Declarative Specification of Z39.50 Wrappers using Description Logics*

Yannis Velegrakis[1]**, Vassilis Christophides[2], Panos Constantopoulos[2,3]

[1] Computer Science Department, University of Toronto,
Toronto, Ontario, Canada M5S-3H5
velgias@cs.toronto.edu
[2] Institute of Computer Science, FORTH,
Vassilika Vouton, P.O.Box 1385, GR 711 10, Heraklion, Greece
[3] Department of Computer Science, University of Crete,
GR 71409, Heraklion, Greece
{christop, panos}@ics.forth.gr

Abstract. Z39.50 is a client/server protocol widely used in digital libraries and museums for searching and retrieving information spread over a number of heterogeneous sources. To overcome semantic and schematic discrepancies among the various data sources the protocol relies on a world view of information as a flat list of fields, called *Access Points* (AP). One of the major issues for building Z39.50 wrappers is to map this unstructured list of APs to the underlying source data. Unfortunately, existing Z39.50 wrappers have been developed from scratch and they do not provide high-level mapping languages with verifiable properties. In this paper, we propose a Description Logic based toolkit for the declarative specification of Z39.50 wrappers. We claim that the conceptualization of AP mappings enables a formal validation of the query translation quality and therefore ensures the quality of the retrieved data. Finally, it allows to tackle a number of Z39.50 pending issues (e.g., metadata retrieval, query failures due to unsupported APs, etc.) by enriching the generated Z39.50 wrappers with a number of added-value services such as conceptual structuring of flat Z39.50 vocabularies and intelligent Z39.50 query assists.

1 Introduction

With the advances in digital processing and communication technologies an increasing number of organizations and individuals are using the Internet for publishing, broadcasting, and exchanging information all over the world. The ability to share, interpret, and manipulate information from multiple sources is a fundamental requirement for large scale applications e.g., digital libraries and museums. A widely used protocol for searching and retrieving information in a distributed environment is Z39.50 [2]. To achieve interoperability [43], Z39.50 (Version 3) relies on (i) standard messages, formats, and procedures governing the communication of clients and servers (system interoperability), (ii) a world view of information as a flat vocabulary of fields, called *Access Points* that abstracts representational details of source data (semantic and schematic interoperability), and (iii) basic textual search primitives to express Boolean queries in the form of field-value pairs (functional interoperability).

* This work was partially supported by the European project AQUARELLE (Telematics Application Programme IE-2005) and the CIMI Interoperability Testbed project.
** Work done while the author was at the ICS-FORTH.

In order to execute Z39.50 queries, sources should wrap their actual data organization, format and access methods according to the Z39.50 specifications established for a specific application, community, etc. These specifications are described in the various *profiles* (i.e. metadata) proposed by national or international bodies (e.g., Library of Congress, CIMI, etc.). It should be stressed that the quality of the established mappings between the source and the Z39.50 view of information is fundamental in order to ensure the quality of the retrieved data (i.e. accuracy, consistency, completeness, etc.). Unfortunately, existing Z39.50 wrappers are developed using some programming language and they do not provide abstract mapping languages with verifiable properties [44, 11, 45]. In this paper, we advocate a Description Logic framework [9] (such as proposed in the context of the DARPA KSE [41]) for the declarative specification of Z39.50 wrappers using high-level concept languages. We claim that modeling the required mappings as first-class citizens, instead of hard coding them in the wrappers (i) allows the formal validation of the translation quality (e.g., ill-defined mappings, inappropriate APs); and (ii) opens unexpected opportunities to tackle a number of Z39.50 pending issues (e.g., metadata retrieval, query failures due to not mapped APs, multiple answer sets handling, etc.).

Building a wrapper for an information source according to a Z39.50 profile (e.g., for digital libraries [34, 33], museums [46, 20], etc.) implies the translation of (i) the Z39.50 Access Points (AP) to the underlying source data structure and semantics, (ii) the Z39.50 Boolean filters to the source query primitives, and (iii) the returned source data from their original format to a predefined Z39.50 record syntax (e.g. GRS-1, XML). For loosely structured sources (e.g., Information Retrieval Systems) wrapping is relatively simple. It essentially requires to define some renaming mappings from the APs to the source data attributes, tags, etc. (e.g., the AP *AU* to the field *author*, etc.). However, for highly structured sources (e.g., Database Management Systems, Knowledge Base Systems) the translation process is considerably more complex. This is mainly due to the fact that there exists a significant mismatch between the Z39.50 flat view of information and the underlying source data model (e.g. relation or class based). In this context, what is really needed is to define for each AP a view on the source data.

To address this issue we introduce an intermediate level between the Z39.50 and the source world, based on advanced knowledge representation and reasoning support, specifically Description Logics (DL). DL provide declarative languages to represent and reason about interrelated sets of objects using modeling primitives such as concepts, roles, and individuals. Starting from a set of primitive concepts and roles representing source conceptualization, we capture the semantics of the AP mappings as derived concepts formed by primitive ones and standard DL concept operators [5]. Since DL can serve both as knowledge representation languages and as query languages [8, 42, 14], derived concepts essentially act as views [15] against which Z39.50 queries are evaluated with source data. Our contribution is twofold : (i) we propose a toolkit for the declarative specification of Z39.50 wrappers using standard DL reasoning mechanisms [23]; and (ii) we enrich the generated Z39.50 wrappers with a number of added-value services such as conceptual structuring of flat Z39.50 vocabularies and intelligent Z39.50 query processing in order to facilitate metadata retrieval and avoid embarrassing query failures due to unsupported, i.e., not mapped, APs.

The rest of the paper is organized as follows. In Section 2 we give an example of a cultural information source and describe the encountered problems to wrap it according to a digital museum Z39.50 profile. In Section 3 we briefly recall the core Description Logic (DL) model and we show how it can be applied for the declarative specification and validation of Z39.50 AP mappings. Section 4 presents the Z39.50 query processing in our DL framework and Section 5 elaborates on the offered added-value wrapping services. The architecture of the Z39.50 wrapper toolkit is presented in Section 6. Finally, we conclude and discuss future work in Section 7.

2 An Example of a Cultural Information Source

In this section we describe the contents and structure of a cultural information source that will be used as running example in the rest of the paper. We focus on the mismatch of the information conceptualization in our test database and a Z39.50 profile for Digital Museums [46, 20], as well as on the consequent problems we have encountered in order to develop a Z39.50 wrapper in the context of the AQUARELLE and CIMIzit projects [38, 37].

2.1 The CLIO system

As a testbed we use the CLIO cultural documentation system, developed at the Institute of Computer Science, Foundation for Research and Technology-Hellas (ICS-FORTH) in close cooperation with the Benaki Museum, Athens and the Historical Museum of Crete, Heraklion. CLIO supports the recording and management of an evolving body of knowledge about ensembles of cultural goods and addresses the needs of museum curators and researchers. The functional kernel of CLIO is the Semantic Index System (SIS) developed by ICS-FORTH [22]. SIS is a persistent storage system based on the object-oriented semantic network data model TELOS [39].

Figure 1 illustrates some features of our example data source inspired by the CLIO system namely *simple* and *multiple classification* as well as *multi-valued* and *optional attributes*. A museum object is represented as an instance of the class "MuseumObject". It may have (*optional attributes*) an owner (class "Owner") and be constructed with the use of one or more (*multi-valued attributes*) materials (class "Material"), processes (class "Process") and techniques (class "Technique"). Each museum object is associated a series of events (class "Event") characterized by their kind, date and involved actor. For instance, the saber of Androutsos (a hero of the 1821 Hellenic Revolution) is made of shaped silver (*multiple instantiation*) and it was constructed by Filimon in 1815. Although not illustrated in our example, SIS-TELOS also supports *simple* and *multiple inheritance*, *unbounded classification*, and treats *attributes as first class citizens* classified on their own.

2.2 Z39.50 wrapping for Digital Museums

Z39.50 [2] is a session oriented and stateful application protocol, based on the client-server architecture. To overcome semantic and schematic discrepancies among the various data sources, Z39.50 relies on a common information model shared by all clients and servers. It consists of a flat list of fields, called *Access Points* (AP) (or more precisely *Use Attributes*), on which queries are expressed. For instance, in the CIMI [20] and AQUARELLE [46] profiles, the supplied APs correspond to general

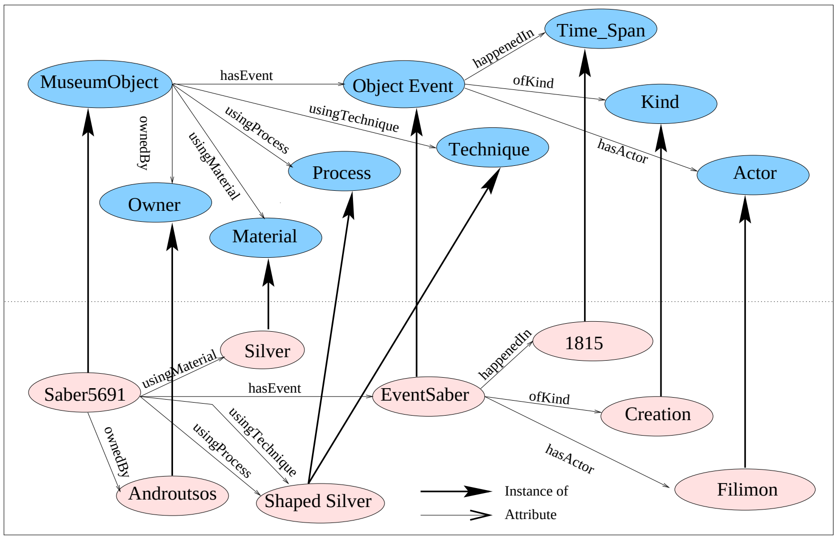

Figure 1: An Example of a Cultural Information Source

information categories like *People* (specific persons or cultural groups), *Dates* of many sorts (including dates of creation, acquisition, exhibition), *Places* (e.g. place of creation, places associated with an event, galleries, provenance), *Subject* (exact description of depicted material), *Style* (including movement and period), *Method* (including process and techniques), *Material*, etc. [31].

This vocabulary of fields is employed by a client in order to search and identify records from the underlying sources and next, to retrieve some or all of them. Z39.50 queries are formulated using Boolean connectors (*and*, *or*, *and-not*), search terms (i.e. Use attribute-value pairs), and qualifiers specifying lexicographical comparisons (e.g., greater than), truncations (e.g. right, left), etc. Going back to our cultural scenario, the following query searches for all the museum objects related with Androutsos, that have been created after 1887 :

Q1: `PersonalName`="Androutsos" `and`
(`DateOfCreation`=1887 `Relation`="GreaterThan")

According to Figure 1, the person *Androutsos* might be the creator (i.e. the actor involved in a creation event), or the owner of the object. This implies that a query on the AP *PersonalName* should be translated by the wrapper into queries on the source *Actor* and *Owner* classes. Furthermore, a query on the AP *DateOfCreation* should be translated into queries on the *Time_Span* class and the associated *Object_Event* and *Kind* classes. Finally, the returned museum objects information, should be formatted/converted by the wrappers according to a common agreed record syntax (e.g. GRS-1, XML) and structure (e.g., elements *ObjectId*, *Title*, *Creator*, etc.).

We believe that the underlying Z39.50 information model is more suitable to query loosely structured text bases than highly structured data sources. Indeed, due to the significant mismatch between the Z39.50 and the source information model, most of the existing structure and semantic richness of the sources is not taken into account during querying while wrapping becomes considerably more complicated. It becomes clear that an AP may be translated to a source query on one or more classes or relations using one or more attribute selections, joins, etc. There is a wide range of Z39.50 mapping cases (see below) and nothing guarantees that the semantics of the specified views correspond to the intended meaning of the APs in the Z39.50 profile: it may be included in the original AP meaning, partially overlapped, disjoint, etc. This is typically the kind of information that is missing from existing Z39.50 wrappers in order to verify the quality of the retrieved data (i.e. accuracy, consistency, completeness, etc.). Two Z39.50 wrapping issues are worth further elaboration and they will be addressed in the rest of the paper.

Unsupported Access Points: Since the AP meaning is defined in a profile without prior knowledge of the source contents, it may correspond to information only implicitly represented in the source or it may not correspond at all to any source information. For example, our cultural source documents objects from the gun collection kept in the *Benaki Museum* and although not explicitly stated, this information could be used to answer queries on the AP *Location*. On the other hand, the AP *Protection Status*, dealing with buildings and monuments, is not at all applicable. According to the protocol both APs are considered as *unsupported* in our source and queries containing them will fail and return diagnostic messages. For large scale applications where queries are generated by a Z39.50 client without a knowledge of wrappers' metadata (i.e. mappings) it is very likely to exist at least one unsupported AP per source. This will result in embarrassing query failures and users risk to obtain no answer from the sources. A commonly used approach to cope with this problem is to omit the unsupported APs from the broadcasted query and try to answer only the supported part. Obviously with this approach users are not aware if the returned answers resulted from the execution of the full query or from a part of it, while the various Z39.50 wrappers behave in an unpredictable manner.

Fixed collections of Retrieved Objects: The information returned in response to a client request is always associated with a specific data collection in the source (e.g. a persistence root). In the rest of the paper we will call this root as *central concept*. No matter what the queried APs are, the answer always correspond to central concept instances (e.g., museum objects) appropriately converted into a record structure having a fixed number of fields (also defined in the profile as *Record Elements*). This implies that all the queried fields are supposed to be connected in a source with the Z39.50 central concept. However, this is not the case with structured sources (relational or object-oriented) where multiple collections are supported and data relationships are not always explicitly stated in the schema (using external keys or object paths). Furthermore, even when such paths are explicitly stated, Z39.50 profiles usually support APs for expressing full-text queries that require navigation over sets of paths. For instance, we may use the AP *Any*, to query on term "Androutsos", without specifying what exactly the related APs are : "Androutsos" may correspond in our cultural source to a person owning an object, a person creating an object, a geographical location, etc. Unless the native query language of the source supports generalized path expressions [18, 19], this kind of mappings cannot easily be expressed in structured sources. It is up to the Z39.50 wrapper administrator to decide query evaluation under these circumstances in a more or less *ad hoc* way.

3 Declarative Specification of Z39.50 Wrappers using DL

Description Logics (DL), also known as terminological logics, has been intensively studied for more than a decade in the field of Knowledge Representation and Reasoning Systems (KRRS). DL provide declarative languages for the representation and reasoning about classes of objects and their relationships, encompassing other well-known formalisms such as entity-relationship or class inheritance models [17]. Recently DL have received considerable attention in the context of information integration systems [3, 35, 16, 27, 6] since it was proved to provide flexible formalisms to model and reason over a large number of data integration views [36]. We follow the same approach to declaratively define the required AP mappings as views over source data. It should be stressed that, compared to previous work on data integration, our context is quite different: (i) Z39.50 wrapping involves only one source at a time (vs. mediation of several sources); (ii) Z39.50 world view of information is intrinsically flat (vs. middleware structured models); and (iii) Z39.50 wrappers support some query processing (vs. simple translations of queries and data). In the sequel, we briefly recall the core DL model that we use to cope with the various Z39.50 wrapping issues presented in the previous section and provide Z39.50 wrappers with formally verifiable mapping specifications.

3.1 The core Description Logic model

The main modeling primitives of Description Logics (DL) are concepts, roles, and individuals. A concept describes a class of elements (individuals) in the domain of interest and is defined by the conditions that must be satisfied by the elements in the class. A role describes a relationship between two individuals. The two basic components of a DL system are the *terminological box* (**TBox**) and the *assertional box* (**ABox**). The former contains the concepts (intentional knowledge) and the latter contains the individuals (extensional knowledge). There exist two types of concepts: *Primitive* and *Derived*. The definition of a primitive concept specifies only the necessary conditions for an individual to be an instance of it. On the other hand, the definition of a derived concept states the necessary and sufficient conditions for an individual to be instance of it. This implies that an individual has to be explicitly defined as instance of a primitive concept, while instances of derived concepts are inferred by the DL system.

The interpretation of a DL knowledge Base Σ is $\mathcal{I} = (\mathcal{I}(\Delta), \mathcal{I}(\cdot))$ where $\mathcal{I}(\Delta)$ denotes a non-empty set of values (the domain) and $\mathcal{I}(\cdot)$ an interpretation function, mapping every concept to a subset of $\mathcal{I}(\Delta)$, every role to a subset of $\mathcal{I}(\Delta) \times \mathcal{I}(\Delta)$, and every individual to an element of $\mathcal{I}(\Delta)$ such that $\mathcal{I}(a) \neq \mathcal{I}(b)$ for different individuals a, b (Unique Name Assumption). Intuitively, the interpretation of a concept C (denoted as $\mathcal{I}(C)$) is the set of objects that are known to belong to that concept. A concept C_1 is said to be *subsumed* by another concept C_2 (denoted as $C_1 \dot{\leq} C_2$) if and only if $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$. Based on this subsumption relation, a set of concepts can form a taxonomy having a *bottom* ($\bot$) and *top* ($\top$) concept.

The part of the **TBox** that contains the primitive concepts is called *schema* part while derived concepts form the *view* part [15]. The **TBox**-*schema* part consists of a finite set of axioms having one of the forms: $A \dot{\leq} D$, $R \dot{\leq} C \times D$, where A, C, D

Name	Concrete Form	Mathem. Repr/tion	Semantics
Concept Name	A	A	$\mathcal{I}(A) \subseteq \mathcal{I}(\Delta)$
Top	TOP	$\top$	Δ
Bottom	BOTTOM	$\perp$	$\emptyset$
Union	(OR C D)	$A \sqcup C$	$\{d_1 \| d_1 \in \mathcal{I}(A) \cup \mathcal{I}(C)\}$
Intersect	(AND C D)	$A \sqcap C$	$\{d_1 \| d_1 \in \mathcal{I}(A) \cap \mathcal{I}(C)\}$
Not	(NOT A)	$\neg A$	$\{d_1 \| d_1 \notin \mathcal{I}(A)\}$
Existential Quantification	(SOME R C)	$\exists R.C$	$\{d_1 \| \exists d_2 : (d_1, d_2) \in \mathcal{I}(R) \wedge d_2 \in \mathcal{I}(C)\}$
Universal Quantification	(ALL R C)	$\forall R.C$	$\{d_1 \| \forall d_2 : (d_1, d_2) \in \mathcal{I}(R) \rightarrow d_2 \in \mathcal{I}(C)\}$
OneOf	(ONEOF $i,\ j, \ldots$)	$\{\ i,\ j, \ldots\}$	$\{\ i,\ j, \ldots\}$
Role name	$R:\ A,\ B$	$_A\|R\|_B$	$\mathcal{I}(R) \subseteq \mathcal{I}(A) \times \mathcal{I}(B))$
Reverse	(REVERSE R)	R^{-1}	$\{(d_1, d_2) \| (d_2, d_1) \in \mathcal{I}(\mathcal{R})$

Table 1. Concept and Role forming operators

are primitive concepts, and R is a role (note that roles have restricted *to* and *from* values). The **TBox**-*view* part consists of a finite set of concepts definitions having the form $A \doteq E$ where A is a derived concept and E is a concept expression formed by other concepts and the operators shown in Table 1. *Cycles* in concept definitions are not allowed (see [40] for formal definitions). In the next subsection we will explain these operators through examples illustrating the mappings of Z39.50 APs to our cultural source. Finally, disjointness of classes in the **TBox** is given by axioms of the form: $A \| C$ (i.e., $\mathcal{I}(A) \cap \mathcal{I}(C) = \emptyset$).

The **ABox** is defined from a finite set of declarations having one of the forms: $C(a)$ and $R(a, b)$. The first one (unary predicates) declares that individual a belongs to the interpretation of the primitive concept C and the second one (binary predicates) declares that there exists a role R from a to b (belonging respectively to the interpretations of concepts C and D in the definition of R). The main reasoning services [23] offered by a DL system Σ are the following:

- **Concept Satisfiability** ($\Sigma \not\models C \equiv \perp$) checking if a concept has not an empty interpretation,
- **Subsumption Checking** ($\Sigma \models C_1 \dot{\leq} C_2$) checking if a concept C_2 subsumes C_1,
- **Instance Checking** ($\Sigma \models C(a)$) checking if an individual a belongs to the interpretation of a concept C.

The above core model corresponds to an almost standard DL framework [5], actually supported by several DL systems[4] e.g., CICLOP[5], FaCT[6], KRIS[7], HAM-ALC[8], etc.

[4] The only subtle issue here is the introduction of restricted and inverse roles as in [28].
[5] http://www-ensais.u-strasbg.fr/LIIA/ciclop/ciclop.htm
[6] http://www.cs.man.ac.uk/˜horrocks/FaCT/
[7] http://www.dfki.uni-sb.de/˜tacos/kris.html
[8] http://kogs-www.informatik.uni-hamburg.de/˜moeller/ham-alc/

3.2 DL concept languages for Z39.50 AP mappings

In a very natural way, source structure and semantics can be represented as primitive concepts and roles, while the AP mappings as derived concepts (i.e. views) defined on top. Figure 2 illustrates the primitive concepts (**TBox**-*schema part*) representing our cultural source schema given in Figure 1 while the derived concepts (**TBox**-*view part*) correspond to the established mappings of the CIMI-AQUARELLE profile APs [20, 46]. The data of our cultural source correspond to the individuals (**ABox**) of the DL System. Note that this is only a logical view of information from the Z39.50 wrappers (see Section 6) and there is no need to actually load source data into the DL system (virtual **Abox**). In the following examples we illustrate the expressive power of the proposed DL concept language (see Table 1) to capture the various kinds of translations involved in Z39.50 wrapping for structured sources (see Section 2).

Example 1: Perhaps the simplest case to map an AP is when its semantics corresponds exactly to one concept of the source schema. For instance, the AP *Date* is translated as follows :

$$Date \doteq Time\_Span$$

Example 2: In most practical cases, APs should be mapped by combining more than one source concepts using the DL *Union* and *Intersect* concept forming operators. For instance, information about persons in our cultural source is represented by the concepts *Actor* and *Owner*, and the AP *PersonalName* is mapped as follows :

$$PersonalName \doteq Actor \sqcup Owner$$

Similarly, the mapping of the AP *Method* is defined as :

$$Method \doteq Process \sqcap Technique$$

Furthermore, mappings of abstract APs like *Who* describing any personal or corporate name that can be found in our source, are defined by using other AP derived concepts such as :

$$Who \doteq PersonalName \sqcup CorporateName$$
$$Any \doteq Who \sqcup What \sqcup When \sqcup Where$$

Finally, APs like *Any*, for full-text queries are easily mapped by considering the definitions of abstract APs like *Who*, *What*, *When* and *Where* (the 4W APs).

Example 3: More complicated situations arise when the AP mapping requires a traversal over the roles associated with aggregated source concepts. For example, to map the AP *DateOfCreation* we need to define the following derived concept using the DL *Inverse Role* operator :

$$DateOfCreation \doteq \exists(happenedIn)^{-1}.(\exists ofKind.\{\text{“}Creation''\})$$

The above expression has three parts: (i) the bracket expression corresponds to a concept having as interpretation only the individual “Creation”, i.e. subsumed by *Kind*, (ii) the parenthesis expression represents the related creation *Event*s, and (iii) the whole expression captures the *Date*s associated with these events. Note that the restriction of a role *to* and *from* values obviates the need to verify that the returned individuals actually belong to the interpretation of *Date*.

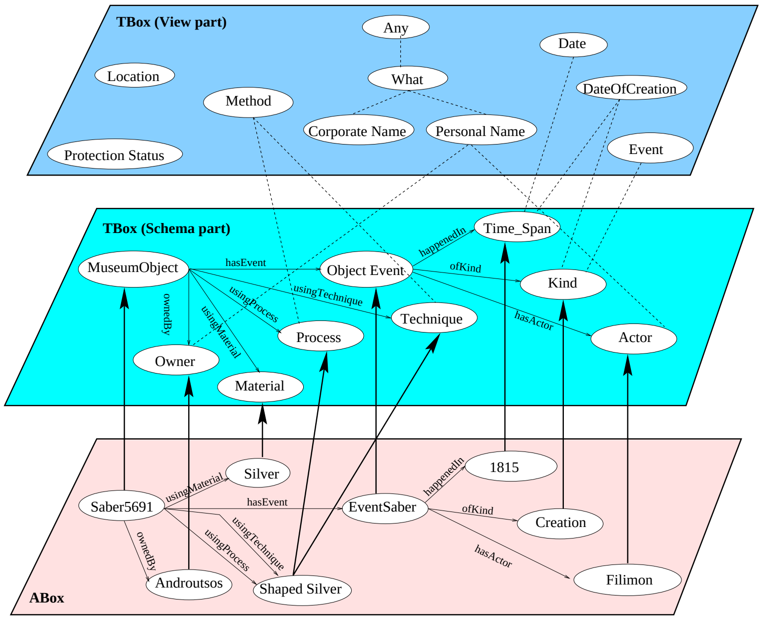

Figure 2: Modeling an Information Source and Z39.50 APs mappings in DL

Example 4: For APs corresponding to information not explicitly stated in a source, the DL *OneOf* concept forming operator is used to translate them. For instance, although not given in our example source, it is known that all objects belong to the Benaki Museum (Athens) gun collection, and hence the APs $CorporateName$, $Location$ and $Collection$ are mapped as follows :

$$CorporateName \doteq \{\text{“}Benaki\ Museum''\}$$
$$Location \doteq \{\text{“}Benaki\ Museum\ Athens''\}$$
$$Collection \doteq \{\text{“}Benaki\ Gun\ Collection''\}$$

This implies that $CorporateName$, $Location$ and $Collection$ are concepts whose interpretation contains only one individual, respectively “Benaki Museum”, “Benaki Museum Athens” and “Gun Collection”.

Example 5: In the case where there is no information in the source corresponding to a specific AP, the related derived concept is defined to be equivalent either to the *Bottom* i.e. the concept with an empty interpretation or the *Top* i.e. the concept whose interpretation contains all the individuals. The decision depends on the expected *precision* and *recall*: the former favors *precision* while the latter *recall*. More precisely, according to the semantics of APs in a Z39.50 profile, we consider that the *Top* for AP mappings is the AP concept Any previously defined (see Example 2). For instance, the AP $ProtectionStatus$ which is used for preserved buildings and cannot be mapped to our cultural source of museum objects, is translated as follows :

$$ProtectionStatus \doteq \bot \text{ (or } ProtectionStatus \doteq Any)$$

In both cases, wrappers are able to smoothly incorporate unsupported APs into the query processing (see SubSection 5.2) and avoid embarrassing query failures.

3.3 Formal validation of Z39.50 wrapping quality

Having defined the mappings of the Z39.50 APs as derived concepts on top of a source schema (i.e. views), standard DL reasoning services like *Concept Satisfiability* can be used to infer if some or all of the APs mappings are ill-defined. Consider, for instance, that the concept *Material* of our culture source is disjoint with the concepts *Technique* and *Process* (see Figure 2). Then, the following mapping of the AP *Method* (see Example 2) is inconsistent:

$$Method \doteq Material \sqcap Process \sqcap Technique$$

Indeed, due to class disjoiness the AP derived concept *Method* describes a necessarily empty set. In our DL framework we can formally check whether $\Sigma \not\models Method \equiv \bot$, i.e *Method* has a contradictory description (i.e. intentional semantics). More generally, we can verify the consistency of all the established mappings (i.e., that are well defined and not mapped to the bottom) without actually accessing the source data, by simply checking whether the **TBox** has at least one model: $\Sigma \not\models$. This kind of quality services are not supported by existing Z39.50 wrappers.

To conclude this section we should note that modeling the AP mappings as DL derived concepts allows to develop Z39.50 wrappers with formally verifiable properties. More precisely, (i) APs whose meaning is not at all or only implicitly represented in the source can be effectively mapped and smoothly incorporated into the query processing; and (ii) consistency of the established APs mapping can be easily checked without accessing the source data. These added value services are quite useful for profile developers, Z39.50 wrappers administrators and end-users.

4 Z39.50 query processing using DL

Since DL can serve both as a knowledge representation language and as a query language [8, 42, 14], Z39.50 queries can also be modeled as derived concepts. More precisely, a query can be seen as a description of the necessary and sufficient conditions that have to be satisfied by the individuals forming its answer set, i.e. its interpretation. Conversely, primitive (i.e., source) or derived concepts (i.e., AP mappings) can be used for data querying by considering their interpretation. In the sequel, we present how the Z39.50 Boolean filters can be (i) translated by the wrappers using the same DL concept language employed to map the Z39.50 APs, and (ii) rewritten by taking into account the defined AP views and the fixed central concept of the data actually returned by a source (see Section 2).

4.1 The core Z39.50 query languages

As we have seen in Section 2, Z39.50 queries are essentially composed of search terms with APs and qualifiers for comparisons, truncations, etc., eventually combined using Boolean connectors. Consider, for instance, the following simple query (i.e. no qualifiers) :

Q2: `PersonalName` = "Androutsos"

Recall that *PersonalName* is an AP, mapped as derived concept (C_{AP}) to the *Actor* and *Owner* concepts, and "Androutsos" a value considered as individual (a). **Q2** can be translated into a basic query to the DL knowledge base Σ using the *Instance Checking* reasoning service ($C_{AP}(a)$) :

$$\Sigma \models PersonalName(\text{"}Androutsos\text{"})$$

If the individual "*Androutsos*" is in the interpretation of the concept *PersonalName* (i.e. the union of the *Actor* and *Owner* interpretations), the knowledge base returns a positive answer and the answer set (i.e., query interpretation) contains only the individual "*Androutsos*". Else the answer set will be empty. More formally, core Z39.50 queries can be defined as DL derived concepts (**Tbox**-query part) that will be interpreted with source individuals (**Abox**) in the following way :

Definition 1. Given a DL knowledge base Σ, an Access Point derived concept C_{AP} and a core Z39.50 query q of the form $AP = a$, the answer set of q is given by the interpretation of the concept $C_q : \mathcal{I}(C_q) = \{a \in \mathcal{O}_\Sigma \mid \Sigma \models C_{AP}(a)\}$, where $\mathcal{O}_\Sigma$ is the set of individuals of Σ.

Note that query answering relies here on some form of closed world assumption [29]. In the style of [24] we make the realistic assumption about complete knowledge of the DL extensional part (i.e., source data) and thus consider in the interpretation of concepts only their known individuals.

Now let us see how we can express Z39.50 queries using relation or truncation qualifiers like, for instance :

Q3: `PersonalName`="Andr" `Truncation`="Right"

These search operators are not directly expressed in a standard DL framework, but they can be captured as external functions. The DL operator **TEST-C** allows to call various *test functions* outside of a DL system. This operator is essentially an escape method from the limits of the DL expressiveness allowing to manipulate individuals using external functions written in some programming language (see e.g., CLASSIC[9] [12]). A *test function* f gets an individual as argument and returns TRUE or FALSE if it satisfies the conditions specified in the body of the function. The interpretation of the expression TEST-C(f) is then all the individuals which, given as argument, the TRUE value is returned by f. Only monotonic functions are considered in this respect. **Q3** can then be translated as follows :

$$\mathcal{I}(C_{Q3}) = \{a \in \mathcal{O}_\Sigma \mid \Sigma \models (PersonalName \sqcap \text{TEST-C}(rtrunc_{\text{"}Andr\text{"}}))\}$$

where $rtrunc_{\text{"}Andr\text{"}}$ is a test function supported by our example source, which performs right truncation on string "Andr".

Finally, the concept forming operators $\sqcap$, $\sqcup$, and $\neg$ (see Table 1) can be straightforward used to capture the Z39.50 Boolean connectors *and*, *or*, and *and-not*.

It should be stressed that when the search operators defined in a Z39.50 profile are not supported by the underlying source, we are confronted with the same problems

[9] http://www.research.att.com/sw/tools/classic/classic.html

as in the case of unsupported APs. To cope with these problems we follow the same approach presented in the previous section, allowing to map unsupported Z39.50 search operators either to the *true* or *false test functions*. The former favors *recall*, since it returns all the individuals of the queried AP concept, while the latter favors *precision*, since it returns the empty set. In both cases, wrappers are able to smoothly incorporate unsupported search operators into the query processing.

4.2 Z39.50 query answering

Unfortunately, the above translation into DL is not sufficient to express the exact semantics of Z39.50 queries as defined in a profile. We have seen in Section 2, that the result of a Z39.50 query is the set of related individuals belonging to a central concept of interest (e.g., the root of museum objects in our cultural scenario), rather than the set of individuals that belong to given AP derived concepts and satisfy the search conditions. To cope with this problem we need to define the central concept (C_C) in the **Tbox** as a derived concept (e.g. $C_C \doteq MuseumObject$) and then introduce *concept path expressions* (P_{AP}) connecting, through roles, the individuals of C_C with the various AP concepts involved in a query. For instance, for the AP derived concept *DateofCreation* used in **Q1** we consider the following path (see Figure 2) :

$$P_{DateofCreation} \doteq \exists hasEvent.(\exists happenedIn.Time\_Span)$$

Since *DateofCreation* is only a simple case and AP derived concepts are usually defined by more complex concept expressions (e.g. *PersonalName*), what is really needed is to declare, for each of the involved primitive concepts (e.g. *Actor*, *Owner*), the corresponding paths to the central concept e.g., (see Figure 2) :

$$P_{PersonalName1} \doteq \exists hasEvent.(\exists hasActor.Actor)$$
$$P_{PersonalName2} \doteq \exists ownedBy.Owner$$

The same approach is followed in order to consider the paths of composite APs (e.g., the 4W APs) defined in terms of others. More formally :

Definition 2. A path expression P_{AP} is a sequence of elements $p = e_1 e_2 \ldots e_n$ such that for $i \in [1, n-1]$: $e_i \in \{\exists\} \cup \{\forall\} \cup \mathcal{R}.$, where $\mathcal{R}$ is the set of the primitive role names (suffixed by ".") and $e_n \in \mathcal{C}$ is the set of primitive concepts.

These paths are then used during Z39.50 query translation to capture the exact answer set (C_{Answer}) with individuals of the central concept. More precisely, we consider the following translation steps :

1. The core Z39.50 queries are initially translated into elementary DL query concepts as described in the previous subsection. For instance, the preliminary translation of **Q1** presented in Section 2 is :
   $\Sigma \models (PersonalName(\text{“}Androutsos\text{”}) \sqcap (DateofCreation \sqcap \text{TEST-C}(gt_{\text{“}1887\text{”}}))$

2. Then the obtained expressions are rewritten into an intermediate *canonical form* by expanding the involved AP derived concepts (**Tbox**-view part) into their constituent primitive ones and introducing the corresponding path expressions (P_{AP}) emanating from the central concept. For instance, the canonical form of **Q1** is :

$$\begin{aligned}\Sigma \models ((\ & \exists hasEvent.(\exists hasActor.Actor(\text{“}Androutsos\text{”}) \sqcup \\ & \exists ownedBy.Owner(\text{“}Androutsos\text{”}))) \sqcap \\ & \exists hasEvent.(\exists happenedIn.(Time\_Span \sqcap \text{TEST-C}(gt_{\text{“}1887\text{”}}))))\end{aligned}$$

3. The final expression of Z39.50 queries (C_{Answer}) is then obtained by considering in the resulting canonical form only the individuals of the central concept (C_C) as follows :

$$\begin{aligned}C_{Answer1} = \{\ a \in \mathcal{O}_\Sigma \mid \Sigma \models & \\ (\ & MuseumObject(a) \sqcap \\ & ((\exists hasEvent.(\exists hasActor.Actor(\text{“}Androutsos\text{”})) \sqcup \\ & \quad \exists ownedBy.Owner(\text{“}Androutsos\text{”})) \sqcap \\ & \quad \exists hasEvent.(\exists happenedIn.(Time\_Span \sqcap \text{TEST-C}(gt_{\text{“}1887\text{”}})))))\}\end{aligned}$$

It should be noted that for Z39.50 queries using full text APs like *Any*, we need to consider the paths to the central concept of all its constituent source concepts. The resulting canonical form essentially represents a set of queries capturing the translation of generalized path expressions at the source schema level [19] without requiring any extension of the underlying source query capabilities. Furthermore, as we will see in next section, the canonical form of Z39.50 queries is a subject of optimization by the wrappers taking into account the subsumption relationships between derived or primitive concepts.

5 Advanced Z39.50 Wrapping Services

In Section 3 we showed the benefits from modeling Z39.50 AP mappings as DL concepts (i.e. views) in order to formally validating their consistency. In this section we focus on the capability of DL-based wrappers to reason about the relationships between the AP views as well as between these views and Z39.50 queries also represented as DL concepts. Specifically, we show (a) how a flat Z39.50 list of APs can be organized in a subsumption taxonomy thus rendering their underlying source-specific conceptual structure; and (b) how Z39.50 queries can be optimized with respect to their intentional semantics without accessing actual source data (virtual **Abox**).

5.1 Conceptual structuring of flat Z39.50 vocabularies

Despite the simplified world view of information as a flat list of APs, Z39.50 profiles are usually developed according to an implicit conceptual structure of the information requested by the users. Indeed, the APs defined in a profile represent real world entities for a particular application, function, or community, at various abstraction levels and with different relationships between them. For example, in the CIMI-AQUARELLE profile [20, 46] we can observe a wide range of APs : from very abstract APs like *Any*, to general ones like *What*, *Who*, *When* and *Where*, (the 4W

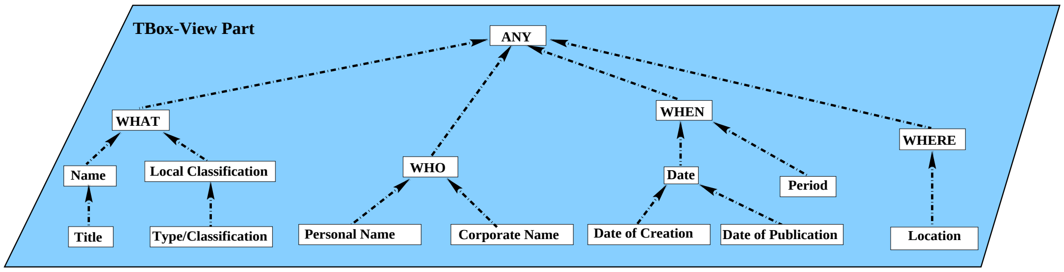

Figure 3: Structuring a Flat Vocabulary of Z39.50 APs

APs) until more specific like *Date* or *DateOfCreation*. Making explicit their relationships in the context of a specific source, is very useful for both end-users and third-party metadata providers. It essentially allows to understand why the conceptual structures of information in a source and a profile differ in order to improve the design of APs, query precision, interpretation of results, etc.

We rely on the DL *Subsumption Checking* reasoning service to organize in a taxonomy the derived concepts capturing the AP mappings for a source. For instance, given the definition of *Date* and *DateOfCreation* (see Section 3) it can be inferred that $DateOfCreation \dot{\leq} Date$ (see [47] for formal definitions). In the simplest case the subsumption relationships are direct consequence of the definitions of composite AP concepts as for instance the 4W APs.

Figure 3 illustrates the subsumption taxonomy of several CIMI-AQUARELLE APs as they are mapped to our example source (**Tbox**-view part). This taxonomy serves as advanced knowledge support about wrapped sources (i.e. metadata) which can be exploited off-line or on-line. In the latter case the Z39.50 `Explain` service[10] can be used. Note that accessing and exchanging source metadata is not a simple task due to the different technologies (DBMS, KBS, etc.) employed by the sources and the various implementation choices made by wrapper administrators. We believe that a DL concept language can also be used to facilitate metadata retrieval (i.e. AP mappings) in a way commonly understood by all clients and independent from the underlying source/wrapper technology.

5.2 Intelligent query processing

In Section 4 we have seen that DL concept languages used to capture the schema of a source and define Z39.50 APs mappings as views on top of it, can also be employed to express the Z39.50 queries against these views. Not surprisingly Z39.50 queries can then be classified into the concept taxonomy using the subsumption relationships between them and the other primitive or derived concepts (**Tbox**). The first benefit from this classification is to determine if a Z39.50 query can be effectively evaluated against the existing source schema and AP views. Indeed, after the translation of Z39.50 queries into a canonical DL form, wrappers are able to check whether the description (intention) of a query is contradictory without accessing the source data

[10] A service allowing Z39.50 clients to retrieve information about servers.

(**ABox**). For instance, the following query can be detected as inconsistent since it uses the AP *ProtectionStatus* mapped to the *bottom* concept.

Q4: `PersonalName` = "Androutsos" `and` `ProtectionStatus` = "Preserved"

If now a query is semantically well-defined it can be appropriately classified by determining the set of its immediate *subsumers* and *subsumees*, i.e. the concepts found above or below in the taxonomy. This classification opens interesting optimization opportunities since it induces a set of semantic transformations in order to locate the exact place of concepts in the taxonomy [7]. Consider, for instance, the following query where the derived concept *Who* subsumes *PersonalName* (see Figure 3) :

Q5: `PersonalName`="Androutsos" `or` `Who`="Androutsos"

Q5 will be rewritten into the following semantically equivalent query that will be actually executed by the source :

Q5': `Who` = "Androutsos"

Recall that according to the semantics of Z39.50 queries, the result is always composed of individuals of a central concept (C_C) like *MuseumObject*. Therefore Z39.50 queries like **Q5** are always classified under C_C defined in the **Tbox**-view part. This enables an intelligent caching of query results [26, 4] by the wrappers and a consequent optimization of Z39.50 queries. If the concept representing a query is found to be equivalent to one already existing in the taxonomy, the interpretation of that concept can be returned as an answer set instead of evaluating it. This is the case of **Q5** assuming that the equivalent query **Q5'** has been previously evaluated and cached. Alternatively, the interpretations of all the immediate subsumers have to be checked against the query conditions. This is extremely useful, as Z39.50 is a stateful protocol and queries are quite often simple refinements of previously issued ones, like for example :

Q6: `Q5'` `and` `When` = 1815

In this case **Q5'** subsumes **Q6** and only the second part of the query needs to be executed by the source (intersection is performed locally by the wrapper). Finally, the results of **Q6** could also be cached in the wrapper. This implies that the cached interpretation of concept **Q5'** will now contain only its proper individuals i.e. those not belonging to the interpretations of its immediate subsumees like **Q6**. Note that supporting several query answer sets proves to be quite expensive with current implementations of Z39.50 wrappers [44, 11, 45] replicating overlapped results.

6 Implementing a DL-based Z39.50 Wrapper Toolkit

The architecture of the DL-based Toolkit we have developed [47] is shown in Figure 4. It is composed of the following five modules :

Module 1 is responsible for *network communication* with the client and is based on the *Yaz* toolkit [30]. When it receives a search request it decodes it into appropriate C structures. More specifically, it produces the *syntax tree* of the query that is included in the search request and sends it to Module 2. When a response has to be sent back to the client, this module is responsible for the transformation of the answer to the appropriate network format.

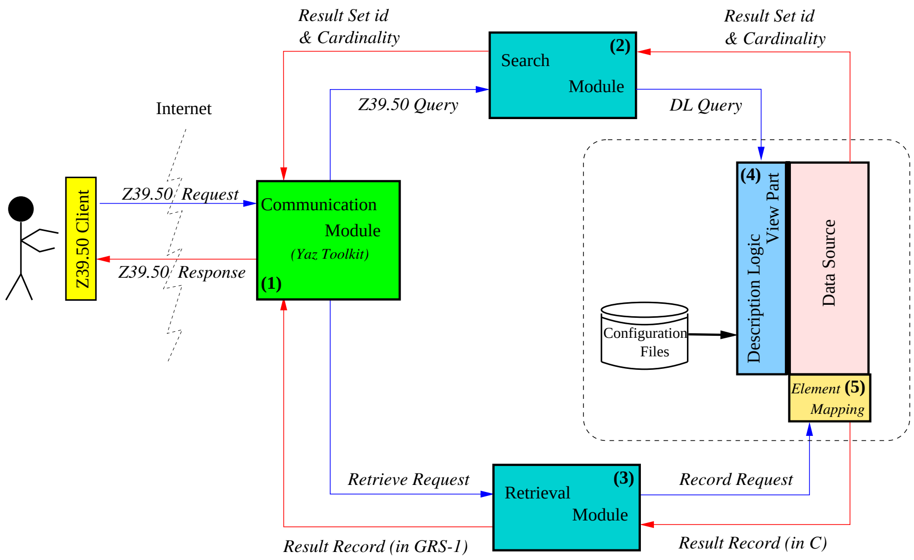

Figure 4: The Z39.50 Wrapper Toolkit Architecture

Module 2 is used only during the *search* process. When it receives the syntax tree of a Z39.50 query, it translates it to a preliminary DL expression (see Section 4) that is sent to Module 4 for evaluation. After the execution, it receives the *id* and the cardinality of the result set (not the data themselves) and forwards this information to Module 1 to be sent back to the client.

Module 3 is used only during the *retrieval* process. After receiving a Z39.50 result set *id* it communicates with Module 5 to get the retrieved records in the form of C++ structures. The task of Module 3 is then to encode the returned C++ structures in one of the record formats defined in the Z39.50 profile (i.e. GRS-1, USMARC or XML) in order to send the retrieved records back to Module 1.

Modules 4 and 5 essentially form the *DL-based wrapper* for the underlying source (see dotted line in Figure 4). Module 4 loads the source schema and the AP mappings (**Tbox**) from a configuration file while the data reside in the source (virtual **Abox**) and can only be cached in the DL system. When it receives a DL query from Module 2, it rewrites it according to the defined AP mappings (see Section 3) and central concept of interest and forwards the resulting expression for evaluation to the underlying source (in our example, SIS). Finally, Module 5 converts the retrieved objects of the central concept by taking into account the mappings of the Z39.50 Record Elements to the source data. Although not presented in this paper, these mappings are defined similarly to the APs.

All modules are operational while Module 4 actually supports only the DL *Instance Checking* service and sources built on top of the SIS-Telos [22]. Due to the similarities between the DL and SIS-Telos query models, the translation of the resulting DL query expressions into our cultural source is straightforward. We plan to extend this interface of Module 4 for other data source technologies, especially relational and object DBMSs (SQL, OQL), as already studied in [10, 13, 32].

To conclude, the modular architecture of the proposed toolkit allows to significantly reduce wrapper development and maintenance costs. First, the DL-based Module 4 can be reused in order to wrap the same source according to multiple, possibly overlapping profiles (e.g., AQUARELLE-CIMI and Dublin Core). This obviate the need to merge different Z39.50 profiles into one, in order to be supported by the existing wrappers [44, 11, 45]. In our approach, the profile becomes a characteristic of the client query, rather than a characteristic of the source. Second, the same Z39.50 server can support several wrapped sources. This is due to the fact that Modules 1,2 and 3 need not be aware of the Z39.50 APs (or Element) mappings to the various source data. This information is requested only by Module 4, i.e. the source wrapper. Hence, a server can support simultaneously sources of different technology, as well as Z39.50 profiles with different APs mappings in each data source.

7 Conclusion and Future Work

In this work we have addressed the declarative specification of Z39.50 wrappers. We have presented a wrapper generation toolkit based on DL concept languages in order to map the Z39.50 world view of information to the underlying source data structure and semantics. The proposed DL mapping language offers a number of advantages : (i) the required views over source data can be easily defined while a wide range of Z39.50 translation cases can be expressed (unlike standard DBMS query languages such as SQL); (ii) it comes equipped with formally verifiable properties allowing to check the consistency of the defined views and therefore ensure the quality of the retrieved data; (iii) it enables reasoning about the relationships between these views and thus rending explicit to Z39.50 profile developers, end-users, etc., the conceptual structure of the Z39.50 vocabularies for a specific source; and (iv) it can serve to translate Z39.50 queries, which opens interesting opportunities for semantic query optimization and caching of results, exploiting as much as possible the stateful nature of the protocol.

Currently, the developed toolkit supports only the DL *Instance Checking* service for evaluating queries and sources built on top of SIS-Telos [22]. We plan to complete the implementation of the toolkit in order to provide full-fledged DL reasoning services. There is on-going study of the available DL systems[11] for possible integration in our toolkit. Furthermore, we intend to validate our approach with several Z39.50 profiles and extend the wrapping facilities to other data source technologies (e.g., DBMS, IRS, etc.). Last but not least, we plan to apply the ideas presented in this paper at a higher level of information integration, in order to build intelligent mediators instead of wrappers [1].

Acknowledgments

We are grateful to the AQUARELLE and CIMI Consortiums for their technical support during this project. We also thank A. Analyti, D. Plexousakis and M. Döerr for helpful comments on a preliminary version of this paper.

[11] http://www.ida.liu.se/labs/iislab/people/patla/DL/systems.html

References

1. B. Amann, V. Christophides, I. Fundulaki, M. Scholl, and A. Vercoustre. Intelligent Mediation of Cultural Information Sources. *ERCIM NEWS*, October 1998.
2. ANSI/NISO. Z39.50 (versions 2 and 3) Information Retrieval: Application Service Definition and Protocol Specification, 1995.
3. Y. Arens, C. Y. Chee, C.-N. Hsu, and C. A. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. *International Journal of Cooperative Information Systems*, 2(2):127–158, 1993.
4. N. Ashish, C.A. Knoblock, and C. Shahabi. Intelligent Caching for Information Mediators: A KR Based Approach. In *Proc. of 5th Workshop KRDB'98*, pages 3.1–3.7, Seattle, Washington, USA, 1998.
5. F. Baader, H. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt, and H. Profitlich. Terminological Knowledge Representation: A proposal for a Terminological Logic. In Nebel B., Luck K. von, and Peltason C., editors, *International Workshop on Terminological Logics*, Dagstuhl, Germany, 1991. DFKI.
6. S. Bergamaschi, S. Castano, and M. Vincini. Semantic Integration of Semistructured and Structured Data Sources". *SIGMOD Record Special Issue on Semantic Interoperability in Global Information*, 28(1), March 1999.
7. S. Bergamaschi, C. Sartori, and M. Vincini. DL Techniques for Intensional Query Answering in OODBs. In *Proc. of 2nd Workshop KRDB'95*, Bielefeld, Germany, September 1995.
8. A. Borgida. Description Logics for Querying Databases. In *Proc. of the 1st Intern. Workshop on Description Logics*, pages 95–96, Bonn, Germany, May 1994.
9. A. Borgida. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, October 1995.
10. A. Borgida and R. J. Brachman. Loading Data into Description Reasoners. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 217–226, June 1993.
11. V. Bouthors, J. Dupuis, and N. T. Huu. Z39.50 Gateway for Mistral and ARF DTD Specification. Aquarelle project, deliverable 5.2, INRIA, France, September 1997.
12. R. Brachman, D. McGuinness, P.P. Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and How to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks — Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann, 1991.
13. P. Bresciani. Uniformly Querying Knowledge Bases and Data Bases. In *Proc. of 1st Workshop KRDB'94*, Saarbrücken, Germany, September 1994.
14. P. Bresciani. Querying Databases from Description Logics. In *Proc. of 2nd Workshop KRDB'95*, Bielefeld, Germany, September 1995.
15. M. Buchheit, F. Donini, W. Nutt, and A. Schaerf. Terminological Systems Revisited: Terminology=Schema + Views. In *Proc. of 1st Workshop KRDB'94*, Saarbrücken, Germany, September 1994.
16. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logic Framework for Information Integration. In *Proc. of the 6st Conf. on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13, Trento, Italy, June 1998.
17. D. Calvanese, M. Lenzerini, and D. Nardi. Description Logics for Conceptual Data Modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, 1998.
18. V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 313–324, Minneapolis, Minnesota, May 1994.

19. V. Christophides, S. Cluet, and G. Moerkotte. Evaluating Queries with Generalized Path Expressions. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 413–422, Montreal, Canada, June 1996.
20. CIMI. The CIMI Profile Release 1.0h: A Z39.50 Profile for Cultural Heritage Information. Technical report, Consortium for the Computer Interchange of Museum Information, Available at http://www.cimi.org/documents/HarmonizedProfile/-HarmonProfile1.htm, November 1998.
21. CNIDR. The Scientific and Technical Attribute and Element Set (STAS). Technical report, Center for Networked Information Discovery and Retrieval at MCNC, Available at http://stas.cnidr.org/STAS.html, 1995.
22. P. Constantopoulos and M. Doerr. The SIS System: A brief presentation. ICS-FORTH, http//www.csi.forth.gr/isst, May 1993.
23. F.D. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasonig in Description Logics. In G. Brewka, editor, *Principles of Knowledge Representation and Reasoning*, Studies in Logic, Language and Information, pages 193–238. CLSI Publications, 1996.
24. F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Queries, Rules and Definitions as Epistemic Sentences in Concept Languages. *Lecture Notes in Computer Science*, 810:113, 1994.
25. FGDC. Z39.50 Application Profile for Geospatial Metadata (GEO). Technical report, U.S. Federal Geographic Data Committee, Available at http://130.11.52.178/-GEO21a.html, 1997.
26. A. Goni, A. Illarramendi, E. Mena, and J.M. Blanco. An Optimal Cache for a Federated Database System. *Journal of Intelligent Information Systems (JIIS)*, 9(2):125–155, 1997.
27. A. Goni, E. Mena, and A. Illarramendi. *Information Modelling and Knowledge Bases*, chapter Querying Heterogeneous and Distributed Data Repositories using Ontologies, pages 19–34. IOS Press, 1998.
28. I. Horrocks and U. Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. In *Proc. of the 5th Intern. Workshop on Description Logics*, Povo-Trento, Italy, June 1998.
29. U. Hustadt. Do we need the Closed World Assumption in Knowledge Representation? In *Proc. of 1st Workshop KRDB'94*, Saarbrücken, Germany, September 1994.
30. Index Data, Available at http://www.indexdata.dk/yaz. *Yaz User's Guide and Reference Manual*, version 1.4 edition, 1997.
31. K. Janney and J. Sledge. A User Model for CIMI Z39.50 Application Profile. September 1995.
32. T. Kessel, M. Schlick, H.-M. Speiser, U. Brinkschulte, and H. Vogelsang. C3L+++: Implementing a Description Logics System on Top of an Object-Oriented Database System. In *Proc. of 3nd Workshop KRDB'96*, Budapest, Hungary, August 1996.
33. LC. Z39.50 Profile for Access to Digital Collections. Technical report, Library of Congress, Available at http://lcweb.loc.gov/z3950/agency/profiles/collections.html, 1996.
34. LC. Application Profile for the Government Information Locator Service (GILS). Technical report, Library of Congress, Available at http://www.gils.net/prof_v2.html, 1997.
35. A.Y. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Sources Using Source descriptions. In *Proc. of Inter. Conf. on Very Large Databases*, pages 251–262, Bombay, India, September 1996.
36. A.Y. Levy and M.C. Rousset. Using Description Logics to Model and Reason About Views. In W. Wahlster, editor, *Proc. of 12th European Conf. in Artificial Intelligence (ECAI)*, Budapest Hungary, August 1996. John Wiley & Sons, Ltd.

37. A. Michard, V. Christophides, M. Scholl, M. Stapleton, D. Sutcliffe, and A-M. Vercoustre. The Aquarelle Ressource Discovery System. *Journal of Computer Networks and ISDN Systems*, 30(13):1185–1200, August 1998.
38. W.E. Moen. Accessing Distributed Cultural Heritage Information. *Comm. of ACM*, 41(4):45–48, April 1998.
39. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, 1990.
40. B. Nebel. Terminological Cycles: Semantics and Computational Properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–362. Morgan Kaufmann, San Mateo, 1991.
41. R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, and R. Neches. The DARPA Knowledge Sharing Effort: Progress Report. In C. Rich, B. Nebel, and W. Swartout, editors, *Proc. of the Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, 1992. Morgan Kaufmann.
42. K. Schild. The use of Description Logics as Database Query Languages. In *Proc. of 2nd Workshop KRDB'95*, Bielefeld, Germany, September 1995.
43. A. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In M.F. Goodchild, M.J. Egenhofer, R. Fegeas, and C.A. Kottman, editors, *Interoperating Geographic Information Systems*. Kluwer Academic Publishers, February 1999.
44. O. Signore and M. Loffredo. Z39.50-SQL Gateways: Technical Description. Aquarelle project, deliverable 5.1, CNR-CNUCE, Italy, April 1997.
45. SSL. Z39.50 version of Index+: Technical Description. Aquarelle project, deliverable 5.3, System Simulation Ltd, UK, October 1997.
46. SSL. Aquarelle Z39.50 Profile. Technical report, Aquarelle: The Information Network on Cultural Heritage, Available at http://aqua.inria.fr/Aquarelle/Public/EN/profile-2.0.html, May 1998.
47. Y. Velegrakis. Declarative Specification of Z39.50 Wrappers using Description Logics. Technical Report FORTH-ICS-TR-225, Computer Science Institute, Foundation of Research and Technology (ICS-FORTH) - Hellas, July 1998. M.Sc. thesis, Computer Science Department, University of Crete.