Three knowledge representation formalisms for content-based manipulation of documents

Rim Al Hulou, Olivier Corby, Rose Dieng-Kuntz, Jérôme Euzenat, Carolina Ramirez, Amedeo Napoli, Raphaël Troncy

Three knowledge representation formalisms for content-based manipulation of documents

Rim Al-Hulou[1], **Olivier Corby**[2], **Rose Dieng-Kuntz**[2], **Jérôme Euzenat**[3], **Carolina Medina Ramirez**[2], **Amedeo Napoli**[1], and **Raphaël Troncy**[4]
[1] LORIA, BP 239, F-54506 Vandœuvre-les-Nancy,
[2] INRIA, B.P. 93, 2004, route des Lucioles, F-06902 Sophia-Antipolis,
[3] INRIA Rhône-Alpes, 655, avenue de l'Europe, F-38330 Montbonnot Saint-Martin,
[4] INA, 4 avenue de l'Europe, F-94366 Bry sur Marne.

Abstract

Documents accessible from the web or from any document base constitute a significant source of knowledge as soon as the document contents can be represented in an appropriate form. This paper presents the ESCRIRE project, whose objective is to compare three knowledge representation (KR) formalisms, namely conceptual graphs, description logics and objects, for representing and manipulating document contents. The comparison relies on the definition of a pivot language based on XML, allowing the design of a domain ontology, document annotations and queries. Each element has a corresponding translation in each KR formalism, that is used for inferencing and answering queries. In this paper, the principles on which relies the ESCRIRE project and the first results from this original experiment are described. An analysis of problems encountered, advantages and drawbacks of each formalism are studied with the emphasis put on the ontology-based annotations of document contents and on the query answering capabilities.

1 Introduction

The goal of this paper is first to propose an original methodology for content-based annotation and querying of documents within a knowledge representation formalism (KR), and then to propose a comparison of three KR formalisms regarding the tasks of annotation and querying. Content-based annotation of (textual) documents is an important task useful in a number of applications involving information retrieval, query answering, content-based inferencing, information extraction, and text mining.

Knowledge representation formalisms are good candidates for representing the content of a document. Three KR formalisms, namely conceptual graphs (CG), description logics (DL), and object-based knowledge representation languages (ObKR), are studied in the following. The present experiment has been carried out within the ESCRIRE project, on a base of 4500 abstracts of biological articles extracted from the NIH Medline public database. For this first experiment, a hundred of abstracts have been considered and manually annotated. The contents extracted from these abstracts are restricted to a list of genes and the genetic interactions between these genes.

In order to manipulate efficiently a set of documents for various purposes, e.g. information retrieval or text mining, it is worthwhile to build an explicit representation of the document contents and to index the documents accordingly. Contrasting with the more classical indexing techniques, i.e. keyword-based, full-text, weighted, or thesaurus-based, an alternative is to index documents by means of terms referencing concepts explicitly present in documents and represented in an ontology. Such an approach enables an ontology-guided information retrieval about/from such documents [5]. Furthermore, the content of documents can be represented within a knowledge representation formalism and then attached to the document (inside or outside the document itself). The representation of the document content can be used for manipulating the document and for reasoning purposes.

Several KR formalisms are available for documentation annotation, and there is no guide for choosing one formalism instead of another. This is one of the goals of the ESCRIRE project to carry on an experimentation for comparing the three knowledge representation formalisms previously mentioned for the content document representation and querying tasks.

The comparison protocol involves the design of a

"bridge" between documents and the KR formalisms through a *pivot language* based on the XML language [12], and called the ESCRIRE language. First, the ontology and the annotations are encoded within the pivot language, then the ontology and the annotations are encoded in each KR formalism, and finally the queries have to be satisfied in each formalism (see figure 1).

This work is original from several points of view. First, it presents a methodology for representing the content of documents in a pivot language, and then in a KR formalism. The use of a pivot language for document content annotation and querying can be likened to database integration techniques based on an intermediate mediator and a KR formalism, where several databases can be accessed through a common interface [1]. Moreover, the ESCRIRE experiment involves a comparison of three KR formalisms, on a real-world basis. More than a competition, this experiment has to be understood as a parallelization of the three formalisms, in order to enlighten the kinds of services that a KR formalism has to provide for an intelligent content-based manipulation of documents. The methodology presented here can be generalized to other present tasks of interest such as information extraction, text analysis, text mining and semantic Web [9] [17].

This paper is divided in three main parts. The first part introduces the global framework of the experiment and the ESCRIRE language. The second part presents the translation from the pivot language within each KR formalism. The last part is a global discussion on the experiment, comparing the behavior of each formalism regarding the annotation and querying tasks. Comparison with related work and research perspectives conclude the paper.

2 The ESCRIRE pivot language

2.1 First definitions

In this paper, a *document* denotes an abstract of a Medline entry making reference to an article in biology, without title, authors and journal name. A *content* refers to the "meaning" of a document that has to be represented. An *annotation* describes potential interesting biological elements present in a document, i.e. *genes* and *interactions* between genes.

Two kinds of entities are manipulated: generic entities involved in the content of a document and corresponding to classes of individuals, such as *gene*, and individual entities, such as *Antennapedia*, corresponding to individual objects. Moreover, the background knowledge available in the ESCRIRE project mainly relies on an *ontology*, i.e. a set of classes organized into a taxonomy and representing the domain concepts, and a set of *assertions* or *facts* involving classes and instances of the classes. For example, the ontology includes the definition of the domain concepts *gene* and *homeotic gene*, while the fact *Antennapedia is a homeotic gene* must be known for a correct understanding of the documents (see [7] for a discussion of these issues).

Within the ESCRIRE language, a document is represented by an XML structure composed by the document itself and a list of annotations, themselves represented as XML structures. The ontology and the assertions are represented both in the ESCRIRE language and within each KR formalism.

2.2 The ontology

The ESCRIRE ontology is based on a DTD where classes, relation classes, and a subsumption relation for organizing these classes in a hierarchy are defined[1]. A class is described by a set of *attributes –roles* for a relation class– whose range can be a class or a concrete type, such as integer, float, string and boolean. A class can be declared either as a *defined class* using the tag `esc:defclass`, i.e. the attributes of the class are considered as necessary and sufficient conditions for an individual to be an instance of the class, or as a *primitive class* using the tag `esc:descclass`, i.e. the attributes of the class are only considered as necessary conditions. The distinction primitive/defined class is the same as that of DL [6].

Below, an example of the primitive class `gene` declaration is given. The `esc:descclass` tag introduces the name of the primitive class `gene`, while the `esc:defattribute` tag introduces the attribute `name`, whose range, namely `string`, is given by the tag `esc:typeref`.

```
<esc:descclass name="gene">
  <esc:defattribute name="name">
    <esc:typeref name="string"/>
  </esc:defattribute>
</esc:descclass>
```

"In Escrire, there is a distinction between classes and individual objects. However, individual objects can have to be included in the ontologies. For instance, for restricting the range of an attribute. Thus..

In the ESCRIRE project, there is a separation between the classes in the ontology and the individual objects. However, there are cases where individual objects have

[1] See `http://escrire.inrialpes.fr/dtd/escrire.dtd`.

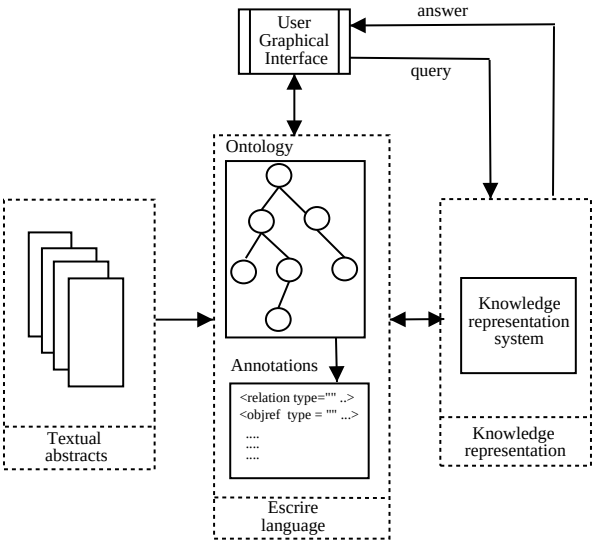

Figure 1: The ESCRIRE comparison protocol

to be included in the ontology, e.g. for restricting the range of an attribute in a class with a list of individuals. Thus, the ontology supports the description of individuals, while the annotations cannot contain explicit references to the classes of the ontology.

Moreover, some generic entities such as a family of genes can be present in a document. It can be useful to represent such a generic entity as a *prototype*, i.e. an individual object representing a class of individuals. These individual objects are *universally* known and can appear in any annotation or any query.

2.3 The annotations

An annotation describes genes and relations between genes present in a document. An annotation is represented by an XML structure, with two main types of tags: *roles* and *attributes*. The roles are used for denoting the domain and the range of the relation involved in the annotation, while the attributes are used to represent the characteristics of the relation. For example, the annotation given below describes an *interaction* between a `promoter`, namely a `gene` called `mam`, and a `target`, namely another `gene` called `N`. The interaction has also an attribute, i.e. `effect`, whose value is `inhibition`.

```
<esc:relation type="interaction">
  <esc:role name="promoter">
    <esc:objref type="gene" id="mam"/>
  </esc:role>
  <esc:role name="target">
    <esc:objref type="gene" id="N"/>
  </esc:role>
  <esc:attribute name="effect">
    <esc:value>inhibition</esc:value>
  </esc:attribute>
</esc:relation>
```

This description can be interpreted as follows: the effect of the interaction described above is that the gene `mam` inhibits the gene `N`.

As another example, the following annotation describes an interaction between genes of type `gap` and the gene `Abd-B`, where the `gap` genes are represented by the prototype `gap`, that is an instance of the class `gene-class`.

```
<esc:relation type="interaction">
  <esc:role name="promoter-class">
    <esc:objref type="gene-class" id="gap"/>
  </esc:role>
  <esc:role name="target">
    <esc:objref type="gene" id="Abd-B"/>
  </esc:role>
</esc:relation>
```

The individuals introduced in an annotation are mainly instances of genes present in the ontology. However, this is possible to have local individuals in an annotation, i.e. attached to a specific document. In this case, such individuals cannot be referenced by other annotations nor by concepts in the ontology.

2.4 The queries

A query is similar to an OQL query [2], based on the triple `SELECT-FROM-WHERE`. The basic logical connectors conjunction, disjunction, negation, as well as universal and existential quantifiers may be used. In the following example, the goal of the query is to search for interactions (between genes) whose target or promoter is a gene called `gt`, and to return the effect and location of such interactions.

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

A query generally makes a reference to an interaction between genes or to a particular gene itself, and is used to retrieve the documents mentioning that interaction or that particular gene. One strength of a KR formalism is its inferencing capabilities, based on a semantics. This is in contrast with the XML structures that are only syntactic structures, without any associated semantics.

In our framework, inferences are local to a document: a query `Q` is interpreted in the context of a particular document `D`. The document `D` is an answer to the query `Q` if `D` satisfies `Q` according to the ontology and to the annotations associated with the `D`. The answer to a query is composed of the `urls` of the documents satisfying the query.

3 From ESCRIRE to the KR formalisms

In this section, the translation process between the ESCRIRE language and each KR formalism is detailed. This process has been performed on the same basis for the three formalisms. Local differences are discussed whenever they are of interest.

3.1 Descriptions logics

Presentation. Description logics constitute a family of KR formalisms based on the notions of concepts (set of individuals), roles (relations in which are involved the concepts) and individuals. Concepts and roles are organized within hierarchies by a subsumption relation. Reasoning is based on concept satisfiability, subsumption and classification. In our framework, we used the RACER system [11].

Translation. Classes and relations in the ESCRIRE ontology are translated respectively into primitive or defined concepts. The set of facts is included in an *ABOX* composed of individual objects and relations present in the ontology or in the documents. A document is represented by an individual, actually an instance of the primitive concept `Document`. Individuals representing objects and relations are linked to individuals representing the documents where they are referenced. For example, the individual introduced in 2.3 represents an instance of the concept `interaction`: it is related to an instance of `gene` identified by `mam` through the role `promoter` and to an instance of `gene` identified by `N` through the role `target`; it has an attribute `effect` whose filler is `inhibition`. Moreover, this individual is related to all individuals representing documents where it is referenced.

RACER is equipped with an API allowing the design, modification and querying of the DL knowledge base. A processor performing the transformation of the ESCRIRE expressions into descriptions and individuals in RACER has been built. The ESCRIRE ontology and the annotations are parsed by this processor, and then concepts and individuals are added respectively in the *TBOX* and *ABOX* of RACER.

Query evaluation. Classification is used as a query processing technique [19]. A query `Q` is translated into one or more concepts C_i in the DL system. Each concept C_i is then classified in the concept hierarchy, and the answer of the query is constituted by the instances satisfying the concept C_i. More precisely, the description of C_i is extracted from the `FROM` and `WHERE` clauses in the query. Each variable v_k in the `FROM` clause has a range given by a class defined in the ontology with a corresponding concept in the hierarchy. Thus, the type of v_k is considered as a potential subsumer of C_i, whose complete description will be constructed according to the conditions given in the `WHERE` clause.

3.2 Conceptual Graphs

Presentation. The conceptual graph (CG) formalism is based on a support made of a concept type hierarchy, a relation type hierarchy, a projection relation for comparing graphs, a set of individual markers for naming the concept instances, a conformity relation between markers and types, and a base of conceptual graphs built on this support [3]. In our framework, we have used CORESE, a search engine based on the CG formalism, and merged in an environment based on RDF and RDFS[2] [4]. The languages RDF(S) –standing for RDF and RDFS– allow the description of any resource on the Web within a hierarchy of classes with

[2] See `http://www.w3.org/`.

their properties.

Therefore, the mappings ESCRIRE→RDF(S) and RDF(S)→CG have been used for the translation from documents to the CG environment. In particular, the support of CG, composed of a type concept hierarchy and of a relation type hierarchy, corresponds to RDF schemas and the CG-fact graphs to RDF statements.

Translation. The ESCRIRE and RDF(S) languages have both an XML syntax. Thus, the mapping ESCRIRE→RDF(S) has been implemented through an XSLT transformation. The ESCRIRE annotations are translated into RDF statements, and the ESCRIRE ontology into RDFS. In particular, relations have been reified, i.e. translated into RDFS classes corresponding to concept types in CG, because there are no n-ary relations in RDF and CG relations do not support attributes and roles as this is the case for ESCRIRE relations. Moreover, roles and attributes of ESCRIRE relations are translated into RDF(S) property definitions, corresponding to CG relation types.

Query evaluation. A query language for CORESE has been developed to support the ESCRIRE queries, that are translated into graphs to be exploited afterwards by projection. The ESCRIRE queries are translated into one or more query graphs (QG). This translation depends on the existence of the logical operator `OR` in the ESCRIRE query. The QG is then mapped on a joint graph composed of individuals and annotations graphs.

3.3 Object-based representation system

Presentation. An object-based representation system relies both on knowledge representation and object-oriented programming characteristics. For the ESCRIRE project, we used the TROEPS system, a frame-based language with additional object-oriented programming capabilities [20]. In TROEPS, an individual object is a set of attribute-value pairs associated to an identifier. The value of an attribute can be known or unknown; it can be an object or a value with a primitive type, or a list of such values. The individual objects are grouped into classes introducing constraints on the attribute values. The classes are organized into tree-structured hierarchies: actually, a hierarchy can be considered as a viewpoint on a concept, and more than one hierarchy can exist for representing that concept. The TROEPS system provides various facilities for manipulating knowledge including filtering queries, similarity queries involving a distance measure, value inference, classification and identification inferences.

Translation. The TROEPS system offers an XML interface for building, deleting, renaming or classifying individual objects from an XML stream. However, a specialized parser has been built for translating ESCRIRE expressions into TROEPS expressions: ESCRIRE classes are naturally mapped into TROEPS classes, and classes without super-classes initiate a new concept. The TROEPS system does not allow the declaration of defined classes: thus, all classes can only be implemented as descriptions (the same holds true for relations). As in DL and CG, relations are mapped into classes whose attributes correspond to the attributes and the roles of the ESCRIRE relation. This provides a natural hierarchy for the relations.

Query evaluation. The queries are processed in the TROEPS system by an evaluation process especially designed for the ESCRIRE project, working as follows:

- The parsing transforms the XML code into a LISP structure analog to the initial query, where the references to classes and objects are replaced with their values.
- The normalization transforms the variables into filters and produces a normalized form of the query (pushing negation inward).
- The preprocessing compiles the path expressions and then evaluates these paths whenever possible.
- The evaluation computes the intersections and joints according to the semantics of the connectors.

Since the document references are attached to objects and the answer to a query must be found within one document, the evaluator does not only compute the answer to the query but also computes the set of documents to be given as answer.

4 Discussion and results

Definition of classes in the ontology. The defined/primitive distinction exists in CG and DL, but not in the ObKR system, where only primitive classes exist. In order to take into account this distinction in TROEPS, it would be necessary to implement a module –not presently available– similar to the classifier of DL. The CG translation is based on RDFS, that does not make a difference between defined and primitive classes. Thus, this distinction could not be taken into account in the ESCRIRE-RDF(S)-CG mapping: an extension of RDFS with defined and primitive classes would be of interest in this case.

In the ESCRIRE experiment, the closed-world assumption is mandatory since the information available is supposed to be complete. Thus, the negation operator in the DL system cannot be directly applied, and the complementary of the extension –set of instances– of a negated defined concept had to be calculated.

For the experiment, the set of concrete domains has been restricted to strings and numbers without specific comparators (except equality). For CG, these concrete domains are taken into account since CORESE can handle comparators between numbers and between strings. The TROEPS system has a sophisticated extension mechanism for handling concrete domains. However, concrete domains are not implemented in the current version of RACER used here, and thus strings and numbers have been represented as concepts. In this way, equality could be managed without particular problem.

Representation of annotations. In the Medline abstracts used for the experiment, a gene class can be interpreted as a class of individuals or as a prototype, i.e. as an individual standing for a set of individuals. This raises a basic problem: should the gene classes be represented as individuals to be manipulated in the annotations, or as classes in the ontology? This is an old debate in knowledge representation studies [21], and this leads to a difficult modeling choice: for example, what should be status of the *Antennapedia/homeotic gene* pair, instance/class, subclass/class or class/metaclass? For the ESCRIRE project, the notion of gene class has been reified: a prototype is used for representing a class of genes to be manipulated in annotations and queries. This reification choice did not raise any problem in any of the three KR formalisms.

Binary relations have been represented as classes, and their properties, e.g. transitivity or symmetry, have to be handled by specific code. Actually, the rational way of solving this problem would be an adequate extension of each KR formalism for representing and manipulating binary as well as n-ary relations. This would lead to implement specific reasoning mechanisms, e.g. for controlling the propagation of attribute values through transitivity. Once more, the RDF(S) model should be extended for enabling reflexivity, symmetry, transitivity and inverse of relations. In DL, binary relations had to be managed by an external process, and the facts inferred from relation properties have been directly added to the *ABOX* of the DL system to be further manipulated.

Reasoning and evaluation of queries. The ESCRIRE language allows references to classes but does not provide any special construction for reasoning on classes in queries. However, it can be useful for example to ask the following question *Are all classes of which Antennapedia is an instance subsumed by the class pair-rule?* Such a query should be represented as the ESCRIRE structure below, and the satisfaction of this query has to be handled by the KR formalism.

```
<esc:subsumedp>
  <esc:classof> <esc:objref type="gene"
                            id="Antp"/>
  </esc:classof>
  <esc:classref name = "pair-rule"/>
</esc:subsumedp>
```

In a query, the variables present in the `FROM` clause are assumed to be universally quantified. However, the variables present in the `WHERE` clause may also be quantified, and this may lead to some problems, because no quantifier has been especially integrated in the KR formalisms for taking into account this kind of quantification. For the TROEPS language, it suggests the need for a more complete query language and query evaluator. For CG, as the logic interpretation of a CG is a first-order logic formula, positive and existentially quantified, the existential quantifier in CG can be taken into account. However, a universal quantification is not so easy to handle: one way could be to rely on the transformation of a universally quantified formula into the negation of a existentially quantified formula, but this would require a complex extension of the projection operator. The same problem applies to DL: an existential quantification can be handled as concept satisfiability, whereas a nested universal quantification would need an extension of the DL system.

5 Related work

The ESCRIRE approach can be compared to several other approaches enabling the description of metadata about a document through semantic annotations. A system where documents are represented in DL and manipulated through their structure (part-of-reasoning) and their content is presented in [15]. The objective of the OIL project is to enable the specification and the exchange of ontologies [9], while systems such as Ontobroker [8] and SHOE [13] propose an ontology-guided retrieval of annotated documents. In the Picsel project, an information integration system is based on the DL system CARIN: querying functionalities similar to those developed for the ESCRIRE project have been studied and designed [10].

The WebKB system [16] relies on CG to represent metadata on Web documents and uses CG as a query

language for information retrieval about annotated documents. The search for documents from a query expressed in CG mainly exploits the specialization relation calculated among CG. The SiRLI system relies on a translation of RDF into F-Logic [18]: querying and inferencing both on RDF schemas and RDF statements are allowed, through arbitrarily complex queries based on the combination of variables and boolean expressions. Finally, the RQL language [14] is an OQL-based query language for RDF(S) taking completely advantage of the RDF(S) functionalities and query optimization techniques for document manipulation.

6 Synthesis and Conclusion

In this paper, we have presented an original methodology for content-based annotation and querying of documents within a KR formalism. A pivot language based on XML has been designed for bridging documents and the KR formalism. A domain ontology, annotations attached to documents, and queries have been designed both in the pivot language and in the KR formalism.

Relying on this methodology, we have proposed elements of comparison between three KR formalisms, namely descriptions logics, conceptual graphs, and object-based knowledge representation. As a first result, the ESCRIRE project shows that each KR formalism can be used with profit for representing the content of documents through annotations and an underlying domain ontology. For example, queries have been correctly answered on a test-base within the three KR formalisms. However, advantages and drawbacks present in each KR formalism, e.g. definitions in ObKR, universal individual in CG, standard concrete domains in DL (numbers, strings, ...), make explicit necessary extensions in each KR formalisms.

This experiment also enlightens the necessity of combining KR formalisms with a language such as XML for important nowadays applications such as content-based document manipulation and semantic Web. At present, a core of constructions –corresponding for the basic part to the RDF(S) constructions– is included in most of the available Web languages (DAML, OIL, ...) for these tasks. This core must still be extended for a large scale content-based manipulation of documents, as well as semantic Web management. Moreover, a general strategy for implementing the mapping between a pivot language based on XML and a concrete KR formalism has to be designed. Local strategies based on the construction of specific parsers are not easy to generalize, and lead to problems in the development and maintenance of applications. In this way, the infrastructure underlying the ESCRIRE experiment can be thought of as a query broker: the queries are answered by several engines simultaneously (and independently). This can be useful when the expressive power and the efficiency of the KR formalisms are different.

Finally, among the future works, we aim at answering the challenging question: is it better to have a very expressive representation language and a simple query language, or the inverse? We are then touching problems linked to the combination of knowledge base and database management: what are the tasks that must be respectively carried out by the KR formalism and by the database manager?

References

[1] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logics Framework for Information Integration. In *Proceedings of KR'98*, pages 2–13, 1998.

[2] R.G.G. Cattell. *The Object Database Standard: ODMG-93.* Morgan Kaufmann, San Francisco, CA, 1994.

[3] M. Chein and M.-L. Mugnier. Conceptual graphs: Fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[4] O. Corby, R. Dieng, and C. Hebert. A Conceptual Graph Model for W3C Resource Description Framework. In *Conceptual Structures: Theory, Tools and Applications, ICCS'2000, Darmstadt, LNAI*, 2000.

[5] Fensel D., Decker S., Erdmann M., and Studer R. Ontobroker in a nutshell. In *European Conference on Digital Libraries*, pages 663–664, 1998.

[6] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, 1996.

[7] J. Euzenat. Eight questions about semantic web annotations. *IEEE Intelligent Systems*, 17(1):2–9, 2002.

[8] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, and A. Witt. On2broker: Semantic-based access to information sources at the WWW. In *WebNet 99*, pages 366–371, 1999.

[9] D. Fensel, F. van Harmelen, I. Horrocks, D.L. McGuinness, and P.F. Patel-Schneider. Oil: An

ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[10] F. Gasdoué, V. Lattès, and M.-C. Rousset. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL Project. *I.J. of Cooperative Information Systems*, 9(4):383–401, 2000.

[11] V. Haarslev, C. Lutz, and R. Möller. A description logic with concrete domains and a role-forming predicate operator. *Journal of Logic and Computation*, 9(3):351–384, 1999.

[12] E.R. Harold. *The XML Bible*. IDG Books, 1999.

[13] J. Heflin, J. Hendler, and S. Luke. SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078, Dept. of Computer Science, University of Maryland at College Park, 1999.

[14] G. Karvounarakis, V. Christophides, and D. Plexousakis. Querying Semistructured (Meta)data and Schemas on the Web: The case of RDF and RDFS. Technical report 269, ICS-FORTH, 2000. http://www.ics.forth.gr/proj/.

[15] P. Lambrix. *Part-Whole Reasoning in an Object-Centered Framework*. Lecture Notes in Artificial Intelligence 1771. Springer, Berlin, 2000.

[16] P. Martin and P. Eklund. Knowledge Retrieval and the World Wide Web. *IEEE Intelligent Systems*, Special issue on Knowledge Management and the Internet, 2000.

[17] S.A. McIlraith, T.C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[18] Decker S., Brickley D., Saarela J., and Angele J. A Query Service for RDF. QL'98. Query Languages Workshop, W3C Workshop, 1998. http://www.w3.org/TandS/QL/QL98/.

[19] Navathe S., Savasere A., Anwar T., Beck H., and Gala S. 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.

[20] Projet Sherpa. Tropes 1.0 reference manual. Technical report, INRIA Rhône-Alpes, Grenoble, 1998. ftp://ftp.inrialpes.fr/pub/sherpa/rapports/.

[21] E.E. Smith and D. Medin. *Categories and Concepts*. Harvard University Press, Cambridge, MA, 1981.