Combining XML and DL for describing and querying documents

Rim Alhulou and Amedeo Napoli
LORIA, Campus Scientifique,
B.P. 239, 54506 Vandoeuvre-les-Nancy,
FRANCE
Email: {alhulou,napoli}@loria.fr

Abstract

This paper is talking about an application of DL related to the Escrire project whose objective is to compare three knowledge representation (KR) formalisms (conceptual graphs, description logics and object-based representation systems), for the content-based representation and manipulation of textual documents. A domain ontology is used in order to annotate documents. The ontology, annotations and queries are described using a pivot format based on XML, and are translated into each of the three formalisms, where inference mechanisms are used for query answering. In this paper, we give a short presentation of the Escrire project, and we give a first synthesis of the translation process from the pivot format into a description logics.

1 Motivation

In order to take advantage of the huge mass of information available on the Web and into document databases, it is necessary to design efficient techniques of retrieval, extraction and querying. Ontologies play an increasingly important role in Information Technology. Originally they have been applied in the Knowledge Engineering field to model domain knowledge for enabling knowledge sharing and reuse. Then, they have been also used to exploit this knowledge in the field of Information Retrieval [2], ontologies have been used for the content-based annotation of documents. The Escrire project follows this line: the objective of this project is to use an ontology to annotate a set of abstracts of biological documents extracted from the NIH Medline public database, and then to query the annotated documents within a KR formalism. In this project, three KR formalisms have been used, namely conceptual graphs (CG), description logics (DL), and object-based knowledge representation language (ObKR). In the following, we restrict our attention to the use of DL within the Escrire Project.

In order to be manipulated by their content, documents are annotated and a domain ontology has been designed. The annotations and the ontology are described within a pivot format based on XML. More precisely, this pivot format relies on a set of syntactic rules describing the ontology and the annotations attached to documents. The syntactic rules are the core of a DTD. Moreover, the pivot format plays the role of a bridge between documents and the DL formalism: every element in the ontology and every annotation have a corresponding element in the DL formalism.

The pivot format is also used to describe SQL-like queries that are also represented within the DL formalism to be handled (by the classifier). One can notice that the pivot format has been especially built for the needs of the application, and is by no means another XML-based description language. The DL system that has been used for the Escrire project is Racer [3].

In this paper, we present the description of the ontology and annotations in section 2. Then, in section 3 we detail the translation to the DL formalism. In section 4, we present the difficulties encountered for this translation. Finally we conclude the paper in section 5.

2 Pivot format

The pivot format is used for describing the ontology, annotating documents and building queries.

2.1 Description of ontology

An ontology is a description of the concepts and relationships (between concepts) existing for a particular domain. In our case, we work in the biological domain, and we are particularly interested by genes (concepts) and interactions between genes (relations between concepts). In the Escrire project, the ontology is defined by a DTD. In this DTD, classes (and classes of relations) are declared and organised in a taxonomy. Each class in the ontology is described by a set of attributes and roles representing the properties of this class. The type of the roles is restricted to a particular class in the ontology, while that of the attributes is restricted to a particular concrete type (string, integer, etc). In the ontology, we have two types of classes; defined classes (with necessary and sufficient conditions) and primitive classes (with only necessary conditions). The pivot format is similar in these aspects to the language recently proposed in [8], with the exception that the pivot format is supposed to describe n-ary relations and SQL-like queries.

For example, the following class represents a part-whole relation. It has two roles: `whole` that introduces the container and `part` that introduces the parts involved in the relation. The range of these two roles is of type `body-part` (another class in the ontology). This relation has no attributes, and it is transitive , reflexive and antisymmetric.

```
  <esc:descbinrel name="subpart" transitive="yes"
     reflexive="yes" antisymmetric="yes">
     <esc:defrole name="whole">
        <esc:classref name="body-part"/>
     </esc:defrole>
     <esc:defrole name="part">
        <esc:classref name="body-part"/>
     </esc:defrole>
</esc:descbinrel>
```

2.2 Annotations

A document is composed of three parts: (1) a textual abstract; (2) a set of classic meta-data (Dublin core); and (3) a set of meta-data concerning the content (annotations). The pivot format is used for representing the third part (annotations) according to the ontology, i.e. the objects and relations (more particularly genes and interactions) referenced in these documents. Objects (instances of classes) and relations (instances of classes of relations) are described by their properties (names and values of roles and attributes), and the class to which they belong. For instance, the following annotation describes an interaction between two genes, `gt` (of type `gap`) which plays the role of `promoter` and `antp` (of type `homeotic`) which plays the role of `target`. The `effect` of this interaction is that `gt` inhibits `antp`.

```
  <esc:relation type="interaction">
     <esc:role name="promoter">
        <esc:objref type="gap" id="gt"/>
     </esc:role>
     <esc:role name="target">
        <esc:objref type="homeotic" id="antp"/>
     </esc:role>
     <esc:attribute name="effect">inhibition</esc:attribute>
</esc:relation>
```

We can notice that a relation may have roles, mainly by a source element and a target element, and attributes describing the properties of the relation.

2.3 Querying

The structure of a query respects the following schema: (1) `SELECT`: the selected features are a list of paths that can be empty; (2) `FROM`: the variables introduced in the query are typed by a class; (3) `WHERE`: the conditions may be combined by logical constructors (conjunction, disjunction, negation) as well as universal and existential quantifiers; (4) `ORDERBY`: the order of the elements in the answer of the query is given by a list of paths. For example, the following query is designed for searching for documents mentioning `interactions` with the gene `gt` as `target` or `interactions` with the gene `antp` as `promoter`.

```
SELECT    I.effect, I.location
FROM      I:interaction
WHERE     I.target=OBJREF(gene,"gt")
     OR   I.promoter=OBJREF(gene,"antp")
ORDERBY   I.location
```

3 Translation from the pivot format into DL

In this section, we present the translation of the ontology and annotations described in the pivot format into the DL system. The Racer system has been used in the Escrire project for a number of reasons: it is based on a very expressive DL language, it has an efficient classifier, the capability of managing individuals (Abox), and it is equipped with a Java interface that is very convenient to our application needs.

Classes and classes of relations

The classes and relations of the ontology are translated into concepts in the Tbox of the DL system. Relations are represented as concepts. All attributes and roles are translated into roles in the DL system. The two types of classes in the ontology (defined and primitive) are transformed into DL concepts according to their status: defined classes become defined concepts in the DL system, while primitive classes become primitive concepts. For example, the relation `subpart` described above is translated into the defined concept `subpart` subsumed directly by `top`, with two roles `whole` and `part`. The properties of this relation (reflexivity, antisymmetry and transitivity) are managed by an external module beside the DL system. The concept `subpart` is described as:

```
subpart:  (and TOP (some whole body-part)(some part body-part))
```

Objects and relations

Each document is represented as an individual in the Racer Abox (an instance of a primitive concept called `Document`). Objects and relations referenced in the ontology or in the documents are translated into individuals of the Racer Abox. An individual is related to all the individuals filling its roles and attributes. Individuals representing objects and relations are linked to those representing the documents where they are referenced. For instance, the relation defined in 2.2 can be represented as an instance of the concept `interaction`, related to an instance of `gap` (identified by `gt`) through the `promoter` role and it to an instance of `homeotic` (identified by `antp`) through the `target` role. The `effect` of this interaction is that `gt` inhibits `antp`: this fact is represented by relating the instance of `interaction` to `inhibition` (instance of `effectrange`) through the role `effect`.

Query evaluation

Classification is one of the DL reasoning mechanisms, and can be used as a query processing technique [5, 1]. A query `Q` can be translated into one or more concepts C_i ($i \geq 1$) in the DL system. A concept C_i will be classified in the hierarchy of concepts, and the answer to the query `Q` will be constituted by the instances of C_i. C_i has its own instances and those of its sub-concepts.

A query is manipulated as follows in the DL system. Firstly, the query `Q` (described in the pivot format) is normalised: the negations are pushed inside the terms using Morgan rules. For example, the following formula in the `WHERE` clause: `(NOT (AND (AND a b) (AND c d)))` is normalised as `(OR (NOT a) (NOT b) (NOT c) (NOT d))`.

For example, the query presented in 2.3 is translated into a defined concept `I` in the DL system, described as:

```
I: (and interaction (or (some target Gtct) (some promoter Antpct)))
```

Two primitive concepts (`Antpct` and `Gtct`) are created because the individuals (`antp` and `gt`) cannot be directly used in concepts descriptions (we don't have the `one-of` constructor in Racer).

4 Difficulties in the translation

Here, we present the difficulties encountered in the translation from the pivot format into the DL system.

4.1 Evaluation of properties of relations

In the Escrire project, relations may have any number of roles and attributes, and may be binary or n-ary. Racer does not provide any special constructor for taking into account such relations. Thus, they are represented as concepts, and their properties i.e. reflexivity, symmetry and transitivity, are managed by an external module besides the DL system. Facts (new assertions) resulting from the manipulation of relations are added to the Abox of the DL.

4.2 Negation of concepts

In our Framework, the set of available information is assumed to be complete, i.e. inferences should be drawn under the closed-world assumption. Actually, this is not the case in the DL system, where there is no way to turn off the open-world reasoning mode. Thus, the negation operator applied to a concept `C` does not give the complementary set of the set of instances of `C`, but instead the set of instances explicitly declared as instances of the negated concept (not `C`).

4.3 Quantifier evaluation

In a query, variables in the `FROM` clause are supposed to be universally quantified. Moreover, variables in the `WHERE` clause may also be quantified. The existential and universal quantifiers (especially the nested ones) in the `WHERE` clause cannot be easily translated into the DL system. In some cases, the existential quantifier could be equivalent to concept satisfiability.

```
SELECT I1
FROM I1:interaction
WHERE ∀ I2:interaction
      and I2.target = OBJREF(gt)
      and I1.target = OBJREF(antp)
      and I2.effect = I1.effect
```

In this query, there is one variable `I1` universally quantified in the `FROM` clause, and it ranges on the class `interaction`, while `I2` is the variable declared in the `WHERE` clause and it is universally quantified. This query can be translated into two defined concepts (`I1` and `I2`):

```
(define-concept I1 (and interaction (some target Antpct)))
(define-concept I2 (and interaction (some target Gtct)))
```

The concepts `I1` and `I2` are added in the Tbox of the DL system. Both have the concept `interaction` mentioned in the `FROM` clause as ascendant.

However, there is no way to represent the universal quantification in the description of `I1`. A first idea could be to use the `(all effect I2)` and the `(exactly n effect I2)` constructions to describe this fact, but this is not enough. Actually, we would need a construction like `(Same-as I1.effect I2.effect)`, and the associated inference mechanism.

4.4 Cycles

Terminological cycles [6] are taken into account in Racer. Thus, Racer classifies and calculates the instances of concepts referring to each other. In our case, if the concepts (extracted from queries) are labelled as defined concepts with cycles, Racer cannot calculate their instances because none of these instances are known in advance. If we take the query in 4.3, and we try to describe the defined concepts using terminological cycles:

```
I1: (and interaction (some target Antpct)(some effect (some ifct I2))))
I2: (and interaction (some target Gtct)(some effect (some ifct I1))))
```

The role `ifct` is the inverse role of `effect`. There is no problem for classifying these two concepts in the Tbox, but the problem is to calculate their instances because the Abox does not contain any assertion about instances of `I1` or `I2`. It would be easier to solve this kind of problem if "variables" were available in DL.

5 Conclusion

The first results of the Escrire project show that the DL Racer can be used for representing an ontology of a domain, for describing and querying a set of documents in an XML-like form. We have presented some difficulties encountered in the use of DL for manipulating documents. The principal difficulties are related to the translation of queries involving the universal quantifier and terminological cycles.

Some DL systems support Abox reasoning with two kind of assertions: concept instantiation and role instantiation. This is not sufficient for taking into account queries involving relations. Another technique based on rolling-up roles was proposed for answering conjunctive queries and has been extended to disjunctions of conjunctive queries [4, 7]. In this technique, a query is reduced to a concept where roles are represented as concepts also. We have to study further this approach in order to know if it can be used to solve the cycle and the universal quantification problems mentioned above.

Finally, one question could be do we have to restrict the query language in the pivot format, or to extend the DL formalism.

6 Acknowledgements

We would like to thank Volker Haarslev and Ralf Möller for their collaboration and the discussions that they furnished.

References

[1] H. W. Beck, S. K. Gala, and S. B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *Proceedings Fifth International Conference on Data Engineering*, pages 572–581, Los Angeles, California, 1989. IEEE Computer Society Press.

[2] N. Guarino, editor. *Formal Ontology in Information Systems.* IOS Press, Amsterdam, 1998.

[3] V. Haarslev and R. Möller. Description of the racer system and its applications. In *Proceedubgs International Workshop on Description Logics (DL-2001)*, Stanford, USA, August 2001.

[4] I. Horrocks and S. Tessaris. Answering conjunctive queries over DL aboxes: A preliminary report. In *Description Logics*, pages 173–182, 2000.

[5] S. Navathe, A. Savasere, T. Anwar, H. Beck, and S.Gala. Object modeling using classification in candide and its applications. In A. Dogac, M. T. Ozsu, A. Biliris, and T. Sellis, editors, *Advances in Object-Oriented Database Systems*, pages 435–476. Springer, Berlin,, 1993.

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

[7] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic.* PhD thesis, University of Manchester, 2001.

[8] F. van Harmelen and I. Horrocks (eds). Reference description of the daml+oil ontology markup language, 2001. http://pride.daml.org/2000/12/reference.html.Jan 2001.