A Terminological Approach to Resource Discovery Mechanisms

David E. Losada[1], Raúl Ramos[1,2], and Alvaro Barreiro[1]

[1] Dept. Computación. Fac. Informática
Universidade da Coruña.
Campus de Elviña 15071 A Coruña, Spain
{losada,barreiro}@dc.fi.udc.es
[2] CERN. European Laboratory for Particle Physics
Information Technology Division
1211 Genève, Switzerland
Raul.Ramos@cern.ch

Keywords: Knowledge Representation, Intelligent Data Retrieval, Terminological Systems, Description Logics.

Abstract. Resource Discovery Mechanisms are commonly used to publish and retrieve information about network-available resources. Description Logics are basically a subset of Predicate Logic where expressiveness and computational complexity have been deeply studied. In this paper we present a formal approach to these mechanisms using Description Logics. This is based on a terminological model of the RDM (Resource Description Messages) mechanism. We provide a translation of the resource description schemas and basic query languages of RDM. This translation allows us to benefit from the inference capabilities of terminological systems without losing the information contained in RDM messages.

1 Introduction

Description Logics (DLs) are formalisms for expressing, organising and manipulating knowledge. In the framework of Artificial Intelligence (AI) and Knowledge Representation (KR), they provide methods and procedures to represent, retrieve and reuse relevant knowledge in a certain domain. DLs usually provide facilities for maintaining and exploiting taxonomies of concepts and roles called terminologies. DLs can be considered as formal successors of semantic networks and more specifically of KL-ONE [7].

After the pioneering work of Levesque and Brachman [12], and during the last 15 years the expressiveness and complexity of DLs have been studied [1, 8, 9, 17, 18, 5, 19]. DLs are implemented by concept languages, the most known ones being CLASSIC [6], Kris [2] and Loom [14]. Practical applications of these languages cover a great variety of domains. In particular, there are applications for database schema modelling and more general knowledge-based management systems [4, 3], conceptual modelling of information sources in global information systems [13] and proposals of terminological models for Information Retrieval [15, 20, 16].

Resource Discovery Mechanisms are used across the Internet to easily access network-available collections of data. They embody languages to describe and query about the resources available at a particular site. With these mechanisms users or applications can retrieve information about the type of available resources before accessing them with other protocols.

In this paper we present a terminological conceptualisation of a typical resource discovery model. The generality of this approach proofs the practical possibility to integrate resource discovery mechanisms in hypothetical DL environments, overcoming the need to rewrite existing resource descriptions in AI formalisms such as DLs. Section 2 is a formal introduction to DLs. Section 3 describes the issues of resource discovery mechanisms in which we are interested in and section 4 presents the target terminological model. The paper ends with our conclusions and the future work opened by our model.

2 Terminological Knowledge Representation Systems

Terminological Knowledge Representation Systems (TKRSs) organise and represent knowledge by means of taxonomies. These systems have also specific inference mechanisms to explore these structures. In TKRSs, the intensional knowledge denotes the global knowledge about a specific domain and is called **TBox** (*Terminological Box*) or Definitional Module. The **ABox** (*Assertional Box*) or Assertional Module contains particular knowledge about a specifc situation in a domain. The TBox is a set of terminological axioms that use expressions of a Concept Language (CL) to construct a taxonomy. In the ABox, the concepts and roles previously defined in the TBox are used in a set of assertional axioms. Objects, here called individuals, introduced with these assertional axioms are classified in the taxonomy. Typical inferences are subsumption, satisfiability, equivalence and disjointness. In fact, subsumption is the basic reasoning task and all other inferences can be reduced to subsumption problems

([9], [19]). The CL is the core of a TKRSs and its expressiveness and computational properties determine the efficiency and practical utility of the representation system.

In a TKRS the world is a set of *individuals*. A *concept* is a subset of individuals and a *role* is a subset of pairs of individuals (binary relation). The TBox is a taxonomy composed of the definition of concept and roles and their subsumption relations. The ABox contains the set of individuals and relations about a specific perception of the world.

2.1 Syntax

Let **A** be a set of atomic concepts and **P** a set of atomic roles. Concepts (C, D) and roles (Q, R) are inductively built from atomic concepts and roles, the universal concept and the empty concept.
(1) An element of **A**, A, is a concept and an element of **P**, P, is a role (atomic concepts and roles)
(2) $\top$ (universal concept)
(3) $\bot$ (empty concept)
(4) $\neg A$ (negation of an atomic concept)
(5) $C \sqcap D$ (concept intersection)
(6) $\forall P.C$ (concept universal role quantification)
(7) $\exists P.\top$ (unqualified existential quantification)
(8) $C \sqcup D$ (Extension $\mathcal{U}$, union of concepts)
(9) $\exists R.C$ (Extension $\mathcal{E}$, qualified existential quantification)
(10) $\neg C$ (Extension $\mathcal{C}$, complement of non-atomic concepts)
(11) $\geq nR$, $\leq nR$ (Extension $\mathcal{N}$, number restrictions)
(12) $Q \sqcap R$ (Extension $\mathcal{R}$, role intersection of roles)
(13) $C \times D$ (role product)
(14) $\leq nR.C, \geq nR.C$ (qualified number restrictions)
(15) $fills.i.R$ (*fills* existential quantification)
(16) nR (exact number restriction)
(17) $nR.C$ (qualified exact number restriction)

2.2 Semantics

The formal meaning of the language is given by a model-theoretic interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The interpretation consists of an arbitrary set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept A in a subset of $\Delta^{\mathcal{I}}$ ($\mathrm{A}^{\mathcal{I}}$) and every role P in a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ ($\mathrm{P}^{\mathcal{I}}$). The predefined concepts $\top$ y $\bot$ have a fixed interpretation, $\Delta^{\mathcal{I}}$ y $\emptyset$ respectively. The meaning of the concept and role expressions described above is as follows:

$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \longrightarrow b \in C^{\mathcal{I}}\}$
$(\exists R.\top)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}$
$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
$(nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid card\{b \mid (a,b) \in R^{\mathcal{I}}\} = n\}$
$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid card\{b \mid (a,b) \in R^{\mathcal{I}}\} \geq n\}$
$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid card\{b \mid (a,b) \in R^{\mathcal{I}}\} \leq n\}$
$(Q \sqcap R)^{\mathcal{I}} = Q^{\mathcal{I}} \cap R^{\mathcal{I}}$
$(C \times D)^{\mathcal{I}} = \{(a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid a \in C^{\mathcal{I}} \wedge b \in D^{\mathcal{I}}\}$
$(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid card\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \leq n\}$
$(\geq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid card\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \geq n\}$
$(nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid card\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} = n\}$
$(fills.i.R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid i \in \Delta^{\mathcal{I}} \wedge (a,i) \in R^{\mathcal{I}}\}$

The semantics guarantees the equivalences $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$, $\exists R.C \equiv \neg \forall R.\neg C$, $(nR) \equiv (\geq nR) \sqcap (\leq nR)$ and $(nR.C) \equiv (\geq nR.C) \sqcap (\leq nR.C)$. The most readable expressions will be used in this paper. Terms (1) to (7) come from the family of $\mathcal{AL}$ languages, and the next terms are extensions of the basic $\mathcal{AL}$ language. The first concept languages, $\mathcal{FL}$ and $\mathcal{FL}^-$ [12], are sublanguages of $\mathcal{ALC}$ and $\mathcal{AL}$ respectively ([19]).

2.3 The TBox

Let D be a concept expression and S a role expression. A TBox or *Terminology* is a finite set of *terminological axioms* that define the concepts A, B and role R:

- Terminological axioms of defined concepts and roles (also called complete definitions): $A = D, R = S$.
- Terminological axioms of primitive concepts and roles (also called incomplete definitions): $A \sqsubseteq D, R \sqsubseteq S$.
- Terminological disjointness axioms: $dis(A, B)$.

and with two restrictions:

- A concept or role cannot appear more than once in the left hand side of a terminological axiom.
- The disjointness axiom must not contain defined concepts.

Note that if the language has the $\top$ concept, the distinction between non-defined concepts (without restrictions in the interpretation) and partially defined concepts (with necessary conditions in the interpretation) is not necessary. A non-defined concept can be seen as partially defined ($A \sqsubseteq \top$).

Let A be a concept, R a role, D a concept expression and S a role expression. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies a terminological axiom, iff: $A^{\mathcal{I}} = D^{\mathcal{I}}(R^{\mathcal{I}} = S^{\mathcal{I}})$ for the terminological axiom $A = D(R = S)$, $A^{\mathcal{I}} \subseteq D^{\mathcal{I}}(R^{\mathcal{I}} \subseteq S^{\mathcal{I}})$ for the terminological axiom $A \sqsubseteq D(R \sqsubseteq S)$ and $A^{\mathcal{I}} \cap B^{\mathcal{I}} = \emptyset$ for the terminological axiom $dis(A, B)$.

Now we can define a *model*. An interpretation $\mathcal{I}$ is a model of a TBox if it satisfies all the terminological axioms.

2.4 The ABox

An ABox contains the individuals and relations that are part of the world defined in the TBox. If C is a name of concept, R is a name of role, and a and b are names of individuals then the sentences $(a.C)$ and $(a.b.R)$ are assertional axioms.

The interpretation function $\cdot^{\mathcal{I}}$ is extended to map the names of the individuals and roles over $\Delta^{\mathcal{I}}$ and $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ respectively. An interpretation $\mathcal{I}$ satisfies an assertional axiom, iff: $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for the assertional axiom $(a.C)$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for the assertional axiom $(a.b.R)$.

An interpretation $\mathcal{I}$ is a model of an ABox if it satisfies all the assertional axioms. Terminological systems usually include the *unique names assumption* but do not include the *closed world assumption*.

3 Resource Description Mechanisms

The growth and popularity of Internet based services makes possible for any user to access lots of data and information. This work is about the **data retrieval** problem which is different from the **information retrieval** problem, although some protocols and tools are common to both. The information retrieval problem is more general and deals with unstructured documents. The data retrieval problem deals with collections of structured data and it is important in places where there are collaborations between several users and groups of people. These collections are provided by services offering different kinds of data, ranging from people directories (white pages) to book stores and catalogued information. In general, users must know where and how to search. Some services provide facilities to help users in their search for data. In this context the data retrieval problem relates to the overall organisation of the data and the uniformity with which the same kind of information should be offered by different providers.

A solution for the data retrieval problem requires protocols that allow the interchange of the offered data and the information associated with it. However, in this paper we only want to present a model for the data and we will not study in depth the required architecture. It is enough to bear in mind that (see Fig. 3):

- There is a network made of two kinds of nodes: **Directory Service Providers** (DSPs) which provide data collections; and **Clients** who access DSPs using some shared access method.
- Every DSP offers a collection of elements and a general description of it. Every element within a collection is described by a sequence of pairs `attribute:value`. Since every DSP contains information related to the collections and elements that it offers, the clients or other agents (eg. WWW search robots) can access general descriptions of its contents (schemas).

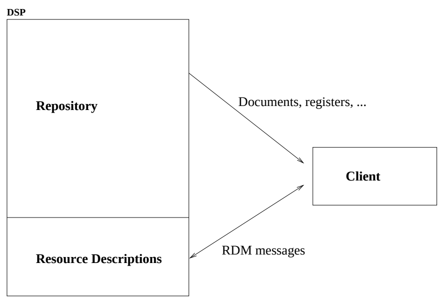

Fig. 1. General schema of RDM

We take the specification of RDM (*Resource Description Messages*) proposed in [10] as a practical implementation of this last idea. RDM is a mechanism to discover and retrieve descriptions of network-accessible resources. A DSP produces resource descriptions (RDs) about its contents and makes them available to clients through RDM. Clients may find out about the contents of any DSP by querying it using RDM without having to access the actual data. This can be done in several steps as shown in the example of Figure 3. Clients may then decide the actual resources in which they are interested and get them using some shared method, avoiding unnecessary traffic through the network

RDM messages are encoded using Harvest's SOIF (*Summary Object Interchange Format*) [11]. Schemas and RDs are encoded in SOIF. RDM also provides predefined schemas to support certain types of interaction with DSPs. For example, RDM defines a schema to ask about RDs (*What RDs have changed since last week?*) or to describe taxonomies of the offered collections.

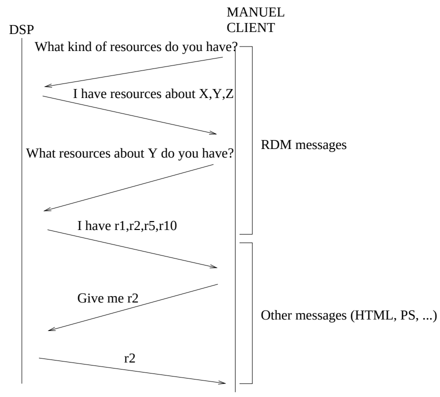

Fig. 2. A possible scenario of use of RDM

4 A Terminological Model of Resource Description Messages

Here, we consider the problem of integrating DSPs into a hypothetical DL environment. RDM is an implementation of a very simple query language and data structure, while DLs provide a generic model for the conceptualisation of domains. Since neither of them is widely spread out nor dominant, in terms of the general public, one may foresee situations where their coexistence may overlap or be optimised. We may consider two situations:

Figure 4 represents a **DL based world** where most of the nodes communicate via DL messages using a hypothetical DL protocol. In order to integrate a DSP whose repository is described with RDM, we need a layer around such DSP to translate incoming DL messages to RDM and outgoing RDM messages to DL. Genuine RDM clients may communicate directly with the DSP avoiding the translation step.

In an **RDM world** a DSP is only accessible via RDM. DL clients are fewer and must use the RDM/DL translation layer to access DSPs as shown in Fig. 4. Obviously DL clients may access other DL nodes with DL messages and RDM clients can communicate directly with DSPs.

So one may pose the task of building these translation to provide mechanisms to overcome potential problems or reuse existing technologies. Observe that the translation is not symmetrical since RDM is simpler and less powerful from the expressiveness point of view. This way an RDM to DL translation will be based on a conceptualisation of the RDM model in DL; while in an RDM world, DL nodes will probably just

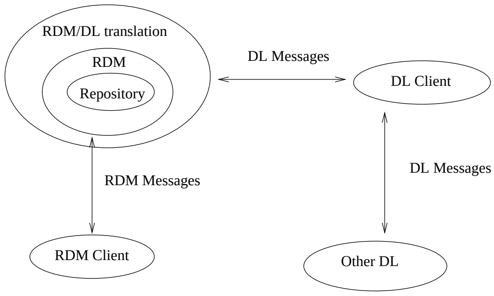

Fig. 3. Integration of a DSP in a DL environment

use RDM to encapsulate DL entities. In the following, we deal with the first problem and leave the second one as an open research line.

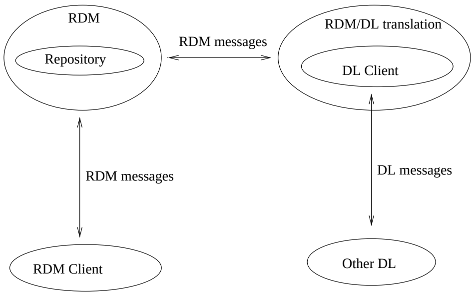

Fig. 4. Integration of a DL client in a RDM world

A basic terminological model of RDM is a previous step to a full RDM/DL translation. The basic model consists of terminological axioms for RDM/SOIF schemes (i.e. general knowledge about resources) and assertional axioms for RDM/SOIF objects (i.e the actual resources).

Let us consider, at a fairly informal level, the process of writing terminological and assertional axioms starting from the RDM/SOIF elements. A *schema* declares a set of *attributes* for every object defined by that schema. For each attribute, a set of general properties that determine its external interpretation are defined. For example, whether the attribute is an index, its data type, its length, etc.

The following is a RDM/SOIF schema which a DSP could contain about a resource *Paper* which has two attributes, *Title* and *Year*. Besides the attributes, additional information about the schema itself is included:

`Schema-name`, schema name; `URL`, location of the schema; `LastModified`, last modification of the schema.

```
@SCHEMA{ -
  Schema-Name:              Paper
  URL:                      http://someplace
  LastModified:             12/2/2003

  SOIF-Attribute-1{x}:      Title
  Description-1{x}:         Contains the title
  Data-Type-1{x}:           String
  Enforce-Uniqueness-1{x}: Yes

  SOIF-Attribute-2{x}:      Year
  Description-2{x}:         A number identifying the paper
  Data-Type-2{x}:           Integer
  Index-Attribute-2{x}:     1
}
```

4.1 Attributes

In DLs attributes are typically modelled as functional roles between individuals. So, an object with a set of attributes is an individual with a set of functional roles. These roles are filled with values that are individuals. Often, these values are occurrences of concepts such as String, Integer, etc. Most concept languages have these predefined concepts (*host* concepts). For example, to model in a TKRS a schema Paper which defines the objects that have a `Title:STRING` and a `Year:INTEGER`, we need the following terminological axioms:

$\mathsf{String} \sqsubseteq \top$
$\mathsf{Integer} \sqsubseteq \top$
$\mathsf{Title} \sqsubseteq \top$
$\mathsf{Year} \sqsubseteq \top$
$\mathsf{Paper} \sqsubseteq \top$
$\mathsf{hasTitle} = \mathsf{Paper} \times \mathsf{Title}$
$\mathsf{hasYear} = \mathsf{Paper} \times \mathsf{Year}$

If we consider an attribute as an object with properties (description, data-type, etc.) the attribute would become an individual and a set of attributes would become a concept Attribute. With this conceptualisation we can exploit these concepts as any concept in a terminological system. For instance, we can specialise, generalise and classify attributes.

Let us consider the introduction of a description for the attribute `Title`. This description will be the String "ContainstheTitle". We need a concept Title (conceptualisation of the attribute) and a role Description. With constructor numbered as (15) in section 2 we can express that this role is filled with a particular individual (the individual "ContainstheTitle", which is an occurrence of the concept String). So, in a TKRS we can model attributes with terminological axioms using the fills constructor. Notice that with this constructor, terminological axioms rely on the existence of individuals making pure intensional knowledge or reasoning impossible.

First, using the following terminological axioms we define a concept Attribute, a concept Schema, and a role Value.

$\mathsf{Schema} \sqsubseteq \top$

$\mathsf{Attribute} \sqsubseteq \top$

$\mathsf{Value} \sqsubseteq \top \times \top$

Now, we define a set of roles to describe the general properties of an attribute (those shared by all of its occurrences):

$\mathsf{Description} = \mathsf{Attribute} \times \mathsf{String}$

$\mathsf{DataType} = \mathsf{Attribute} \times \mathsf{String}$

$\mathsf{IndexAttribute} = \mathsf{Attribute} \times \mathsf{Integer}$

$\mathsf{EnforceUniqueness} = \mathsf{Attribute} \times \mathsf{Boolean}$

Next, for each SOIF attribute we create a concept subsumed by the concept Attribute. We also have to create the individuals to fill in the roles associated with the attribute properties. We consider the schema shown above in this section. Assuming the existence of the host concepts String and Boolean, we create the individuals "ContainstheTitle" and True with the assertional axioms ("Contains the Title".String) and (True.Boolean). In order to define the concept Title we fill in the roles of attribute characteristics with the proper individuals. Many concept languages include functional roles as primitive elements of the language. However this is not strictly necessary, and we can simply add a qualified number restriction. This is done in the following terminological axiom:

$\mathsf{Title} = \mathsf{Attribute} \sqcap \mathsf{fills}.\text{“}\mathsf{Contains\ the\ Title}\text{”}.\mathsf{Description} \sqcap$
$\quad \mathsf{fills}.\mathsf{True}.\mathsf{EnforceUniqueness} \sqcap \mathsf{1Value}.\mathsf{String}$

Finally, let us write the terminological axiom to define the role hasTitle using the concept Title: $\mathsf{hasTitle} = \mathsf{Paper} \times \mathsf{Title}$

Analogously, we use the following terminological axioms to define the concept Year:

("Contains the Year".String)

$\mathsf{Year} = \mathsf{Attribute} \sqcap \mathsf{fills}.\text{“}\mathsf{Contains\ the\ Year}\text{”}.\mathsf{Description} \sqcap$
$\quad \mathsf{fills}.\mathsf{True}.\mathsf{EnforceUniqueness} \sqcap \mathsf{1Value}.\mathsf{Integer}$

$\mathsf{hasYear} = \mathsf{Paper} \times \mathsf{Year}$

4.2 Schemas

After expressing SOIF attributes with terminological axioms we still have to represent additional information about schemas. We again fill in the properties (roles) of a concept (the schema we are defining) with

individuals. We need the terminological axioms, $\mathsf{URL} = \mathsf{Schema} \times \mathsf{String}$ and $\mathsf{LastModified} = \mathsf{Schema} \times \mathsf{Date}$.
and the assertional axioms, (“http://someplace”.String) and (12/02/2003.Date). Finally, we define a Paper with another terminological axiom,
$\mathsf{Paper} = \mathsf{Schema} \sqcap \mathsf{fills.}$“http://someplace”$\mathsf{.\ URL} \sqcap$
$\quad \mathsf{fills.12/02/2003.LastModified} \sqcap \mathsf{1hasTitle} \sqcap \mathsf{1hasYear}$

4.3 Objects

Now we can introduce the objects defined by the schema as individuals of the concepts previously defined. For example, for the following SOIF object (which follows the previously defined *schema*),

```
@Paper{ -
  Title:    New Worlds
}
```

we will introduce the following assertional axioms (where p1, t1 and y1 are unique names given to the individuals):
(t1.Title), (“New Worlds”.String), (t1.”New Worlds”.Value)
(y1.Year), (2004.Integer), (y1.2004.Value)
(p1.Paper), (p1.t1.hasTitle), (p1.y1.hasYear)

4.4 Queries

In RDM we can build queries by using two languages:

- The *Schema-Basic* query language is used by RDM clients to retrieve schema descriptions from the RDM server. The requests an RDM client can issue in this context are called *Schema-Description Requests*. RDM does not directly provide a method to obtain the description of a certain schema (i.e. you can not query about the schema *Paper*). This kind of information has to be recovered indirectly (i.e. you must query about the schemas that have an attribute called *Author*, or simply, query about the schemas that have any defined attribute).
- The *Gatherer* query language can be used to send requests and receive answers that involve a set of RDs. These requests are called *RD-Requests*. Basically, RDM allows only to retrieve all the RDs of all the schemas of the RDM server. Our translation to DLs assumes not only this basic query but also the retrieval of all the RDs of the schemas that have a certain attribute.

Notice that the expressiveness of RDM queries is very reduced, unlike the DL case. As a result of this, in a DL environment a DL client can not obtain the same information from a RDM node than from another DL node. The RDM node will be seen in a DL environment as a *poor node*. On the other hand, in a RDM environment a DL client must restrict its questions to the reduced expressiveness of RDM queries.

Translation Let *Paper* be a schema with attributes *Title* and *Journal*, and *Technical-Report* another schema with attributes *Title* and *Department*. The Schema-Basic query language allows to ask for the schemas that have a certain attribute, for example *Title*. Bearing in mind the previous conceptualisation of *Title*, in DLs we must only ask if there are defined concepts that are subsumed by the expression concept $(1.\mathsf{hasTitle})$. This question can be formulated in any concept language. In our assumed extension to the basic *Gatherer* queries it would be possible to request the RDs that have any defined attribute, for example *Title*. This can be done in DLs by retrieving the individuals of the defined concepts subsumed by the expression concept $(1.\mathsf{hasTitle})$. Again, this question can be expressed in any concept language.

5 Conclusions and Future Work

The basic terminological model proposed allows us to express RDM messages in a terminological language, obtaining the advantages of the terminological systems. In essence, what we have done is using DLs as a medium to carry RDM messages. The translation is structural and DLs are basically blind to the RDM semantic content.

Once the model is completed we expect the automatic generation of terminological axioms and assertions will not present great difficulty. The integration on the client side of results returned by different DSPs is still more interesting. Moreover, this task could be done in the general case of DL nodes, developing a communication protocol between DL clients, servers and systems which can deal with distributed terminologies.

Acknowledgements:

This work was supported in part by project 10503B96 from Xunta de Galicia.

References

1. F. Baader, M. Buchheit, and B. Hollunder. Cardinality restriction on concepts. *Artificial Intelligence*, 88:195–213, 1996.
2. F. Baader and B. Hollunder. KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14, 1991.
3. A. Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.
4. A. Borgida and R.J. Brachman. Loading data into description reasoners. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 217–226, Washington, DC., USA, May 1993.
5. A. Borgida and P. Patel-Schneider. A semantic and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
6. A. Borgida, Brachman R.J., D.L. McGuiness, and L.A Resnick. CLASSIC: A structural data model for objects. In *Proc. 1989 ACM SIGMOD Conference on Management of Data*, pages 59–67, June 1989.
7. R.J. Brachman and J.G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April, 1985.
8. F. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.
9. F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.
10. D. Hardy. Resource Description Messages(RDM). Technical report, World Wide Web Consortium, 1996.
11. Darren R. Hardy, Michel F. Schwartz, and Duane Wessels. Harvest v1.4 user's manual: Effective use of Internet information. Technical report cu-cs-743-94, University of Colorado at Boulder, 1994.
12. H.J. Levesque and R.J. Brachman. A fundamental tradeoff in knowledge representation and reasoning (revised version). In Ronald J. Brachman and Hector J. Levesque, editors, *Readings in knowledge representation*, pages 817–823. Morgan Kauffmann, Los Altos, CA, 1985.
13. A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Systems, Special Issue on Networked Information Discovery and Retrieval*, 5(2):1–23, 1995.
14. R. MacGregor and R. Bates. The LOOM knowledge representation language. Technical Report ISI/RS-87-188, USC/Information Sciences Institute, Marina del Rey, CA, 1987.
15. C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In *Proc. of SIGIR-93, the 16th ACM Conference on Research and Development in Information Retrieval*, pages 298–307, Pittsburgh, PA, 1993.
16. C. Meghini and U. Straccia. A relevance terminological logic for information retrieval. In *Proc. of SIGIR-96, the 19th ACM Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, August, 1996.

17. B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34:371–383, 1988.
18. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
19. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
20. F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval - SIGIR-94*, Dublin, Ireland, 1994.