A description logic with concrete domains for a metaclass level of an interoperable data system

J. Bermúdez A. Illarramendi
Departamento de Lenguajes y Sistemas Informáticos.
Universidad del País Vasco UPV-EHU.
San Sebastián. http://siul02.si.ehu.es

Abstract

The meta level notion has been applied to different contexts. In this paper we present its application to an interoperable system. The incorporated meta level mechanism permits to improve the management of the system by: 1. allowing to exploit the existing mapping information and, 2. enhancing the query processing task through the use of query patterns. Mapping information is the abstraction that provides the link between the terms that appear in the semantic view (class level) offered to the user of the interoperable system and the instances of these terms actually supported by stored data (instance level) in distributed and heterogeneous data sources. Our mechanism allows one to describe mapping expressions and reasoning with them. Query patterns are associated to already obtained generic plans. Our mechanism recognizes a user query (description in the class level) as an instance of a query pattern (description in the metaclass level). Then, a proper instantiation of the correspondent generic plan solves the plan generation step of the query processing task.

Dealing with a meta level, one aspect that requires a particular attention is the kind of language selected to specify metaclasses. This paper presents a logic-based language that allows the specification of structural and semantic constraints of objects that belong to the class level. Our proposal is founded on the *Description Logics* (DL) technology. We have defined a DL with two combined languages: one to specify the schema part in the metaclass level, and another for the view part. The description of semantic constraints on class level objects is achieved by the integration of suitable predicates of *Concrete domains* into the DL language. The trade-off between expressivity and computational complexity of reasoning has been taken into account when selecting the adequate constructors for those languages.

1 Introduction

Computer networks make possible accessing to a large number of heterogeneous, autonomous and distributed information sources. It is desirable that those information sources maintain complete autonomy but —at the same time— their locations, data models, contents and query interfaces should not burden the users posing queries. Several works, [13, 12, 1, 20, 21], have addressed the previous problem on the basis of a shared framework to overcome those difficulties found by the users. The goal of all such works is to provide users with an interface to access the sources in such a manner that would permit them to focus just on the expression of the query in terms of a known vocabulary, presuming their knowledge of the application domain. The general problems that must be solved in order to accomplish the pursued goal are:

- The generation of a semantic view over the underlying data sources.

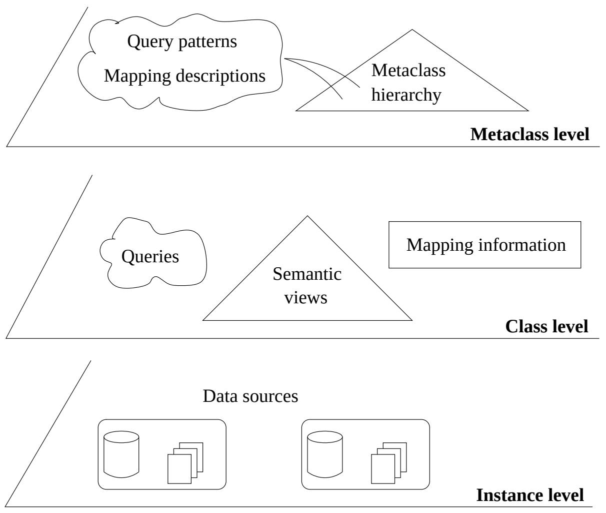

Figure 1: Three levels architecture

- The specification of a mapping information that relates the semantic view with the underlying data sources.
- The definition of a query processing strategy that allows answering to queries formulated over the semantic view by accessing the corresponding data sources.

The goal of this paper is to present a description logic that we have defined with a two fold purpose: 1. to exploit mapping information that links semantic views with data sources and, 2. to manage query patterns in order to enhance the query processing task performance. The elements of the metaclass level describe queries and mapping expressions from the class level bellow. So, in the level hierarchy we find first, the *instance level*, corresponding to the elements of the data sources. Then, the *class level*, corresponding to the elements of the semantic views (defined over the data sources) plus queries and mapping information. Finally, the *metaclass level*, corresponding to specifications describing objects in the class level and characterizations of query patterns and, in general, queries over the class level (see figure 1).

Dealing with a meta level, one aspect that requires a particular attention is the kind of language selected to specify metaclasses. This paper presents a logic-based language that allows not only the specification of structural constraints of objects in the class level, but of semantic constraints too. The trade-off between expressivity and computational tractability has been taken into account when selecting the adequate constructors for that language. Our proposal is founded on the *Description Logics* technology [26, 17, 15]. Description Logic systems are knowledge representation systems founded on Description Logics (DL), which are fragments of first order logic. The general architecture of a DL knowledge base is composed of two levels of knowledge: the terminological level and the assertional level. The terminological level, usually called TBox, is used for the introduction of classes of individuals and their mutual relationships, called *concepts* and *roles* in this context. Concepts and roles can be seen as unary and binary predicates respectively. DLs provide a family of languages by means of selected constructors to build concept and role expressions (table 1 shows some of them with their set theoretic semantics). The TBox represents intensional knowledge, it is the schema counterpart of a semantic model. The assertional level, usually called ABox, contains membership assertions of individuals to concepts and pairs of individuals to roles. The ABox represents extensional knowledge, aiming an instantiation of the schema.

There are several services that can be accomplished by a DL knowledge base. The most characteristic one is *classification*: the automatic process of inserting a concept into the

CONCEPT	INTERPRETATION
ANYTHING	$\Delta^{\mathcal{I}}$
NOTHING	$\emptyset$
and$(C,\, D)$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
or$(C,\, D)$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
not(C)	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
all$(r,\, C)$	$\{d \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
some$(r,\, C)$	$\{d \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
at-least$(\mathrm{n},\, r)$	$\{d \in \Delta^{\mathcal{I}} \mid \sharp(r^{\mathcal{I}}(d)) \geq n\}$
at-most$(\mathrm{n},\, r)$	$\{d \in \Delta^{\mathcal{I}} \mid \sharp(r^{\mathcal{I}}(d)) \leq n\}$
one-of$(a_1,\ldots,a_n)$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
same-as$(r,\, s)$	$\{d \in \Delta^{\mathcal{I}} \mid r^{\mathcal{I}}(d) = s^{\mathcal{I}}(d)\}$
fills$(r,\, a)$	$\{d \in \Delta^{\mathcal{I}} \mid a^{\mathcal{I}} \in r^{\mathcal{I}}(d)\}$

ROLE	INTERPRETATION
IDENTITY	$\{(d, d) \mid d \in \Delta^{\mathcal{I}}\}$
domain(C)	$C^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
range(C)	$\Delta^{\mathcal{I}} \times C^{\mathcal{I}}$
role-and$(r,\, s)$	$r^{\mathcal{I}} \cap s^{\mathcal{I}}$
role-or$(r,\, s)$	$r^{\mathcal{I}} \cup s^{\mathcal{I}}$
role-not(r)	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \backslash r^{\mathcal{I}}$
inverse(r)	$\{(d, d') \mid (d', d) \in r^{\mathcal{I}}\}$
restrict$(r,\, C)$	$\{(d, d') \in r^{\mathcal{I}} \mid d' \in C^{\mathcal{I}}\}$
compose$(r,\, s)$	$r^{\mathcal{I}} \circ s^{\mathcal{I}}$

Table 1: Concept and role constructors

hierarchy (*subsumption*) of concepts maintained in the TBox. Another important service is *recognition*: to derive the concepts to which an individual belongs due to the facts asserted in the ABox. Query answering is directly related to that service.

We have tried to select a minimal DL with enough expressivity looking for computational tractability. Unfortunately, if there are complementary predicates in the *concrete domain* we loose tractability in the worst case. Finally, we mention some works that, although they pursue different goals, they deal with subjects to which the notion of a meta level can be applied. In [10] it is studied the problem of modeling semi-structured data. The CDL-Schemas proposed might be an alternative language for a metaclass level. Nevertheless, subsumption is EXPTIME-hard and semantic relationships between any components of a structured object cannot be expressed. In [5, 2] it is investigated the matching of concept descriptions, a new inference service for maintaining DL knowledge bases. Patterns are specified as descriptions with variables. A pattern might be considered as a metaclass concept whose individuals are those concepts that are successfully matched by the pattern. A notion of subsumption between patterns is missing. [19] deals with the learning of concepts, i.e. select a concept description that satisfies a set of observations. A combination of observations is a metaclass concept. Observations involve subsumption, membership and part-of relations. Metaclass concepts have normal form but nothing is said about their subsumption relationships.

The remainder of this paper is structured as follows. Section 2 explains briefly the framework of this work. Section 3 presents the notion of admisible concrete domains, the description logic integrating such concrete domains, and the reasoning services we are interested in. Section 4 is dedicated to the soundness and completeness proof of our calculus and analyze its complexity. Section 5 presents the conclusions.

2 Framework

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 sources. Our approach is based on the use of *Ontologies* as semantic views capturing the information content of the underlying data sources [21]. 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 sources. 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 sources. We use a *Description Logic* (DL) language to describe terms of ontologies. DL technology provides well known advantages [11]. In spite of some weaknessess, DL languages deserve consideration as query languages in this context. In this paper, user queries are DL expressions.

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[1]. The precise syntax and semantics used for mapping expressions is defined in [4]. 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`). 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.

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. Since concrete objects with their own theory are involved, for example attributes with their dependency constraints, and ontology terms expressions with subsumption based predicates, we integrate *Concrete Domains* into our language. Hybrid subsumption is co-NP-hard relative to the consequence problem for the concrete domain.

3 The languages

This section introduce two abstract languages for the metaclass level: one to specify primitive concepts and roles, that will form the schema part of the metaclass level, and another to characterize defined concepts, that will compose the view part. Two guidelines have determined the design of both languages: sufficient expressivity for modeling relevant information concerning our two goals —efficient detection of query plans and mapping information management— and maintaining bounded complexity of reasoning as possible.

We assume that queries posed to an Ontology are descriptions in a concept language. We want to define a hierarchy of query patterns in such a way that instances of each pattern may be answered using the same generic plan. Consequently our metaclass descriptions pay attention to relevant features accounting for plan generation: operators constructing the query, semantic interrelationships of its components, and location, capabilities or properties of the data sources supporting actual extensions of the concepts and roles considered by the query. Concerning mapping information management, a similar kind of information is in fact relevant: structure of mapping expressions, interrelationships among components of those expressions and properties about repositories and sources. Then, we conclude that the selected structural

[1]Data repositories are composed by interrelated data sources. They are independent data pools with specific data organization

descriptors are useful to both objectives.

With respect to complexity of reasoning, the pioneering work [8] showed the computational cliff encountered when apparently minor changes are made to the representation language. In [16, 18, 14, 24] languages with reasoning problems from polynomial to undecidable are reported. With this in mind, we have included in the languages just the necessary to satisfy the goal.

Summarizing, we use concept constructors that focus primarily on structural properties. Structure is described specifying several kinds of constraints on roles. Different groupings of various constraints provide different expressiveness and complexity of reasoning, in particular [24] shows the undecidability caused by the *role value map*[2] operator discussed in [6]. Our interest in not discarding tractability have guided us to select a collection of constructors, used in a different context of query optimization [9], that were proved polynomial. However, those constructors are not enough to express the interesting semantic relationships in which components of objects from our class level are involved, so we extend that language with existential quantification of those interesting relationships. Certainly, there are relationships that are difficult to express in description logics, e.g. functional dependencies between attributes, or merely it is rewardless to do so because reasoning with such expressions is too hard and by no means better than using the specific theories suitably developed and efficiently implemented for that purpose. To overcome those difficulties we follow the steps defined on [3] to integrate *concrete domains* into our language. The predicates of these concrete domains represent the useful semantic relationships. Informally, the idea is that predicates over concrete domains are encapsulated as oracles in separate modules and metaclass 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 considered about items in our class level. 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.

Let us formalize the notion of concrete domain and specify the properties needed for its successful integration.

Definition 3.1 *A* concrete domain $\mathcal{D}$ *consists of a set* $dom(\mathcal{D})$*, the domain of* $\mathcal{D}$*, and a set* $pred(\mathcal{D})$*, the predicate names of* $\mathcal{D}$*. Each predicate name* $\mathcal{P}$ *is associated with an arity* n *and an* n*-ary predicate* $\mathcal{P}^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$*.*

The following are examples of concrete domains. The concrete domain $\mathcal{NAT}$, with the set of natural numbers as its domain and with predicates $>$, $<$ and $=$. Let DB be a relational database and QL an appropriate query language for it. The set of values of that database may be considered the domain of the concrete domain $\mathcal{DB}$. All the relations that can be defined with the query language QL over DB are the predicates. Let DB be, again, a relational database and AT the set of attributes defined in DB. The concrete domain $\mathcal{AT\_TUPLES}$ is defined as follows: the domain is the set of tuples of AT, and the only predicate is the functional dependency predicate between tuples in AT.

Adequate interaction with concrete domains requires a suitable property of concrete domains to be able to implement oracles. We need to know if a proposition is true in the theory of the concrete domain, on the basis that some other propositions 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 displayed finite conjunction

[2]It is the **same-as** constructor of table 1

$$\bigwedge_{i=1}^{k} \mathcal{P}_i(\underline{x}^{(i)}) \models \mathcal{P}_0(x_0)$$

when every assignment of elements of $dom(\mathcal{D})$ to the variables that satisfy the finite conjunction, also satisfy $\mathcal{P}_0(x_0)$. Effective procedures must exist to compute the answer to the *consequence problem.*

Definition 3.2 *A concrete domain $\mathcal{D}$ is* admissible[3] *if and only if the consequence problem is decidable. Moreover, for technical reasons, we need the empty predicate* False *in pred($\mathcal{D}$), or one way to express a contradiction by finite conjunctions of predicates.*

For example, the previously defined concrete domains $\mathcal{NAT}$ and $\mathcal{AT\_TUPLES}$ are admissible. Admissibility of $\mathcal{DB}$ depends on the query language QL. In addition, various admissible concrete domains may be jointly integrated. Informally, a disjoint sum of admissible concrete domains constructs an admissible concrete domain.

Definition 3.3 *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be admissible concrete domains with dom($\mathcal{D}_1$) disjoint from dom($\mathcal{D}_2$) and pred($\mathcal{D}_1$) disjoint from pred($\mathcal{D}_2$). Then the concrete domain $\mathcal{D}_1 \oplus \mathcal{D}_2$ is constructed as follows:*

$$\begin{aligned} dom(\mathcal{D}_1 \oplus \mathcal{D}_2) &= dom(\mathcal{D}_1) \cup dom(\mathcal{D}_2) \\ pred(\mathcal{D}_1 \oplus \mathcal{D}_2) &= pred(\mathcal{D}_1) \cup pred(\mathcal{D}_2) \end{aligned}$$

and each predicate name $\mathcal{P}$ maintaining its previous n-arity and associated to the corresponding n-ary predicate $\mathcal{P}^{\mathcal{D}_1 \oplus \mathcal{D}_2} = \mathcal{P}^{\mathcal{D}_1}$ if $\mathcal{P} \in$ pred($\mathcal{D}_1$), $\mathcal{P}^{\mathcal{D}_2}$ if $\mathcal{P} \in$ pred($\mathcal{D}_2$).

Lemma 3.1 *If $\mathcal{D}_1$ and $\mathcal{D}_2$ are admissible concrete domains then $\mathcal{D}_1 \oplus \mathcal{D}_2$ is an admissible concrete domain.*

Proof: Obviously, it is possible to express a contradiction with predicates in $pred(\mathcal{D}_1 \oplus \mathcal{D}_2)$.

Let the pair $\bigwedge_{i=1}^{k} \mathcal{P}_i(\underline{x}^{(i)})$ and $\mathcal{P}_0(x_0)$ be an instance of the consequence problem with predicate names in $pred(\mathcal{D}_1 \oplus \mathcal{D}_2)$. Assume, without loss of generality, that $\mathcal{P}_0 \in pred(\mathcal{D}_1)$. Let S be the set of predicate names from $\mathcal{P}_1, \ldots, \mathcal{P}_k$ that are in $pred(\mathcal{D}_1)$. Then $\mathcal{P}_0(x_0)$ is a consequence of $\bigwedge_{i=1}^{k} \mathcal{P}_i(\underline{x}^{(i)})$ if and only if $\mathcal{P}_0(x_0)$ is a consequence of $\bigwedge_{\mathcal{P} \in S} \mathcal{P}(x^{(i)})$, and the latter is decidable because $\mathcal{D}_1$ is admissible. □

Certainly, the disjoint sum construction $\oplus$ of a finite set of admissible concrete domains is analogous. Therefore, there is no loss of generality when our presentation consider the integration of only one admissible concrete domain into the language.

Next we introduce the two languages for the metaclass level: first the schema language $\mathcal{SL}(\mathcal{D})$ and second the view language $\mathcal{VL}(\mathcal{D})$.

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

The schema language $\mathcal{SL}(\mathcal{D})$ permits the description of the abstract structure of elements in the class level such as terms of the DL used to specify the ontologies, mapping expressions and user queries; $\mathcal{D}$ is the name of the admissible concrete domain. From now on we will look at the class level as filled with structured individuals that are members of the concepts in the metaclass level. Concepts in $\mathcal{SL}(\mathcal{D})$ are built out of *primitive concepts* and three kinds of *role constraints*: range restriction to primitive concepts, existential quantification and single valuation. In addition, the existence of features with values on the concrete domain $\mathcal{D}$. Primitive concepts are used for specifying different categories of elements within the

[3]We adapt the definition of admissible given in [3]. We do not need that names of predicates be closed under negation, neither the existence of a universal predicate.

class level. Usually those categories are not fully characterized and only necessary properties of their elements are expressed. We assume three alphabets of symbols: one for primitive concepts, another for primitive roles, and the last for features (monovalued roles). The 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. The letter C —maybe with subscripts— denotes $\mathcal{SL}(\mathcal{D})$ concepts according to the following syntax rule:

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

An individual belongs to a range restriction $\forall P.A$ when every individual related with it by P is in concept A. An object is a member of $\exists P$ (resp. $\exists P_\mathcal{D}$) if there is an object related with it by P (resp. $\exists P_\mathcal{D}$). When the number of individuals related by P is less than or equal to one then it belongs to $\leq 1P$. Following, we present a set theoretic formalization of the semantics.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ for $\mathcal{SL}(\mathcal{D})$ 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 $P_\mathcal{D}$ to a partial function from $\Delta^\mathcal{I}$ to $dom(\mathcal{D})$. The interpretation function is extended to concept expressions as follows[4]:

$$\begin{aligned}
(\forall P.A)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \forall e.(d,e) \in P^\mathcal{I} \rightarrow e \in A^\mathcal{I}\} \\
(\exists P)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \exists e.(d,e) \in P^\mathcal{I}\} \\
(\leq 1P)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \sharp\{e \mid (d,e) \in P^\mathcal{I}\} \leq 1\}
\end{aligned}$$

The introduction of primitive concepts and roles in the metaclass level is accomplished by a finite set of *structure axioms* of the following forms:

$$A \sqsubseteq C \qquad P \sqsubseteq A \times B \qquad P_\mathcal{D} \sqsubseteq A \times \mathcal{D}$$

which respectively assert that any member of A is in C, role P has domain A and range B, and feature $P_\mathcal{D}$ has domain A. Hence, they specify necessary properties of members of primitive concepts and roles. Structure axioms admit cycles; i.e. concept name A may appear in concept expression C or in axioms introducing concept names that appear in C.

We will write

$$A \sqsubseteq C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$$

to abbreviate the set of axioms $A \sqsubseteq C_1$, $A \sqsubseteq C_2, \ldots$, $A \sqsubseteq C_n$. Conjunction is not included as a concept constructor to simplify the rules of the calculus presented in section 4.

An interpretation $\mathcal{I}$ *satisfies* an axiom $A \sqsubseteq C$ if $A^\mathcal{I} \subseteq C^\mathcal{I}$. $\mathcal{I}$ satisfies an axiom $P \sqsubseteq A \times B$ (resp. $P_\mathcal{D} \sqsubseteq A \times \mathcal{D}$) if $P^\mathcal{I} \subseteq A^\mathcal{I} \times B^\mathcal{I}$ (resp. $P_\mathcal{D}^\mathcal{I} \subseteq A^\mathcal{I} \times dom(\mathcal{D})$). A finite set of structure axioms $\mathcal{S}$ is called a *Schema*. An interpretation that satisfies all axioms of a schema $\mathcal{S}$ is called a *model* of $\mathcal{S}$. The Schema restricts the interpretations that we consider.

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

Next we show the *view language* $\mathcal{VL}(\mathcal{D})$, that is not an independent language indeed. Primitive concepts and roles from the *schema language* $\mathcal{SL}(\mathcal{D})$ are the building blocks for $\mathcal{VL}(\mathcal{D})$ concepts and roles. The view language $\mathcal{VL}(\mathcal{D})$ is devoted to characterize metaclasses specifying necessary and sufficient properties of their members in the class level. The properties considered are structural conditions and semantic constraints over parts of the objects. The

[4]For a homogeneous treatment, we write $(d,e) \in P_\mathcal{D}^\mathcal{I}$ meaning $P_\mathcal{D}^\mathcal{I}(d) = e$. In the semantic equations we only refer to P

$\mathcal{SL}(\mathcal{D})$ schema shows the structure of objects in the class level; $\mathcal{SL}(\mathcal{D})$ primitive roles play a featuring part. Now, several operators allow us to write complex $\mathcal{VL}(\mathcal{D})$ roles. Specifically, inverse of abstract 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. The reason to avoid inverses of concrete valued features is that the concept language is not for defining classes in a concrete domain. Following is the syntax for $\mathcal{VL}(\mathcal{D})$ roles:

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

where the letters R, S range over $\mathcal{VL}(\mathcal{D})$ roles. The letter P denotes a primitive $\mathcal{SL}(\mathcal{D})$ role, P^{-1} is its inverse. 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 $R \cdot S$. Formal semantics is specified after the presentation of the syntax for concepts.

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

As previously mentioned, $\mathcal{SL}(\mathcal{D})$ primitive concepts are basic elements for the $\mathcal{VL}(\mathcal{D})$ concept language. In addition there also exist the universal concept, singleton concepts, intersection of concepts, existential quantification of paths, and existential constraints over paths. The constraints are the equality predicate plus any predicate from the concrete domain $\mathcal{D}$. Following is the syntax rule, where C and D denote $\mathcal{VL}(\mathcal{D})$ concepts, A denotes primitive $\mathcal{SL}(\mathcal{D})$ concepts, a represents names for individuals and $\mathcal{P}$ denotes names for predicates of the concrete domain $\mathcal{D}$:

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

Informally, a class object may be viewed as a rooted directed graph with inner nodes labeled with metaclasses, arcs labeled with roles, and nodes without leaving arcs labeled with names of individuals or elements of a concrete domain. Role paths have the obvious graphical representation. An object is in $\exists p$ if the path p, starting at the root, matches the structure of the object and there is an object reached by such a path. The objects in $\exists p = q$ are those on which paths p and q can be satisfactorily followed from the root and agree on the object reached following them. The concept constructor $\mathcal{P}(p_1, \ldots, p_n)$ allows the connection with concrete domains; it formalizes the way to interact with those oracles cited at the beginning of this section. 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 [3] or path agreement in [7].

$\mathcal{SL}(\mathcal{D})$ interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ are adapted to be $\mathcal{VL}(\mathcal{D})$ *interpretations*. The interpretation function $\cdot^{\mathcal{I}}$ is extended to associate a different element of $\Delta^{\mathcal{I}}$ to each different individual name (a restriction usually called *Unique Name Assumption*); concrete predicates names are interpreted as their associated predicates $\mathcal{P}^{\mathcal{I}} = \mathcal{P}^{\mathcal{D}}$ and concept and role expressions are interpreted as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\exists p)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in p^{\mathcal{I}}\} \\
(\exists p = q)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in p^{\mathcal{I}} \wedge (d,e) \in q^{\mathcal{I}}\} \\
\mathcal{P}(p_1, \ldots, p_n)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists \delta_i \in dom(\mathcal{D}).(d, \delta_i) \in p_i^{\mathcal{I}} (i = 1, \ldots, n) \wedge \mathcal{P}^{\mathcal{I}}(\delta_1, \ldots, \delta_n)\}
\end{aligned}$$

$$\begin{array}{rcl}(P^{-1})^{\mathcal{I}} & = & \{(d,e) \mid (e,d) \in P^{\mathcal{I}}\} \\ \varepsilon^{\mathcal{I}} & = & \{(d,d) \mid d \in \Delta^{\mathcal{I}}\} \\ (R:C)^{\mathcal{I}} & = & \{(d,e) \mid (d,e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \\ (R \cdot S)^{\mathcal{I}} & = & \{(d,e) \mid \exists f.(d,f) \in R^{\mathcal{I}} \wedge (f,e) \in S^{\mathcal{I}}\}\end{array}$$

Notice that a $\mathcal{VL}(\mathcal{D})$ interpretation is completely determined by the interpretation of individual names, primitive concepts, primitive roles, and features.

Defined concepts are introduced through *view axioms*:

$$V = C$$

where V is a name and C is a $\mathcal{VL}(\mathcal{D})$ expression. There is only one definition for each name V and cycles are not allowed in view axioms. Thus V is an abbreviation for C. Then, the interpretation of a name V is $V^{\mathcal{I}} = C^{\mathcal{I}}$. A finite set of view axioms $\mathcal{V}$ is called a *View Taxonomy*.

Let $\mathcal{S}$ be a $\mathcal{SL}(\mathcal{D})$ schema. We say that C is $\mathcal{S}$-*subsumed* by D

$$C \sqsubseteq_{\mathcal{S}} D$$

when $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ that is model of $\mathcal{S}$. With a schema and a view taxonomy we have a terminology (TBox) in the metaclass level. That terminology represents intensional knowledge. In the next subsection we formalize the assertions of the ABox which represent extensional knowledge.

3.3 The world description

In our context, facts that conform a world description are stated through explicit knowledge about structure and relationships of items in the class level. These assertions are stated when concepts, roles, user queries or mapping informations are introduced into our class level. In abstract, the language for these assertions is the following:

$$A(a) \qquad P(a,b) \qquad P_{\mathcal{D}}(a,\delta) \qquad \mathcal{P}(\delta_1,\ldots,\delta_n)$$

where A is a primitive concept, P is a primitive role, $P_{\mathcal{D}}$ is a feature, a, b are individuals, δ and δ_i's are elements of the concrete domain and $\mathcal{P}$ is a concrete n-ary predicate. Informally, $A(a)$ asserts that a is an instance of concept A, $P(a,b)$ (resp. $P_{\mathcal{D}}(a,\delta)$) asserts that a is related to b through role P (resp.a, δ and $P_{\mathcal{D}}$), and $\mathcal{P}(\delta_1,\ldots,\delta_n)$ states a base fact about predicate $\mathcal{P}$.

A $\mathcal{VL}(\mathcal{D})$ interpretation $\mathcal{I}$ satisfies $A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, it satisfies $P(a,b)$ (resp. $P_{\mathcal{D}}(a,\delta)$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$ (resp. $P_{\mathcal{D}}^{\mathcal{I}}(a^{\mathcal{I}}) = \delta$) and it satisfies $\mathcal{P}(\delta_1,\ldots,\delta_n)$ if $\mathcal{P}^{\mathcal{I}}(\delta_1,\ldots,\delta_n)$.

A *World Description* $\mathcal{W}$ is a finite set of those assertions. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{W}$ if it satisfies every assertion in $\mathcal{W}$.

A *Knowledge Base* is a triple $\Sigma = (\mathcal{S},\mathcal{V},\mathcal{W})$ where $\mathcal{S}$ is a $\mathcal{SL}(\mathcal{D})$ Schema, $\mathcal{V}$ is a $\mathcal{VL}(\mathcal{D})$ View Taxonomy, 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 $\mathcal{S}$, $\mathcal{V}$ and $\mathcal{W}$. $\Sigma = (\mathcal{S},\mathcal{V},\mathcal{W})$ is *satisfiable* if it has a Σ-model.

We are interested in classifying $\mathcal{VL}(\mathcal{D})$ concepts that represent query patterns and in recognizing instances of $\mathcal{VL}(\mathcal{D})$ concepts. Since definitions in the View Taxonomy are not cyclic we only pay attention to $\mathcal{VL}(\mathcal{D})$ concept expressions, thus the component $\mathcal{V}$ of a knowledge base is taken out of our considerations. However, in our context, the two other components of a knowledge base are relevant for the cited tasks. The Schema $\mathcal{S}$ may contain cycles and the descriptive semantics given to $\mathcal{S}$ restrict the models of the corresponding knowledge base. Furthermore, it is well known 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 [23]. Formally, we are interested in the following reasoning services.

Definition 3.4 *Let* $\Sigma = (\mathcal{S}, \mathcal{V}, \mathcal{W})$ *be a knowledge base,* C *and* D *concepts from* $\mathcal{VL}(\mathcal{D})$ *and* a *an individual.*

(1) Σ logically implies $C \sqsubseteq_{\mathcal{S}} D$ ($\Sigma \models C \sqsubseteq_{\mathcal{S}} D$) *if every* Σ*-model* $\mathcal{I}$ *satisfies* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Hybrid subsumption *is the problem of checking whether* $\Sigma \models C \sqsubseteq_{\mathcal{S}} D$.

(2) Σ logically implies $C(\mathrm{a})$ ($\Sigma \models C(\mathrm{a})$) *if every* Σ*-model satisfies* $C(\mathrm{a})$. Instance checking *is the problem of checking whether* $\Sigma \models C(\mathrm{a})$.

Hybrid subsumption is the basic operation to the more complex service of classification, i. e. given $\Sigma = (\mathcal{S}, \mathcal{V}, \mathcal{W})$ and C, to determine whether $\Sigma \models D \sqsubseteq_{\mathcal{S}} C$ or $\Sigma \models C \sqsubseteq_{\mathcal{S}} D$ for every concept D defined in $\mathcal{V}$. Instance checking is the basic operation for retrieving all the individuals which are instances of a concept.

Notice that, when concept languages include individuals, terminological subsumption ($C \sqsubseteq_{\mathcal{S}} D$) implies hybrid subsumption ($\Sigma \models C \sqsubseteq_{\mathcal{S}} D$) but the converse direction does not hold. Moreover, the qualified existential quantification operator **some**(R,C)[5] may enforce instance checking to be strictly computationally harder than terminological subsumption, see [22]. Nevertheless we show that, within our language, instance checking is reducible to hybrid subsumption.

Lemma 3.2 *Let* Σ *be a knowledge base,* C *a* $\mathcal{VL}(\mathcal{D})$ *concept and* a *an individual. Then*

$$\Sigma \models C(\mathrm{a}) \ \Leftrightarrow\ \Sigma \models \{\mathrm{a}\} \sqsubseteq_{\mathcal{S}} C$$

Proof: If Σ is unsatisfiable, then it is trivially true $\Sigma \models C(a)$ and $\Sigma \models \{a\} \sqsubseteq C$. If Σ is satisfiable, then for every Σ-model $\mathcal{I}$ it is evident that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ if and only if $\{a^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$. □

4 The calculus

This section presents a calculus to decide hybrid subsumption. The calculus is based on syntactic objects named *constraints*, and it is a technique that has been used broadly, for instance in [25, 14, 16]; in general, the technique looks for unsatisfiability of concepts and it relies on a complement operator of the concept language. The lack of that operator in our language urge us to modify the technique. Our calculus is a proper adaptation of the one presented in [9], 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 syntactic entities that resemble complex assertions in a world description. For this purpose we introduce two disjoint alphabets of variable symbols: VA (ranged over by x, y) and VC (ranged over by v, w). The VA variables are assigned elements in the abstract domain of an interpretation and VC variables are assigned values in a concrete domain. It is important to remark that the Unique Name Assumption do not apply to variables. We call *abstract objects* (ranged over by s, t) to members of the union of VA and

[5]See its semantics in table 1

the set of names of individuals NA, and *concrete objects* (ranged over by k) to members of the union of VC and the concrete domain $\mathcal{D}$ (we simply say *objects* when the context avoids confusion).

A *constraint* is a syntactic entity of one of the following forms:

$$s : C \qquad sRt \qquad sRk \qquad \mathcal{P}(k_1, \ldots, k_n) \qquad k : \mathcal{D}$$

where C is a $\mathcal{VL}(\mathcal{D})$ concept, R is a $\mathcal{VL}(\mathcal{D})$ role and $\mathcal{P}$ is a n-ary concrete predicate. A finite nonempty set of constraints is a *constraint system.*

We extend the semantics to constraint systems as follows. Given a $\mathcal{VL}(\mathcal{D})$ interpretation $\mathcal{I}$, we define an $\mathcal{I}$-assignment α as a function that maps every variable in VA to an element of $\Delta^{\mathcal{I}}$, every variable in VC to an element of $dom(\mathcal{D})$, every individual in NA to its interpretation (i.e. $\alpha(a) = a^{\mathcal{I}}$) and every element of the concrete domain to itself (i.e. $\alpha(k) = k$ when $k \in dom(\mathcal{D})$).

A pair $(\mathcal{I}, \alpha)$ *satisfies* a constraint $s : C$ if $\alpha(s) \in C^{\mathcal{I}}$, a constraint $sR\delta$ if $(\alpha(s), \alpha(\delta)) \in R^{\mathcal{I}}$, a constraint sRk if $(\alpha(s), \alpha(k)) \in R^{\mathcal{I}}$, and a constraint $\mathcal{P}(k_1, \ldots, k_n)$ if $\mathcal{P}^{\mathcal{I}}(\alpha(k_1), \ldots, \alpha(k_n))$. A constraint system CS is *satisfiable* if there is a pair $(\mathcal{I}, \alpha)$ that satisfies every constraint in CS. If there is a Σ-model $\mathcal{I}$, and assignment α such that $(\mathcal{I}, \alpha)$ satisfies CS we say that CS is Σ-*satisfiable* and $(\mathcal{I}, \alpha)$ is a Σ-*model* of CS.

A world description $\mathcal{W}$ can be translated into a constraint system $\text{CS}_{\mathcal{W}}$ 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 $\text{CS}_{\mathcal{W}}$ are model equivalent in the sense that their models are the same.

A notion of logical entailment is defined on constraint systems.

Definition 4.1 *Let* C_1 *and* C_2 *be two constraint systems and* Σ *a knowledge base. We say that* C_1 Σ-entails C_2 ($\text{C}_1 \models_{\Sigma} \text{C}_2$) *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 4.1 *Let* $\Sigma = (\mathcal{S}, \mathcal{V}, \mathcal{W})$ *be a satisfiable knowledge base,* C *and* D *two* $\mathcal{VL}(\mathcal{D})$ *concepts and* x *a variable from* VA. *Then*

$$\Sigma \models C \sqsubseteq_{\mathcal{S}} D \;\Leftrightarrow\; \text{CS}_{\mathcal{W}} \cup \{x : C\} \models_{\Sigma} \{x : D\}$$

Proof:

$\Rightarrow$ Suppose $\Sigma \models C \sqsubseteq_{\mathcal{S}} D$ and $\text{CS}_{\mathcal{W}} \cup \{x : C\} \not\models_{\Sigma} \{x : D\}$. Then, there exists a Σ-model $(\mathcal{I}, \alpha)$ such that $\alpha(x) \in C^{\mathcal{I}}$ and $\alpha(x) \notin D^{\mathcal{I}}$. This contradicts the hypothesis that for every Σ-model $\mathcal{I}$: $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

$\Leftarrow$ Suppose $\text{CS}_{\mathcal{W}} \cup \{x : C\} \models_{\Sigma} \{x : D\}$ and $\Sigma \not\models C \sqsubseteq_{\mathcal{S}} D$. Then, there exists a Σ-model $\mathcal{I}$ and an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$ and $d \notin D^{\mathcal{I}}$. We can construct the pair $(\mathcal{I}, \alpha)$ with $\alpha(x) = d$ that is obviously a Σ-model of $\text{CS}_{\mathcal{W}} \cup \{x : C\}$, and by the hypothesis $\alpha(x) \in D^{\mathcal{I}}$ which contradicts $d \notin D^{\mathcal{I}}$. $\square$

The Σ-entailment problem $\text{CS}_{\mathcal{W}} \cup \{x : C\} \models_{\Sigma} \{x : D\}$ is decided with the calculus. The rules of the calculus work on ordered pairs $F.G$ of constraint systems. We call 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 $\text{CS}_{\mathcal{W}} \cup \{x : C\}.\{x : D\}$, 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. A general

DO1:	$F.G$	$\rightarrow$	$\{s:\top\} \cup F.G$
		if	$s:C$ is in F
DO2:	$F.G$	$\rightarrow$	$\{s:\top\} \cup F.G$
		if	sRt is in F
DO3:	$F.G$	$\rightarrow$	$\{k:\mathcal{D}\} \cup F.G$
		if	$sP_{\mathcal{D}}k$ is in F
DO4:	$F.G$	$\rightarrow$	$\{k_1:\mathcal{D},\ldots,k_n:\mathcal{D}\} \cup F.G$
		if	$\mathcal{P}(k_1,\ldots,k_n)$ is in F
DO5:	$F.G$	$\rightarrow$	$F.G \cup \{s:\top\}$
		if	$s:C$ is in G
DO6:	$F.G$	$\rightarrow$	$F.G \cup \{s:\top\}$
		if	sRt is in G
DO7:	$F.G$	$\rightarrow$	$F.G \cup \{k:\mathcal{D}\}$
		if	$sP_{\mathcal{D}}k$ is in G
DO8:	$F.G$	$\rightarrow$	$F.G \cup \{k_1:\mathcal{D},\ldots,k_n:\mathcal{D}\}$
		if	$\mathcal{P}(k_1,\ldots,k_n)$ is in G

Table 2: The domain rules

condition for the application of any rule is that the resulting pair after rule application must be different from the departure pair. This simplifies the description of the rules and is needed to guarantee termination of the derivation procedure. 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 and are presented in tables 2 to 6. Any concept of the form $\exists p = q$ can be translated to an equivalent one $\exists p' = \varepsilon$, just composing inverses of primitive roles when fillers of p and q are abstract objects. In the case that fillers of p and q are concrete objects, $\exists p = q$ is the same as $=_{\mathcal{D}} (p,q)$ if the equality predicate $=_{\mathcal{D}}$ is in $\mathcal{D}$. To simplify the calculus, the rules only consider the $\exists p' = \varepsilon$ variant. The substitution of every occurrence of the variable y with s in the pair $F.G$ is denoted by $(F.G)[y/s]$. An informal review of each group of propagation rules follows.

The domain rules (see table 2) add constraints that bind objects to the abstract or to the concrete domain.

The decomposition rules (see table 3) work on facts. They add constraints derived from the compositional structure of concept and role descriptions. Notice that the inverse operator is admitted only with primitive roles, thus the symbol Q^- in rule D2 is only a name for the primitive role P when Q is the complex role P^{-1} and a name for P^{-1} when Q is the primitive role P. New variables are introduced to name objects whose existence is a must.

The schema rules (see table 4) augment facts. They add constraints that pursue the satisfaction of every axiom in the schema $\mathcal{S}$. However, it is important to notice the condition on rules S5 and S6. That condition prevents an excessive growth of variable population in facts; a new variable is introduced only if by demand of a goal constraint.

The goal rules (see table 5) add constraints to goals that become targets for the composition rules.

The composition rules (see table 6) resemble a bottom-up evaluation of targeted concepts within a constraint system.

To limit the appearance of new objects in the set of facts is a key to maintain controlled the complexity of the calculus. Decomposition rules D4, D6, D9 and schema rules S5, S6 may increase the number of variables. Since variables introduced by decomposition rules carry more specific information than variables created by schema rules a control strategy is adopted: a schema rule can be applied only if no decomposition rule is applicable.

Let us mention some technical details and appoint some notation for the rest of the section. We always refer to a satisfiable knowledge base $\Sigma = (\mathcal{S}, \mathcal{V}, \mathcal{W})$ and concepts C and D. An

D1:	$F.G$	$\rightarrow$	$\{s : C,\ s : D\} \cup F.G$
		if	$s : C \sqcap D$ is in F
D2:	$F.G$	$\rightarrow$	$\{sQt\} \cup F.G$
		if	$tQ^{-}s$ is in F
D3:	$F.G$	$\rightarrow$	$(F.G)[y/a]$
		if	$y : \{a\}$ is in F
D4:	$F.G$	$\rightarrow$	$\{spy\} \cup F.G$
		if	$s : \exists p$ is in F
			and there is no t with spt in F and y is a fresh variable
D5:	$F.G$	$\rightarrow$	$\{sps\} \cup F.G$
		if	$s : \exists p = \varepsilon$ is in F
D6:	$F.G$	$\rightarrow$	$\{sQy, y : C, ypt\} \cup F.G$
		if	$s(Q : C)pt$ is in F,
			and there is no t' such that sQt', $t' : C$, $t'pt$ are all in F,
			and $p \neq \varepsilon$, and y is a fresh variable
D7:	$F.G$	$\rightarrow$	$\{sQt, t : C\} \cup F.G$
		if	$s(Q : C)t$ is in F
D8:	$F.G$	$\rightarrow$	$(F.G)[k/P_{\mathcal{D}}^{\mathcal{I}}(s)]$
		if	$sP_{\mathcal{D}}k$ is in F and k is a variable
D9:	$F.G$	$\rightarrow$	$\{sp_1k_1, \ldots, sp_nk_n, \mathcal{P}(k_1, \ldots, k_n)\} \cup F.G$
		if	$s : \mathcal{P}(p_1, \ldots, p_n)$ is in F
			and there are no k_i' such that sp_ik_i' and $\mathcal{P}(k_1', \ldots, k_n')$ are in F.
			The k_i are fresh variables for concrete objects

Table 3: The decomposition rules

S1:	$F.G$	$\rightarrow$	$\{s : A_2\} \cup F.G$
		if	$s : A_1$ is in F and $A_1 \sqsubseteq A_2$ is in Σ
S2:	$F.G$	$\rightarrow$	$\{t : A_2\} \cup F.G$
		if	$s : A_1$ and sPt are in F,
			and $A_1 \sqsubseteq \forall P.A_2$ is in Σ
S3:	$F.G$	$\rightarrow$	$\{s : A_1,\ t : A_2\} \cup F.G$
		if	sPt is in F, and $P \sqsubseteq A_1 \times A_2$ is in Σ
S4:	$F.G$	$\rightarrow$	$(F.G)[y/t]$
		if	$s : A,\ sPy,\ sPt$ are in F,
			and $A \sqsubseteq (\leq 1 P)$ is in Σ
S5:	$F.G$	$\rightarrow$	$\{sPy\} \cup F.G$
		if	$s : \exists(P : C)p$ is in G or $s : \exists(P : C)p = \varepsilon$ is in G,
			or else $s : \mathcal{P}(p_1, \ldots, p_n)$ is in G with $p_i = (P : C)q$,
			and there is no t with sPt in F,
			and there is an A with $s : A$ in F and $A \sqsubseteq \exists P$ in Σ,
			and y is a fresh variable
S6:	$F.G$	$\rightarrow$	$\{sP_{\mathcal{D}}k\} \cup F.G$
		if	$s : \exists P_{\mathcal{D}}$ is in G or $s : \mathcal{P}(\ldots, P_{\mathcal{D}}, \ldots)$ is in G,
			and there is no k with $sP_{\mathcal{D}}k$ in F,
			and there is an A with $s : A$ in F and $A \sqsubseteq \exists P_{\mathcal{D}}$ in Σ,
			and k is a fresh variable

Table 4: The schema rules

G1:	$F.G$	$\rightarrow$	$F.G \cup \{s : C,\ s : D\}$
		if	$s : C \sqcap D$ is in G
G2:	$F.G$	$\rightarrow$	$F.G \cup \{t : C\}$
		if	$s : \exists(Q : C)$ is in G or $s : \exists(Q : C) = \varepsilon$ is in G,
			and sQt is in F
G3:	$F.G$	$\rightarrow$	$F.G \cup \{t : C,\ t : \exists p\}$
		if	$s : \exists(Q : C)p$ is in G or $s : \exists(Q : C)p = \varepsilon$ is in G
			or $s : \mathcal{P}(p_1, \ldots, p_n)$ is in G with $p_i = (Q : C)q$
			and sQt is in F

Table 5: The goal rules

C1:	$F.G$	$\rightarrow$	$\{s : C \sqcap D\} \cup F.G$
		if	$s : C$ and $s : D$ are in F, and $s : C \sqcap D$ is in G
C2:	$F.G$	$\rightarrow$	$\{s : \top\} \cup F.G$
		if	$s : \top$ is in G
C3:	$F.G$	$\rightarrow$	$\{s : \exists p\} \cup F.G$
		if	$p = \varepsilon$ or there exist t with spt in F
			and $s : \exists p$ is in G
C4:	$F.G$	$\rightarrow$	$\{s : \exists p = \varepsilon\} \cup F.G$
		if	$p = \varepsilon$ or sps is in F, and $s : \exists p = \varepsilon$ is in G
C5:	$F.G$	$\rightarrow$	$\{s(Q : C)pt\} \cup F.G$
		if	there is a t' with sQt', $t' : C$ and $t'pt$ in F, such that
			$s : \exists(Q : C)p$ or $s : \exists(Q : C)p = \varepsilon$ or $s : \mathcal{P}(\ldots,(Q : C)p,\ldots)$ is in G
C6:	$F.G$	$\rightarrow$	$\{s(Q : C)t\} \cup F.G$
		if	sQt, $t : C$ are in F, and
			$s : \exists(Q : C)$ or $s : \exists(Q : C) = \varepsilon$ is in G
C7:	$F.G$	$\rightarrow$	$\{s : \mathcal{P}(p_1,\ldots,p_n)\} \cup F.G$
		if	there exist k_i $i = 1 \ldots n$ $(n \geq 1)$ with sp_ik_i in F
			and $s : \mathcal{P}(p_1,\ldots,p_n)$ is in G
			and $\bigwedge_{j \in J} \mathcal{Q}_j(\ldots) \models \mathcal{P}(k_1,\ldots,k_n)$ with $\mathcal{Q}_j(\ldots)(j \in J)$ being the predicate assertions in F

Table 6: The composition rules

examination of the goal rules should convince the reader that, for every pair derivable from $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\}.\{x : D\}$ the set of goals has exactly one constraint that allocate an object in concept D, possibly the name of the object is different from x due to renaming rules D3 or S4 (i.e. $\{s : D\}$). Furthermore, since D3 and S4 apply simultaneously to facts and goals, if the constraint $\{s : D\}$ is in goals then the same object name s is allocated in C inside facts (i.e. $\{s : C\}$ is in facts). 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. $o : D \in G_D$).

In what follows we show, first the soundness and next the completeness of the calculus. Then, its computational complexity is pointed out.

Lemma 4.2 *Let $F.G$ be a pair derived from $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\}.\{x : D\}$ and $F'.G'$ a pair obtained by the application to $F.G$ of a propagation rule. Then*

(a) For every Σ-model $(\mathcal{I}, \alpha)$ of F, we can construct a Σ-model $(\mathcal{I}, \alpha')$ of F', perhaps modifying the assignment of fresh variables.

(b) Moreover, if $s : D \in G$ and $s' : D \in G'$ then α' can be defined with $\alpha'(s') = \alpha(s)$.

Proof:

(a) It can be shown by individual analysis of each rule. For instance, with rule D4, if $(\mathcal{I}, \alpha)$ satisfies $s : \exists p$ then let $d \in \Delta^{\mathcal{I}}$ be one of the elements such that $(\alpha(s), d) \in p^{\mathcal{I}}$. Then $\alpha' = \alpha$ except $\alpha'(y) = d$ is sufficient. Analogously with the rest of the rules.

(b) It is enough to inspect rules D3 and S4. Notice that $\alpha' = \alpha$ and $(\mathcal{I}, \alpha)$ is Σ-model of F in both cases. Concerning D3, $y : \{a\} \in F$ thus $\alpha'(a) = \alpha(a) = \alpha(y)$. With regards to S4, it is sufficient to define $\alpha'(t) = \alpha(y)$ and $\alpha' = \alpha$ for any other object. □

Corollary 4.1 *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 $\alpha'(o) = \alpha(x)$.*

Proof: By induction on the length of the sequence of rules used to reach $F_C^{\mathcal{W}}$ and applying lemma 4.2. □

Corollary 4.2 $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\} \models_{\Sigma} \{x : D\} \;\Leftrightarrow\; F_C^{\mathcal{W}} \models_{\Sigma} \{o : D\}$.

Proof:

$\Rightarrow$ Suppose $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\} \models_\Sigma \{x : D\}$ and $F_C^{\mathcal{W}} \not\models_\Sigma \{o : D\}$. So, there exists a Σ-model $(\mathcal{I}, \alpha)$ of $F_C^{\mathcal{W}}$ such that $\alpha(o) \notin D^{\mathcal{I}}$. Since $o : D \in G_{\mathcal{D}}$ we have $o : C \in F_C^{\mathcal{W}}$ and $\alpha(o) \in C^{\mathcal{I}}$. Moreover, $(\mathcal{I}, \alpha')$ is Σ-model of $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\}$ with $\alpha'(x) = \alpha(o)$ and $\alpha' = \alpha$ for any other object. Then $\alpha'(x) \notin D^{\mathcal{I}}$ contradicts the hypothesis.

$\Leftarrow$ Suppose $F_C^{\mathcal{W}} \models_\Sigma \{o : D\}$ and $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\} \not\models_\Sigma \{x : D\}$. Hence, there exists a Σ-model $(\mathcal{I}, \alpha)$ of $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\}$ such that $\alpha(x) \notin D^{\mathcal{I}}$. By corollary 4.1, we have a Σ-model $(\mathcal{I}, \alpha')$ of $F_C^{\mathcal{W}}$ such that $\alpha'(o) = \alpha(x)$. Since $o : D \in F_C^{\mathcal{W}}$, $\alpha(x) \in D^{\mathcal{I}}$ yields a contradiction. $\Box$

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 we can build an interpretation, that we call *canonical.*

Let CS be a clash-free constraint system. Let u be an special abstract object not appearing in CS and $u_{\mathcal{D}}$ a singular concrete object added to $dom(\mathcal{D})$. We recall that a clever control was devised to prevent schema rules S5 and S6 from introducing a new object every time. The purpose of u and $u_{\mathcal{D}}$ is to satisfy the schema axioms in $\mathcal{S}$ of the form $A \sqsubseteq \exists P$ for those objects such that $s : A \in$CS and there is no P-filler for s in CS. Then we define the *canonical interpretation pair* $(\mathcal{I}_{\mathrm{CS}}, \alpha)$ as follows:

$$\begin{array}{ll}
\Delta^{\mathcal{I}_{\mathrm{CS}}} & = \{s \mid s \text{ is an abstract object in CS }\} \cup \{u\} \\
s^{\mathcal{I}_{\mathrm{CS}}} & = s, \text{ for every abstract object in CS.} \\
A^{\mathcal{I}_{\mathrm{CS}}} & = \{s \mid s : A \in \mathrm{CS}\} \cup \{u\} \\
P^{\mathcal{I}_{\mathrm{CS}}} & = \{(s,t) \mid sPt \in \mathrm{CS}\} \cup \{(u,u)\} \cup \\
& \quad \{(s,u) \mid \text{there is no } sPt \text{ in CS,} \\
& \quad \text{but for some } A,\ s : A \in \mathrm{CS} \text{ and } A \sqsubseteq \exists P \in \mathcal{S}\} \\
& \quad \text{for every } P \sqsubseteq A \times B \in \mathcal{S} \\
P_{\mathcal{D}}^{\mathcal{I}_{\mathrm{CS}}} & = \{(s,k) \mid sP_{\mathcal{D}}k \in \mathrm{CS}\} \cup \{(u,u_{\mathcal{D}})\} \cup \\
& \quad \{(s,u_{\mathcal{D}}) \mid \text{there is no } sP_{\mathcal{D}}k \text{ in CS,} \\
& \quad \text{but for some } A,\ s : A \in \mathrm{CS} \text{ and } A \sqsubseteq \exists P_{\mathcal{D}} \in \mathcal{S}\} \\
& \quad \text{for every } P_{\mathcal{D}} \sqsubseteq A \times \mathcal{D} \in \mathcal{S}
\end{array}$$

The assignment α is any assignment such that satisfies the finite conjunction of all the $\mathcal{P}^{\mathcal{I}_{\mathrm{CS}}}(\alpha(k_1), \ldots, \alpha(k_n))$ for which $\mathcal{P}^{\mathcal{I}_{\mathrm{CS}}}(k_1, \ldots, k_n) \in CS$.
There exists such an assignment, since CS is clash-free.

The following two lemmas establish the completeness of the calculus. The canonical interpretation built on the clash-free facts $F_C^{\mathcal{W}}$ of a complete pair of constraint systems provide a prototypical Σ-model. For the conclusion of the decision procedure it just remains checking membership of a singular object in the concept D within this prototypical Σ-model.

Lemma 4.3 *Let* $F_C^{\mathcal{W}}.G_D$ *be a complete pair derived by the calculus from* $\mathrm{CS}_{\mathcal{W}} \cup \{x : C\}.\{x : D\}$. *If the facts* $F_C^{\mathcal{W}}$ *is a clash-free constraint system then the canonical interpretation* $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ *is a* Σ*-model of* $F_C^{\mathcal{W}}$.

Proof: By definition, $\mathcal{I}_{F_C^{\mathcal{W}}}$ satisfies the Unique Name Assumption because every object is interpreted by itself.

We must show that $\mathcal{I}_{F_C^{\mathcal{W}}}$ satisfies every axiom in the schema $\mathcal{S}$ and every assertion in the world description $\mathcal{W}$ of the knowledge base Σ. The proof proceeds by case analysis of the axioms in $\mathcal{S}$. For instance, let $A \sqsubseteq \forall P.B$ be an axiom in $\mathcal{S}$ and $s \in A^{\mathcal{I}_{F_C^{\mathcal{W}}}}$. If s is u then $u \in (\forall P.B)^{\mathcal{I}_{F_C^{\mathcal{W}}}}$ since $(u,u) \in P^{\mathcal{I}_{F_C^{\mathcal{W}}}}$ and u is the only P-filler of u and $u \in B^{\mathcal{I}_{F_C^{\mathcal{W}}}}$. Otherwise s is not u; if there is any t such that $sPt \in F_C^{\mathcal{W}}$ then $t \in B^{\mathcal{I}_{F_C^{\mathcal{W}}}}$ due to the application of rule S2 and we are done. The study of the other kinds of axioms proceeds analogously. With respect to the world description $\mathcal{W}$, notice that it is translated into the constraint system $\mathrm{CS}_{\mathcal{W}}$. Therefore, it is sufficient to show that $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ satisfies every constraint in $F_C^{\mathcal{W}}$.

Every constraint in $F_C^{\mathcal{W}}$ of the form $s{:}A$, sRt, sRk or $\mathcal{P}(k_1,\ldots,k_n)$ is satisfied by definition of $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$. If a constraint of the form $s : \{a\}$ is in $F_C^{\mathcal{W}}$, since $F_C^{\mathcal{W}}$ is clash-free and complete (i.e. D3 has been applied) we have $s = a$ and the constraint is trivially satisfied. If $sQ^{-}t$ is in $F_C^{\mathcal{W}}$ then, by rule D2, tQs is in $F_C^{\mathcal{W}}$ also and we are in the first case. Straightforward induction on the length of the concepts involved in each constraint proves that $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ satisfies every constraint in $F_C^{\mathcal{W}}$. $\square$

Lemma 4.4 *Let $F_C^{\mathcal{W}}.G_D$ be a complete pair derived by the calculus from $\mathrm{CS}_{\mathcal{W}} \cup \{x{:}C\}.\{x{:}D\}$. Let $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ be a canonical interpretation of $F_C^{\mathcal{W}}$ and $s{:}E$ a constraint in G_D. If $F_C^{\mathcal{W}}$ is clash-free then*

$$(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha) \text{ satisfies } s{:}E \;\Rightarrow\; s{:}E \in F_C^{\mathcal{W}}.$$

Proof: By induction over the structure of the concept E. The details are based on the hypothesis and the consideration of the composition rules. $\square$

Theorem 4.1 *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 $o : D \in G_D$:*

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

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 lemma 4.4, $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ does not satisfy $o : D$ (i.e. $\alpha(o) \notin D^{\mathcal{I}_{F_C^{\mathcal{W}}}}$). By lemma 4.3, $(\mathcal{I}_{F_C^{\mathcal{W}}}, \alpha)$ is a Σ-model of $F_C^{\mathcal{W}}$ and constructing and assignment 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, by corollary 4.2, we are done. 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 corollary 4.1, 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 [9], 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 [22].

5 Conclusions

In this paper we have presented a description logic to be used in a meta level of an interoperable system that permits: 1. exploiting the existing mapping information and 2. enhancing the query processing task. The logic includes two languages, one to define the structure of the elements of the class level (Schema language), and another to define queries over the class level (View language).

Concepts in the Schema language are built out of primitive concepts and three kinds of role constraints: range restriction to primitive concepts, existential quantification and single valuation. The *structure axioms* specify necessary properties of members of primitive concepts and roles. A finite set of structure axioms forms a Schema at the metaclass level.

The View language consists of primitive concepts, the universal concept, singleton concepts, conjunction of concepts, existential quantification of paths, and existential constraints over paths. The constraints are the equality predicate plus any predicate from an admissible *concrete domain*. The latter construct provides the way to incorporate to the View language some descriptions concerning semantic properties which are interesting in the context we are considering, for instance subsumption of terms or functional dependency of attributes.

Expressions from the View language may be used to browse the class level looking for knowledge such as relationships stated among elements from the different categories defined at the Schema or structural properties of the elements themselves. Moreover, the View language enables the description of *patterns* of the queries formulated by the users. The reasoning capabilities provided by the logic allows first, the arrangement of a taxonomy of patterns partially ordered by the subsumption relation and second, the recognition of the patterns which a user query match. The solution to both issues is reduced to the hybrid subsumption problem of concepts from the View language, and it has been proved that it is co-NP-hard relative to the consequence problem of the concrete domain. Moreover, we associate generic plans to patterns and the taxonomy of patterns serves as an index to locate a plan to answer properly a user query: just an instantiation of some parameters is needed.

Finally, we can mention two relevant open problems. First, the design of an optimized algorithm for the implementation of the presented calculus. Second, the study of the eventual necessity of higher expressiveness in the metaclass level. Surely, adding description constructs will increase algorithmic complexity of reasoning if completeness is desired. Worst case algorithmic intractability face us with the problem of empirical evaluation of implementations hoping that pathological inputs which produce worst cases are unlikely to arise in practice. Nevertheless, the three level architecture and its usage proposed in this paper can perfectly remain.

References

[1] Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[2] F. Baader, A. Borgida, and D. L. McGuiness. Matching in description logics: Preliminary results. In *Proc. of the 1998 International Workshop on Description Logics (DL'98). Povo-Trento. Italy*, pages 21–29, 1998.

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

[4] 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.

[5] A. Borgida and D. McGuiness. Asking Queries about Frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning. (KR-96). Cambridge. Mass.*, pages 340–349, 1996.

[6] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[7] 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.

[8] R.J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *AAAI-84, Austin*, pages 34–37, 1984.

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

[10] D. Calvanese, G. De Giacomo, and M. Lenzerini. Semi-structured data with constraints and incomplete information. In *Proc. of the 1998 International Workshop on Description Logics (DL'98). Povo-Trento. Italy*, pages 11–20, 1998.

[11] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logic Framework for Information Integration. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning. (KR-98). Trento. Italy.*, pages 2–13, 1998.

[12] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th IPSJ, Tokyo, Japan*, 1994.

[13] C. Collet, M. Huhns, and W. Shen. Resource Integration using a Large Knowledge Base in Carnot. *IEEE Computer*, (24):55–62, December 1991.

[14] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. M. Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

[15] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation, Studies in Logic, Language and Information*, pages 193–238. CSLI Publications, 1996.

[16] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning. (KR-91)*, pages 151–162, 1991.

[17] R. Mac Gregor. The evolving technology of classification-based knowledge representation systems. In J.F.Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 385–400. Morgan Kaufmann, San Mateo, CA, 1991.

[18] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. Technical report, DFKI Research Report RR-90-04, 1990.

[19] P. Lambrix and P. Larocchia. Learning composite objects. In *Proc. of the 1998 International Workshop on Description Logics (DL'98). Povo-Trento. Italy*, pages 147–152, 1998.

[20] A.Y. Levy, A. Rajamaran, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the VLDB 96.*, 1996.

[21] 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.

[22] 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.

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

[24] M. Schmidt-Schauss. Subsumption in KL-ONE is Undecidable. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning. (KR-89). Toronto, Ont.*, pages 421–431, 1989.

[25] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[26] W.A. Woods and J.G. Schmolze. The KL-ONE family. *Computers Math. Applic.*, 23(2):133–177, 1992.