

Normalization of relations and ontologies

LULE AHMEDI

Department of Computer Engineering
University of Prishtina
Kodra e diellit pn, 10000 Prishtinë
REPUBLIC OF KOSOVA
lule.ahmedi@fiek.uni-pr.edu

EDMOND JAJAGA

Department of Computer Science
South East European University
Ilindenska pn, 1200 Tetove
FYR MACEDONIA
ej16374@seeu.edu.mk

Abstract: - Although few systems for normalization of relations are already in place to support schema refinement, they are rarely used by database practitioners, or as a teaching aid at universities. Meanwhile, the Semantic Web potential for novice implementations understood by both humans and machines Web-wide has just recently urged the need to reinterpret systems that are yet in the mainstream of standalone or traditional Web systems. That has motivated us to consider the design and implementation of a database normalization system as integral part of the machine-understandable knowledge base on the Web, as conceived by the Semantic Web. This paper presents the ontology layer of our normalization system, as well as the initial findings in building the rule layer of this system using Semantic Web technologies.

Key-Words: - normalization of relations, ontologies, rules in Semantic Web, OWL/SWRL, logic programming, Prolog.

1 Introduction

In practice, a critical point in providing a robust database solution is its level of optimization which may in the first place be ensured through a well-defined design. Our work considers refinement of the database design given a set of relation schemas and functional dependencies (FDs) holding over them at the input. Few systems for normalization of relations are already in place [5; 6; 7; 8; 9] to support schema refinement, although rarely used by database practitioners, or as a teaching aid at universities.

Meanwhile, the Semantic Web potential for novice implementations understood by both humans and machines Web-wide has just recently urged the need to reinterpret systems that are yet in the mainstream of standalone or traditional Web systems. That has motivated us to investigate the use of Semantic Web technologies in developing a database normalization system, thus aligning-well with the idea of Tim Berners-Lee [12] for integrating as much data and algorithms as possible into a machine-understandable knowledge base on the Web.

We present here the kernel of our normalization system consisting of the ontology layer and some enabling algorithms for normalizing relations, like finding the attribute closure. Further, the initial findings in using Semantic Web towards completing the rule layer of our normalization system are listed.

The paper is organized as follows: Section 2 outlines related work; the ontology layer of our

system and issues regarding the structuring of data in lists and n-ary predicates are treated in Section 3; Section 4 introduces the kernel of our rule layer, and reveals the main challenges we are facing in covering all algorithms of the normalization theory in our system.

2 Related work

A number of systems for normalization of relations in languages like Prolog [5] and Mathematica [6] have already been developed in order to ease the deployment of the theory of normalization. NORMIT [7] is a Web-enabled tutor for database normalization. Few other works exist as well which have addressed the same theory [8; 9; 13].

Observing the development of Semantic Web rule systems like the Semantic Web Rule Language (SWRL) [14] which is a prototype rule language for the future Web and build heavily upon the Description Logic (DL), and moreover, due to the intersection of DL with Logic Programming (LP), we have decided to examine the Prolog normalization system developed by Ceri and Gottlob [5] and draw mappings between rules in Prolog [5] and SWRL when concerning the rule layer of our Semantic Web normalization system.

ProWLog [15] and SWORIER [16] are two hybrid approaches which laid Prolog on top of Web Ontology Language (OWL) [17; 18], thus addressing the issue of capturing open-world semantics of OWL into Prolog. The SWORIER

team translates rules of SWRL and RuleML [19; 20] into Prolog prior to reasoning. If rules were found by SWORIER not expressible in any of SWRL or RuleML, they represented them straight in Prolog. During the OWL-into-Prolog translation, solutions were provided [16] to problems also encountered in the work of Volz et al. in 2003 [21; 22] like: negation, complementary classes, disjunctive heads, open world assumption, enumerated classes, and equivalent individuals. These issues are our concern as well, but from another perspective, i.e. the Prolog-into-SWRL translation.

3 The Ontology Layer

We developed an ontology in OWL to encode the theory of normalization of relations in Semantic Web. Following we describe classes and object properties defined in our ontology, as well as their meaning in terms of the normalization theory.

Class Relation models relations of a database schema. A property `has_schema` assigns an instance of class Schema to a relation. Thus, `has_schema` has Relation as its domain and Schema as its range. The Schema individuals are restricted to allow only values of range Attribute through `has_attr` property. The `has_attr` property lists all attributes which constitute (1) the schema of a given relation schema if its domain is the class Schema, or (2) the left-hand side or right-hand side of a given FD if its domain is one of the classes LHS or RHS respectively. Classes LHS and RHS are subclasses of class Side capturing both sides of a FD - LHS for the left-hand side and RHS for the right-hand side. Since `has_attr` has as its domain the Side class, by inference classes LHS and RHS are related with Attribute class through this property. A class named FD (cf. Fig.1) models functional dependencies that hold over a given relation, which is captured through an existential quantifier over `holds_over` property to contain some values of the Relation class. A property relating class FD (the domain value) is `has_side` which has two subproperties: `has_lhs` and `has_rhs`, which are functional, infer all definitions given above for the `has_side` property.




-  (has_lhs **some** LHS) and (has_lhs **exactly** 1)
-  (has_rhs **some** RHS) and (has_rhs **exactly** 1)
-  holds_over **some** Relation

Fig.1 A set of necessary restrictions for the FD class defined in Protégé

Classes `in3nf` and `inbcnf` are both subclasses of the Relation class, and are meant to classify relations which are in third normal form (3NF), or

in Boyce-Codd normal form (BCNF), respectively.

To capture the semantics of computing a closure of a given set of FDs for a given relation a class named `AttrClosure` is defined. Its individuals link a set of attributes with its attribute closure (attribute set) through properties `clo_attr` and `closure` respectively. It is the responsibility of the rule layer of our ontology to calculate the instances of this property, as will be introduced in the next section.

The running example we will use throughout this paper consists of a relation schema and a set F of FDs as follows:

$\text{rel}(A, B, C, D, E, F)$

$F = \{AB \rightarrow C, C \rightarrow A, D \rightarrow E, DE \rightarrow F, E \rightarrow D, E \rightarrow F\}$

The same instance expressed in Prolog [5] looks as follows:

`schema(rel, [a, b, c, d, e, f]).`

`fd(rel,[a,b],[c]). fd(rel,[c],[a]). fd(rel,[d],[e]).`

`fd(rel,[d,e],[f]). fd(rel,[e],[d]). fd(rel,[e],[f]).`

whereas its representation in our normalization ontology is depicted in Figure 2.

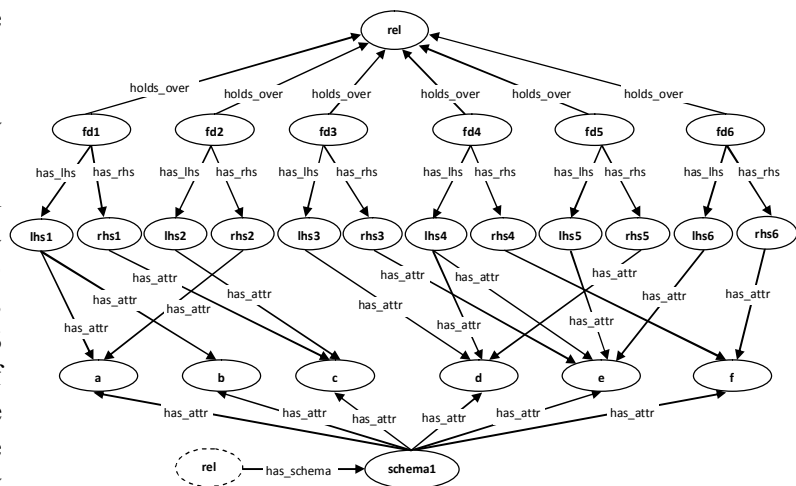


Fig.2 The running example represented in our normalization system

3.1 N-ary Relations in our Ontology

In the Prolog normalization framework [5], attributes represented through the Attribute set of values of the LHS and Schema classes on the `has_attr` property constitute an ordered sequence in order to achieve efficiency in Prolog, but the order of attributes within the Attribute sequence has certainly no semantic meaning. The order of attributes in our ontology within the Attribute sequence be it for the LHS class or the Schema class, does also not matter, which is fully in conformance with the semantics of the relational database model.

4 The Rule Layer

To reason over the ontology layer of our normalization system described in the previous section we have considered two approaches: hybrid and a pure Semantic Web approach.

For the *ontology layer*, following the pure Semantic Web approach simply use the ontology described above, while regarding the hybrid approach it is required a translation into Prolog, which is not complex since both OWL and Prolog base on the same subset of logic (Horn Logic) [16].

Regarding the *rule layer* of our normalization system, the hybrid approach would simply require to adopt Prolog rules available in [5]. According to the second approach we have expressed Prolog rules with SWRL, some of which will be introduced in the next section.

Although the hybrid approach promises to require less efforts since there are already theories defined for the OWL-into-Prolog translation in general, we merely tend to introduce a rather normalization system solely with Semantic Web technologies at both layers. We will in the next section describe a set of SWRL rules which constitute the core of our normalization rule layer, as well as in Section 4.2 list some initial findings towards building a complete normalization system following always the *pure Semantic Web approach*.

4.1 SWRL Rules in our Normalization System

In the theory of normalization of relations, the algorithm of finding the *closure of a set of attributes* presents the main building block of all other algorithms, like that of finding all keys of a relation, or of decomposing a relation into 3NF using Bernstein's algorithm.

The attribute closure of a given set X of attributes with respect of a set of FDs is implemented in Prolog with the following couple of rules:

```
closure(REL,X,CLO_OF_X):- fd(REL,LHS,RHS),
                           subset(LHS,X),
                           not subset(RHS,X),
                           union(W, X,RHS,REL),!,
                           closure(REL,W,CLO_OF_X).
closure(REL,X,CLO_OF_X) :- CLO_OF_X = X.
```

Consider the relation instance *rel* and a set of FDs as provided in our example. If we pose a query for finding the closure CLO_OF_X of the attribute set $[a,d,e]$ to the Prolog normalization system:

```
?- closure(rel,[a,d,e],CLO_OF_X).
```

the result returned will be:

```
CLO_OF_X = [a,d,e,f]. (1)
```

In our normalization system, the algorithm for finding the closure of a set of attributes is implemented through two SWRL rules as given in Fig.4.

```
1 AttrClosure(?clo)^clo_attr(?clo,?attrs) → closure(?clo,?attrs)
2 AttrClosure(?clo)^closure(?clo,?attrs)^sqwrl:makeBag(?sk,?attrs)^
3 has_lhs(?fd,?lhs) ^ has_attr(?lhs,?at) ^ sqwrl:makeBag(?sl,?at)^
4 has_rhs(?fd,?rhs) ^ has_attr(?rhs,?bt) sqwrl:makeBag(?sr,?bt) ^
5 sqwrl:groupBy(?sk,?clo,?fd) ^
6 sqwrl:groupBy(?sl,?clo,?fd) ^
7 sqwrl:groupBy(?sr,?clo,?fd) ^
8 sqwrl:contains(?sk,?sl) ^ sqwrl:notContains(?sk,?sr) ^
9 sqwrl:union(?u,?sr,?sk) ^ sqwrl:element(?k,?u) →
10 closure(?clo,?k)
```

Fig.4 The SWRL implementation of the attribute closure algorithm

Both rules evaluate once for each instance $?clo$ of the *AttrClosure* class which owns two properties:

- the *clo_attr* property which holds the set of input attributes (see 1 in Fig.4), and
- the *closure* property which yields the set of attributes constituting the closure (see *CLO_OF_X* in Prolog rules) of attributes given in *clo_attr*.

The first rule (line 1) initializes the closure set to the set of input attributes for which the closure should be computed. In the second rule, we use three attribute sets: $?sk$ consists of the set of the currently computed attribute closure (line 2) initially set equal to the set of input attributes (first rule), $?sl$ collection consists of LHS attributes of the current FD (line 3), and $?sr$ consists of RHS attributes of the current FD (line 4). Once we have constructed collections, we apply the *groupBy* built-in operator of the *SQWRL* library [23] which constitutes groups for each (closure, FD) pair on each of the three collections (lines 5-7). Groups created enable that we run solely the second rule once per each closure to be computed, but recursively (in a loop) over all dependencies since each FD requires the currently computed closure as its input. On each group (see the *'^'* operator for performing over a group) we test whether the current dependencies' LHS is a subset of the currently computed attribute closure $?sk$, and whether its RHS is not a subset of that same $?sk$ collection (line 8). If these two built-in subgoals contains and notContains of *SQWRL* succeed, we then build the union $?u$ of the RHS attributes with the actual attribute closure collection, and retrieve all elements of that union's result collection $?u$ through the built-in *sqwrl:element* clause (line 9).

If we compute the attribute closure for the same input data as in the running example, this time in our normalization system, we will gain the same result

set (cf. Fig.5) as (1).

We have tested the correctness of the attribute closure implementation in our normalization system through a set of experiments summarized in the following table:

Test no.	Number of attributes within the relation schema	Number of FDs	Number of closure inputs	Number of OWL axioms exported to Jess	Number of axioms inferred (Jess Rule engine)	Ceri and Gottlob's system vs. our system results
Test1	6	5	5	45	16	equal
Test2	5	5	5	45	21	equal
Test3	6	6	5	49	19	equal
Test4	7	8	6	64	24	equal
Test5	9	10	6	83	17	equal

The chart of Fig.6 illustrates the complexity distribution among five tests run in our system.

4.2 Challenges in Building the Rule Layer of our Normalization System

Following are some of the challenges encountered while building the rule layer of our system which have also been identified by other researchers when investigating the correspondences between DL and LP [21; 22], or are due to Semantic Web rule systems being yet under development [14]. We next propose alternative approaches to addressing these concerns which are evident in the Semantic Web.

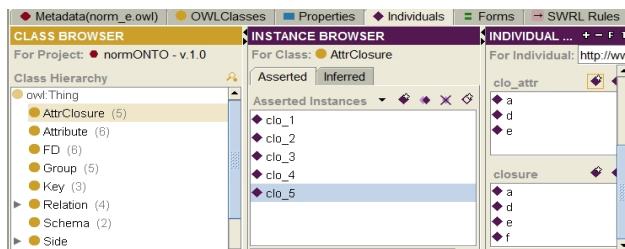


Fig.5 The individual clo_5 holding the closure adef of the ade attribute set

4.2.1 Open-World Assumption: Enumeration

SWRL together with OWL shares the open world assumption, while Prolog embraces the closed world assumption, thus returning false when failing to satisfy the goal. Because of the OWA rules that attempt to enumerate individuals or property values in an ontology are possible only when OWL states those numbers explicitly.

When decomposing a relation into BCNF with the Prolog system [5], if a subgoal $X=[_,_]$ succeeds the relation contains two attributes and it is classified as a relation in BCNF. The same test is not expressible in SWRL [24], but to perform close world enumeration we provided the following alternative using the size operator to compute the number of attributes in a given relation schema,

using the Jess rule engine [10] for reasoning:

```
Relation(?r)^has_schema(?r,?s)^has_attr(?s,?attr)°
sqwrl:makeSet(?ss,?attr)^sqwrl:groupBy(?ss,?r)°
sqwrl:size(?n,?ss)^swrlb:lessThan(?n,3)→inbcnf(?r)
```

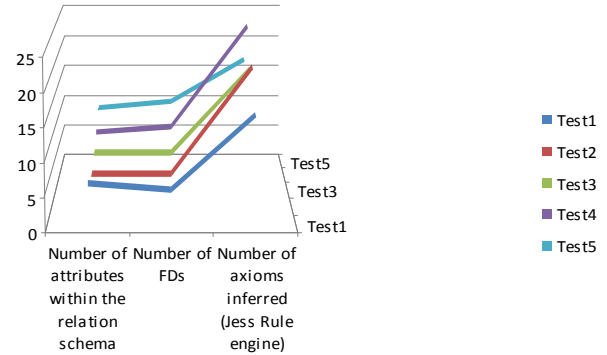


Fig.6 The chart view of the evaluation of the rule closure

4.2.2 Unique Name Assumption

OWL's open world semantics does not allow one to assume that two individuals are automatically distinct if they have different names, i.e., OWL does not have a unique name assumption (UNA). Additionally, due to the normal rule pattern matching, two variables can also match the same individual in a rule. SWRL supports UNA, thus extending OWL capabilities in this direction. SWRL supports the sameAs, differentFrom and allDifferent clauses to determine if individuals refer to the same underlying individual or are distinct. In Prolog, UNA is enabled with operators not and equal (=).

If we wish to capture the semantics that two attributes A and B of a relation are different to each other, we can write $\text{not } A=B$ in Prolog, whereas in SWRL the same is expressed through $A \text{ owl:differentFrom } B$. In our system, we rather state that all individuals of the Attribute class are distinct to each other by using a single owl:allDifferent annotation in OWL.

4.2.3 Nonmonotonicity: Fact Assertion, Modification and Retraction

Like OWL, SWRL supports monotonic inference only. Hence, SWRL rules cannot modify or retract information in an ontology [24]. Asserting new facts to OWL using SWRL is allowed as long as that implies only adding new individuals, no way of retracting any of existing ones.

In the Prolog system [5] facts: $\text{fd}/3$, $\text{inbcnf}/1$, $\text{in3nf}/1$, $\text{key}/2$, $\text{clo}/3$, etc. are asserted and retracted from the database dynamically as needed. For example, in the second step of the Bernstein's

algorithm for decomposing relations to a 3NF, when partitioning the set of FDs into groups with identical LHS, a new fact group(REL, LHS) is asserted in the base of facts. In the third step, groups with equivalent keys are merged, which implies that both group facts are retracted from the base of facts and a new group fact is asserted consisting of both keys. This is not allowed in SWRL.

The SWORIER team [16] has developed an extension module to their system which is able to assimilate dynamic changes that are provided at run time, including adding new facts, or removing facts. A similar workaround may be adopted for our system to support the modification and retraction of facts dynamically as needed.

4.2.4 Nonmonotonicity: Negation as Failure

Obviously there is no support for negation as failure (NAF) in Semantic Web. Translating Prolog's NAF into SWRL is among main issues addressed when developing our system.

A typical example of NAF in the Prolog system [5] is the clause `not subset(RHS,X)` of the attribute closure rule mentioned in the previous section. To "close the world" in SWRL we have arranged members of both sets RHS and X into SQWRL collections `makeSet` or `makeBag` (cf. Fig.4).

Another rationale would lead to deploying the recently available OWL 2 construct for asserting negative facts about an individual [4]. This is usually costly since it involves asserting explicitly all known negative facts to a database: in the above example, the "not subset" relationship for each pair of possible combination of attributes in sets.

4.2.5 Nonmonotonicity: Classical Negation

While SWRL does not support negated atoms or NAF, classical negation is possible in OWL/SWRL through the use of the `owl:complementOf` class description in OWL or SWRL [24].

An OWL `complementOf` axiom which states that, if a key is not a member of the class `KnownKey`, then it should be classified as a member of the class `NonKey`, is asserted in our system:

`NonKey owl:complementOf KnownKey`

Of course, with OWL's (and SWRL's) open world assumption, this conclusion can only be reached for individuals for which it may definitely be concluded that they cannot be members of the class `KnownKey`. A SWRL rule which reasons over complementary classes `KnownKey` and `NonKey` may be written as follows:

```
KnownKey(?x)^tbox:isComplementOf(?y,?x)
-> NonKey(?y)
```

4.2.6 Recursion

Recursion is not directly supported in SWRL since we cannot use results of rules when reasoning over a set of rules. Since we use Jess to reason over SWRL rules, the recursion is supported enabling thus the use of rule results at any level of recursion.

In order to determine a closure of a set of attributes we must consider every input FD. We cannot find the closure through one rule which will loop over all dependencies, since each FD requires the currently computed closure as its input. Thus we are forced to compute the same rule once per each FD until all FDs are exhausted, and every FD will eventually contribute its RHS if certain conditions are fulfilled (cf. Fig. 4).

4.2.7 Disjunction and Alternatives

When translating Prolog rules for normalization of relations into DL, we do not have the problem of disjunction in the head, since every rule is Horn-like.

An example Prolog rule consisting of alternative atoms is applied when finding a minimal cover for a set of FDs [5]. The rule named `elimredundfdfs` is employed for eliminating redundant dependencies, if their RHS is a subset of their LHS closure. The FD is retracted from the base of facts depending on the success of the given alternative clauses.

Before translating this rule into SWRL, we simply rewrite it into two rules with equivalent heads [21; 3] `elimredundfdfs`, eliminating thus the need for explicitly expressing alternatives in SWRL.

5 Conclusion and Future Work

In this research we have explored the ability of the Semantic Web technologies to support the development of a system for normalization of relations.

ProWLog and SWORIER [16][15] have shown that a hybrid approach is quite a natural and fruitful step. We rather introduced a novel approach in building a pure Semantic Web system for normalization of relations led in the first place by the work conducted by Ceri and Gottlob [5] and the Semantic Web vision for the future Web.

A translation bridge from LP into DL may well have been of use when developing our system targeted to rely solely on the Semantic Web technologies which have their foundations in DL.

Our work lays some initial findings in mapping between these two distinct logic languages, i.e., the SWRL into Prolog mapping. There are companies and researchers who have translated RDF and OWL into Prolog as described in [21; 16; 15]. Thea [3] is an example which supports translation of a rather restricted form of Prolog (unary and binary) into SWRL.

We believe in the first place that the development of our system will be useful for understanding normalization algorithms, and their applicability in solving day to day database design problems. In addition, we hope this work will encourage further integration of existing desktop and traditional Web applications into Semantic Web, hence making the data, like corporate data and hidden Web relational databases, and the Semantic Web applications understood by machines supplement each other.

References:

- [1] M. Marchiori, Introduction to the Special Issue on Logic Programming and the Web, *TPLP*, Vol.8, Nr.3, 2008, pp.247-248
- [2] M. Marchiori, Towards a People's Web: Metalog, Web Intelligence, *IEEE Computer Society*, 2004, pp. 320-326
- [3] V. Vassilades, J. Wielemaker & C. Mungall, Processing OWL2 ontologies using Thea: An application of logic programming, *OWLED, CEUR Workshop Proceedings*, Vol.529, 2009
- [4] C. Golbreich, E. K. Wallace & P. F. Patel-Schneider, *OWL 2 Web Ontology Language New Features and Rationale, W3C Recommendation*, Retrieved from <http://www.w3.org/TR/owl2-new-features/>, 2009
- [5] C. Stefano & G. Georg, Normalization of relations and Prolog, *Communications of the ACM*, Vol.29, No.6, 1986, pp. 524-544
- [6] A. Yazici & Z. Karakaya, JMathNorm: A Database Normalization Tool Using Mathematica, *ICCS (2), Lecture Notes in Computer Science*, Vol.4488, 2007, pp. 186-193
- [7] A. Mitrovic, NORMIT: A Web-Enabled Tutor for Database Normalization, *International Conference on Computers in Education (ICCE)*, 2002, pp. 1276-1280
- [8] M. Bouzeghoub, G. Gardarin & E. Metais, Database design tools: An expert system approach, *11th VLDB*, 1985, pp. 82-95
- [9] L. Kerschberg, Expert database systems, *The 1st International Conference on Expert Database Systems*, 1984
- [10] *Jess: the rule engine for the Java platform*, (n.d.), Retrieved from <http://www.jessrules.com/>, 2010
- [11] B. Parsia, E. Sirin, B. C. Grau, E. Ruckhaus D. Hewlett, Cautiously approaching SWRL, *Preprint submitted to Elsevier Science*, 2005
- [12] T. Berners-Lee, J. Hendler & O. Lassila, The Semantic Web, *Scientific American*, Vol.284, Nr.5, 2001, pp. 34-43
- [13] R. Fagin, Functional dependencies in a relational database and propositional logic. *IBM J. Res. Dev.*, Vol.21, Nr.6, 1977, pp. 534-544
- [14] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Groszof, M. Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleM, W3C Member Submission*, Retrieved from <http://www.w3.org/Submission/SWRL/>, 2004
- [15] T. Matzner, & P. Hitzler, Any-World Access to OWL from Prolog, *Lecture Notes in Computer Science*, Vol.4667, 2007, pp. 84-98
- [16] K. Samuel, I. L. Obrst, S. Stoutenburg, K. Fox, P. Franklin, A. Johnson, Translating OWL and semantic web rules into prolog: Moving toward description logic programs, *TPLP*, Vol.8, Nr.3, 2008, pp. 301-322
- [17] M. K. Smith, C. Welty & D. L. McGuinness, *OWL Web Ontology Language Guide*, Retrieved from <http://www.w3.org/TR/owl-guide/>, 2004
- [18] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, *OWL Web Ontology Language Reference, W3C Recommendation*, Retrieved from <http://www.w3.org/TR/owl-ref/>, 2004
- [19] H. Boley & S. Tabet, *Rule Markup Language*, Retrieved from www.dfki.uni-kl.de/ruleml, 2001
- [20] H. Boley, S. Tabet & G. Wagner, Design rationale of RuleML: A markup language for semantic Web rules, *The International Semantic Web Working Symposium*, 2001
- [21] R. Volz, S. Decker, & D. Oberle, *Bubo - Implementing OWL in rule-based systems*, Retrieved from <http://www.daml.org/listarchive/joint-committee/att-1254/01-bubo.pdf>, 2003
- [22] R. Volz, *Web Ontology Reasoning with Logic Databases*, PhD Thesis, AIFB, University of Karlsruhe, 2004
- [23] *CollectionsSQWRL*, (n.d.), Retrieved from <http://protege.cim3.net/cgi-bin/wiki.pl?CollectionsSQWRL>, 2010
- [24] M. O'Connor, *SWRLLanguageFAQ*, Retrieved from <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ>, 2010