Domain Concept-Based Queries for Cancer Research Data Sources

Alejandra González Beltrán, Anthony Finkelstein
Department of Computer Science, University College London
London, UK
{a.gonzalezbeltran,a.finkelstein}@cs.ucl.ac.uk

J. Max Wilkinson
NCRI Informatics Initiative
London, UK
max.wilkinson@ncri.org.uk

Jeff Kramer
Deparment of Computer Science, Imperial College London
London, UK
j.kramer@imperial.ac.uk

Abstract

Biomedical scientists generate, access, validate and interpret multiple distributed and heterogeneous data sets. Semantic annotations for these data sets are paramount for exchanging and using the data, and take the form of concepts from a domain ontology. ONIX is a platform that facilitates the access to cancer research data resources and one of its goals is to interoperate with caGrid – a grid computing infrastructure for data sharing. In this paper, we present the ONIX approach to building a semantic layer with support for concept-based queries, which exploit semantic annotations of resources, focusing on caGrid resources. The main contributions of this work are: the automatic generation of OWL ontologies from resources' metadata; concept-based query construction and validation; rewriting and translation from concept-based queries to the caGrid query language.

1. Introduction

The UK National Cancer Research Institute (NCRI) Informatics Initiative is developing a platform called ONIX, the ONcology Information eXchange, to facilitate access to distributed data resources generated from cancer research. ONIX aims to interoperate with, and is based on components from, the caGrid infrastructure [13], which is part of the cancer Biomedical Informatics Grid (caBIG$^{®}$)[1] programme[2].

caGrid is a metadata-aware grid environment equipped with three core services: an XML schema repository, vocabulary services and a metadata registry [13]. The data resources available on the Grid, in the form of caGrid data services, follow a Model-Driven Architecture approach that uses a UML representation of their structure [11]. The Global Model Exchange (GME) is the core service managing a registry of XML Schemas corresponding to UML models in the Grid. The Enterprise Vocabulary Service (EVS) manages controlled vocabularies and ontologies across the infrastructure. The NCI thesaurus (NCIt) is the main vocabulary used. The third core service is the cancer Data Standards Repository (caDSR), which manages common data elements (CDEs) derived from annotating UML models with concepts from NCIt. These specialised models are called domain information models [11]. A CDE is an atomic unit of metadata whose meaning is well-defined – through mappings to concepts of a domain ontology, which provide unanmbiguous definitions – and can be reused for the benefit of interoperability. Specifically, CDEs in caDSR have information about the mappings that associate UML classes, UML attributes and UML associations with one or more concepts from the NCIt ontology [11].

Each caGrid data service exposes a caGrid/Common Query Language (CQL) interface, which is backed by a CQL query processor [13]. This processor translates CQL queries to the native query language of the resource and converts the results. CQL is a simple, though not very expressive, object-oriented query language. It is based on the syntactic and structural information of the underlying object model of the data resources. Thus, caGrid data services build a structural abstraction layer over the data sources. Each data service exposes metadata, which is independently maintained by the three main services: EVS, caDSR and GME described above. While CQL allows for queries to a

[1]`https://cabig.nci.nih.gov`
[2]Note: a glossary of terms is included as an appendix, listing all the acronyms used and their meanings.

single resource, DCQL (for distributed CQL) is a extended language for multiple resources. A Federated Query Processor (FQP) service deals with DCQL queries, translates them into CQL and submits them to target resources. Then, FQP retrieves the results for each independent resource.

Though caGrid is based on a rich metadata infrastructure, it does not support semantic queries nor data integration: two crucial problems for cancer researchers. Semantic queries are queries based on the relationships between entities rather than simply their syntax or structure. Data integration entails combining data from different resources to provide a unified view of the data.

In this paper, we present an approach that extends the caGrid infrastructure with a semantic layer that takes advantage of Semantic Web technologies by building ontological representations of the metadata. This approach deals with both problems – semantic queries and data integration. It has been been designed within the ONIX system so that non-caGrid resources can potentially be incorporated in the semantic layer, as long as these data resources provide well-annotated information models. In particular, in this paper we show the components required to support high-level and descriptive semantic queries, expressed using domain concepts, for caGrid resources. By high-level query, we refer to a query that can be written without specific details on the structure of the target resource. By descriptive query, we mean queries that provide the criteria for the desired data rather than the procedure to find the data. This works presents the theoretical framework and details on the prototype implementation for semantic queries. Moreover, our approach has also been applied for the problem of data integration over caGrid data services, and we refer the reader to [12] for more details.

This paper is organised as follows. Below, we present a motivating example for domain concept-based queries, which highlights the importance of this functionality. In Section 2, we outline the approach and present the system architecture to support semantic concept-based over caGrid. Sections 3 and 4 describe the two main components of the system. We analyse related work in Section 5 and conclude the paper in Section 6.

Motivating Example. To show the utility of our approach, we consider a motivating example given from the perspective of a biomedical scientist. Single Nucleotide Polymorphisms (SNPs) occur throughout the genome and have been shown to have associations with, and on rare occasion cause, a particular disease. SNPs can be characterised by their polymorphism, their location or where they exist within a gene. The gene *Transforming Growth Factor Beta 1* (TGFB1) has been implicated in many diseases, including cancer. A scientist is interested in examining the effect of all or certain SNPs present in the gene of TGFB1. To start, it is necessary to identify all the SNPs available for TGFB1.

As a target resource, the scientist can use the cancer Bioninformatics Infrastructure Objects (caBIO) caGrid data service [11]. Figure 1 shows the part of the caBIO UML model version 4.0 relevant for the query. Currently in caGrid, the scientist must know the structure of target resources in order to build a query to run against them. They must manually determine possible, or rational, connections within the model and/or between different models. In caBIO, it is necessary to determine the path associating a SNP with a gene. Figure 1 shows that the *SNP* class is associated with *RelativeLocation*, which is a generalisation of *MarkerRelativeLocation* and *GeneRelativeLocation*. In other words, *RelativeLocation* is the parent class of the other two. In UML, associations are inherited and thus, the class *SNP* is associated with the two children classes: *MarkerRelativeLocation* and *GeneRelativeLocation*. In turn, the *GeneRelativeLocation* class is associated with the class *Gene*. Consequently, SNPs associated with genes can be retrieved by considering the following path between the UML classes: *SNP* → *GeneRelativeLocation* → *Gene*. Listing 1 shows the resulting CQL query, where the path of associations and attribute conditions have to be stated. Thus, this query is procedural rather than descriptive. At the time of writing, this query cannot be written using the caGrid portal[3], because inherited associations are not considered in the query builder.

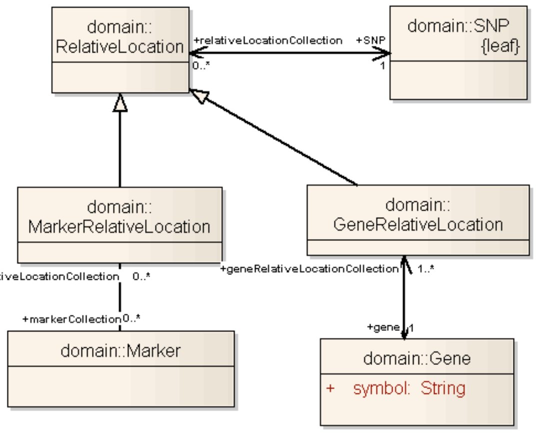

Figure 1: Part of the caBIO 4.0 model

We note that while UML models are annotated with concepts from NCIt, these concepts are not used for query construction, as shown above. Our aim is to allow scientists to build queries based on domain concepts, rather than on syntax and structure of the resources. Considering NCIt[4] concepts, SNPs correspond to the concept *Single_Nucleotide_Polymorphism* (C18279), and genes corre-

[3] `http://cagrid-portal.nci.nih.gov`, version 2.2

[4] We are considering the current OWL version of NCIt, namely version 9.02d.

spond to the concept *Gene* (C16612). From now on, we will refer to the concept *Gene* as *GeneConcept*, to distinguish it from the *Gene* class. Additionally, the symbol of a gene should have concept *Gene_Symbol* (C43568). Consequently, a scientist can think much more naturally about the query as: **find objects annotated with *Single_Nucleotide_Polymorphism* that have an association with objects annotated with *GeneConcept*, which in turn have an attribute whose concept is *Gene_Symbol***. In the rest of the paper, we present several processing steps to translate from a concept-based query, similar to the natural language expression above, into a CQL query. We will revisit the motivating example in each step as a way of exemplifying the process, which involves, among other things, finding possible paths between classes and will be automated.

Listing 1: CQL Query

```
<ns1:CQLQuery xmlns:ns1="http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLQuery">
<ns1:Target name="gov.nih.nci.cabio.domain.SNP">
  <ns1:Association name="gov.nih.nci.cabio.domain.GeneRelativeLocation" roleName=
    "relativeLocationCollection">
   <ns1:Association name="gov.nih.nci.cabio.domain.Gene" roleName="gene">
    <ns1:Attribute name="symbol" predicate="EQUAL_TO" value="TGFB1"/>
   </ns1:Association>
  </ns1:Association>
</ns1:Target>
</ns1:CQLQuery>
```

2. Approach and System Architecture

Supporting concept-based queries, as in the example, involves reasoning over resources' metadata. Semantic Web technologies deal with knowledge representation and reasoning. Reasoning refers to the process of making explicit knowledge that is otherwise implicit in the given representation [1]. The Web Ontology Language (OWL)[5] is a W3C recommendation for knowledge modeling, which has formal foundations in Description Logics (DLs) [1]. To be able to reason about metadata, we propose building a semantic layer within ONIX, on top of the current caGrid structural layer. This semantic layer is based on OWL ontological representations of the caGrid information models, which offer an integrated view of their syntax, structure and semantics. The approach can be applied to other systems as long as they deal with annotated UML models.

Figure 2 shows the extension of the caGrid service-oriented architecture with novel semantic services (shaded). In particular, in this paper we will present two components:

OWL generation service : in charge of developing OWL ontologies from information models (annotated UML models)

Semantic query service : responsible for rewriting, translating and processing semantic queries at different levels of abstraction with CQL/DCQL as a target language.

We have developed prototype implementations of these components, using the OWLAPI[6] and the Pellet reasoner[7]. The OWL generator uses as metadata provider either a particular caGrid data service, given its URL, or the caDSR service, given the project name under which the model is registered in caDSR. The semantic query prototype supports queries over single caGrid data services, which are transformed to CQL queries.

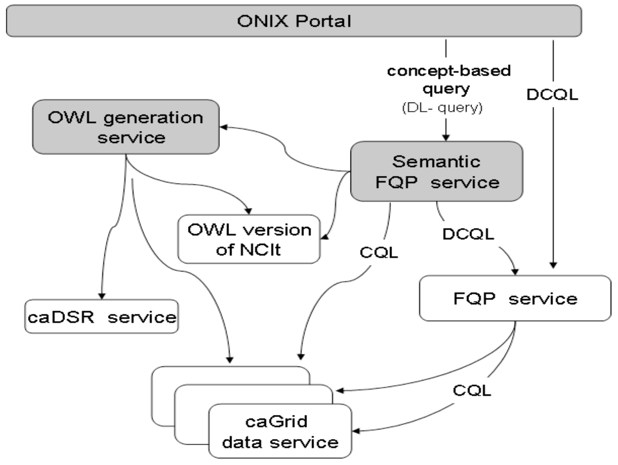

Figure 2: System Architecture

3. OWL generation from caGrid Information Models

UML and OWL are representation languages designed to address the needs of object-oriented development and semantic web, respectively. While both languages share some common constructs (e.g. the class element, generalisations, associations), there are significant differences be-

[5] `http://www.w3.org/TR/owl-ref/`
[6] OWLAPI,`http://owlapi.sourceforge.net/`
[7] Pellet, `http://clarkparsia.com/pellet/`

tween them. UML is a visual language with no formal underpinnings, while OWL has formal semantics based on DLs [1]. Additionally, UML uses a Closed World Assumption (CWA) but OWL uses an Open World Assumption (OWA)[1]. Basically, the distinction lies in the treatment of absence of information: what is not known to be true in CWA is assumed to be false, but no assumptions are made in OWA about its truth value.

There have been several approaches converting UML into OWL and vice versa. A review is presented in Section 5. All the approaches map UML classes to OWL classes, UML attributes to datatype properties and UML associations to object properties. To the best of our knowledge, semCDI [15] is the only UML-to-OWL conversion over caGrid models: it adopts the previously mentioned mapping but also considers annotated UML classes. A NCIt concept is mapped to an OWL class and the UML class/concept relationship is modeled as subsumption, i.e. the UML class as a subclass of the concept class. However, only isolated concepts from NCIt are included in the ontology and no other annotations of UML constructs are considered (e.g. annotations for UML attributes).

semCDI representation has some issues. If NCIt is imported, using subsumption for UML class/concept relationship may result in an inconsistent ontology. This is due to the UML classes annotated with disjoint NCIt concepts. For example, in caBIO 4.0, the *RelativeLocation* class has the concepts *Location*, *Relative_Value* and *Chromosome*, where the first two subcasses of *Properties_or_Attributes* in NCIt and *Chromosome* is subclass of *Anatomic_Structure_System_or_Substance*, and these two higher-level concepts are disjoint. If UML attributes are modeled as datatype properties, as in semCDI, their annotated concepts could only be included as OWL annotations properties, which are used to represent metadata on OWL constructs, but are not considered in reasoning. Thus, in semCDI representation, it is not possible to reason about UML attributes' annotations.

For the reasons stated above, in our annotated-UML-to-OWL transformation, UML attributes are mapped to OWL classes and the UML construct/concept relationship is modeled as an object property. We follow a modular approach for the development of the ontologies, which preserves NCIt semantics and it is described below.

We generate two alternative information model ontologies, following the same pattern, according to the reasoning needs for semantic queries or data integration support. For example, for data integration, we include cardinality restrictions to represent multiplicity of UML associations [12]. Below, we focus on the generated ontology for semantic queries and will show that it cannot contain cardinality restrictions. The generated ontologies model services' metadata but can also be extended to contain instance data as individuals, as shown in [12].

NCIt semantics. To perform semantic queries and data integration, it is required to derive knowledge from NCIt onto the information models. This can be achieved by importing NCIt and using reasoning. However, NCIt is a very large ontology, which covers the entire biomedical domain. In terms of performance, reasoning over the whole NCIt is expensive. Each information model is only concerned with a subset of the knowledge represented by NCIt. Thus, for efficiency and succinctness, we build NCIt modules using the methodology from [9], which balances the trade-off between the properties of safety, coverage and economy. Safety ensures that the semantics of imported concepts is not changed [9]. Coverage refers to importing everything that is relevant and economy stipulates that only what is relevant for the chosen terms should be imported [9]. It is observed that using the relevant module from NCIt for a particular information model is enough for formulating all possible concept-based queries over it: due to the coverage criterion, if a concept has been used to annotate the information model, it will be present in the extracted module.

Semantic Query ontology. We developed an ontology defining the semantic query vocabulary. This is a combination of UML constructs and semantic annotations. We included OWL classes called *UMLClass* and *UMLAttribute* to represent the base class for all UML classes and attributes, respectively. The object property *hasConcept* is used to model UML construct/NCIt concept relationship, *hasAssociation* to represent UML associations between classes, *hasAttribute* to represent a relationship between a *UMLClass* and a *UMLAttribute*. The property *hasAssociation* is defined as transitive to allow inference of paths between classes. We also included a datatype property *hasValue* to represent the relationship between a *UMLAttribute* individual to its data value.

Data Service ontology. To generate a data service ontology, we use its metadata to extend the semantic query ontology (see Figure 3). All UML classes and UML attributes are defined as subclasses of *UMLClass* and *UMLAttribute*, respectively. The properties *hasConcept*, *hasAssociation* and *hasValue* link classes and attributes as explained above, using existential relationships. Each UML association is defined as subproperty of *hasAssociation*. Inherited associations are represented explicitly (e.g the OWL class *SNP* is associated with the OWL class *GeneRelativeLocation* through the object property *relativeLocation*, where *relativeLocation* is a subproperty of *hasAssociation*). Consequently, we have built a hierarchy of properties deriving from *hasAssociation*, a transitive property. To ensure decidability, OWL-DL imposes that no cardinality restrictions

[8]http://www.w3.org/TR/owl-ref/#OWLDL

can be placed on transitive properties or their inverses or any of their subproperties[8]. Thus, we use existential restrictions and build a different ontology for data integration [12], if cardinality restrictions between individuals needs to be represented.

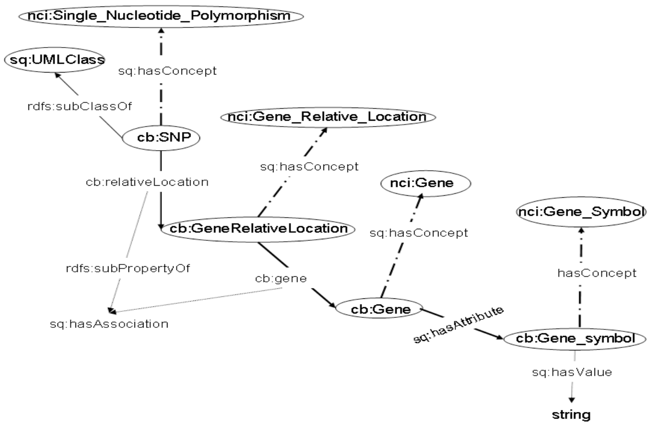

Figure 3: Extract from caBIO 4.0 ontology

Motivating Example Revisited. Considering the motivating example, we build the ontology from caBIO data service metadata (see Figure 3). A DL-query is a query specified as an OWL class description. Using Manchester OWL (MOWL) Syntax [8], the DL-query denoting SNPs associated with Genes whose symbol is "TGFB1" is as shown in Listing 2. This is how the query written by the scientist looks like in our system. We note that this query is similar to the natural language expression presented when introducing the example in Section 1, especially when contrasted with the CQL query from Listing 1.

This DL-query is expressed just using concepts from NCIt and, in contrast to the CQL query, it does not explicitly state the path to follow from a SNP to a Gene. The advantage of building the query using concepts from NCIt is that these are concepts from the biomedical domain the scientist is familiar with. A benefit of MOWL Syntax is that uses infix notation (hasConcept some GeneConcept) rather than prefix notation of DLs (∃ hasConcept.GeneConcept). Thus, it is closer to the natural language query from Section 1. This query is high-level and descriptive: the query is not based on the structure of the target resources and it does not specify the procedure to find the data, i.e. the path between UML classes, but the criteria for the desired data. Even though building a DL-query requires some training, this process could be guided through an appropriate user interface. Additionally, a controlled natural language layer could be built on top of the semantic layer.

In the next section, we will show the different steps required to translate from this type of DL-query to one or more corresponding CQL queries (if more than one path can be followed between two classes).

Listing 2: Concept-based query given by the user

```
hasConcept some Single_Nucleotide_Polymorphism
        and hasAssociation some (hasConcept some GeneConcept and
                    hasAttribute some (hasConcept some Gene_Symbol and hasValue value "TGFB1"))
```

4. Query Rewriting and Translation

In order to translate from DL-queries to D/CQL, we follow several steps involving query rewriting and translation. There are approaches translating DL-queries [14] and SPARQL queries [4] into object-oriented languages. Below, we describe an approach similar to [14], converting DL-queries into CQL, via a formalism for object-queries called the Monoid Comprehension Calculus (MCC) [6]. Using MCC ensures support for optimisations and easy treatment for other target languages (including future versions of CQL). Peim *et al* [14] require that an acyclic set of definitions is defined to be able to rewrite a DL query through an expansion algorithm. In contrast, we support DL-query rewriting through justifications of entailments [10] within the information model ontology.

4.1. DL Query Validation and Rewriting

To be valid, the DL-query must entail an OWL Class C, subclass of *UMLClass*, which is the type of the objects to be retrieved. As the data service ontology only contains terminology expression but no individuals, we first eliminate the expressions using datatype properties but keep track to which DL expression they are associated with, obtaining:

```
Q ≡ hasConcept some Single_Nucleotide_Polymorphism
and hasAssociation some (hasConcept some GeneConcept
and hasAttribute some (hasConcept some Gene_Symbol)
```

such that $SNP \sqsubseteq Q$. Given Q, which uses NCIt terms, we will rewrite it into a DL-query only using terms of a specific data service. Our rewriting method obtains queries that represent a subset of Q and uses justifications of the entailment $SNP \sqsubseteq Q$, in the case of the example. A justification is a minimal set of axioms sufficient to produce an entailment [10]. Kalyanpur et al [10] present algorithms for computing all the justifications of an entailment in OWL-DL ontologies. While justifications were designed to improve understandability of ontologies and debugging, we use them to derive the paths connecting UML classes in the model. Below, we analyse the rewriting process with respect to the motivating example.

Motivating example revisited. Classifying the ontology using an OWL-DL reasoner, such as Pellet [16], SNP is inferred as a subclass of Q. The explanation set [10] for the subsumption $SNP \sqsubseteq Q$ includes the axioms presented in Listing 3[9].

Then, we filter the explanation set so that axioms using only NCIt concepts are not included. In this case, axioms from caBIO are 1, 4 and 9 above. We combine left-hand-side expressions with right-hand-side expressions. Then, we incorporate expressions using datatype properties that were eliminated in the validation step. The rewritten DL-query is given in Listing 4.

Through this query rewriting process, from a domain-concept-based query we obtained a DL-query using concepts from the caBIO ontology. We note that the reasoning process (using explanations) inferred that the SNP class is associated with the Gene class through the GeneRelative-Location class (see Listing 4), while this was not explicitly stated in the original DL-query (see Listing 2).

It is noted that, as there might be several explanations for an entailment, a single original DL query may result in several rewritten DL queries (an eventually several CQL queries). These will be following different paths through the associations on the model and thus, each rewritten query will have distinct meaning. The user will be presented with the alternative rewritten query to use, ie. which association path within the model is of interest.

Listing 3: Explanation set

```
1  SNP ⊑ relativeLocationCollection some GeneRelativeLocation
2  relativeLocationCollection ⊑ hasAssociation
3  hasAssociation is transitive
4  GeneRelativeLocation ⊑ gene some Gene
5  gene ⊑ hasAssociation
6  Gene_symbol ⊑ hasConcept some Gene_Symbol
7  Gene ⊑ hasConcept some GeneConcept
8  SNP ⊑ hasConcept some Single_Nucleotide_Polymorphism
9  Gene ⊑ hasAttribute some Gene_symbol
10 Q ≡ hasAssociation some (hasConcept some GeneConcept)
```

Listing 4: DL-query using only elements from caBIO ontology

```
SNP and relativeLocationCollection some (GeneRelativeLocation and gene some (Gene
     and hasAttribute some Gene_symbol and hasValue value "TGFB1")
```

4.2. DL query to MCC Translation

The monoid comprehension calculus (MCC) is a formal framework expressing collection types to support object queries optimizations [6]. Object queries involve collections (e.g. sets, lists, bags), whose semantics can be captured by monoid comprehensions (MC). In this paper, we provide informal definitions of the entities we use here and refer the reader to [6] for more details.

A monoid of type T is an algebraic structure defined by the pair $(\oplus, Z_{\oplus})$ where $\oplus : T \times T \to T$ is an associative function and $Z_{\oplus}$ is the left and right identity of $\oplus$. A collection monoid is a monoid for a collection type (e.g. lists or bags) and must also specify a unit function building a singleton collection.

A MC takes the form $\oplus\{e \ [\!]\ \overline{q}\}$, where $\oplus$ is a monoid operator called the *accumulator*, e is the *header* and $\overline{q} = q_1, \ldots, q_n, n \geq 0$ is a sequence of *qualifiers*. A qualifier can take the form of a *generator*, $v \leftarrow e'$ with v a range variable and e' an expression constructing a collection, or a *filter* predicate. The symbol $\uplus$ denotes the accumulator for bags. As bags are unordered collections of values that may have duplicates, $\uplus$ is commutative but not idempotent [6]. For example, $\uplus\{x \ [\!]\ x \leftarrow \{1,2\}\}$ is the monoid comprehension representing the bag $\{\{1,2\}\}$.

In the table below, we present the basic transformations from Manchester OWL (MOWL) Syntax to MCC, where D_p denotes the domain of property p and A' denotes the transformation of expression A. This transformation is similar to [14]. After applying the transformation, the MC is normalised following Fegaras' algorithm [6] to obtain an unnested MC expression.

MOWL Syntax	MCC
p some C	$\uplus\{d \ [\!]\ d \leftarrow D_p, c \leftarrow C'\}$
A and B	$\uplus\{a \ [\!]\ a \leftarrow A', b \leftarrow B'\}$
A or B	$(\uplus\{a \ [\!]\ a \leftarrow A'\}) \uplus (\uplus\{b \ [\!]\ b \leftarrow B'\})$

Motivating example revisited. Using the above transformation over the example query and the normalisation algorithm, we obtain the following MCC expression:

[9] $\sqsubseteq$ represents subclass and subproperty relation

Listing 5: MCC expression corresponding to the DL-query

```
⊎ \{ s ⟦ s ← SNP, r ← s.relativeLocationCollection , r ← GeneRelativeLocation ,
              g ← r.gene , g ← Gene, a ≡ g.symbol , a ← Gene_symbol}
```

4.3. MCC to CQL

As mentioned in the introduction, CQL is the common query language used across caGrid data services. The current version of CQL allows users to retrieve objects of a single class, indicating predicates on its attributes, associations and predicates on the associated classes. In this section, we will show the translation from a MCC expression to CQL. Considering the motivating example, the translation converts the MCC of the previous section (Listing 5) to the CQL query in the introduction (Listing 1).

The main components of a CQL query are[10]:

Target : indicating the type of objects to be retrieved

Attribute : indicating a restriction for an attribute of a particular object (both the Target or an object referred through an association)

Association : indicating a related object

Additionally, CQL can contain Query Modifiers, indicating to return a 'count only' result, the whole objects, a single Distinct Attribute or a selection of Attribute Names for the target object. These modifiers cannot be expressed in the DL query. The translation from DL-query to CQL, via MCC, will consider retrieving full objects as default. Before submitting the resulting CQL query, the user will be asked to set the query modifiers.

CQL has some known caveats[11]: it retrieves just objects of the Target type (not even subclasses of the Target type), only attributes with simple XML schema types are allowed to restrict the query or as return values, the association objects can be used to impose restrictions on the result type but cannot be returned.

Translating the MCC expression into CQL amounts to: include as Target in CQL the expression for the variable in the head; include an Association per each pair of generators, one determining the name (the class to which they belong) and the other identifying the role name; include an Attribute restriction for each filter.

5. Related Work

UML-to-OWL transformations Existing transformations were motivated by different applications and were specified in varying levels of detail.

Berardi *et al* [2] developed a description logics representation of UML class diagrams to detect inconsistency and redundancy by reasoning over them. Their transformation is not exhaustive, since their focus is on performance analysis of reasoning over UML diagrams.

The Object Management Group developed the Ontology Definition Metamodel (OMG ODM)[12], comparing and contrasting UML and OWL and presenting an exhaustive transformation.

Djurić *et al* [7] extended UML by defining an UML profile for OWL, based on OMG ODM, which extends UML models with tagged values and stereotypes. These extended UML models can be automatically transformed to an OWL representation.

The UMLBackend[13] plug-in for Protégé is an implementation translating UML to OWL and vice versa. It considers a subset of UML constructs.

Evermann [5] presents a conversion between UML and OWL, and vice versa, with the objective of describing Bunge's upper-level ontology, specified in natural language, and allows its wider use.

Shironoshita *et al* [15] presented semCDI, which uses a UML-to-OWL transformation for caGrid information models. We analysed this transformation in detail in Section 3.

Semantic queries over grid systems Other systems have exploited semantic web technologies to support semantic queries for distributed databases in grid environments. DartGrid [3] builds RDF views of legacy databases to support RDF queries over the instance data. ACGT [17] uses grid middleware and supports SPARQL queries over resources. semCDI [15] extends caGrid and supports SPARQL querying by translating SPARQL to CQL. The details of this transformation are not described and using MCC as intermediate languages provides generality and support for optimisations, even for SPARQL [4]. Additionally, none of these systems have focused on providing a high-level query language support over metadata.

6. Conclusions and Future Work

This paper presented a semantic layer approach, designed within the ONIX system, to support high-level concept-based queries over semantically annotated data sources. This functionality is important for ONIX success within the biomedical community.

We applied the approach to the caGrid infrastructure, where running queries against data services requires to

[10]http://cagrid.org/display/dataservices12/CQL
[11]http://cagrid.org/display/dataservices12/CQL
[12]http://www.omg.org/cgi-bin/apps/doc?ptc/07-09-09.pdf
[13]http://protege.cim3.net/cgi-bin/wiki.pl?UMLBackend

know their syntax and structure. We showed that DL-queries, expressed using domain concepts can be automatically translated into CQL queries. DL-queries are high-level and descriptive: they specify the criteria for the desired data based on relationships at the concept level. The translation process determines the relevant paths between UML classes that specify how to find the data in a CQL query. We described the prototype implementation for this functionality.

The main contributions are: a) the automatic generation of OWL ontologies from annotated-UML models, which preserves UML and the domain ontology (NCIt) semantics; b) DL-queries validations and rewriting from the concept space into the model space through justifications; c) subsequent DL query translation to MCC and then to CQL.

We observed that the approach is general, in the sense that has the potential to support non-caGrid data resources, as long as they provide appropriate metadata, i.e. annotated UML models. The module extraction method can be applied to other domain ontologies, apart from NCIt. While we presented a translation from DL-queries to CQL, one of the advantages of using the MCC is that only the last step is dependent on the caGrid query language. Other query languages can be supported by specifying the translation rules from MCC.

In future work, we aim to extend the translation of DL-queries over multiple resources to DCQL, as well as, to consider other semantic queries languages, such as SPARQL and SPARQL-DL. While the prototype implementation and experiments of translating queries over the generated ontologies have served as a proof-of-concept for the approach, we intend to perform a detailed performance evaluation of the architecture.

Acknowledgments The authors are grateful to Cancer Research UK and the UK National Cancer Research Institute Informatics Initiative for support for their research.

References

[1] F. Baader et al., editors. *The Description Logic Handbook: Theory, Implementation, and Applications* . Cambridge University Press, 2003.

[2] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.

[3] H. Chen, Z. Wu, Y. Mao, and G. Zheng. DartGrid: a semantic infrastructure for building database Grid applications. *Concurrency and Computation: Practice and Experience*, 18:1811–1828, 2006.

[4] W. Corno, F. Corcoglioniti, I. Celino, and E. D. Valle. Exposing Heterogeneous Data Sources as SPARQL Endpoints through an Object-Oriented Abstraction. In *ASWC*, pages 434–448, 2008.

[5] J. Evermann. A UML and OWL description of Bunges upper-level ontology model. *Software and Systems Modeling*, page 16191366, 2008.

[6] L. Fegaras and D. Maier. Optimizing object queries using an effective calculus. *ACM Trans. Database Syst.*, 25(4):457–516, 2000.

[7] D. Gašević, D. Djurić, and V. Deved. MDA-based Automatic Ontology Development. *STTT*, 9(2):103–117, 2007.

[8] M. Horridge, N. Drummond, J. Goodwin, et al. The Manchester OWL Syntax. In *OWLED*, 2006.

[9] E. Jiménez-Ruiz, B. C. Grau, U. Sattler, et al. Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support. In *ESWC*, pages 185–199, 2008.

[10] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding All Justifications of OWL DL Entailments. In *ISWC/ASWC*, pages 267–280, 2007.

[11] G. A. Komatsoulis, D. B. Warzel, F. W. Hartel, et al. caCORE version 3: Implementation of a model driven, service-oriented architecture for semantic interoperability. *Journal of Biomedical Informatics*, 41(1):106–123, 2008.

[12] J. P. McCusker, J. A. Phillips, A. González-Beltrán, A. Finkelstein, and M. Krauthammer. Semantic data warehouse for caGrid. *BMC Bioinformatics*, In press., 2009.

[13] S. Oster, S. Langella, S. Hastings, et al. caGrid 1.0: A Grid Enterprise Architecture for Cancer Research. In *AMIA Annual Symposium*, 2007.

[14] M. Peim, E. Franconi, N. W. Paton, and C. A. Goble. Query processing with description logic ontologies over object-wrapped databases. In *SSDBM*, pages 27–36, 2002.

[15] E. P. Shironoshita, , Y. R. Jean-Mary, R. Bradley, and M. R. Kabuka. semCDI: Semantic Query Formulation for caBIG. *JAMIA*, 15(4):559–568, 2008.

[16] E. Sirin. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.

[17] M. Tsiknakis, M. Brochhausen, J. Nabrzyski, et al. A Semantic Grid Infrastructure Enabling Integrated Access and Analysis of Multilevel Biomedical Data. *IEEE Transactions on Information Technology in Biomedicine*, 12(2):2, 2008.

Appendix: Glossary of terms

Acronym	Meaning
ACGT	Advancing Clinico-Genomic Trials on Cancer
caBIG®	cancer Biomedical Informatics Grid
caBIO	cancer Bioinformatics Infrastructure Objects
caDSR	cancer Data Standars Repository
CDE	Common Data Element
CQL	caGrid/Common Query Language
DCQL	Distributed caGrid/Common Query Language
DL	Description Logic
EVS	Enterprise Vocabulary Service
FQP	Federated Query Processor
GME	Global Model Exchange
MCC	Monoid Comprehension Calculus
NCI	National Cancer Institute
NCIt	NCI thesaurus
NCRI	National Cancer Research Institute
ONIX	ONcology Information eXchange
OWL	Web Ontology Language
RDF	Resource Description Framework
semCDI	SEMamntic Cabig Data Integration
SNP	Single Nucleotide Polymorphism
SPARQL	SPARQL Protocol and RDF Query Language
TGFB1	Transforming Growth Factor Beta 1
UML	Unified Modeling Language
XML	eXtensible Markup Language