A database normalization tool using Semantic Web technologies

Lule Ahmedi and Edmond Jajaga

Abstract — The Semantic Web technologies have just recently urged the need to touch and reinterpret many application areas. On the other side, there are currently few systems for normalization of relations within a database, which are also rarely used, either by database designers, or as a teaching aid at universities.

This paper introduces a system for normalization of relations as integral part of the machine-understandable knowledge base on the Web, as conceived by the Semantic Web. We have adopted the semantics for the ontology layer of our normalization system and made some findings regarding the rule layer of our system. The main challenges appear at the rule layer, since there is not a single rule system which satisfies all of our needs. The solutions are provided in different rule systems, mainly on the Semantic Web Rule Language, for issues like: knowledge base modifications, negation, open world assumption, and disjunction.

Keywords — Logic programming, Normalization of relations, Ontologies, Prolog, Rules in Semantic Web.

I. Introduction

It has been estimated that more than 80 percent of all computer programming is database-related [1]. Moreover, studies has shown [2] that the vast majority of the content in the WWW resides in the Deep Web sources which store their content in backend databases and are ever growing.

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. The main concern in the database design is avoiding data redundancy. FDs provide useful information for avoiding data redundancy and data manipulation anomalies. To limit the complexity of our research, we do not consider other important classes of integrity constraints like multi-valued dependencies, or join dependencies, which sometimes reveal redundancies that cannot be detected using FDs alone [3]. Dependencies other than FDs may however be added in our system incrementally at any later stage to reflect their impact in design decisions. A measure of the redundancy within a relation is called normal form, a concept introduced in the early 70s by Codd [4]. A relation has to fulfill the required conditions in order to be in a particular normal form. A mechanism, named decomposition, will eventually be applied over a relation if it is not in a required normal form, thus replacing it with smaller relations. Few systems for normalization of relations are already in place [5]-[11] to support schema refinement, although rarely used be it 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 [13] for integrating as much data and algorithms as possible into a machine-understandable knowledge base on the Web.

In this paper, we present 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 technologies towards completing the rule layer of our normalization system are listed, accompanied always with case studies drawn by our system.

The paper is organized as follows: Section 2 outlines related work focusing on those with more commonalities to our approach; 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 in details the main challenges we are facing in covering all algorithms of the normalization theory in our system.

II. Related Work and Our Approach

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 which is otherwise complex to apply and hence avoided by most practitioners and students at universities. NORMIT [7] is a Web-enabled tutor for database normalization. Few other works exist as well which have addressed the same theory [8]-[11], [14].

Observing the development of Semantic Web rule systems like the Semantic Web Rule Language (SWRL) [15] which is a prototype rule language for the future Web and build heavily upon the Description Logic, and moreover, due to the intersection of Description Logics (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 [16] and SWORIER [17] are two hybrid approaches which combine the Web Ontology Language (OWL) [19], [20] with logic programming languages like Prolog when building a Semantic Web rule system: they both laid Prolog on top of OWL, thus addressing the issue of capturing open-world semantics of OWL into Prolog. The SWORIER team translates rules of SWRL and RuleML [21], [22] into Prolog prior to reasoning. The translation is fairly straightforward due to both SWRL/RuleML and Prolog being based on the same subset of logic (Horn Clause) [17]. 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 [17] to problems also encountered in the work of Volz et al. in 2003 [23], [24] 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.

III. The Ontology Layer

We developed an ontology in OWL to encode the theory of normalization of relations in Semantic Web. Following are *classes* defined in our ontology, as well as their meaning in terms of the normalization theory:

- Class Relation: models relations of a database schema.
- Class Attribute: models attributes contained in relations of a database schema.
- Class FD (cf. Fig. 1): models functional dependencies that hold over a given relation. A restriction is defined for each instance on properties has_rhs and has_lhs of this class to be of cardinality one. Also an existential quantifier requires that each instance on property holds_over of this class should contain some values of the Relations class for which the given functional dependency is defined.
- Classes Side, RHS, and LHS: the Side class captures both sides of a functional dependency - the left-hand side through its LHS subclass, and the right-hand side through its RHS subclass.

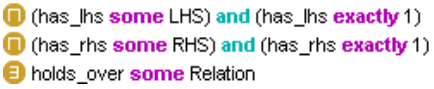

Fig. 1 a set of necessary restrictions for the FD class defined in Protégé

- Classes in3nf, and inbcnf: are both subclasses of the Relation class, and are meant to classify relations which are in third normal form, or in Boyce-Codd normal form, respectively, once the functional dependencies that hold over them are considered. It is the rule layer of our ontology which should infer the instances of these classes (if any).
- Class AttrClosure: models the attribute closure (its property closure) for a given set of attributes (the clo_attr property).

Object properties defined in our ontology are as follows:

- Property has_side, has_lhs, and has_rhs: the has_side property is defined for the FD class (the domain value). Properties has_lhs of range LHS and has_rhs of range RHS are both functional properties and subproperties of the has_side, and hence infer all definitions given above for the has_side property.
- Property holds_over: explains which functional dependency holds over which relation. It is defined for the FD class and has values of range Relation.
- Property has_schema: assigns a schema to a relation. It is defined for the Relation class, and is restricted to allow only values of range Schema.
- Property has_attr: is defined for the Side class, hence of RHS and LHS as well due to inference, as well as for the Schema class, and is restricted to allow only values of range Attribute. It 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 functional dependency if its domain is one of the classes LHS or RHS respectively.
- Properties clo_attr and closure: are meant to capture semantics of a closure over a given set of functional dependencies for a given relation. 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.

Example 1 The running example we will use throughout this paper consists of a relation schema and a set of functional dependencies as follows:

rel(A, B, C, D, E, F).
F = {AB → C, C → A, D → E, DE → F, E → D, E → 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]).
```

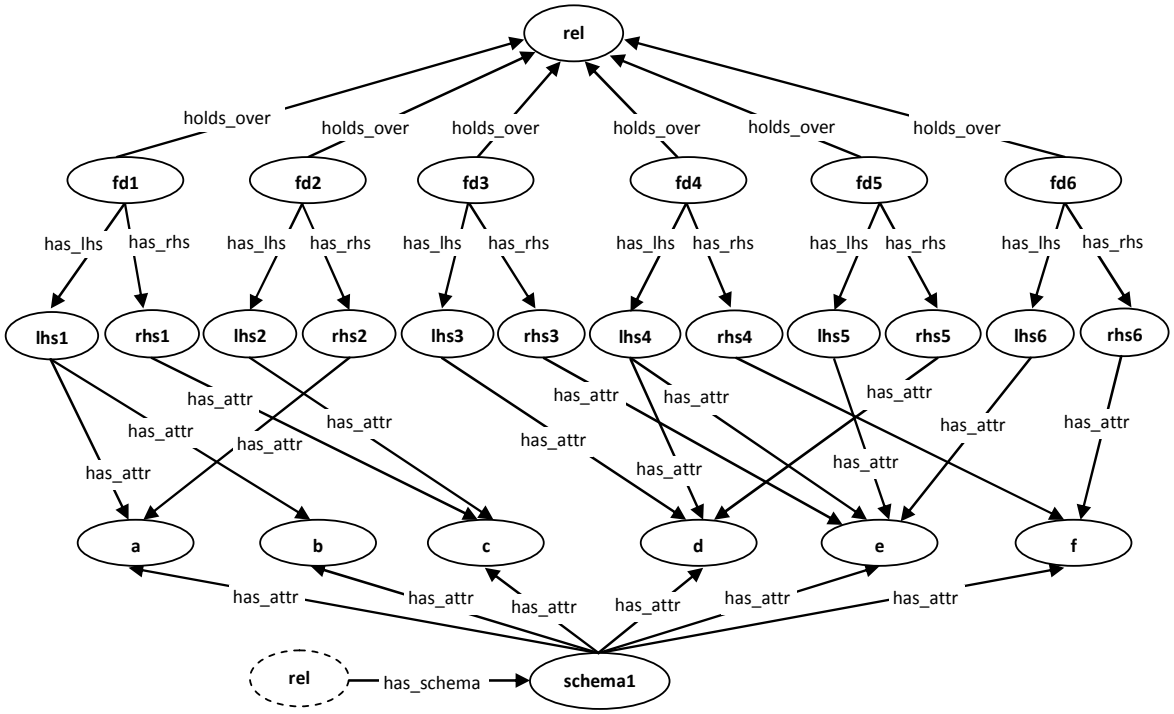

Fig. 2 The running example *Example 1* represented in our normalization system

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

A. *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, are also irrelevant in terms of semantics. There is anyway only a loose support currently in OWL to deal with ordered sequences as discussed below.

OWL supports by default the representation of binary relations through properties. For instance, the property has_rhs is a binary relation between an individual of the FD class, say fd1, and another individual, say rhs1 of the RHS class. On the other side, if following the Prolog normalization framework, the has_schema and has_attr properties would require the definition of their range to be of more complex structures like *n-ary relations*, also referred to as sequences. There are currently three alternatives in OWL for expressing sequences: RDF lists [35], OWL lists [38], and OWL n-ary relations [37]. *RDF lists* retain order of individuals in a sequence, but can be reasoned with only when applying OWL Full reasoners. *OWL lists* are well suited for representing and reasoning with individuals in a sequence but are still in their infancy level as regards popularity. *N-ary relations* on the other side provide a pattern for representing sequences which retain order and can be reasoned with as well, but are more general and cost an additional superfluous effort for dealing with their generality. Fig. 3 illustrates the use of OWL n-ary relations for representing Attribute values on the has_attr property of the LHS class as an ordered sequence.

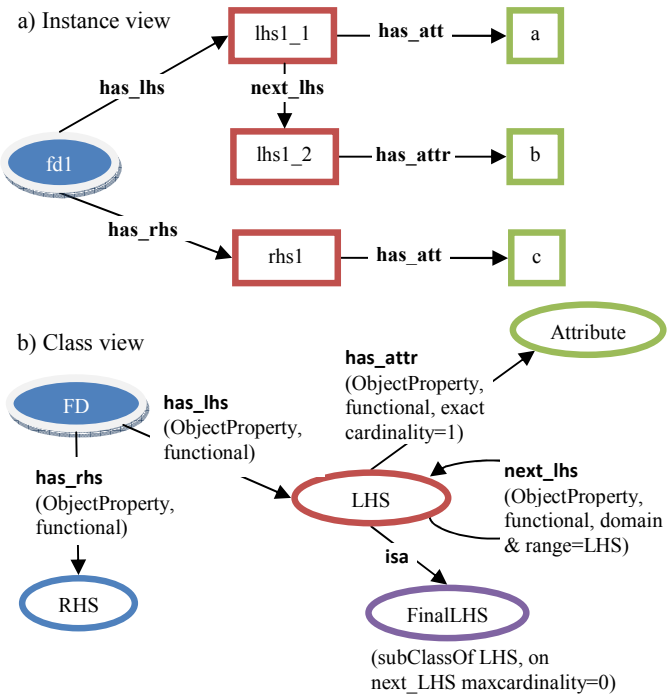

Fig. 3 n-ary representation in OWL of the Attribute sequence (a, b) given FD: ab→c.

None of the three considered representations of sequences in OWL has a fully-pledged implementation of SWRL constructs to reason with. We have hence decided to model `Attribute` instances as sets of values where the order of its members does not matter, which is fully in conformance with the semantics of the relational database model. Future considerations might cover dealing with attributes as ordered sets of values within a database schema or the left-hand side of a functional dependency.

IV. The Rule Layer

Observing the development of Semantic Web rule systems with their formal foundations on Description Logics, and having in mind the intersection that exists between Description Logics and Logic Programming, we decided to consider the Prolog system for normalization of relations developed by Ceri and Gottlob [5] as a starting point when designing our normalization system.

Hybrid approach. A straightforward approach would then be to follow the hybrid architecture of layering LP languages like Prolog on top of the OWL ontology layer as in PrOWLog [16] or SWORIER[17], i.e.:

- Ontology layer - Translate the OWL ontology layer introduced in the previous section into Prolog as advised by Volz et al. (2003) [23], [24] for the OWL-into-Prolog translation in general inclusive subtle translation issues like: negation, complementary classes, disjunctive heads, open world assumption (OWA), enumerated classes, equivalent individuals, duplicate facts, and cardinality. This translation is not complex since both OWL and Prolog base on the same subset of logic (Horn Clause) [17].
- Rule layer - Simply adopt Prolog rules available in [5] for building the rule layer of our normalization system.

A pure Semantic Web approach. Although the above approach promises to require less efforts for building our normalization system since there are already theories defined for the OWL-into-Prolog translation in general, we merely tend to introduce a rather pure Semantic Web approach which builds solely upon Semantic Web technologies at both layers:

- Ontology layer - There is no need to translate the OWL ontology into Prolog: simply use the ontology defined in the previous section.
- Rule layer - Here, instead of adopting Prolog rules, provide a Semantic Web rule system to reason over the OWL layer that is introduced in the previous section. We have hence expressed them in SWRL rules which were initially defined in Prolog. This is much like dealing with Prolog-into-SWRL translation.

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*.

A. 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 Third Normal Form (3NF) using Bernstein's algorithm.

Recall the definition of the attribute closure algorithm given a set X of attributes with respect to a group F of dependencies (cf. Fig. 4) and its implementation in Prolog as provided in [5] (cf. Fig. 5).

```
Algorithm (closure(X, F))
1  Let X be a given set of attributes over the set F of FDs
2  CLOSURE_OF_X = X;
3  repeat until there is no change: {
4       if there is a FD: LHS -> RHS in F such that
5          (LHS subset of CLOSURE_OF_X and
6           RHS not subset of CLOSURE_OF_X),
7       then set CLOSURE_OF_X = CLOSURE_OF_X U RHS
8  }
```

Fig. 4 the attribute closure algorithm

The correspondences between the Prolog implementation (Fig. 5) and the algorithm (Fig. 4) are as follows: algorithm line 2: line 6 in Prolog; algorithm line 3: the recursive call of the closure subgoal (line 5), and the cut operator '!' in Prolog; algorithm lines 4, 5, 6: the fd subgoal in line 1, line 2, and line 3 in Prolog; algorithm line 7: line 4 in Prolog.

```
1 closure(REL,X,CLOSURE_OF_X):- fd(REL,LHS,RHS),
2                               subset(LHS,X),
3                               not subset(RHS,X),
4                               union(W, X,RHS,REL),!,
5                               closure(REL,W,CLOSURE_OF_X).
6 closure(REL,X,CLOSURE_OF_X):- CLOSURE_OF_X = X.
```

Fig. 5 the Prolog implementation of the attribute closure algorithm

Example 2 Consider the relation instance rel and a set of functional dependencies as provided in Example 1. If we pose a query for finding the closure CLOSURE_OF_X of the attribute set [a,d,e] to the Prolog normalization system (cf. rules in Fig. 5):

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

the result returned will be:

```
CLOSURE_OF_X = [a,d,e,f].
```

How is this result inferred? The rule closure examines every FD of the base of facts whose LHS is a subset of the current attribute closure X=[a,d,e], and whose RHS is not a subset of X=[a,d,e]. The first FD which satisfies these two subgoals is DE → F. The built-in operator union then computes the union W of X=[a,d,e] and RHS=[f] resulting into W=[a,d,e,f], following

with the recursive call of the predicate closure with now W instead of X as the attribute set at the input. The recursive invocation of the predicate closure follows whenever there is a FD in F which makes all subgoals preceding the predicate closure in the rule body evaluate to true (the cut operator '!') .

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. 6.

```
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. 6 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. 6), and
- the closure property which yields the set of attributes constituting the closure (see CLOSURE_OF_X in Fig. 5) 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 left-hand side attributes of the current FD (line 3), and ?sr consists of right-hand side attributes of the current FD (line 4). Once we have constructed collections, we apply the groupBy built-in operator of the SQWRL library [25] which constitutes groups for each (closure, FD) pair on each of the three collections ?sk, ?sl, and ?sr (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).

Example 3 If we compute the attribute closure for the same input data as in Example 1, this time in our normalization system, we will gain the same result set (cf. Fig. 7) as in Example 2. The clo_5 instance, say, of the AttrClosure class will hold information about computed attribute closure through two properties: clo_attr which holds the information for input attribute set ade, and closure which will bind clo_5 to the computed attribute set closure adef (Fig. 7).

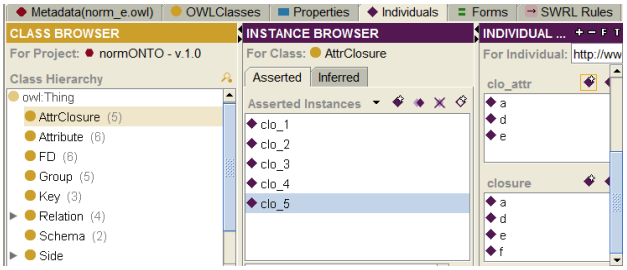

Fig. 7 the individual clo_5 holding the closure adef of the ade attribute set

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 below (cf. Fig. 8) illustrates the complexity distribution among five tests run in our system.

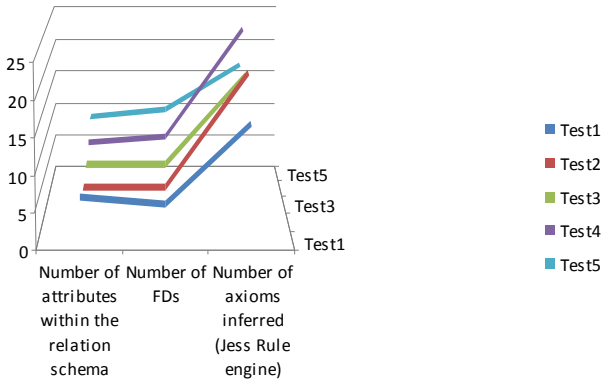

Fig. 8 the chart view of the evaluation of the rule closure

B. Rule Layer Challenges

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 [23], [24], or are due to Semantic Web rule systems being yet under development [15]. We next propose alternative approaches to addressing these concerns which are evident in the Semantic Web.

1) Open-World Assumption: Enumeration

SWRL together with OWL shares the open world assumption (OWA) [26]. This is not the case with Prolog which embraces the closed world assumption. This distinction is not surprising, since SWRL roughly belongs to the union of DL and Horn Logic [27], while Prolog is a language of Logic Programming. If Prolog fails to find some instance satisfying the goal it will simply return false.

Because of the OWA, SWRL rules that attempt to enumerate individuals or properties in an ontology are not always possible. One cannot write a rule that makes an inference based on, say, the number of individuals or property values in an ontology unless OWL statements state those numbers explicitly.

Example 4 When decomposing a relation into BCNF, the algorithm tests the relation whether it has two attributes; if it is true, then the relation is already in BCNF.

This test in Prolog has been captured by checking whether the list `X` contains two elements using the following statement [5]:

```
X = [_,_]
```

The rule fails if the list `X` has more than two elements.

The same test is not expressible in SWRL [26]. A possible approach in overcoming this shortage in expressivity of OWL is using the property cardinality restriction. Hence, a rule that classifies an individual as a BCNF relation (a relation with two attributes) finds that a given individual is a member of the Relation class, and has the (exact) cardinality two on the property has_attr as follows:

```
Relation(?r)^has_schema(?r,?s)^(has_attr=2)(?s)
→ inbcnf(?r)
```

Because of the OWL's (and SWRL's) open world assumption, this rule shall actually match individuals that may have no values for the has_attr property in the current ontology, but for which the existence of such values may be meanwhile deduced from OWL axioms. It is not possible to express this type of match in SWRL [26] unless we "close the world" as readily provided in our system since using the Jess rule engine [29] for reasoning. Another alternative SWRL rule may also perform closed world enumeration as required for classifying a relation to be a BCNF relation:

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

The above SWRL rule uses the size SQWRL built-in operator to compute the number of attributes in a given relation schema, and if that size is less than 3 (see the lessThan SWRL built-in operator), the relation is asserted as a new individual to the inbcnf class in our OWL layer.

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 (=).

Example 5 If we wish to capture the semantics that two attributes `A` and `B` of a relation are different to each other, we can write not A=B in Prolog, whereas in SWRL the same is expressed through A 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.

3) Nonmonotonicity: Fact Assertion, Modification and Retraction

Like OWL, SWRL supports monotonic inference only. Hence, SWRL rules cannot be used to modify information in an ontology. If SWRL rules allowed ontology modifications, nonmonotonicity would ensue. For this reason, it is also not possible to modify or retract information in an ontology using SWRL [26]. 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.

Example 6 In the Prolog system for normalization of relations [5], the following facts:

```
fd/3, inbcnf/1, remember/1, tnfdecomp/2, group/2, key/2,
clo/3, schema/2, fdj/3, allkeys/1, decomp/2, in3nf/1,
bcnfdecomp/2
```

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 third normal form, when partitioning the set of FDs into groups with identical left hand sides, 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 [17] 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.

An alternative approach in addressing this issue is using the latest W3C standard for rule systems, namely Rule Interchange Format (RIF) [39] or rather its production rule dialect (RIF-PRD) [40] since it is best suited for our application. RIF-PRD supports knowledge base modifications through 'Assert', 'Retract' and 'Modify' actions. The lack of an RIF-PRD

implementation, and, on the other side, the availability of several implementations for SWRL have guided us towards currently relying only on SWRL systems when designing our approach.

4) Nonmonotonicity: Negation as Failure

Another important distinctive feature between Prolog and SWRL is negation as failure (NAF). This is the consequence of SWRL's monotonicity. Translating the Prolog's NAF into SWRL is among the main issue addressed when developing our system since there is obviously no support for negation as failure in Semantic Web.

Example 7 When determining the closure of a set of attributes (Fig. 5) in Prolog, the clause not subset(RHS,X) is applied which is a typical example of the NAF [5]. In our system, we overcame the lack of support for NAF in SWRL by deploying pre-defined SQWRL predicates as follows: we "closed the world" by arranging members of both sets RHS and X into collections using the built-in predicates makeSet or makeBag, and then compare two collections through the notContains operator of SQWRL extension of SWRL (see SWRL rule in Fig. 6).

Another rationale would lead to deploying the recently available OWL 2 construct for asserting negative facts about an individual [30]. 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.

5) Nonmonotonicity: Classical Negation

While SWRL does not support negated atoms or negation as failure, classical negation is possible in OWL/SWRL through the use of the owl:complementOf class description in OWL or SWRL which states that the class extension consists of those individuals that are NOT members of the class extension of the complement class [26].

Example 8 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)

It will assert individuals to the NonKey class extension whenever there is an individual found belonging to the KnownKey class extension, and a complementOf class description asserted to hold between classes KnownKey and NonKey in our ontology.

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.

Example 9 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. 6).

7) Disjunction and Alternatives

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

In the Prolog normalization system [5], there are rules which require expressing alternatives instead of a conjunction of atoms. Prolog solves this problem with the "**;**" operator. Yet another way of representing alternatives in Prolog exists, i.e., describing each alternative in a separate clause.

Example 10 An example Prolog rule consisting of alternative atoms is applied when finding a minimal cover for a set of dependencies [5]. The rule named elimredundfds is employed for eliminating redundant dependencies, and looks as follows:

```
1 elimredundfds(REL):-
2     retract(fd(REL,LHS,RHS)),
3     closure(REL,LHS,Z),
4     (not(subset(RHS,Z)),
5         asserta(fd(REL,LHS,RHS)));
6     (subset(RHS,Z)),
7     fail.
8 elimredundfds(REL).
```

The first four clauses of this rule (lines 2-5) will succeed if a dependency's right hand side is not a subset of its left hand side closure. If this test fails, the clauses in lines 2, 3 and 6 will succeed, and the dependency will be eliminated from the base of facts since it is redundant and can thus be implied from the FD set.

Before translating this rule into SWRL, we simply rewrite it into two rules with equivalent heads [23], [31], [32] elimredundfds, eliminating thus the need for explicitly expressing alternatives in SWRL. Seems like it would take less efforts to implement this rule in RIF-PRD with the use of disjunction operator *Or*. The presentation syntax of this rule would look as in Fig. 7.

```
Prefix ( ex <http://normonto.org/ex1#>)
Prefix(pred <http://www.w3.org/2007/rif-builtin-predicate#>)
(* ex:rule_elimredundfds *)
Forall ?fd such that And( ?fd #ex1: FD
                                ?fd [ex1:has_lhs ?lhs]
                                ?fd [ex1:has_rhs ?rhs])
   (If
      AND(?lhs [ex1: has_clo List(?z)]
            External (pred:list-contains(List(?z), ?rhs)))
   )
   Then retract(?fd)
```

Fig. 7 the RIF-PRD implementation of the rule elimredundfds

V. Conclusion

In this research we have explored the ability of the Semantic Web technologies to support the development of a system for normalization of relations. Few systems for normalization of relations are already in place [5]-[11] to support schema refinement, but are rarely used.

Early examples, like the Metalog [33] system in the World Wide Web Consortium (W3C), has shown that combining logic programming and Semantic Web is a quite natural and fruitful step: and in fact, the burst of research in Semantic Web developments has eventually started to touch, connect and reinterpret many topics that were and are mainstream of the logic programming area [34].

Further, hybrid systems like SWORIER [17] and PrOWLog [16] exist which provide ontologies and a Prolog engine to reason over, as well as a study by Volz et. al. [23], [24] which elaborates the mapping from description logic languages like SWRL into logic programming languages like Prolog. One could follow such an approach, the hybrid approach of combining DL and LP languages to support ontologies and reasoning over them as is required for implementing the normalization system. 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. In [5], a system for normalization of relations by using Prolog rules has been developed.

A translation bridge from logic programming into description logics may well have been of use when developing our system targeted to rely solely on the Semantic Web technologies which have their foundations in description logics. 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 [23], [17], [16]. Thea [31] is an example which supports translation of a rather restricted form of Prolog (unary and binary) into SWRL. RIF-PRD provides a promising solution for making our normalization system complete, which as for now is limited due to no support in place in SWRL for the knowledge base modification actions such as those assumed to be provided by any RIF-PRD implementations, namely 'Assert', 'Retract' and 'Modify'.

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. The system may particularly be well suited for the design of small database applications, and also as a teaching aid. 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.

Appendix

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
<!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
<!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
<!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
<!ENTITY swrla "http://swrl.stanford.edu/ontologies/3.3/swrla.owl#" >
<!ENTITY tbox "http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#"
>
<!ENTITY abox "http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#"
>
<!ENTITY rdfb "http://swrl.stanford.edu/ontologies/built-ins/3.4/rdfb.owl#"
>
<!ENTITY swrlx "http://swrl.stanford.edu/ontologies/built-
ins/3.3/swrlx.owl#" >
<!ENTITY swrlm "http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlm.owl#" >
<!ENTITY sqwrl "http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl#" >
<!ENTITY swrlxml "http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlxml.owl#" >
<!ENTITY temporal "http://swrl.stanford.edu/ontologies/built-
ins/3.3/temporal.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/norm_e.owl#"
   xml:base="http://www.owl-ontologies.com/norm_e.owl"
   xmlns:rdfb="http://swrl.stanford.edu/ontologies/built-ins/3.4/rdfb.owl#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlx="http://swrl.stanford.edu/ontologies/built-
ins/3.3/swrlx.owl#"
   xmlns:abox="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#"
   xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:swrlm="http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlm.owl#"
   xmlns:swrlxml="http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlxml.owl#"
   xmlns:temporal="http://swrl.stanford.edu/ontologies/built-
ins/3.3/temporal.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
```

```
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:tbox="http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#">
   <owl:Ontology rdf:about="">
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/abox.owl"/>
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.4/rdfb.owl"/>
      <owl:imports   rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl"/>
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlxml.owl"/>
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.4/swrlm.owl"/>
      <owl:imports
rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/temporal.owl"/>
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/swrlx.owl"/>
      <owl:imports     rdf:resource="http://swrl.stanford.edu/ontologies/built-
ins/3.3/tbox.owl"/>
   </owl:Ontology>
   <owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Collection">
         <rdf:Description rdf:about="#schema1"/>
      </owl:distinctMembers>
   </owl:AllDifferent>
   <owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Collection"/>
   </owl:AllDifferent>
   <owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Collection">
         <rdf:Description rdf:about="#lhs1"/>
         <rdf:Description rdf:about="#lhs2"/>
         <rdf:Description rdf:about="#lhs3"/>
         <rdf:Description rdf:about="#lhs4"/>
         <rdf:Description rdf:about="#lhs5"/>
         <rdf:Description rdf:about="#lhs6"/>
      </owl:distinctMembers>
   </owl:AllDifferent>
   <owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Collection">
         <rdf:Description rdf:about="#fd1"/>
         <rdf:Description rdf:about="#fd2"/>
         <rdf:Description rdf:about="#fd3"/>
         <rdf:Description rdf:about="#fd5"/>
      </owl:distinctMembers>
   </owl:AllDifferent>
   <owl:AllDifferent>
      <owl:distinctMembers rdf:parseType="Collection">
         <rdf:Description rdf:about="#rhs1"/>
         <rdf:Description rdf:about="#rhs2"/>
         <rdf:Description rdf:about="#rhs3"/>
         <rdf:Description rdf:about="#rhs4"/>
         <rdf:Description rdf:about="#rhs5"/>
         <rdf:Description rdf:about="#rhs6"/>
      </owl:distinctMembers>
   </owl:AllDifferent>
   <swrl:Variable rdf:ID="at"/>
   <swrl:Variable rdf:ID="attr"/>
   <swrl:Variable rdf:ID="attrs"/>
   <swrl:Variable rdf:ID="bt"/>
   <swrl:Variable rdf:ID="clo"/>
   <swrl:Variable rdf:ID="elsk"/>
   <swrl:Variable rdf:ID="fd"/>
   <swrl:Variable rdf:ID="i"/>
   <swrl:Variable rdf:ID="k"/>
   <swrl:Variable rdf:ID="l"/>
   <swrl:Variable rdf:ID="lhs"/>
   <swrl:Variable rdf:ID="n"/>
   <swrl:Variable rdf:ID="r"/>
   <swrl:Variable rdf:ID="rhs"/>
   <swrl:Variable rdf:ID="s"/>
   <swrl:Variable rdf:ID="sk"/>
   <swrl:Variable rdf:ID="sl"/>
   <swrl:Variable rdf:ID="slel"/>
   <swrl:Variable rdf:ID="sm"/>
   <swrl:Variable rdf:ID="sr"/>
   <swrl:Variable rdf:ID="srel"/>
   <swrl:Variable rdf:ID="ss"/>
   <swrl:Variable rdf:ID="st"/>
   <swrl:Variable rdf:ID="t"/>
   <swrl:Variable rdf:ID="u"/>
   <swrl:Variable rdf:ID="un"/>
   <swrl:Variable rdf:ID="x"/>
   <swrl:Variable rdf:ID="y"/>
   <Attribute rdf:ID="a"/>
   <owl:Class rdf:ID="AttrClosure"/>
   <swrl:Imp rdf:ID="rule-closure-initial">
      <swrl:body>
         <swrl:AtomList>
            <rdf:first>
               <rdf:Description>
                  <rdf:type rdf:resource="&swrl;ClassAtom"/>
                  <swrl:argument1>
                     <rdf:Description rdf:about="#clo"/>
                  </swrl:argument1>
                  <swrl:classPredicate rdf:resource="#AttrClosure"/>
               </rdf:Description>
            </rdf:first>
            <rdf:rest>
               <swrl:AtomList>
                  <rdf:first>
                     <rdf:Description>
                        <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
                        <swrl:argument2>
                           <rdf:Description rdf:about="#attrs"/>
                        </swrl:argument2>
                        <swrl:argument1>
                           <rdf:Description rdf:about="#clo"/>
                        </swrl:argument1>
                        <swrl:propertyPredicate rdf:resource="#clo_attr"/>
                     </rdf:Description>
                  </rdf:first>
                  <rdf:rest rdf:resource="&rdf;nil"/>
               </swrl:AtomList>
            </rdf:rest>
         </swrl:AtomList>
      </swrl:body>
      <swrl:head>
         <swrl:AtomList>
            <rdf:first>
               <rdf:Description>
                  <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
                  <swrl:argument2>
                     <rdf:Description rdf:about="#attrs"/>
                  </swrl:argument2>
                  <swrl:argument1>
                     <rdf:Description rdf:about="#clo"/>
                  </swrl:argument1>
                  <swrl:propertyPredicate rdf:resource="#closure"/>
               </rdf:Description>
```

```
          </rdf:first>
          <rdf:rest rdf:resource="&rdf;nil"/>
        </swrl:AtomList>
      </swrl:head>
<swrla:isRuleEnabled>rdf:datatype="&xsd;boolean">false
</swrla:isRuleEnabled>
  </swrl:Imp>
  <owl:Class rdf:ID="Attribute"/>
  <Attribute rdf:ID="b"/>
  <Attribute rdf:ID="c"/>
  <AttrClosure rdf:ID="clo_1">
    <clo_attr rdf:resource="#a"/>
    <clo_attr rdf:resource="#b"/>
    <clo_attr rdf:resource="#d"/>
  </AttrClosure>
  <AttrClosure rdf:ID="clo_2">
    <clo_attr rdf:resource="#e"/>
  </AttrClosure>
  <AttrClosure rdf:ID="clo_3">
    <clo_attr rdf:resource="#d"/>
  </AttrClosure>
  <AttrClosure rdf:ID="clo_4">
    <clo_attr rdf:resource="#d"/>
    <clo_attr rdf:resource="#e"/>
  </AttrClosure>
  <AttrClosure rdf:ID="clo_5">
    <clo_attr rdf:resource="#a"/>
    <clo_attr rdf:resource="#d"/>
    <clo_attr rdf:resource="#e"/>
  </AttrClosure>
  <owl:ObjectProperty rdf:ID="clo_attr">
    <rdfs:domain rdf:resource="#AttrClosure"/>
    <rdfs:range rdf:resource="#Attribute"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="closure">
    <rdfs:domain rdf:resource="#AttrClosure"/>
    <rdfs:range rdf:resource="#Attribute"/>
  </owl:ObjectProperty>
  <Attribute rdf:ID="d"/>
  <Attribute rdf:ID="e"/>
  <Attribute rdf:ID="f"/>
  <owl:Class rdf:ID="FD">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#holds_over"/>
        <owl:someValuesFrom rdf:resource="#Relation"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#has_lhs"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
            </owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#has_lhs"/>
            <owl:someValuesFrom rdf:resource="#LHS"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#has_rhs"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
            </owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#has_rhs"/>
            <owl:someValuesFrom rdf:resource="#RHS"/>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </rdfs:subClassOf>
  </owl:Class>
  <FD rdf:ID="fd1">
    <has_lhs rdf:resource="#lhs1"/>
    <has_rhs rdf:resource="#rhs1"/>
    <holds_over rdf:resource="#rel"/>
  </FD>
  <FD rdf:ID="fd2">
    <has_lhs rdf:resource="#lhs2"/>
    <has_rhs rdf:resource="#rhs2"/>
    <holds_over rdf:resource="#rel"/>
  </FD>
  <FD rdf:ID="fd3">
    <has_lhs rdf:resource="#lhs3"/>
    <has_rhs rdf:resource="#rhs3"/>
    <holds_over rdf:resource="#rel"/>
  </FD>
  <FD rdf:ID="fd4">
    <has_lhs rdf:resource="#lhs4"/>
    <has_rhs rdf:resource="#rhs4"/>
    <holds_over rdf:resource="#rel"/>
  </FD>
  <FD rdf:ID="fd5">
    <has_lhs rdf:resource="#lhs5"/>
    <has_rhs rdf:resource="#rhs5"/>
    <holds_over rdf:resource="#rel"/>
  </FD>
  <FD rdf:ID="fd6">
    <has_lhs rdf:resource="#lhs6"/>
    <has_rhs rdf:resource="#rhs6"/>
    <holds_over rdf:resource="#rel"/>
  </FD>
  <owl:ObjectProperty rdf:ID="has_attr">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Schema"/>
          <owl:Class rdf:about="#Side"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#Attribute"/>
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:ID="has_lhs">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
    <rdfs:range rdf:resource="#LHS"/>
    <rdfs:subPropertyOf rdf:resource="#has_side"/>
  </owl:FunctionalProperty>
  <swrl:Imp rdf:ID="rule_closure">
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <rdf:Description>
            <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
            <swrl:argument2>
              <rdf:Description rdf:about="#lhs"/>
```

```
      </swrl:argument2>
      <swrl:argument1>
        <rdf:Description rdf:about="#fd"/>
      </swrl:argument1>
      <swrl:propertyPredicate rdf:resource="#has_lhs"/>
    </rdf:Description>
  </rdf:first>
  <rdf:rest>
    <swrl:AtomList>
      <rdf:first>
        <rdf:Description>
          <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
          <swrl:argument2>
            <rdf:Description rdf:about="#rhs"/>
          </swrl:argument2>
          <swrl:argument1>
            <rdf:Description rdf:about="#fd"/>
          </swrl:argument1>
          <swrl:propertyPredicate rdf:resource="#has_rhs"/>
        </rdf:Description>
      </rdf:first>
      <rdf:rest>
       <swrl:AtomList>
        <rdf:first>
         <rdf:Description>
          <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
          <swrl:argument2>
            <rdf:Description rdf:about="#at"/>
          </swrl:argument2>
          <swrl:argument1>
             <rdf:Description rdf:about="#lhs"/>
          </swrl:argument1>
           <swrl:propertyPredicate rdf:resource="#has_attr"/>
         </rdf:Description>
        </rdf:first>
<rdf:rest>
    <swrl:AtomList>
      <rdf:first>
        <rdf:Description>
         <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
           <swrl:argument2>
             <rdf:Description rdf:about="#bt"/>
           </swrl:argument2>
           <swrl:argument1>
             <rdf:Description rdf:about="#rhs"/>
           </swrl:argument1>
          <swrl:propertyPredicate rdf:resource="#has_attr"/>
       </rdf:Description>
      </rdf:first>
       <rdf:rest>
       <swrl:AtomList>
         <rdf:first>
           <rdf:Description>
             <rdf:type rdf:resource="&swrl;ClassAtom"/>
             <swrl:argument1>
               <rdf:Description rdf:about="#clo"/>
              </swrl:argument1>
             <swrl:classPredicate rdf:resource="#AttrClosure"/>
       </rdf:Description>
    </rdf:first>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:first>
        <rdf:Description>
          <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
    <swrl:argument2>
          <rdf:Description rdf:about="#attrs"/>
      </swrl:argument2>
      <swrl:argument1>
        <rdf:Description rdf:about="#clo"/>
      </swrl:argument1>
      <swrl:propertyPredicate rdf:resource="#closure"/>
  </rdf:Description>
</rdf:first>
<rdf:rest>
  <swrl:AtomList>
    <rdf:first>
     <rdf:Description>
       <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
        <swrl:arguments>
          <rdf:List>
             <rdf:first>
               <rdf:Description rdf:about="#sk"/>
             </rdf:first>
             <rdf:rest>
              <rdf:List>
                <rdf:first>
                  <rdf:Description rdf:about="#attrs"/>
                </rdf:first>
                <rdf:rest rdf:resource="&rdf;nil"/>
              </rdf:List>
             </rdf:rest>
           </rdf:List>
         </swrl:arguments>
         <swrl:builtin>
          <rdf:Description rdf:about="&sqwrl;makeBag"/>
         </swrl:builtin>
        </rdf:Description>
       </rdf:first>
       <rdf:rest>
        <swrl:AtomList>
         <rdf:first>
           <rdf:Description>
            <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
            <swrl:arguments>
             <rdf:List>
             <rdf:first>
               <rdf:Description rdf:about="#sk"/>
             </rdf:first>
             <rdf:rest>
             <rdf:List>
             <rdf:first>
              <rdf:Description rdf:about="#clo"/>
             </rdf:first>
             <rdf:rest>
              <rdf:List>
               <rdf:first>
                 <rdf:Description rdf:about="#fd"/>
               </rdf:first>
               <rdf:rest rdf:resource="&rdf;nil"/>
              </rdf:List>
             </rdf:rest>
            </rdf:List>
           </rdf:List>
          </rdf:rest>
         </rdf:List>
        </swrl:arguments>
        <swrl:builtin>
        <rdf:Description rdf:about="&sqwrl;groupBy"/>
       </swrl:builtin>
      </rdf:Description>
     </rdf:first>
     <rdf:rest>
     <swrl:AtomList>
      <rdf:first>
```

```
<rdf:Description>
  <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
    <swrl:arguments>
      <rdf:List>
        <rdf:first>
           <rdf:Description rdf:about="#sl"/>
        </rdf:first>
        <rdf:rest>
            <rdf:List>
              <rdf:first>
                 <rdf:Description rdf:about="#at"/>
               </rdf:first>
               <rdf:rest rdf:resource="&rdf;nil"/>
             </rdf:List>
          </rdf:rest>
         </rdf:List>
         </swrl:arguments>
         <swrl:builtin>
           <rdf:Description rdf:about="&sqwrl;makeBag"/>
         </swrl:builtin>
        </rdf:Description>
        </rdf:first>
        <rdf:rest>
        <swrl:AtomList>
         <rdf:first>
          <rdf:Description>
             <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
             <swrl:arguments>
               <rdf:List>
                 <rdf:first>
                  <rdf:Description rdf:about="#sl"/>
                 </rdf:first>
                 <rdf:rest>
                  <rdf:List>
                    <rdf:first>
                      <rdf:Description rdf:about="#clo"/>
                    </rdf:first>
                    <rdf:rest>
                      <rdf:List>
                        <rdf:first>
                          <rdf:Description rdf:about="#fd"/>
                         </rdf:first>
                         <rdf:rest rdf:resource="&rdf;nil"/>
                       </rdf:List>
                      </rdf:rest>
                    </rdf:List>
                 </rdf:rest>
               </rdf:List>
             </swrl:arguments>
             <swrl:builtin>
              <rdf:Description rdf:about="&sqwrl;groupBy"/>
             </swrl:builtin>
           </rdf:Description>
         </rdf:first>
         <rdf:rest>
          <swrl:AtomList>
           <rdf:first>
             <rdf:Description>
              <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
              <swrl:arguments>
                <rdf:List>
                 <rdf:first>
                    <rdf:Description rdf:about="#sr"/>
                 </rdf:first>
                 <rdf:rest>
                  <rdf:List>
                    <rdf:first>
                      <rdf:Description rdf:about="#bt"/>
                    </rdf:first>
                        <rdf:rest rdf:resource="&rdf;nil"/>
                  </rdf:List>
                    </rdf:rest>
                 </rdf:List>
              </swrl:arguments>
              <swrl:builtin>
                    <rdf:Description rdf:about="&sqwrl;makeBag"/>
              </swrl:builtin>
             </rdf:Description>
           </rdf:first>
           <rdf:rest>
          <swrl:AtomList>
             <rdf:first>
              <rdf:Description>
               <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
                <swrl:arguments>
                 <rdf:List>
                 <rdf:first>
                    <rdf:Description rdf:about="#sr"/>
                 </rdf:first>
                 <rdf:rest>
                 <rdf:List>
                  <rdf:first><rdf:Description rdf:about="#clo"/>
                  </rdf:first>
                  <rdf:rest>
                   <rdf:List>
                    <rdf:first><rdf:Description rdf:about="#fd"/>
                    </rdf:first>
                    <rdf:rest rdf:resource="&rdf;nil"/>
                  </rdf:List>
                 </rdf:rest>
                </rdf:List>
              </rdf:rest>
            </rdf:List>
          </swrl:arguments>
          <swrl:builtin>
             <rdf:Description rdf:about="&sqwrl;groupBy"/>
          </swrl:builtin>
        </rdf:Description>
      </rdf:first>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:first>
            <rdf:Description>
              <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
              <swrl:arguments>
                <rdf:List>
                  <rdf:first><rdf:Description rdf:about="#sk"/>
                  </rdf:first>
                  <rdf:rest>
                    <rdf:List>
                      <rdf:first>
                        <rdf:Description rdf:about="#sl"/>
                      </rdf:first>
                      <rdf:rest rdf:resource="&rdf;nil"/>
                    </rdf:List>
                  </rdf:rest>
                </rdf:List>
              </swrl:arguments>
              <swrl:builtin>
                <rdf:Description rdf:about="&sqwrl;contains"/>
              </swrl:builtin>
            </rdf:Description>
          </rdf:first>
          <rdf:rest>
```

```
      <swrl:AtomList>
    <rdf:first>
      <rdf:Description>
        <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
        <swrl:arguments>
          <rdf:List>
            <rdf:first><rdf:Description rdf:about="#sk"/>
            </rdf:first>
        <rdf:rest>
           <rdf:List>
            <rdf:first><rdf:Description rdf:about="#sr"/>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;nil"/></rdf:List>
        </rdf:rest>
      </rdf:List>
    </swrl:arguments>
     <swrl:builtin>
      <rdf:Description rdf:about="&sqwrl;notContains"/>
    </swrl:builtin>
  </rdf:Description>
</rdf:first>
 <rdf:rest>
    <swrl:AtomList>
      <rdf:first>
        <rdf:Description>
          <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
          <swrl:arguments>
            <rdf:List>
              <rdf:first><rdf:Description rdf:about="#u"/>
              </rdf:first>
              <rdf:rest>
                <rdf:List>
                  <rdf:first><rdf:Description rdf:about="#sr"/>
                   </rdf:first>
                  <rdf:rest>
                    <rdf:List>
                      <rdf:first>
                        <rdf:Description rdf:about="#sk"/>
                      </rdf:first>
                      <rdf:rest rdf:resource="&rdf;nil"/>
                    </rdf:List>
                  </rdf:rest>
                </rdf:List>
              </rdf:rest>
            </rdf:List>
          </swrl:arguments>
          <swrl:builtin>
            <rdf:Description rdf:about="&sqwrl;union"/>
          </swrl:builtin>
        </rdf:Description>
    </rdf:first>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:first>
          <rdf:Description>
            <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
             <swrl:arguments>
              <rdf:List>
                <rdf:first><rdf:Description rdf:about="#k"/>
                </rdf:first>
              <rdf:rest>
                <rdf:List>
                  <rdf:first><rdf:Description rdf:about="#u"/>
                  </rdf:first>
                  <rdf:rest rdf:resource="&rdf;nil"/>
                </rdf:List>
              </rdf:rest>
                </rdf:List>
              </swrl:arguments>
              <swrl:builtin>
                <rdf:Description rdf:about="&sqwrl;element"/>
              </swrl:builtin>
            </rdf:Description>
          </rdf:first>
          <rdf:rest rdf:resource="&rdf;nil"/>
        </swrl:AtomList>
        </rdf:rest>
        </swrl:AtomList>
        </rdf:rest>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
   </swrl:AtomList>
  </rdf:rest>
   </swrl:AtomList>
   </rdf:rest>
   </swrl:AtomList>
   </rdf:rest>
   </swrl:AtomList>
   </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
  </rdf:rest>
  </swrl:AtomList>
 </rdf:rest>
</swrl:AtomList>
</swrl:body>
<swrl:head>
  <swrl:AtomList>
    <rdf:first>
      <rdf:Description>
        <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
        <swrl:argument2>
          <rdf:Description rdf:about="#k"/>
        </swrl:argument2>
        <swrl:argument1>
          <rdf:Description rdf:about="#clo"/>
        </swrl:argument1>
        <swrl:propertyPredicate rdf:resource="#closure"/>
      </rdf:Description>
    </rdf:first>
    <rdf:rest rdf:resource="&rdf;nil"/>
  </swrl:AtomList>
</swrl:head>
<swrla:isRuleEnabled rdf:datatype="&xsd;boolean">false
</swrla:isRuleEnabled>
</swrl:Imp>
<owl:FunctionalProperty rdf:ID="has_rhs">
  <rdf:type rdf:resource="&owl;ObjectProperty"/>
  <rdfs:range rdf:resource="#RHS"/>
  <rdfs:subPropertyOf rdf:resource="#has_side"/>
</owl:FunctionalProperty>
<owl:ObjectProperty rdf:ID="has_schema">
  <rdfs:domain rdf:resource="#Relation"/>
```

```
  <rdfs:range rdf:resource="#Schema"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_side">
  <rdfs:domain rdf:resource="#FD"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="holds_over">
  <rdfs:domain rdf:resource="#FD"/>
  <rdfs:range rdf:resource="#Relation"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="in3nf">
  <rdfs:subClassOf rdf:resource="#Relation"/>
</owl:Class>
<owl:Class rdf:ID="inbcnf">
  <rdfs:subClassOf rdf:resource="#in3nf"/>
</owl:Class>
<owl:Class rdf:ID="LHS">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_attr"/>
      <owl:allValuesFrom rdf:resource="#Attribute"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Side"/>
</owl:Class>
<LHS rdf:ID="lhs1">
  <has_attr rdf:resource="#a"/>
  <has_attr rdf:resource="#b"/>
</LHS>
<LHS rdf:ID="lhs2">
  <has_attr rdf:resource="#c"/>
</LHS>
<LHS rdf:ID="lhs3">
  <has_attr rdf:resource="#d"/>
</LHS>
<LHS rdf:ID="lhs4">
  <has_attr rdf:resource="#d"/>
  <has_attr rdf:resource="#e"/>
</LHS>
<LHS rdf:ID="lhs5">
  <has_attr rdf:resource="#e"/>
</LHS>
<LHS rdf:ID="lhs6">
  <has_attr rdf:resource="#e"/>
</LHS>
<Relation rdf:ID="rel">
  <has_schema rdf:resource="#schema1"/>
</Relation>
<Relation rdf:ID="rel_a">
  <has_schema rdf:resource="#schema2"/>
</Relation>
<Relation rdf:ID="rel_b"/>
<Relation rdf:ID="rel_key"/>
<owl:Class rdf:ID="Relation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_schema"/>
      <owl:allValuesFrom rdf:resource="#Schema"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<swrl:Imp rdf:ID="rule-inbcnf">
  <swrl:body>
    <swrl:AtomList>
      <rdf:first>
        <rdf:Description>
          <rdf:type rdf:resource="&swrl;ClassAtom"/>
          <swrl:argument1>
            <rdf:Description rdf:about="#r"/>
          </swrl:argument1>
          <swrl:classPredicate rdf:resource="#Relation"/>
        </rdf:Description>
      </rdf:first>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:first>
            <rdf:Description>
              <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
              <swrl:argument2>
                <rdf:Description rdf:about="#s"/>
              </swrl:argument2>
              <swrl:argument1>
                <rdf:Description rdf:about="#r"/>
              </swrl:argument1>
              <swrl:propertyPredicate rdf:resource="#has_schema"/>
            </rdf:Description>
          </rdf:first>
          <rdf:rest>
           <swrl:AtomList>
            <rdf:first>
             <rdf:Description>
              <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
               <swrl:argument2><rdf:Description rdf:about="#attr"/>
                </swrl:argument2>
                <swrl:argument1><rdf:Description rdf:about="#s"/>
                 </swrl:argument1>
                 <swrl:propertyPredicate rdf:resource="#has_attr"/>
              </rdf:Description>
              </rdf:first>
              <rdf:rest>
                <swrl:AtomList>
                  <rdf:first>
                    <rdf:Description>
                      <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
                      <swrl:arguments>
                        <rdf:List>
                          <rdf:first>
                            <rdf:Description rdf:about="#ss"/>
                          </rdf:first>
                          <rdf:rest>
                            <rdf:List>
                              <rdf:first>
                                <rdf:Description rdf:about="#attr"/>
                              </rdf:first>
                              <rdf:rest rdf:resource="&rdf;nil"/>
                            </rdf:List>
                          </rdf:rest>
                        </rdf:List>
                      </swrl:arguments>
                      <swrl:builtin>
                        <rdf:Description rdf:about="&sqwrl;makeSet"/>
                      </swrl:builtin>
                    </rdf:Description>
                  </rdf:first>
                  <rdf:rest>
                    <swrl:AtomList>
                      <rdf:first>
                        <rdf:Description>
                          <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
                          <swrl:arguments>
                            <rdf:List>
                              <rdf:first>
                                <rdf:Description rdf:about="#ss"/>
                              </rdf:first>
```

```
                    <rdf:rest>
                      <rdf:List>
                        <rdf:first>
                          <rdf:Description rdf:about="#r"/>
                        </rdf:first>
                        <rdf:rest rdf:resource="&rdf;nil"/>
                      </rdf:List>
                    </rdf:rest>
                  </rdf:List>
                </swrl:arguments>
                <swrl:builtin>
                <rdf:Description rdf:about="&sqwrl;groupBy"/>
    </swrl:builtin>
 </rdf:Description>
</rdf:first>
<rdf:rest>
 <swrl:AtomList>
  <rdf:first>
   <rdf:Description>
    <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
     <swrl:arguments>
      <rdf:List>
        <rdf:first><rdf:Description rdf:about="#n"/>
        </rdf:first>
        <rdf:rest>
          <rdf:List>
            <rdf:first><rdf:Description rdf:about="#ss"/>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;nil"/>
          </rdf:List>
        </rdf:rest>
       </rdf:List>
      </swrl:arguments>
      <swrl:builtin>
        <rdf:Description rdf:about="&sqwrl;size"/>
      </swrl:builtin>
    </rdf:Description>
   </rdf:first>
   <rdf:rest>
     <swrl:AtomList>
       <rdf:first>
         <rdf:Description>
           <rdf:type rdf:resource="&swrl;BuiltinAtom"/>
             <swrl:arguments>
               <rdf:List>
                 <rdf:first>
                  <rdf:Description rdf:about="#n"/>
                 </rdf:first>
                  <rdf:rest>
                    <rdf:List>
                      <rdf:first rdf:datatype="&xsd;long"> 3
                          </rdf:first>
                      <rdf:rest rdf:resource="&rdf;nil"/>
                    </rdf:List>
                  </rdf:rest>
                </rdf:List>
         </swrl:arguments>
         <swrl:builtin>
          <rdf:Description rdf:about="&swrlb;lessThan"/>
         </swrl:builtin>
        </rdf:Description>
       </rdf:first>
        <rdf:rest rdf:resource="&rdf;nil"/>
      </swrl:AtomList>
     </rdf:rest>
     </swrl:AtomList>
    </rdf:rest>
                  </swrl:AtomList>
                 </rdf:rest>
                </swrl:AtomList>
               </rdf:rest>
              </swrl:AtomList>
             </rdf:rest>
            </swrl:AtomList>
           </rdf:rest>
          </swrl:AtomList>
         </rdf:rest>
        </swrl:AtomList>
       </rdf:rest>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <rdf:Description>
            <rdf:type rdf:resource="&swrl;ClassAtom"/>
            <swrl:argument1>
              <rdf:Description rdf:about="#r"/>
            </swrl:argument1>
            <swrl:classPredicate rdf:resource="#inbcnf"/>
          </rdf:Description>
        </rdf:first>
        <rdf:rest rdf:resource="&rdf;nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrla:isRuleEnabled rdf:datatype="&xsd;boolean"> false
    </swrla:isRuleEnabled>
</swrl:Imp>
<owl:Class rdf:ID="RHS">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_attr"/>
      <owl:allValuesFrom rdf:resource="#Attribute"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Side"/>
</owl:Class>
<RHS rdf:ID="rhs1">
  <has_attr rdf:resource="#c"/>
</RHS>
<RHS rdf:ID="rhs2">
  <has_attr rdf:resource="#a"/>
</RHS>
<RHS rdf:ID="rhs3">
  <has_attr rdf:resource="#e"/>
</RHS>
<RHS rdf:ID="rhs4">
  <has_attr rdf:resource="#f"/>
</RHS>
<RHS rdf:ID="rhs5">
  <has_attr rdf:resource="#d"/>
</RHS>
<RHS rdf:ID="rhs6">
  <has_attr rdf:resource="#f"/>
</RHS>
<owl:Class rdf:ID="Schema">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_attr"/>
      <owl:allValuesFrom rdf:resource="#Attribute"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<Schema rdf:ID="schema1">
  <has_attr rdf:resource="#a"/>
  <has_attr rdf:resource="#b"/>
  <has_attr rdf:resource="#c"/>
```

```
    <has_attr rdf:resource="#d"/>
    <has_attr rdf:resource="#e"/>
    <has_attr rdf:resource="#f"/>
  </Schema>
  <Schema rdf:ID="schema2">
    <has_attr rdf:resource="#c"/>
    <has_attr rdf:resource="#e"/>
  </Schema>
  <owl:Class rdf:ID="Side"/>
</rdf:RDF>
```

REFERENCES

[1] R. Stephens, *Beginning Database Design Solutions*. Wiley Publishing, Nov. 2009.

[2] J. Madhavan, A. Halevy, S. Cohen, X. Dong, S. Jeffery, and D. Ko Structured Data Meets the Web: A Few Observations. *Data Engineering,* vol. 4, no. 31, 2006.

[3] R. Ramakrishnan, and J. Gehrke, *Database Management Systems*. 2nd ed. McGrawHill, 1998.

[4] E. F. Codd, "Further Normalization of the Data Base Relational Model," in *Database systems*, Englewood Cliffs, N.J.: Prentice-Hall. 1972, pp. 33–64.

[5] C. Stefano, and G. Georg, "Normalization of relations and Prolog," *Communications of the ACM ,* vol. 29, no. 6, pp. 524-544, 1986.

[6] A. Yazici, and Z. Karakaya, "JMathNorm: A Database Normalization Tool Using Mathematica," *ICCS (2), Lecture Notes in Computer Science*, vol. 4488, pp. 186-193, 2007.

[7] A. Mitrovic, "NORMIT: A Web-Enabled Tutor for Database Normalization," *International Conference on Computers in Education (ICCE)*, pp. 1276-1280. Auckland, New Zealand, Dec. 2002.

[8] M. Bouzeghoub, G. Gardarin, and E. Metais, "Database design tools: An expert system approach," *11th VLDB*, pp. 82-95, Stockholm, Sweden, Aug. 21, 1985.

[9] L. Kerschberg, "Expert database systems," in *The 1st International Conference on Expert Database Systems,* Kiawah Island, S.C., Oct. 1984.

[10] R. M. Lee, "Database inferencing for decision support," *Decis. Support Syst. ,* vol. 1, no. 1, pp. 57-68, 1985.

[11] W. D. Potter, "DESIGN-PRO: A multi-model schema design tool in Prolog," in *The 1st International Conference on Expert Database Systems*, pp. 747-759, Kiawah Island, S.C., Oct. 1984.

[12] V. Hirankitti and T. M. Xuan, "A Meta-logical Approach for Reasoning with Ontologies and Rules Expressed In OWL 2", *WSEAS Intl. Conf. on Applied Computer Science*, pp. 360-366, October 2010.

[13] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American ,* vol. 284, no. 5, pp. 34-43, 2001.

[14] R. Fagin, "Functional dependencies in a relational database and propositional logic," *IBM J. Res. Dev. ,* vol. 21, no. 6, pp. 534-544, 1977.

[15] I. Horrocks, P. F. Patel-Schneider, H. Boley, S.Tabet, B.Grosof, and M. Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission.* [Online]. Available: http://www.w3.org/Submission/SWRL/, May 21, 2004.

[16] T. Matzner, and P. Hitzler, "Any-World Access to OWL from Prolog," *Lecture Notes in Computer Science*, vol. 4667, pp. 84-98, 2007.

[17] 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, no. 3, pp. 301-322, 2008.

[18] R. A. Mena, "Towards a Semantic Web: Ontology Development based on the Extraction of Semantic Concepts from Digital Documents", *WSEAS Intl. Conf. on Computers*, pp. 519-525, July 2009.

[19] M. K. Smith, C. Welty, and D. L. McGuinness, (2004, February 10). *OWL Web Ontology Language Guide.* [Online]. Available: http://www.w3.org/TR/owl-guide/

[20] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. (2004, February 10). *OWL Web Ontology Language Reference.* W3C Recommendation. [Online]. Available: http://www.w3.org/TR/owl-ref/

[21] H. Boley, and S. Tabet (2001). *Rule Markup Language.* [Online]. Available: www.dfki.uni-kl.de/ruleml

[22] H. Boley, S. Tabet, and G. Wagner, "Design rationale of RuleML: A markup language for semantic Web rules," *The International Semantic Web Working Symposium (Stanford University).* July 30 – Aug. 1, 2001.

[23] R. Volz, S. Decker, and D. Oberle, (2003). "Bubo - Implementing OWL in rule-based systems," [Online]. Available: http://www.daml.org/listarchive/joint-committee/att-1254/01-bubo.pdf

[24] R. Volz, *Web Ontology Reasoning with Logic Databases.* PhD Thesis, AIFB, University of Karlsruhe, 2004.

[25] *CollectionsSQWRL*. (n.d.). [Online]. Available: http://protege.cim3.net/cgi-bin/wiki.pl?CollectionsSQWRL

[26] M. O'Connor, (2010, February 12). *SWRLLanguageFAQ.* [Online]. Available: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ

[27] B. Parsia, E. Sirin, B. C. Grau, E. Ruckhaus, and D. Hewlett, "Cautiously approaching SWRL," in *Preprint submitted to Elsevier Science.* 2005.

[28] F. M. Pinto, and M. F. Santos, "Ontological Assistance for Knowledge Discovery in Databases Process", *WSEAS Intl. Conf. on Computers*, pp. 453-458, July 2009.

[29] *Jess: the rule engine for the Java platform.* (n.d.). (2010, May 16). [Online]. Available: http://www.jessrules.com/

[30] C. Golbreich, E. K. Wallace, and P. F. Patel-Schneider, (2009, October 27). *OWL 2 Web Ontology Language New Features and Rationale.* W3C Recommendation. [Online]. Available: http://www.w3.org/TR/owl2-new-features/

[31] V. Vassilades, , J. Wielemaker, and C. Mungall, "Processing OWL2 ontologies using Thea: An application of logic programming," *OWLED, CEUR Workshop Proceedings*, vol. 529, 2009.

[32] G. Antoniou, and F. van Harmelen, *A Semantic Web Primer*, 2nd ed. MIT Press, 2008.

[33] M. Marchiori, "Towards a People's Web: Metalog, Web Intelligence," *IEEE Computer Society*, pp. 320-326, 2004.

[34] M. Marchiori, "Introduction to the Special Issue on Logic Programming and the Web," *TPLP*, vol. 8, no. 3, pp. 247-248, 2008.

[35] D. Brickley, R.V. Guha, and B. McBride, (2004, Feb. 10). *RDF Vocabulary Description Language 1.0: RDF Schema.* W3C Recommendation. [Online]. Available: http://www.w3.org/TR/rdf-schema/.

[36] I. Pah, I. (M.) Maniu, G. Maniu, and S. Damian, "A conceptual framework based on ontologies for knowledge management in e-learning systems", *WSEAS Intl. Conf. on Education and Educational Technology*, pp. 283-286, November 2007.

[37] N. Noy, R. Rector, P. Hayes, and C. Welty, (2006, April 12). *Defining N-ary Relations on the Semantic Web*. W3C Working Group Note. [Online]. Available at http://www.w3.org/TR/swbp-n-aryRelations/

[38] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. (2004, Feb. 10) *OWL Web Ontology Language Reference.* W3C Recommendation. [Online]. Available: http://www.w3.org/TR/owl-ref/

[39] M. Kifer, and H. Boley, (2010, June 22). *RIF Overview*. W3C Working Group Note. [Online]. Available: http://www.w3.org/TR/rif-overview/

[40] C. S. Marie, G. Hallmark, and A. Paschke, (2010, June 22). RIF Production Rule Dialect. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/rif-prd/

[41] D. Chiribuca, D. Hunyadi, and E. M. Popa, "The Educational Semantic Web", *WSEAS Intl. Conf. on Applied Informatics and Communications*, pp. 314-219, August 2008.