A DL to describe classes of queries and mapping information of an interoperable data system

J. Bermúdez A. Illarramendi
LSI depart., UPV. San Sebastián. Spain. http://siul02.si.ehu.es

Abstract

The aim of this paper is to present a DL that allows one: to express and exploit the mapping information associated with an interoperable data system, and to express classes of queries for reasoning with, in order to enhance the performance of the interoperable data system query processing.

1 Introduction

Interoperable Data Systems constitute the general framework for the ideas presented in this paper. The main goal is to provide users with a system that allows them to query a Global Information System without being aware of the site, structure, query language and semantics of the particular data repositories. Our context consists of many loosely coupled wide-opened data repositories. So, we differ from the Data Warehousing perspective where a subject oriented approach and a data replication by materialized views are used [6, 8]. Our approach is based on the use of *Ontologies* as semantic views capturing the information content of the underlying data repositories [9]. Thus, the main purpose of the ontologies is to make explicit the information content in a manner independent of the underlying data structures that are used to store the information in the data repositories. Each ontology is on top of a collection of data repositories, providing a semantic model. Ontologies contain a set of interrelated terms relevant to a particular application domain, so they provide a semantic vocabulary to formulate queries to the information system. Users should be able to deal with ontologies instead of dealing with the heterogeneous and distributed data repositories. We use a *Description Logic* (DL) language to describe terms of ontologies. We are convinced of the DL adequacy for representing enough semantics, and we rely on its reasoning capabilities. In spite of some weaknesses, DL languages deserve also consideration as query languages in our open context. We found particularly useful the classification of queries within the ontology terms [10]. For example, when users are not enterely familiar with the source contents, the system has an opportunity to help them tunning (i.e. generalizing or specializing) their queries.

Taking into account that the instances of the ontology terms are actually stored in the underlying data sources, it is necessary to access those sources in order to obtain the query answer. Therefore, there is a need to bridge the semantic distance between the expression of the query and the values of the answer. *Mapping information* is the abstraction that provides the link between the terms that appear in the ontology and the instances of these terms supported by stored data. Particularly, in our context mapping information links terms of an ontology with relational expressions defined over the logical schema of a data repository. The precise syntax and semantics used for mapping expressions is defined in [3]. In this paper, we advocate for using ABox assertions to express the existing mapping information. Notice that, from this point of view, terms and mapping expressions are structured individuals of suitable concepts defined in the TBox (e.g. `Document is-a CONCEPT`, `entitled is-a ROLE`, `the_titles is-a MAPPING_EXPRESSION`, where `CONCEPT`, `ROLE` and `MAPPING_EXPRESSION` are concepts, and `Document`, `entitled` and `the_titles` are individuals). Suitable roles are defined to represent the relevant structure of individuals (e.g. `entitled has-dom Document`) and the relevant relationships between them (e.g. `entitled is-supported-by the_titles`). Then, DL technology allows us to exploit such ABox and TBox.

Moreover, in order to enhance the efficiency of the query processing task, we propose a mechanism that sometimes avoids the query plan generation process. We use a DL to describe *query patterns*, i.e. classes of queries, which are associated with already obtained generic plans. User queries are recognized as instances of query patterns and the appropriate instantiation of the generic plan associated with is the selected query plan.

The aim of this paper is to present a DL that is able to deal with the two mentioned goals. We integrate *Concrete Domains* into our language since concrete objects with their own theory are involved, for example attributes of a logical schema with their dependency constraints, and ontology terms expressions with subsumption based relationships. We have tried to define a minimal language with enough expressivity looking for computational tractability. Unfortunately, if there are complementary predicates in the concrete domain we loose tractability in the worst case. The following section presents first, our adapted notion of admisible concrete domains, next the description logic integrating such concrete domains, and then the reasoning services we are interested in. Section 3 reviews briefly the soundness and completeness proof of our calculus and point out its complexity. Conclusions appear in section 4. The lack of space prevent us of presenting the proofs. We refer the interested reader to [2].

2 The description logic

Our TBox performs two functions: first, to declare the frame-like structures of the considered individuals. Second, to characterize classes of individuals (e.g. *query patterns*). Therefore, the TBox is split into two components with different languages and statements to achieve both goals. The first language is simpler and its statements introduce names for primitive concepts and express necessary conditions on the structure of their instances (e.g.$\mathtt{ROLE} \sqsubseteq \mathtt{TERM} \sqcap \exists \mathtt{has\_dom} \sqcap \leq 1 \mathtt{has\_dom} \sqcap \exists \mathtt{has\_ran} \sqcap \leq 1 \mathtt{has\_ran}$) . Moreover, names for atomic roles and features are introduced by expressing their domain and range (e.g. $\mathtt{has\_dom} \sqsubseteq \mathtt{ROLE} \times \mathtt{CONCEPT}$). Expressions in the second language state necessary and sufficient conditions for the instances of the classes defined. Complex roles allow the navigation of the structure of individuals (e.g. $\mathtt{ROLE} \sqcap \exists \mathtt{has\_dom} : \mathtt{NON\_PRIMITIVE} \sqcap \mathit{Subsumed\_by\_Document}(\mathtt{has\_dom})$). The languages adopted for our TBox are those presented in [5], augmented with constraints (i.e. predicates) from the concrete domain.

Predicates of concrete domains represent useful semantic relationships among components of our individuals. Informally, the idea is that predicates over concrete domains are encapsulated as oracles in separate modules and our descriptions interact with these oracles in a well defined way. Obviously, oracles add extraordinary descriptive power. However, we propose to use the oracles with caution: only for decidable predicates of the theories of items considered. For example: subsumption between terms in ontologies (or any other service provided by the DL system supporting the ontologies), functional or inclusion dependencies among tuples of attributes, and alike.

A *concrete domain* $\mathcal{D}$ consists of two sets: the domain $dom(\mathcal{D})$ and the predicate names $pred(\mathcal{D})$. Each predicate name $\mathcal{P}$ is associated with an arity n and an n-ary predicate $\mathcal{P}^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$. We adapt to our own needs the notion of admissible concrete domain given in [1]. We need to know if a predicate is true in the theory of the concrete domain, on the basis that some other predicates are true. This is the *consequence problem*, that is formalized next.

Let $\mathcal{P}_0, \ldots, \mathcal{P}_k$ be k+1 (not necessarily different) predicate names in $pred(\mathcal{D})$ of arities $n_0, \ldots, n_k$ respectively. Let $\underline{x}^{(i)}$ stands for an n_i-tuple $(x_1^i, \ldots, x_{n_i}^i)$ of variables. It is important to remark that neither all variables in one tuple nor those in different tuples are assumed to be distinct. We say that $\mathcal{P}_0(x_0)$ is a *consequence* of the finite conjunction $\bigwedge_{i=1}^k \mathcal{P}_i(\underline{x}^{(i)})$ when every assignment of elements of $dom(\mathcal{D})$ to the variables that satisfy the finite conjunction, also satisfy $\mathcal{P}_0(x_0)$.

A concrete domain $\mathcal{D}$ is *admissible* if and only if the consequence problem is decidable. Moreover, for technical reasons, we need the empty predicate *False* in $pred(\mathcal{D})$.

Different admissible concrete domains need to be considered in a real appli-

$$
\begin{array}{rcl}
(\forall P.A)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \forall e.(d,e) \in P^{\mathcal{I}} \rightarrow e \in A^{\mathcal{I}}\} \\
(\exists P)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in P^{\mathcal{I}}\} \\
(\leq 1P)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \sharp\{e \mid (d,e) \in P^{\mathcal{I}}\} \leq 1\} \\
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\{a\}^{\mathcal{I}} & = & \{a^{\mathcal{I}}\} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\exists p)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in p^{\mathcal{I}}\} \\
(\exists p = q)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in p^{\mathcal{I}} \wedge (d,e) \in q^{\mathcal{I}}\} \\
\mathcal{P}(p_1,\ldots,p_n)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists \delta_i \in dom(\mathcal{D}).(d,\delta_i) \in p_i^{\mathcal{I}}(i=1,\ldots,n) \wedge \mathcal{P}^{\mathcal{I}}(\delta_1,\ldots,\delta_n)\} \\
(P^{-1})^{\mathcal{I}} & = & \{(d,e) \mid (e,d) \in P^{\mathcal{I}}\} \\
\varepsilon^{\mathcal{I}} & = & \{(d,d) \mid d \in \Delta^{\mathcal{I}}\} \\
(R : C)^{\mathcal{I}} & = & \{(d,e) \mid (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \\
(R \cdot S)^{\mathcal{I}} & = & \{(d,e) \mid \exists f.(d,f) \in R^{\mathcal{I}} \wedge (f,e) \in S^{\mathcal{I}}\}
\end{array}
$$

Figure 1: Semantic equations

cation but, similarly to [1], we proved that a disjoint sum of admissible concrete domains constructs an admissible concrete domain. Therefore, there is no loss of generality when our presentation consider the integration of only one admissible concrete domain into the concept language.

2.1 The schema language $\mathcal{SL}(\mathcal{D})$

This language is for describing the frame-like structure of individuals in our application domain (e.g. ontology terms, mapping expressions, attribute tuples). The letter $\mathcal{D}$ represents the admissible concrete domain, letters A, B denote primitive concepts, the letter P denotes a role, that in $\mathcal{SL}(\mathcal{D})$ is always primitive, and $P_{\mathcal{D}}$ denotes features (i.e. monovalued roles). The letter C denotes $\mathcal{SL}(\mathcal{D})$ concepts according to the following syntax rule:

$$C \longrightarrow A \mid \forall P.A \mid \exists P \mid \leq 1P \mid \exists P_{\mathcal{D}}$$

The introduction of primitive concepts and roles is accomplished by a *Schema*: a finite set of *structure axioms* of the following forms, $A \sqsubseteq C$, $P \sqsubseteq A \times B$, $P_{\mathcal{D}} \sqsubseteq A \times \mathcal{D}$. Structure axioms admit cycles, we give them descriptive semantics [11]. Semantics is given by interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of the abstract domain $\Delta^{\mathcal{I}}$, that is a set disjoint from $dom(\mathcal{D})$, and the interpretation function $\cdot^{\mathcal{I}}$ that maps every primitive concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, every role P to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and every feature $P_{\mathcal{D}}$ to a partial function $P_{\mathcal{D}}^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $dom(\mathcal{D})$. In figure 1 we show the semantic equations.

2.2 The view language $\mathcal{VL}(\mathcal{D})$

This language is for specifying classes of individuals and is the vehicle for exploiting the mapping information and for driving the first steps in the query processing task. In fact, it is not an independent language; primitive $\mathcal{SL}(\mathcal{D})$

concepts and roles are basic elements for the $\mathcal{VL}(\mathcal{D})$ language. Several operators allow us to write complex $\mathcal{VL}(\mathcal{D})$ roles. Specifically, inverse of primitive $\mathcal{SL}(\mathcal{D})$ roles and role composition are the basic tools to navigate the structure of objects. The inverse operator is allowed only for primitive roles (i.e. not for features). Following is the syntax rule, where the letters R, S range over $\mathcal{VL}(\mathcal{D})$ roles:

$$R, S \longrightarrow P_{\mathcal{D}} \mid P \mid P^{-1} \mid \varepsilon \mid (R : C) \mid R \cdot S$$

The letter ε denotes the identity role. The restricted role $(R : C)$ constraints the pairs of R to those whose second component is in the $\mathcal{VL}(\mathcal{D})$ concept C. The composition of roles is expressed by $R \cdot S$.

Paths are chains of iterated role composition. Notice that the restricted role construct $(R : C)$ permits free nestings of $\mathcal{VL}(\mathcal{D})$ roles and concepts; therefore paths can be complex expressions. Letters p, q —maybe with subscripts— denote paths.

Following is the syntax rule for $\mathcal{VL}(\mathcal{D})$ concepts:

$$C, D \longrightarrow A \mid \top \mid \{a\} \mid C \sqcap D \mid \exists p \mid \exists p = q \mid \mathcal{P}(p_1, \ldots, p_n)$$

A denotes primitive $\mathcal{SL}(\mathcal{D})$ concepts, $\top$ denotes the universal concept and $\{a\}$ denotes singletons. Every individual name is associated to a different element of $\Delta^{\mathcal{I}}$ (unique name assumption). The concept constructor $\mathcal{P}(p_1, \ldots, p_n)$ connects to concrete domains. An object is in $\mathcal{P}(p_1, \ldots, p_n)$ if it admits paths $p_1, \ldots, p_n$ that reach elements of the concrete domain which satisfy predicate $\mathcal{P}$. It is worth noticing that composition of any $\mathcal{VL}(\mathcal{D})$ role is allowed in paths, not just functional roles as in concrete constraints of [1] or path agreement in [4]. Semantic equations are shown in table 1. Notice that a $\mathcal{VL}(\mathcal{D})$ interpretation is completely determined by the interpretation of individual names and primitive concepts and roles.

2.3 The Knowledge Base

As said in the introduction, we are interested in DL tools to support our approach to query processing of global information systems. We advocate for using a DL language to specify the terms of the ontologies. We consider ontologies as the semantic descriptions of the underlying data and we think they provide users with a description of the information contents and an adequate semantic vocabulary to formulate queries. In contrast, some other related works face the users with models more operational in nature, or closer to the data structures of the source model [7, 12, 15].

Moreover, we think that term definitions and their interrelations, assertions of mapping information, and other relationships established among elements participating in the process of the integration of the information, can be suitably represented as ABox assertions in a proper DL Knowledge Base. Those assertions are the base facts that conform our *World Description*, and the language to express them is the following: $A(a)$, $P(a,b)$, $P_{\mathcal{D}}(a,\delta)$ and $\mathcal{P}(\delta_1,\ldots,\delta_n)$, where a,b,δ are constants (notice that only primitive concepts and roles are used).

A $\mathcal{VL}(\mathcal{D})$ interpretation $\mathcal{I}$ satisfies $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, it satisfies $R(a,b)$ (resp. $R(a,\delta)$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}},\delta) \in R^{\mathcal{I}}$) and it satisfies $\mathcal{P}(\delta_1,\ldots,\delta_n)$ if $\mathcal{P}^{\mathcal{D}}(\delta_1,\ldots,\delta_n)$. An interpretation $\mathcal{I}$ is a *model* of a *World Description* $\mathcal{W}$ if it satisfies every assertion in $\mathcal{W}$.

The corresponding TBox includes appropriate descriptions for different categories of indiviuals (i.e. concepts, roles, mapping expressions, attributes, etc.). In our case, it is represented by a $\mathcal{SL}(\mathcal{D})$ schema. Then, a *Knowledge Base* is a pair $\Sigma = (\mathcal{S},\mathcal{W})$ where $\mathcal{S}$ is a $\mathcal{SL}(\mathcal{D})$ Schema and $\mathcal{W}$ is a World Description. A $\mathcal{VL}(\mathcal{D})$ interpretation $\mathcal{I}$ is a Σ-*model* if it is a model of $\mathcal{S}$ and $\mathcal{W}$. $\Sigma = (\mathcal{S},\mathcal{W})$ is *satisfiable* if it has a Σ-model.

We are interested in exploiting the existing mapping information and enhancing the query processing task. The foundation for both goals is the classification of $\mathcal{VL}(\mathcal{D})$ concepts that represent query patterns and the recognition of instances of $\mathcal{VL}(\mathcal{D})$ concepts, on the basis of the stated world description. Formally, we are interested in two reasoning services: 1. *Hybrid subsumption* ($\Sigma \models C \sqsubseteq_{\mathcal{S}} D$), that is the problem of checking whether every Σ-model $\mathcal{I}$ satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. And 2. *Instance checking* ($\Sigma \models C(a)$), that is the problem of checking whether every Σ-model satisfies $C(a)$. It is remarkable that when individuals are permitted in the concept language —as is our case— the assertions in the World Description $\mathcal{W}$ do affect the subsumption relation, in general the computational complexity of instance checking is different from that of subsumption, see [14, 13]. Nevertheless, within our language, instance checking is reducible to hybrid subsumption.

Lemma 1 $\Sigma \models C(\mathrm{a}) \;\Leftrightarrow\; \Sigma \models \{\mathrm{a}\} \sqsubseteq_{\mathcal{S}} C$.

3 The calculus

Due to the lack of space, this section only presents a brief review of the calculus to decide hybrid subsumption (we refer to [2] for proofs). Our calculus is a proper adaptation of the one presented in [5], so we follow the same course for the presentation and proofs.

Intuitively, to check $\Sigma \models C \sqsubseteq_{\mathcal{S}} D$, we try to construct a particular Σ-model using suitable *propagation rules* over the hypothetical presence of an element s in C, then s is checked as a member of D over the eventually developed Σ-model. The negative output brings us with a counterexample justifying $\Sigma \not\models C \sqsubseteq_{\mathcal{S}} D$, and a positive output means $\Sigma \models C \sqsubseteq_{\mathcal{S}} D$ due to the prototypical character of the interpretation constructed.

The propagation rules operate on *constraints* that resemble complex assertions in a world description. A *constraint* is a syntactic entity of one of the following forms: $s:C$, sRt, sRk, $\mathcal{P}(k_1,\ldots,k_n)$, $k:\mathcal{D}$, where C is a $\mathcal{VL}(\mathcal{D})$ concept, R is a $\mathcal{VL}(\mathcal{D})$ role, $\mathcal{P}$ is a n-ary concrete predicate, and s, t, k are variables or constants. A finite nonempty set of constraints is a *constraint system*.

We extend the semantics to constraint systems as it would be expected. Every variable is mapped to an element in $\Delta^{\mathcal{I}}$ or $dom(\mathcal{D})$ by a function α. $(\mathcal{I},\alpha)$ is a Σ-*model* of a constraint system if it satisfies every of its constraints.

A world description $\mathcal{W}$ can be translated into a constraint system $\mathrm{CS}_{\mathcal{W}}$ by replacing every assertion $A(a)$, $P(a,b)$, $P_{\mathcal{D}}(a,\delta)$ and $\mathcal{P}(\delta_1,\ldots,\delta_n)$ with the constraint $a:A$, aPb, $aP_{\mathcal{D}}\delta$ and $\mathcal{P}(\delta_1,\ldots,\delta_n)$ respectively. Furthermore, $\mathcal{W}$ and $\mathrm{CS}_{\mathcal{W}}$ are model equivalent in the sense that their models are the same.

We say that a constraint system C_1 Σ-*entails* another one C_2 ($C_1 \models_\Sigma C_2$) if every Σ-model of C_1 is a Σ-model of C_2. The following lemma states the reduction of the hybrid subsumption problem to a Σ-entailment problem.

Lemma 2 *Let $\Sigma = (\mathcal{S},\mathcal{W})$ be a satisfiable knowledge base, C and D two $\mathcal{VL}(\mathcal{D})$ concepts and x a variable. Then* $\Sigma \models C \sqsubseteq_{\mathcal{S}} D \;\Leftrightarrow\; \mathrm{CS}_{\mathcal{W}} \cup \{x:C\} \models_\Sigma \{x:D\}$.

The Σ-entailment problem $\mathrm{CS}_{\mathcal{W}} \cup \{x:C\} \models_\Sigma \{x:D\}$ is decided with the calculus. The rules of the calculus work on ordered pairs $F.G$ of constraint systems: F the *facts* and G the *goals*. Application of the rules add constraints to the facts or to the goals. Beginning with the ordered pair $\mathrm{CS}_{\mathcal{W}} \cup \{x:C\}.\{x:D\}$, the derivation procedure adds to the facts the constraints inferred by the eventual existence of an individual x in C; the definitional structure of C and schema axioms from $\mathcal{S}$ in the knowledge base Σ are the first motivation for progress, then goal constraints act as targets to guide the propagation of constraints. A control strategy for applying the rules is followed to avoid the numerical explosion of individuals. When no rule is applicable we say that the pair is *complete*. Finally, an exploration of the facts of a complete pair is sufficient to decide the answer.

The *propagation rules* are organized into five groups (see the appendix). The pair $F_C^{\mathcal{W}}.G_{\mathcal{D}}$ designates the complete pair derivable from $\mathrm{CS}_{\mathcal{W}} \cup \{x:C\}.\{x:D\}$ (that is unique up to variable renaming because all the rules are deterministic). We call o to the object allocated in D within G_D (i.e. $o:D \in G_D$).

Lemma 3 (Soundness)
(1) Given a Σ-model $(\mathcal{I},\alpha)$ of $\mathrm{CS}_{\mathcal{W}} \cup \{x:C\}$, we can construct a Σ-model $(\mathcal{I},\alpha')$ of $F_C^{\mathcal{W}}$. Furthermore, $(\mathcal{I},\alpha')$ can be constructed such that $\alpha'(o)=\alpha(x)$.
(2) $\mathrm{CS}_{\mathcal{W}} \cup \{x:C\} \models_\Sigma \{x:D\} \;\Leftrightarrow\; F_C^{\mathcal{W}} \models_\Sigma \{o:D\}$.

Constraints in the facts $F_C^{\mathcal{W}}$ of a complete pair are devised in order to build a particular Σ-model. Nevertheless, constraint systems are not necessarily Σ-satisfiable. Obvious contradictions are formalized with the notion of a *clash*. A *clash* is a constraint system having one of the following forms:

1. $\{a : \{b\}\}$ with $a \neq b$.
2. $\{sPa, sPb, s : A\}$ with $A \sqsubseteq \leq 1P \in \mathcal{S}$ and $a \neq b$.
3. $\{\mathcal{P}_1(\underline{k}^{(1)}), \ldots, \mathcal{P}_n(\underline{k}^{(n)})\}$ with $\bigwedge_{i=1}^{n} \mathcal{P}_i(\underline{k}^{(i)}) \models False$.
4. $\{s : \top, s : \mathcal{D}\}$

It is evident that a constraint system containing a clash is not Σ-satisfiable. We say that a constraint system is *clash-free* if it does not contain a clash. From a clash-free constraint system CS we build a *canonical interpretation pair* $(\mathcal{I}_{\mathrm{CS}}, \alpha)$.

Lemma 4 (Completeness) *If $F_C^{\mathcal{W}}$ is a clash-free constraint system then:*
(1) The canonical interpretation $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ is a Σ-model of $F_C^{\mathcal{W}}$.
(2) If, in addition, $s{:}E$ is a constraint in G_D then:
$(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ satisfies $s{:}E \;\Rightarrow\; s{:}E \in F_C^{\mathcal{W}}$.

Theorem 1 *Let $F_C^{\mathcal{W}}.G_D$ be a complete pair derived by the calculus from $\mathrm{CS}_{\mathcal{W}} \cup \{x{:}C\}.\{x{:}D\}$ and let $o{:}D \in G_D$:*

$$\mathrm{CS}_{\mathcal{W}} \cup \{x{:}C\} \models_\Sigma \{x{:}D\} \;\Leftrightarrow\; o{:}D \in F_C^{\mathcal{W}} \textit{ or } F_C^{\mathcal{W}} \textit{ contains a clash.}$$

Proof: $\Rightarrow$ Suppose that $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\} \models_\Sigma \{x : D\}$, $F_C^{\mathcal{W}}$ is clash-free and $o : D \notin F_C^{\mathcal{W}}$. By part (2) of lemma 4, $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ does not satisfy $o : D$ (i.e. $\alpha(o) \notin D^{\mathcal{I}_{F_C^{\mathcal{W}}}}$). By part (1) of lemma 4, $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ is a Σ-model of $F_C^{\mathcal{W}}$ and constructing and assignment function α' equal to α except $\alpha'(x) = \alpha(o)$, we have that $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha')$ is a Σ-model of $\mathrm{CS}_{\mathcal{W}} \cup \{x{:}C\}$. Thus, $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha')$ satisfies $x{:}D$ contradicting $\alpha(o) \notin D^{\mathcal{I}_{F_C^{\mathcal{W}}}}$.

$\Leftarrow$ If $o{:}D \in F_C^{\mathcal{W}}$ then $F_C^{\mathcal{W}} \models_\Sigma o{:}D$, and part (2) of lemma 3 brings us the conclussion. Suppose that $F_C^{\mathcal{W}}$ contains a clash and $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\} \not\models_\Sigma \{x : D\}$. Therefore, it exists a Σ-model $(\mathcal{I}, \alpha)$ of $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\}$ and, by part (1) of lemma 3, we can construct a Σ-model $(\mathcal{I}, \alpha')$ of $F_C^{\mathcal{W}}$ getting a contradiction. $\square$

Finally we point out the computational complexity of hybrid subsumption in our language. The concrete predicate construct in $\mathcal{VL}(\mathcal{D})$ allows to express a limited form of negation if there exist complementary predicates in the concrete domain. For example, predicates on numbers $\leq\_18(x)$ and $>\_19(x)$. Then, according to [5], the $\mathcal{VL}(\mathcal{D})$ concepts $\mathtt{PERSON} \sqcap \leq\_18(age_{\mathcal{D}})$ and $\mathtt{PERSON} \sqcap >\_19(age_{\mathcal{D}})$ with $\mathtt{PERSON} \sqsubseteq \exists age_{\mathcal{D}}$ leads to a co-NP-hard hybrid subsumption. This can be proved using the results in [13].

4 Conclusions

In this paper we have presented a description logic, to be used in the framework of Interoperable Information Systems, with the following two goals: 1. exploiting the existing mapping information between terms of an ontology and values in data sources, and 2. enhancing the query processing task. The required reasoning is founded on the hybrid subsumption problem, which is proved to be co-NP-hard relative to the consequence problem in the integrated concrete domain if there are complementary predicates in the concrete domain.

References

[1] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. Technical report, DFKI Research Report RR-91-10, 1991.

[2] J. Bermúdez and A. Illarramendi. A description logic with concrete domains for a metaclass level of an interoperable data system. Technical report, UPV/EHU/LSI/TR 07-2000, San Sebastián, Spain, May 2000. http://siul02.si.ehu.es/PUBLICATIONS/tr07-2000.ps.

[3] J.M. Blanco, A. Goñi, and A. Illarramendi. Mapping among Knowledge Bases and Databases: Precise definition of its syntax and semantics. *Information Systems*, 24(4):275–301, 1999.

[4] R. J. Brachman, D. L. McGuiness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living With Classic: When and How to Use a KL-ONE-like language. In J.F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann, San Mateo, CA, 1991.

[5] M. Buchheit, M. A. Jeusfeld, W. Nutt, and M. Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A Principled Approach to Data Integration and Reconciliation in Data Warehousing. In *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99). Heidelberg. Germany. CEUR Workshop Proceedings*, volume 19, pages 16/1–16/11, 1999.

[7] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of het-

erogeneous information sources. In *Proc. of the 10th IPSJ, Tokyo, Japan*, 1994.

[8] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The stanford data warehousing project. *IEEE Data Engineering Bulletin*, 1995.

[9] E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed And Parallel Databases (DAPD).*, 8(2):223–272, 2000.

[10] E. Mena, V. Kashyap, A. Illarramendi, and A. Seth. Imprecise answers on highly open and distributed environments: An approach based on information loss for multi-ontology based query processing. To be published in International Journal of Cooperative Information Systems (IJCIS).

[11] B. Nebel. Terminological cycles: Semantics and computational properties. In J.F.Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann, San Mateo, CA, 1991.

[12] M. Tork Roth and P. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. of the 23rd VLDB Conference, Athens, Greece*, pages 266–275, 1997.

[13] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

[14] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13:141–176, 1994.

[15] A. Tomasic, L. Raschid, and P. Valduriez. Scaling Access to Heterogeneous Data Sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, 10(5), September/October 1998.