negation as failure (“not”) in the query. Recall that both “¬” and “not” are allowed in the head and body of LP rules, thus we may extend SMW-ML and SMW-QL to include the two types of negations. The corresponding semantics, thus, will be generalized from the minimal Herbrand model semantics towards the *coherent stable model semantics* as used in answer set programming [10].

5 Related Work

Semantic wikis have gained significant attention in recent years. Some early examples of implementations include Platypus Wiki [5], WikiOnt [2] and Rise [8]. Some of the more recent work is compiled in the SemWiki workshops [25, 18, 19]. For a short survey of semantic wiki systems, please see the survey section of [3]. Here we only mention a few recent work that is closely related to this paper, in addition to the Semantic MediaWiki [24] framework we reviewed.

Note that our focus is on using wikis to represent formal knowledge, thus we will not be addressing work on using wikis to discover knowledge, such as DBpedia²⁴. We also do not consider the non-formal aspects of semantic wikis, like user interface design.

Rhizome [22] support ZML, a script language for creating semantic content, and RxML, an alternative language to RDF for content display. ZML can be translated into RDF. Rhizome does not have a query language of its own. **AceWiki** [15] allows knowledge representation in Attempto Controlled English (ACE) through an interactive editor. ACE texts can be unambiguously translated into a syntactic variant of first-order logic, and it is also possible to bidirectionally translate a subset of ACE into and from OWL DL.

Some other semantic wiki systems use “standard” semantic web languages for knowledge representation and querying. **IkeWiki** [21] and its derivation

²⁴ <http://dbpedia.org>

SWiM [17] use RDF and OWL for storing and reasoning with formal knowledge. **SweetWiki** [3] can use RDFa, RDFS or OWL-Lite for knowledge representation and uses SPARQL for query answering. **OntoWiki** [1] also uses RDF as the modeling language, with a SPARQL querying shell. The advantage of this approach is that those languages are familiar to the Semantic Web community, and many tools are available for processing knowledge modeled by those standard languages. The semantics of those languages is well-understood thus does not need special attention.

However, to Web end users in general, and to the wiki community in particular, understanding and using a KR language like ACE, or semantic web languages like RDF, OWL or SPARQL, is not always an easy task. For such communities, the adoption cost could be significantly reduced if, as with SMW, the changes are only minimal and incremental from systems that have already been widely accepted, e.g., Wikipedia. This factor may be among the leading reasons for the fast growing of the SMW user community²⁵.

6 Conclusion

Semantic wikis provide a low-cost, user friendly means for light-weight modeling and querying of knowledge on the semantic web. However, despite their growing popularity, little attention has been paid to the formal aspects of semantic wikis. This includes the syntax and semantics of their modeling and querying languages, the potentials and limitations of various systems, and useful design patterns for expressivity extensions or concept modeling.

This paper presents our initial results on the formal investigation of semantic wiki knowledge representation and querying with a special attention to the Semantic MediaWiki (SMW). In particular, we defined a model theoretical semantics for the SMW-ML language, and a closed world, minimal Herbrand model semantics for the SMW-QL language by a reduction to positive logic programs. We discovered that entailment inference in SMW is a NL-complete problem, thus established a formal proof for the tractability of SMW systems. We also presented a sound and complete entailment system for SMW that can be natively implemented without external tool support. We also show that query answering in SMW is P-complete in general and in L if subqueries are disallowed. In the light of the presented semantics, we also discussed several critical problems in semantic wikis, e.g., view definition, closed/open world assumption, and rule inferences. While our discussion is on SMW, we believe the analysis methodology and many of our results (e.g., entailment rule inference) can be applied to other semantic wiki systems as well.

Some future directions for our work have been given in the discussion section. Other ongoing work includes: 1) the investigation of open questions in semantic wiki languages, e.g., studying their complexity; 2) query answering in semantic

²⁵ Please note that, while RDF is not the modeling language of SMW, it can still be used as its exchange language for exposing the knowledge in the wiki to other semantic web tools, or to importing an ontology in RDF into the wiki.

wikis based on LP solvers; 3) design patterns in semantic wikis, e.g., relational modeling and object-oriented modeling; and 4) investigating the role of templates in semantic wiki modeling.

7 Acknowledgments

We thank Fangkai Yang, Yaron Koren, Denny Vrandečić and Markus Krötzsch for helpful discussion about the paper, and James Michaelis for proof reading and editing the paper.

This work is partially supported by NSF #0524481, DARPA #FA8650-06-C-7605, #FA8750-07-D-0185, #55-002001, #F30602-00-2-0579, and ITA project W911NF-06-3-0001.

References

1. S. Auer, S. Dietzold, and T. Riechert. Ontowiki - a tool for social, semantic collaboration. In *International Semantic Web Conference*, pages 736–749, 2006.
2. J. Bao and V. Honavar. Collaborative ontology building with wiki@nt - a multi-agent based ontology building environment. In *ISWC 2004 Workshop on Evaluation of Ontology-based Tools (EON)*, pages 37–46, 2004.
3. M. Buffa, F. L. Gandon, G. Ereteo, P. Sander, and C. Faron. Sweetwiki: A semantic wiki. *J. Web Sem.*, 6(1):84–97, 2008.
4. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data Complexity of Query Answering in Description Logics. In *KR*, pages 260–270, 2006.
5. S. E. Campanini, P. Castagna, and R. Tazzoli. Platypus wiki: a semantic wiki web. In *1st Italian Semantic Web Workshop*.
6. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. In *IEEE Conference on Computational Complexity*, pages 82–101, 1997.
7. J. de Bruijn and S. Heymans. Logical foundations of (e)rdf(s): Complexity and reasoning. In *ISWC/ASWC*, pages 86–99, 2007.
8. B. Decker, E. Ras, J. Rech, B. Klein, and C. Hoecht. Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis. *Proceedings of the Workshop on Semantic Web Enabled Software Engineering (SWESE), held at the 4th International Semantic Web Conference (ISWC 2005) November 6th-10th, 2005*.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080, 1988.
10. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.
11. B. C. Grau, B. Motik, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. World Wide Web Consortium (W3C) Candidate Recommendation, 2009.
12. P. Hayes, editor. *RDF Semantics*. W3C Recommendation. World Wide Web Consortium, February 2004.

13. P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington*, pages 205–213. ACM Press, 1998.
14. M. Krötzsch, D. Vrandečić, M. Völkel, H. Haller, and R. Studer. Semantic wikipedia. *J. Web Sem.*, 5(4):251–261, 2007.
15. T. Kuhn. AceWiki: Collaborative Ontology Management in Controlled Natural Language. In *Proceedings of the 3rd Semantic Wiki Workshop*. CEUR Workshop Proceedings, 2008.
16. R. E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.
17. C. Lange. Towards scientific collaboration in a semantic wiki. In *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*, pages 119–126, 2007.
18. C. Lange, S. Schaffert, H. Skaf-Molli, and M. Volkel, editors. *Proceedings of the 3rd Semantic Wiki Workshop (SemWiki 2008) at the 5th European Semantic Web Conference (ESWC 2008), Tenerife, Spain, June 2nd, 2008*, volume 360 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
19. C. Lange, S. Schaffert, H. Skaf-Molli, and M. Volkel, editors. *Proceedings of the 4th Semantic Wiki Workshop (SemWiki 2009) at the 6th European Semantic Web Conference (ESWC 2009)*, volume 464 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
20. V. Lifschitz. Foundations of logic programming. pages 69–127, 1996.
21. S. Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *WETICE*, pages 388–396, 2006.
22. A. Souzis. Bringing the wiki-way to the semantic web with rhizome. In *SemWiki*, 2006.
23. M. Y. Vardi. The complexity of relational query languages (extended abstract). In *STOC*, pages 137–146, 1982.
24. M. Volkel, M. Krötzsch, D. Vrandečić, H. Haller, and R. Studer. Semantic wikipedia. In *WWW*, pages 585–594, 2006.
25. M. Volkel and S. Schaffert, editors. *SemWiki2006, First Workshop on Semantic Wikis - From Wiki to Semantics, Proceedings, co-located with the ESWC2006, Budva, Montenegro, June 12, 2006*, volume 206 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
26. D. Vrandečić and M. Krötzsch. Reusing ontological background knowledge in semantic wikis. In *SemWiki*, 2006.
27. A. Wigderson. The complexity of graph connectivity. In *MFCS*, pages 112–132, 1992.