INTEGRATING IDENTIFICATION CONSTRAINTS IN WEB ONTOLOGY

Thi Dieu Thu Nguyen and Nhan Le-Thanh
I3S laboratory (CNRS - UNSA)
Les Algorithmes - Euclide B Building, 2000 route des Lucioles – B.P. 121
F-06903 Sophia Antipolis Cedex, France

Keywords: Integrating relational data sources, Web ontology languages, Semantic Web, Description Logics, Identification constraints.

Abstract: In recent years, there has been a growing interest in semantic integration in the Semantic Web environment, whose goal is to access, relate and combine knowledge from multiple sources. The need of integrating the semantics from relational data sources into this environment, therefore, has also emerged. However, there is one important aspect of database schemas that OWL up to now has not captured yet, namely identification constraints. To address this problem, this paper introduces a decidable extension of OWL-DL, namely OWL-K, that supports such constraints.

1 INTRODUCTION

In recent years, the problem of interoperability and semantic integration of heterogeneous data sources in the Web environment has attracted distinctive attention of many researchers in various domains, which can be brought together to the so-called Semantic Web (SW). One of the main features of the SW is the ability to describe data sources on the Web by declarative annotations. The latter should be expressive enough to deal with the problem of heterogeneity of sources and well formalized to be used in automated reasoning. OWL-DL (Ontology Web Language-Description Logic) is a popular resource description language proposed by W3C[1] for this purpose. Particularly, to integrate relational data sources on the Semantic Web environment, OWL-DL must be capable of expressing the notion of *Identification Constraints* (ICs), which is so called (primary and foreign) keys in relational models. However, up to now this important feature of database schemas has not been fully captured yet.

Motivated by this problem, we introduce ICs into OWL-DL, resulting in a new Web ontology language OWL-K. This extension is a decisive step to integrate relational data sources into the SW.

[1]World Wide Web Consortium, http://www.w3.org/

2 OVERVIEW OF OWL-DL

In order to support desirable computational properties for reasoning systems, OWL-DL is designed with the same set of constructors as for OWL (Bechhofer et al., 2004), but restricted to be used in a way satisfying decidable inference. What making OWL-DL a Semantic Web language, however, is not its semantics, which are quite standard for Description Logics (DLs), but its RDF/XML exchange syntax besides an abstract frame-like syntax.

The abstract syntax, DL syntax and semantics of OWL-DL descriptions and axioms can be seen in table 1 and 2, where A is a class URI reference; $C, C_1, ..., C_n$ are class descriptions; S is an object property (whose value is an individual) URI reference; $R, R_1, ..., R_n$ are object property descriptions; $o, o_1, ..., o_n$ are individual URI references; d is a data range; U is a datatype property (whose value is a data literal); $\sharp$ denotes cardinality; $\mathcal{I}$ is the interpretation function; Δ^I is the individual domain and $\Delta_{\mathbf{D}}$ is the domain of data values. The semantics of OWL-DL is based on $\mathcal{SHOIN}(\mathbf{D})$ DL, an extension of $\mathcal{SHOQ}(\mathbf{D})$ (Horrocks and Sattler, 2001) with inverse

Dieu Thu Nguyen T. and Le-Thanh N. (2007).
INTEGRATING IDENTIFICATION CONSTRAINTS IN WEB ONTOLOGY.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 338-343
DOI: 10.5220/0002397803380343

Table 1: OWL-DL class, property descriptions.

Abstract syntax	DL Syntax	Semantics
Class(A)	A	$A^I \subseteq \Delta^I$
Class(owl:Thing)	$\top$	owl:Thing$^I = \Delta^I$
Class(owl:Nothing)	$\bot$	owl:Nothing$^I = \emptyset$
intersectionOf($C_1 ... C_n$)	$C_1 \sqcap ... \sqcap C_n$	$(C_1 \sqcap ... \sqcap C_n)^I = C_1^I \cap ... \cap C_2^I$
unionOf($C_1 ... C_n$)	$C_1 \sqcup ... \sqcup C_2$	$(C_1 \sqcup ... \sqcup C_n)^I = C_1^I \cup ... \cup C_2^I$
complementOf(C)	$\neg C$	$(\neg C)^I = \Delta^I \backslash C^I$
oneOf($o_1 ... o_n$)	$\{o_1\} \sqcup ... \sqcup \{o_n\}$	$(\{o_1\} \sqcup ... \sqcup \{o_n\})^I = \{o_1^I, ..., o_n^I\}$
restriction(R someValuesFrom(C))	$\exists R.C$	$(\exists R.C)^I = \{x \in \Delta^I \mid \exists y. \langle x,y \rangle \in R^I \wedge y \in C^I\}$
restriction(R allValuesFrom(C))	$\forall R.C$	$(\forall R.C)^I = \{x \in \Delta^I \mid \forall y. \langle x,y \rangle \in R^I \rightarrow y \in C^I\}$
restriction(R hasValue(o))	$\exists R.\{o\}$	$(\exists R.\{o\})^I = \{x \in \Delta^I \mid \langle x, \{o\}^I \rangle \in R^I\}$
restriction(R minCardinality(n))	$\geq nR$	$(\geq nR)^I = \{x \in \Delta^I \mid \sharp\{y. \langle x,y \rangle \in R^I\} \geq n\}$
restriction(R maxCardinality(n))	$\leq nR$	$(\leq nR)^I = \{x \in \Delta^I \mid \sharp\{y. \langle x,y \rangle \in R^I\} \leq n\}$
restriction(U someValuesFrom(d))	$\exists U.d$	$(\exists U.d)^I = \{x \in \Delta^I \mid \exists y. \langle x,y \rangle \in U^I \wedge y \in d^{\mathbf{D}}\}$
restriction(U allValuesFrom(d))	$\forall U.d$	$(\forall U.d)^I = \{x \in \Delta^I \mid \forall y. \langle x,y \rangle \in U^I \rightarrow y \in d^{\mathbf{D}}\}$
restriction(U hasValue(v))	$\exists U.v$	$(\exists U.v)^I = \{x \in \Delta^I \mid \langle x,v \rangle \in U^I\}$
restriction(U minCardinality(n))	$\geq nU$	$(\geq nU)^I = \{x \in \Delta^I \mid \sharp\{y. \langle x,y \rangle \in U^I\} \geq n\}$
restriction(U maxCardinality(n))	$\leq nU$	$(\leq nU)^I = \{x \in \Delta^I \mid \sharp\{y. \langle x,y \rangle \in U^I\} \leq n\}$
ObjectProperty(S)	S	$S^I \subseteq \Delta^I \times \Delta^I$
inverseOf(S)	S^-	$(S^-)^I = \{\langle x,y \rangle \mid \langle y,x \rangle \in S^I\}$
DatatypeProperty(U)	U	$U^I \subseteq \Delta^I \times \Delta_{\mathbf{D}}$

roles and restricted to unqualified number restrictions. See (Horrocks and Patel-Schneider, 2004) for more details of OWL-DL.

As shown in table 1 and 2, OWL-DL can declare classes and organize them in a subsumption (subclass) hierarchy. OWL classes can be specified as logical combinations (intersections, unions, or complements) of others, as enumerations of specified objects or as restrictions on a particular property so that all the values for the property in instances of the class must belong to a certain class (or datatype); at least one value must come from a certain class (or datatype); there must be at least certain specific values; there must be at least or at most a certain number of distinct values. OWL-DL can also declare properties with their domains and ranges, can organize them into a subproperty hierarchy. It can also state that a property is transitive, symmetric, functional, or inverse w.r.t another.

3 OWL-DL WITH ICS

3.1 Limitations of OWL-DL in Representing ICs

Although supporting considerable expressive power to the Semantic Web, the mechanism to express ICs in OWL-DL is seriously limited. In particular, OWL-DL provides two constructors Functional and InverseFunctional to link individuals together.

If a property P is tagged as Functional then $\forall$ x, y, z: P(x,y) and P(x,z) implies y = z. For example, if the property hasFlag is characterized as Functional, then each nation has at most one flag. Vice versa, if P is tagged as InverseFunctional then $\forall$ x, y and z: P(y,x) and P(z,x) implies y = z. For example, if characterizing hasFlag as InverseFunctional, then two nations could be inferred to be identical based on having the same flag. See the illustration of Functional and InverseFunctional in figure 1.

Thus, one can think of InverseFunctional as defining a unique key in the database sense. However, it does not require that all elements of the domain have values. Furthermore, two values may infer to the same element. Essentially, InverseFunctional does not represent a 1-1 but 1-n relation between elements of the domain and those of the range of a property. Therefore, this constructor does not fully capture the meaning of ICs (which will be more explained in the next section). Similarly to InverseFunctional, Functional does not represent the notion of ICs but a functional dependency. One can think of using both of these constructors to represent the notion of ICs. However, the relations are not compulsory to all elements of the domain. So that the meaning of ICs is not fully described. Figure 1 illustrates the relations put by these constructors.

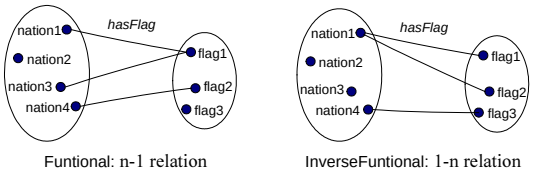

Figure 1: Limitations of OWL-DL in representing ICs.

Table 2: OWL-DL axioms.

Abstract syntax	DL Syntax	Semantics
Class(A partial $C_1...C_n$)	$A \sqsubseteq C_1 \sqcap ... \sqcap C_n$	$A^I \subseteq C_1^I \cap ... \cap C_n^I$
Class(A complete $C_1...C_n$)	$A \equiv C_1 \sqcap ... \sqcap C_n$	$A^I \equiv C_1^I \cap ... \cap C_n^I$
EnumeratedClass(A $o_1...o_n$)	$A \equiv \{o_1\} \sqcap ... \sqcap \{o_n\}$	$A^I \equiv \{o_1^I, ..., o_n^I\}$
SubClassOf(C_1, C_2)	$C_1 \sqsubseteq C_2$	$C_1^I \subseteq C_2^I$
EquivalentClasses($C_1...C_n$)	$C_1 \equiv ... \equiv C_n$	$C_1^I = ... = C_n^I$
DisjointClasses($C_1...C_n$)	$C_i \sqsubseteq \neg C_j$, $(1 \leq i < j \leq n)$	$C_i^I \cap C_i^I = \emptyset$, $(1 \leq i < j \leq n)$
DatatypeProperty(U super(U_1) ... super(U_n))	$U \sqsubseteq U_i, (1 \leq i \leq n)$	$U^I \subseteq U_i^I, (1 \leq i \leq n)$
domain(C_1) ... domain(C_m)	$\geq 1U \sqsubseteq C_i, (1 \leq i \leq m)$	$U^I \subseteq C_i^I \times {\Delta_\mathbf{D}}^I, (1 \leq i \leq m)$
range(d_1) ... range(d_k)	$\top \sqsubseteq \forall U.d_i, (1 \leq i \leq k)$	$U^I \subseteq \Delta^I \times d_i^I, (1 \leq i \leq k)$
[Functional]	$\top \sqsubseteq \leq 1U$	$\{\langle x,y\rangle \mid \sharp\{y.\langle x,y\rangle \in U^I\} \leq 1 \forall x \in \Delta^I\}$
SubPropertyOf(U_1, U_2)	$U_1 \sqsubseteq U_2$	$U_1^I \subseteq U_2^I$
EquivalentProperties($U_1...U_n$)	$U_1 \equiv ... \equiv U_n$	$U_1^I = ... = U_n^I$
ObjectProperty(R super(R_1) ... super(R_n))	$R \sqsubseteq R_i, (1 \leq i \leq n)$	$R^I \subseteq R_i^I, (1 \leq i \leq n)$
domain(C_1) ... domain(C_m)	$\geq 1R \sqsubseteq C_i, (1 \leq i \leq m)$	$R^I \subseteq C_i^I \times \Delta^I, (1 \leq i \leq m)$
range(C_1) ... range(C_k)	$\top \sqsubseteq \forall R.C_i, (1 \leq i \leq k)$	$R^I \subseteq \Delta^I \times C_i^I, (1 \leq i \leq k)$
[Symmetric]	$R \equiv R^-$	$R^I = (R^-)^I$
[Functional]	$\top \sqsubseteq \leq 1R$	$\{\langle x,y\rangle \mid \sharp\{y.\langle x,y\rangle \in R^I\} \leq 1 \forall x \in \Delta^I\}$
[InverseFunctional]	$\top \sqsubseteq \leq 1R^-$	$\{\langle x,y\rangle \mid \sharp\{y.\langle x,y\rangle \in (R^-)^I\} \leq 1 \forall x \in \Delta^I\}$
[Transitive]	Trans(R)	$R^I = (R^I)^+$
SubPropertyOf(R_1, R_2)	$R_1 \sqsubseteq R_2$	$R_1^I \subseteq R_2^I$
EquivalentProperties($R_1...R_n$)	$R_1 \equiv ... \equiv R_n$	$R_1^I = ... = R_n^I$
AnnotationProperty(R)		
Individual(o type(C_1) ... type(C_n)	$o \in C_i$	$o^I \in C_i^I$
value(R_1 o_1)...value(R_n o_n))	$\langle o, o_i\rangle \in R_i$	$\langle o^I, o_i^I\rangle \in R_i^I$
SameIndividual($o_1...o_n$)	$o_1 = ... = o_n$	$o_1^I = ... = o_n^I$
DifferentIndividuals($o_1...o_n$)	$o_i \neq o_j$	$o_i^I \neq o_j^I$

Another limitation of OWL-DL is that these constructors relate only two elements while in database schemas, ICs can be put on a set of elements as shown in the next section.

3.2 Motivation of Adding ICs

In the sense of OWL-DL, ICs can be defined as to state that a certain set of properties uniquely identifies instances of a given class. Essentially, these constraints put a 1-1 relation between sets of values of properties and instances of a class. Example 1 shows that ICs cannot be represented in OWL-DL.

*Example 1:*One would like to state that instances of the class NationHistory are uniquely identified by a couple of properties (hasFlag, onDate), where hasFlag is an ObjectProperty whose values are flags and onDate is a DatatypeProperty whose values are of datatype *date*.

However, OWL-DL has no constructors to describe that an "instance" of the couple (hasFlag, onDate) uniquely identifies an instance of the class NationHistory. It is even impossible to describe this expression with any combination of constructors in OWL-DL.

From the example, we can see that properties in the set identifying instances of a concept are relations between either instances of concepts or instances of a concept and values of a datatype, showing that ICs allow to express the relations not only between individuals but also between individuals and values.

As a result, to express ICs, a new mechanism is required to support both datatype and object properties. The next section will introduce a such mechanism which results in an extended language called OWL-K.

4 MODELING ICS IN OWL-K

4.1 Vocabulary

The representation of ICs in OWL-DL, as shown in section 3.1 cannot express the constraints in example 1. Therefore we extend OWL-DL with IC assertions resulting in OWL-K. In this language, IC assertions are modeled as entities. They are neither classes (or concepts) nor properties (or roles). Hence we define IC assertions as instances of a new class owl:ICAssertion, which is a subclass of rdfs:Resource (see figure 2).

Figure 2 shows the class hierarchy using a "nodes and arcs" graph representation of the RDF data

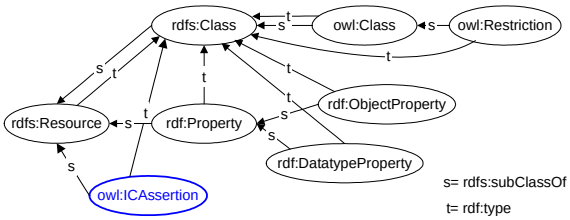

Figure 2: Class Hierarchy for OWL-K.

model. If a class is a subset of another, then there is an rdfs:subClassOf arc from the node representing the former class to the node representing the latter. Similarly, if a resource is an instance of a class, then there is an rdf:type arc from the resource to the node representing the class. (Note that not all resources, classes and such arcs are shown. We only show the principle resources, classes and arcs relating to our extension. The rest remains the same as for OWL-DL.)

An IC puts a constraint on a set of properties. Let us see how OWL-DL provides property restrictions. First, property restrictions are defined as classes while ICs do not create new classes. Second, OWL-DL distinguishes two kinds of property restrictions: value and cardinality constraints. *Value constraints* put constraints on the range of *a* property when applied to a particular class description. *Cardinality constraint* puts constraints on the number of values *a* property can take, in the context of a particular class description. So restriction constructs in OWL-DL put restrictions on only *one* property while an IC puts a restriction on a collection of properties. Restriction constructs in OWL-DL, therefore, do not agree with ICs. To express IC assertions, we introduce new kinds of restriction constructs, namely owl:onClass and owl:byProperty. The former is used to specify the class an IC is applied to. The latter is used to specify the property in the collection of properties identifying the instance of a given class. Since ICs can be applied both to datatype and object properties, `owl:byProperty` is designed to have range both of them. Table 3 shows the vocabulary extension of OWL-K compared with OWL-DL.

4.2 Abstract Syntax

The abstract syntax is used to facilitate access to and evaluation of the language. It is specified by means of a version of Extended BNF, which is defined in section 2 of (Horrocks and Patel-Schneider, 2004).

IC assertions in OWL-K ontologies must be identifiable and referable. Hence as for classes, properties and instances in OWL-DL ontologies, we associate with each IC assertion in an OWL-K ontology an identifier, which is a URI reference.

As the axioms modeled in OWL-DL, a new kind of axioms to express IC assertions is added as follows.

```
ICAssertionID ::= URIreference
        axiom ::= 'ICAssertion(' ICAssertionID
                    description
                    propertyID {propertyID}')'
   propertyID ::= datavaluedPropertyID |
                  individualvaluedPropertyID
```

The IC in example 1 is then written in the abstract syntax as follows:

```
ICAssertion(NatHisIC NationHistory hasFlag onDate)
```

4.3 Semantics

The semantics of the abstract syntax above is defined by definition 4.1. Actually, this definition is provided by a constructor called **Idfor**, which is added to $\mathcal{SHOIN}(\mathbf{D})$, the underpinning of OWL-DL, producing the underpinning of OWL-K, the DL $\mathcal{SHOINK}(\mathbf{D})$ (see section 5). Note that in DLs we talk about *concepts*, *abstract roles* and *concrete roles* while in Web ontology languages we usually call them *classes*, *object properties* and *datatype properties* respectively.

Definition 4.1 (Identification constraint) *An identification constraint is defined as:*

$$(R_1, ..., R_n \,\mathbf{Idfor}\, C) \qquad (1)$$

*where C is a concept, R_i is a simple role (*abstract *or* concrete*)* $\forall 1 \leq i \leq n$, **Idfor** *is the constructor specifying the constraint. The semantics of this definition is formally defined by an interpretation $\mathcal{I}$ that satisfies the definition $(R_1, ..., R_n \,\mathbf{Idfor}\, C)$ iff $\forall s, s' \in C^{\mathcal{I}}$ and $\forall 1 \leq i \leq n, \langle s, t_i \rangle, \langle s', t_i' \rangle \in R_i^{\mathcal{I}}$, we have $t_i = t_i'$ then $s = s'$.*

Intuitively, this definition indicates that two instances of a concept C never have the same participation in these n roles. To ensure the decidability of the reasoning algorithm, the roles must be *simple* (see

Table 3: Vocabulary extension for OWL-K.

rdfs:Class	rdfs:subClassOf
owl:ICAssertion	rdfs:Resource

Property name	Type
owl:onClass	rdf:ObjectProperty
owl:byProperty	rfd:Property

Property name	rdfs:domain	rdfs:range
owl:onClass	owl:ICAssertion	owl:Class
owl:byProperty	owl:ICAssertion	owl:Property

Table 4: Mapping OWL-K abstract syntax to DL syntax and semantics.

Abstract syntax	DL syntax	Semantics
ICAssertion (ICAssertionID $C\ R_1...R_n$)	$(R_1,...,R_n$ **Idfor** $C)$	$I \models (R_1,...,R_n$ **Idfor** $C)$ iff $\forall s,s' \in C^I$ and $\langle s,t_i\rangle, \langle s',t_i'\rangle \in R_i^I$ $\forall 1 \leq i \leq n, t_i = t_i'$ $\forall 1 \leq i \leq n$ then $s = s'$

section 5). The abstract, DL syntax and semantics of ICs are presented in Table 4.

For example, the constraint in example 1 will be represented in DL as the definition: (*hasFlag, onDate* **Idfor** *NationHistory*), where *hasFlag* is the abstract role, *onDate* is the concrete role, *NationHistory* is the concept.

4.4 RDF Graphs

An OWL ontology is an RDF graph, which is in turn a set of RDF triples. Hence it is necessary to relate specific abstract syntax ontologies with specific RDF/XML documents and their corresponding graphs. We provide a mapping from the abstract syntax for OWL-K to the exchange syntax, i.e. RDF/XML syntax. Since OWL-K is the extension of OWL-DL, it inherits the mapping from OWL-DL (Horrocks and Patel-Schneider, 2004). Thus, we introduce here only the mapping for IC assertions (see table 5). As a result, our extension preserves the normative relationship between the abstract syntax and the exchange syntax. The IC in example 1 is represented in the exchange syntax as follows:

```
<owl:ICAssertion rdf:ID = "NatHisIC">
  <owl:onClass rdf:resource = "NationHistory" />
  <owl:byProperty rdf:resource = "#hasFlag"/>
  <owl:byProperty rdf:resource = "#onDate"/>
</owl:ICAssertion>
```

5 DECIDABILITY OF OWL-K

Now we show that OWL-K is decidable by presenting its underpinning $\mathcal{SHOINK}(\mathbf{D})$ language (Nguyen and Le-Thanh, 2007; Nguyen and Le-Thanh, 2006), which is an extension of $\mathcal{SHOIN}(\mathbf{D})$ by combining it with ICs. We show $\mathcal{SHOINK}(\mathbf{D})$-concept satisfiability w.r.t a knowledge base, guaranteeing the decidability of OWL-K. Note that in DLs, *domain of data values* is called *concrete domain*.

Table 5: Transformation of IC assertion to triples.

Abstract syntax S	Transformation - T(S)
ICAssertion(ICAssertionID description $propertyID_1$ ... $propertyID_n$)	ICAssertionID rdf:type owl:ICAssertion. ICAssertionID rdf:type rdfs:Resource. [opt] ICAssertionID owl:onClass T(description). ICAssertionID owl:byProperty T($propertyID_1$). ... ICAssertionID owl:byProperty T($propertyID_n$).

5.1 $\mathcal{SHOINK}(\mathbf{D})$

5.1.1 Datatypes

Datatypes are used to represent literal values such as numbers or strings. They compose a *concrete domain* $\mathbf{D}$ as introduced for $\mathcal{SHOQ}(\mathbf{D})$ (Horrocks and Sattler, 2001). Each datatype $d \in \mathbf{D}$ is associated with a set $d^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$, where $\Delta_{\mathbf{D}}$ is the domain of interpretation of all datatypes. For example, a datatype $\geq_{21}$ in $\mathbf{D}$ defines a set $\geq_{21}{}^{\mathbf{D}}$ of *integer* values greater than or equal to 21.

5.1.2 Syntax and Semantics

As for any DL language, the basic syntactic building blocks of $\mathcal{SHOINK}(\mathbf{D})$ are atomic *concepts*, atomic *roles*, and individuals. Concepts are interpreted as sets of individuals (subsets of the interpretation domain) and roles as sets of pairs of individuals. Expressions are then built from these basics by using several kinds of constructors. For example, the *conjunction* of concepts $C \sqcap D$ denotes the set of individuals obtained by intersecting the sets of individuals belonging to C and D.

$\mathcal{SHOINK}(\mathbf{D})$ syntax and semantics can be seen in table 1, 2 and 4, where C and D are concept descriptions; o is nominal, i.e. singleton concept; $R, R_1, ..., R_n$, S are roles; Δ^I is the interpretation domain disjoint from the concrete interpretation domain $\Delta_{\mathbf{D}}$. The inclusion relationship is denoted by $\sqsubseteq$. In number restrictions, i.e. $\leq nR$ and $\geq nR$, R is a *simple role* which does not have transitive subroles. The formal definition of $\mathcal{SHOINK}(\mathbf{D})$ can be found in (Nguyen and Le-Thanh, 2007; Nguyen and Le-Thanh, 2006).

5.2 $\mathcal{SHOINK}(\mathbf{D})$ Reasoning

The decidability of OWL-K is addressed by proposing a decision procedure for $\mathcal{SHOINK}(\mathbf{D})$. This is an extension of the algorithm introduced in (Horrocks and Sattler, 2005) by adding new expansion rules to support ICs with some refinement techniques. The algorithm is proved to be decidable (Nguyen and Le-Thanh, 2007; Nguyen and Le-Thanh, 2006).

Theorem 5.1 *The $\mathcal{SHOINK}(\mathbf{D})$-concept satisfiability problem w.r.t a knowledge base is decidable.*

Since OWL-K corresponds to $\mathcal{SHOINK}(\mathbf{D})$ DL, we have the following corollary.

Corollary 5.1 *The OWL-K concept satisfiability problem w.r.t a knowledge base is decidable.*

According to Tobies (Tobies, 2001), if $\mathcal{L}$ is a DL that provides the nominal constructor, knowledge base satisfiability can be polynomially reduced to the concept satisfiability w.r.t a knowledge base. So that we obtain the following theorem.

Theorem 5.2 *The knowledge base satisfiability problem of OWL-K is decidable.*

6 DISCUSSION AND PERSPECTIVES

We have proposed OWL-K as a decidable extension of OWL-DL. This language supports ICs, which cannot be represented in OWL-DL. The underpinning DL of the latter, $\mathcal{SHOIN}(\mathbf{D})$, is already known with the complexity of NExpTime-complete (Tobies, 2000). Consequently, OWL-K has a difficult entailment problem. The extension of the language presented here, therefore, are built like detachable components for various needs.

Considering recent works adressing the same problem, (Dou et al., 2006) introduced an ontology-based framework, OntoGrate, using the web ontology language Web-PDDL to incorporate database schemas. This language allows IC representation. However, it is written in RDF, an older formalism with a lower capability than that of OWL. Even wrapping one more layer translating Web-PDDL to OWL syntax to facilitate worldwide use, the language is undecidable because its semantics is based on First Order Logic (FOL), which is well known to be undecidable.

Some other works are discussed in (Chen et al., 2006; Kalfoglou et al., 2005). However, to the best of our knowledge, there exists no integrating framework at the database schema level that provides a formal semantic web ontology language, supporting OWL and IC representation, and at the same time affording a decidable reasoning procedure.

Our work comes within the perspective of a global project to construct a methodology of integration of relational data sources into the Semantic Web environment. This integration will be realized through the model ORM (Object Role Modeling) (Halpin, 2006). The project consists of the study of the mapping *without loss of semantics* of ORM schemas to a Web ontology language, i.e. OWL-K introduced here. Future work will turn towards the research on the automated translation of requests from OWL-K to SQL by using the above mentioned mapping.

REFERENCES

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). OWL Web Ontology Language Reference.

Chen, H., Wang, Y., Wang, H., Mao, Y., Tang, J., Zhou, C., Yin, A., and Wu, Z. (2006). Towards a Semantic Web of Relational Databases: a Practical Semantic Toolkit and an In-Use Case from Traditional Chinese Medicine. In *4th International Semantic Web Conference (ISWC'06)*, LNCS, pages 750–763, Athens, USA. Springer-Verlag. Best Paper Award.

Dou, D., LePendu, P., Kim, S., and Qi, P. (2006). Integrating Databases into the Semantic Web through an Ontology-Based Framework. In *ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)*, page 54, Washington, DC, USA. IEEE Computer Society.

Halpin, T. (2006). Object-role modeling (ORM/NIAM). In *Handbook on Architectures of Information Systems, 2nd edition*, pages 81–103. Springer, Heidelberg.

Horrocks, I. and Patel-Schneider, P. F. (2004). OWL Web Ontology Language, Semantics and Abstract Syntax. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/syntax.html.

Horrocks, I. and Sattler, U. (2001). Ontology Reasoning in the $\mathcal{SHOQ}$(D) Description Logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, Los Altos.

Horrocks, I. and Sattler, U. (2005). A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453.

Kalfoglou, Y. Hu, B., Reynolds, D., and Shadbolt, N. (2005). Semantic Integration Technologies survey. Technical report.

Nguyen, T. D. T. and Le-Thanh, N. (2006). La contrainte d'identification dans la Logique de description $\mathcal{SHOIN}(\mathbf{D})$. ISRN I3S/RR- 2006-34-FR, Laboratoire I3S, Universite de Nice Sophia-Antopolis, France.

Nguyen, T. D. T. and Le-Thanh, N. (2007). Identification constraints in $\mathcal{SHOIN}(\mathbf{D})$. In *Proc. of the 1th Int. Conf. on Research Challeges in Information Science (RCIS 2007)*. Accepted to publish.

Tobies, S. (2000). The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:199–217.

Tobies, S. (2001). *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, Rheinisch-Westfalischen Hochschule Aachen.