

Las Bases de Datos y la Web Semántica: Ficción o Realidad

María del Mar Roldán-García

Dept. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Campus de Teatinos s/n
29071 Málaga
mmar@lcc.uma.es

José F. Aldana-Montes

Dept. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Campus de Teatinos s/n
29071 Málaga
jfam@lcc.uma.es

Resumen

La Web Semántica ha revolucionado la investigación en diversos campos, siendo los más destacables las Bases de Datos y la Inteligencia Artificial (AI). Mientras las bases de datos se centran en la consulta de grandes volúmenes de información, la AI se centra en desarrollar algoritmos de razonamiento correctos y completos para pequeñas ontologías. La combinación de los resultados de ambas disciplinas es fundamental para el desarrollo práctico de la Web Semántica. Sin embargo, esta combinación no es trivial. Con este artículo pretendemos, por un lado, aclarar la terminología utilizada por estas dos comunidades científicas, y por otro, identificar algunos puntos de investigación relevantes en los que la comunidad de bases de datos tiene mucho que decir.

1. Introducción

La Web Semántica ha revolucionado la investigación en diversos campos, siendo los más destacables las Bases de Datos y la Inteligencia Artificial (AI). Las ontologías juegan un papel fundamental en el desarrollo de esta nueva generación de aplicaciones, y hasta el momento, en el campo de la Inteligencia Artificial y Representación del Conocimiento (KR) se ha estudiado la representación del conocimiento mediante ellas y se han definido mecanismos de razonamiento correctos y completos que permiten inferir nuevo conocimiento a partir del conocimiento representado explícitamente. Las lógicas de descripciones son el formalismo utilizado para representar el conocimiento mediante ontologías. Son varios los problemas que aparecen al intentar aplicar los resultados obtenidos por esta comunidad a la Web

Semántica. En primer lugar, los algoritmos de razonamiento están orientados a memoria principal y no tienen en cuenta grandes volúmenes de información, es decir, grandes cantidades de instancias, y por tanto, no son escalables. Por otro lado, la mayoría del esfuerzo durante las últimas décadas se ha dedicado al desarrollo de algoritmos de razonamiento sobre la estructura de la ontología (lo que se denomina en lógica de descripciones la Tbox). Sin embargo, las aplicaciones en la Web Semántica necesitan un método de recuperación de instancias y consultas, que además de permitir recuperar el conocimiento, sea capaz de inferir información no sólo sobre la estructura de la ontología, sino también sobre las instancias (lo que en lógica de descripciones se denomina Abox). Esta información inferida debe poder utilizarse para mejorar el resultado de las consultas. Por todo esto, la comunidad de AI & KR comenzó hace unos años a estudiar posibles lenguajes de consulta para los sistemas que se habían desarrollado. Estos estudios dieron como resultado diversas propuestas para realizar consultas conjuntivas sobre bases de conocimiento descritas mediante lógicas de descripciones [1]. Posteriormente, con la aparición de OWL se comenzó a desarrollar un lenguaje de consultas específico para este lenguaje, que pretende convertirse en estándar, el lenguaje OWL-QL [2] y recientemente se comenzó a desarrollar SPARQL [3] para consultar RDF. El sistema RACER [4] es el sistema basado en lógica de descripciones más representativo. Según sus autores y según podemos observar en la literatura, en la actualidad es el sistema más completo. No sólo implementa mecanismos de razonamiento sobre la Tbox, sino que además proporciona un lenguaje de consultas, el nRQL (new Racer Query Language) [5] que permite realizar consultas conjuntivas. Además, también implementa, en su

última versión, mecanismos de razonamiento sobre la Abox. Sin embargo, el sistema no ofrece persistencia y los mecanismos de razonamiento se implementan mediante complejas técnicas de reducción a satisfabilidad.

Es lógico pensar, que en el momento en que se habla de grandes volúmenes de información y de consultas, los resultados obtenidos por la comunidad de base de datos no puede ser obviados. Durante las últimas décadas esta comunidad se ha dedicado al estudio de sistemas que almacenen de forma persistente grandes cantidades de información, y a consultarla mediante lenguajes de consulta eficientes y escalables. Por tanto, los investigadores en bases de datos han comenzado a desarrollar sistemas que almacenan ontologías de forma persistente, y lenguajes de consulta para RDF, como RQL [6], RDQL [7], SeRQL [8] y Squish [9] entre otros. Sin embargo, estos sistemas están muy enfocados a proporcionar persistencia a las ontologías y un lenguaje de consultas al estilo de las bases de datos, pero tienen una cuenta pendiente con los mecanismos de razonamiento, sobre todo, con los mecanismos para inferir conocimiento a partir de instancias.

La situación hoy en día, es que estas dos comunidades suelen trabajar por separado, y se producen confusiones terminológicas importantes, ya que ambas comunidades entienden de forma diferente los conceptos de consulta y de razonamiento. En este artículo pretendemos, por un lado, aclarar la terminología utilizada por ambas comunidades, y por otro, estudiar la semántica de los mecanismos de razonamiento y de las consultas sobre lógicas de descripciones, con el objetivo de que la comunidad de bases de datos tenga una base sobre la que trabajar, y sea más fácil identificar los puntos en los que los investigadores en bases de datos podemos realizar aportaciones, sin duda muy necesarias, para el desarrollo práctico de la Web Semántica.

2. Definiciones

Para ayudar a la comprensión de los siguientes capítulos, vamos a definir aquellos términos que se van a utilizar.

Definición 1: Lógicas de descripciones. Las lógicas de descripciones forman una familia de formalismos lógicos, relacionados con las redes

semánticas y los sistemas basados en marcos, para la representación y el razonamiento sobre clases complejas de individuos (denominados conceptos) y sus interrelaciones (representadas mediante relaciones binarias denominadas roles). Típicamente se distingue entre conceptos atómicos (o primitivos) y conceptos complejos definidos mediante los constructores del lenguaje. Los lenguajes de las diferentes Lógicas de Descripciones varían en el conjunto de constructores que proporcionan. El lenguaje OWL está basado en una de esas lógicas de descripciones [10], donde los conceptos y roles se denominan respectivamente clases y propiedades.

Definición 2: Base de Conocimiento. Mediante ontologías definimos lo que se denomina una base de conocimiento. Una base de conocimiento tiene dos componentes, una parte terminológica (Tbox) que consiste en conceptos, roles y construcciones complejas a partir de éstos, y que se corresponde con el conocimiento intensional, y una parte extensional (Abox) que son afirmaciones sobre individuos «concretos». La Tbox de una base de conocimiento se estructura en forma de jerarquía de conceptos, definiéndose qué conceptos subsumen o son subsumidos por otros.

Definición 3: Mecanismos de razonamiento sobre la Tbox. Son los que nos permiten consultar la estructura del conocimiento (Tbox) e inferir información a partir de ella. También son conocidos como mecanismos de razonamiento estructurales o intensionales. El mecanismo de razonamiento estructural por excelencia es la subsunción de conceptos, que permiten calcular todos los subconceptos de un concepto dado o consultar si un concepto es subconcepto de otro. Los razonamientos usuales en la Tbox son: Consistencia (satisfabilidad), que comprueba si el conocimiento tiene sentido o no; Subsunción, que es la comprobación de si todos los individuos que pertenecen a un concepto (el subsumido) también pertenecen a otro concepto (el que subsume); y Equivalencia que comprueba si dos clases denotan el mismo conjunto de instancias o individuos. Todos estos razonamientos son reducibles al problema de la satisfabilidad de fórmulas lógicas siempre que usemos un lenguaje de definición de conceptos que sea cerrado con respecto a la negación.

SubclaseDe	Si x es una instancia de C y $C \sqsubseteq D$, entonces x es instancia de D .
ClasesEquivalentes	Si x es instancia de C y C es equivalente a D , entonces x es instancia de D .
SubPropDe	Si (x,y) es una instancia de P y $P \sqsubseteq Q$, entonces (x,y) es una instancia de Q .
PropiedadTransitiva	Si (x,y) es una instancia de P y (y,z) es también una instancia de P y P es transitiva, entonces (x,z) es instancia de P .
PropiedadSimetrica	Si (x,y) es una instancia de P y P es simétrica, entonces (y,x) es instancia de P .
PropiedadInversa	Si (x,y) es una instancia de P y Q se define como la inversa de P , entonces (y,x) es instancia de Q .
PropiedadFuncional	Si (x,y) es una instancia de P , (x_2,y) es también instancia de P , y P es funcional, entonces x_1 y x_2 son la misma instancia.
Union	Si x es instancia de C_i , $i:1..n$, entonces x es instancia de $C_1 \sqcup \dots \sqcup C_n$.
Interseccion	Si x es instancia de $C_1 \sqcap \dots \sqcap C_n$, entonces x es instancia de C_i , $i:1..n$.
Dominio	Si (x,y) es una instancia de P , y C es la clase dominio de P , entonces x es instancia de C .
Rango	Si (x,y) es una instancia de P , y D es la clase rango de P , entonces y es instancia de D .
Allvalues	Si (x,y) es una instancia de P , C es la clase dominio de P , x es instancia de C , y existe una restricción sobre la clase C del tipo $\forall P.D$, entonces y es instancia de D .

Tabla 1. Mecanismos de Razonamiento sobre la Abox

Definición 4: Mecanismos de razonamiento sobre la Abox. Son los que nos permiten inferir nuevas instancias a partir de las definidas de forma explícita en la Abox. Estos mecanismos se denominan también mecanismos de razonamiento extensionales. Típicamente los razonamientos básicos en la Abox son: Comprobación de Instancias, que consiste en verificar cuando un determinado individuo es una instancia de un concepto específico; Consistencia de la Base de Conocimiento, que implica verificar cuando cada concepto que existe en la Base de Conocimiento admite, al menos, una instancia o individuo; y la Realización que encuentra el concepto más específico del que un individuo es instancia.

3. Implementación de mecanismos de razonamiento sobre la Abox en RACER

Como ya hemos mencionado anteriormente, creemos que las aplicaciones en la Web Semántica van a necesitar mecanismos de razonamiento sobre la Abox, que permitan inferir conocimiento a partir de las instancias de la ontología, y aprovechar este conocimiento inferido para mejorar el resultado de las consultas. En este apartado describiremos brevemente los mecanismos de razonamiento sobre la Abox que creemos necesario que implemente cualquier sistema basado en ontologías. Todos estos mecanismos están implementados en la última versión de RACER, lo que demuestra que la comunidad de IA&KR está realizando grandes

esfuerzos para acercar sus sistemas a las necesidades de la Web Semántica. Sin embargo, queda aun un gran esfuerzo por hacer en materia de persistencia, escalabilidad y eficiencia de dichos mecanismos. En [11] se realiza un estudio más exhaustivo sobre los mecanismos de razonamiento. Por tanto, presentaremos aquí los mecanismos de forma resumida tal y como se muestran en la tabla 1.

4. Recuperación de Instancias vs. Consultas

Desde hace bastante tiempo, se viene poniendo de manifiesto la necesidad de definir un lenguaje de consultas para recuperar instancias de una base de conocimiento, más allá del mecanismo de recuperación de instancias definido para las lógicas de descripciones¹. Como hemos mencionado anteriormente, la comunidad de base de datos por un lado, y la comunidad de Inteligencia Artificial por otro, han desarrollado sus propuestas. Sin embargo, tras estudiar estas propuestas y tras realizar algunas pruebas con el sistema RACER, hemos extraído algunas conclusiones interesantes, que nos permiten observar cómo el concepto de consultas no es el mismo para ambas comunidades.

¹ Este mecanismo nos devuelve las instancias de un concepto dado. Este conjunto de instancias estará formado por las instancias de ese concepto más las instancias que se puedan inferir mediante la jerarquía de conceptos.

Para los investigadores en bases de datos, una consulta significa recuperar aquellas tuplas que cumplen unas determinadas restricciones. Estas restricciones se aplican a los atributos de las tablas. La forma de resolver una consulta en una base de datos consiste en localizar las tuplas que cumplen esas restricciones y devolverlas². En lógica de descripciones, son dos las aproximaciones que se han seguido para intentar dotar a estos sistemas de un lenguaje de consulta. Por un lado en [12] se propone usar los constructores de conceptos complejos para realizar consultas. Si para definir un concepto complejo a partir de un concepto atómico usamos constructores que lo restringen, podríamos pensar en usar ese mismo procedimiento para consultar la Abox. Por otro lado, se han desarrollado trabajos para realizar consultas conjuntivas sobre la Abox [13]. Estas consultas serían similares a las consultas conjuntivas en bases de datos. Sin embargo, mientras que en una base de datos es obligatorio que existan instancias, no es obligatorio en una ontología definir de forma explícita a qué concepto pertenece una determinada instancia, ya que los mecanismos de razonamiento que soporte el lenguaje nos permitirán inferir información no definida de forma explícita. En una base de datos, existe una diferencia muy clara entre el esquema (los metadatos) y las instancias (los datos) y el esquema tiene que estar completamente definido antes de que podamos definir instancias del mismo. Esta separación no es obligatoria cuando trabajamos con ontologías, ya que las instancias pueden definirse en cualquier momento. Por otro lado, en una base de datos, el esquema define qué es nuestro dominio de aplicación, y las instancias son válidas sólo si cumplen completamente con las definiciones y restricciones especificadas en el esquema. En otro caso, son rechazadas. En el caso de las ontologías, las instancias son aceptadas siempre que no contradigan de forma explícita el conocimiento del dominio de aplicación definido por la ontología, sin obligar a que estas tengan definidas todas las características que se esperan. En el caso de que un mecanismo de inferencia encuentre alguna instancia parcialmente definida, simplemente asume que no sabe la definición para

² Es decir, una consulta es una fórmula lógica cuantificada existencialmente e interpretada luego bajo la hipótesis del mundo cerrado (CWA).

esa característica en concreto. En otras palabras, las bases de datos trabajan con la hipótesis del mundo cerrado, mientras que las ontologías trabajan con la hipótesis del mundo abierto.

Analizaremos por separado las dos aproximaciones para consultar una base de conocimiento.

4.1. Consultas Conjuntivas

Las consultas conjuntivas aportan la misma semántica que las consultas conjuntivas en una base de datos siempre y cuando no utilicemos ningún mecanismo de inferencia y obliguemos a que todas las variables de la consulta tengan que ligarse. Los lenguajes de consulta en bases de datos designan un subconjunto de variables en la consulta cuyas ligaduras se incluyen en la respuesta. En los lenguajes de representación del conocimiento, incluido OWL, una base de conocimiento puede deducir la existencia de una respuesta para una consulta pero no una ligadura para cada variable de la misma.

A pesar de que se han propuesto otros lenguajes de consulta, a día de hoy nRQL[5], el lenguaje del sistema RACER es el más completo y el único que está implementado. Este lenguaje obliga a ligar todas las variables en la consulta. La otra gran apuesta de los investigadores en Inteligencia Artificial es OWL-QL[2], aunque no parece que esté teniendo mucho éxito. Más que un lenguaje de consultas con una sintaxis concreta, OWL-QL especifica formalmente las relaciones semánticas entre una consulta, su respuesta y la base de conocimiento. Este lenguaje define tres tipos de variables, *must-bind*, *may-bind* y *don't-bind*. Las respuestas de las consultas deben proporcionar ligaduras para todas las variables *must-bind*, pueden suministrar ligaduras para las *may-bind*, y no tienen que proporcionarlas para las *don't-bind*.

Por ejemplo, imaginemos que en la base de conocimiento definimos una restricción que indica que cada persona tiene exactamente un padre. Supongamos que definimos la siguiente consulta

tienePadre(?x,?y)

- Si *?y* es una variable *don't-bind*, entonces el resultado de la consulta serían todas las personas de la base de conocimiento, ya que en este caso, no estamos interesados en saber quien es el padre, sino sólo queremos las personas que tiene padre, y por la restricción

que hemos definido, sabemos que cada persona tiene un padre.

- Si ?y es una variable must-bind, el resultado de la consulta es aquellas personas para las que se ha especificado quien es su padre, devolviendo pares (persona, padre).
- Si ?y es una variable may-bind, el resultado de la consulta será todas aquellas personas de la base de conocimiento, y para los que esté especificado, el nombre del padre.

Los mecanismos de inferencia son los que realmente hacen que resolver una consulta conjuntiva sobre una base de datos y sobre una base de conocimiento se devuelvan resultados diferentes. Nos vamos a centrar en el mecanismo de subsunción de conceptos, que es el que como mínimo ofrece cualquier sistema de lógica de descripciones, pero el proceso es exactamente el mismo para cualquier otro tipo de inferencia. El sistema RACER infiere automáticamente que cuando una clase es subclase de otra, las instancias de la clase subsumida son también instancias de la clase que subsume. Por tanto, las instancias de la clase subsumida se consideran a la hora de resolver la consulta. En una base de datos, esto no ocurre al menos que implementemos dicho mecanismo de inferencia. Veámoslo con un ejemplo. Supongamos la siguiente consulta,

```
x <- Persona (x) ∧ estaMatriculado(x,'Bases de Datos')
```

Queremos recuperar aquellas personas que están matriculados en la asignatura de bases de datos.

En principio, la consulta debería devolver el mismo conjunto de instancias en una base de conocimiento que define la clase Persona y la propiedad estaMatriculado y en una base de datos que define una tabla Persona y una tabla estaMatriculado. Sin embargo, supongamos que definimos la clase Alumno como una subclase de la clase Persona. En una base de conocimiento, el sistema infiere que las instancias de Alumno son también instancias de Persona, y por tanto, devolvería aquellas instancias de Alumno que estén matriculados en la asignatura bases de datos. En una base de datos, tendríamos varias opciones para implementar esta relación de subsunción.

1. Crear una tabla para la clase Alumno y migrar la clave primaria de Persona como clave externa a la tabla Alumno. De esta forma, antes de

insertar una tupla en la tabla Alumno, debemos insertarla en la tabla persona. Esto rompe con la filosofía de las ontologías, ya que éstas permiten definir instancias de cualquier clase, sin necesidad de cumplir ninguna restricción salvo la de no contradecir de forma explícita el conocimiento definido por la ontología. Por tanto, esta opción queda descartada.

2. Para resolver el problema anterior podríamos omitir la restricción de clave externa y disparar un trigger (en el caso de que el gestor de bases de datos lo permita) que cuando se inserte una tupla en la tabla Alumno, automáticamente inserte la tupla en la tabla Persona. Esta opción conlleva duplicar la información en las dos tablas, y además, si suponemos una jerarquía de clases con un número muy elevado de ellas, queda claro que esta solución se vuelve inviable. Además, una supuesta modificación de la jerarquía conllevaría actualizar todos los triggers.
3. Finalmente, la solución más razonable es crear una tabla que almacene la jerarquía de clases una vez precomputada por un razonador. De esta forma, tenemos almacenadas en la base de datos las subclases de cada clase, pudiendo recuperarlas con una consulta simple. Sin embargo, esto obliga a que cada vez que se lance una consulta conjuntiva, haya que reescribir dicha consulta para que se evalúe sobre las tablas que representan las subclases de cada una de las clases implicadas en la consulta. Con esta solución conseguimos que las consultas conjuntivas tengan la misma semántica en ambas representaciones. Para el resto de los mecanismos de razonamiento habría que almacenar la información necesaria para implementarlos en la base de datos.

4.2. Conceptos Complejos como Consultas

Las consultas conjuntivas son consultas simples. Sabemos que en una base de datos podemos realizar consultas mucho más complejas. Sin ir más lejos, las consultas conjuntivas no nos permiten establecer restricciones de cardinalidad, de cuantificación universal (para todo), etc. En un principio, parece lógico pensar que el uso de conceptos complejos para consultar la Abox puede resolver este problema. Sin embargo, la semántica de estas consultas en lógica de

descripciones es muy distinta a la de las consultas en bases de datos. Para demostrarlo, vamos a usar un ejemplo. En OWL tenemos el constructor `owl:allValuesFrom`, cuya semántica se define como:

$$(\forall P.C)^I = \{x \in \Delta^I \mid \forall b. (x,y) \in P^I \rightarrow y \in C^I\}$$

siendo C una clase, P una propiedad y x,y instancias. Es decir, define la clase de individuos x para los cuales se cumple que si el par (x,y) es instancia de la propiedad P , entonces y debe ser una instancia de la clase C . Por tanto, el concepto complejo `∀ estaMatriculado.CursoDoctorado` define aquellos individuos que están matriculados exclusivamente de cursos de doctorado. Otro constructor es `owl:intersectionOf`, cuya semántica se define como:

$$(C \sqcap D)^I = C^I \cap D^I$$

siendo C y D clases, que define aquellos individuos que son instancias de C y D . Por tanto, el concepto complejo

$$\text{Persona} \sqcap \forall \text{ estaMatriculado.CursoDoctorado}$$

define aquellas personas que están matriculadas exclusivamente de cursos de doctorado.

Si utilizamos este concepto complejo como consulta, estaríamos preguntando por aquellas personas que están matriculadas exclusivamente de cursos de doctorado. En una base de datos, se buscarían en la tabla correspondiente a las personas, se miraría los cursos en los que están matriculados, y se comprobaría si todos ellos son cursos de doctorado, devolviendo aquellas personas que cumplen la restricción. La forma intuitiva de resolver esta consulta en un sistema de lógica de descripciones como RACER, consistiría en construir este concepto complejo, y después consultar sus instancias. Sin embargo, la definición del concepto lo único que hace es colocarlo en la jerarquía de conceptos donde corresponda (mediante el mecanismo de clasificación), pero no infiere qué instancias pertenecen a dicho concepto. Por tanto, cuando consultamos las instancias del nuevo concepto, no obtenemos los resultados esperados (para la idea de consulta en una base de datos). El razonador intenta inferir de la Tbox si hay algún concepto que cumpla esta restricción basándose exclusivamente en la estructura de la ontología, es decir, miraría si según la estructura, podemos inferir que existe una restricción que obligue a las

instancias de un concepto determinado a estar matriculado únicamente de cursos de doctorado, y entonces, devolvería la intersección de esas instancias con las de la clase `Persona`, pero nunca consulta la Abox para comprobar si alguna de las instancias cumple esta restricción, como haríamos en una base de datos.

5. Conclusiones

Tanto la comunidad de Inteligencia Artificial como la comunidad de Bases de Datos están realizando grandes esfuerzos por dotar de viabilidad práctica a la Web Semántica. Es indudable que para conseguirlo, es necesario desarrollar sistemas basados en ontologías, que nos permitan gestionarlas de forma eficiente. Estos sistemas deben permitir almacenar de forma persistente grandes ontologías (con grandes cantidades de instancias), razonar sobre su estructura (razonamientos en la Tbox), razonar de forma estructural (p.e. clasificar conceptos), consultar sus instancias permitiendo imponer restricciones complejas (al estilo de las de las bases de datos) para especificar el conjunto de instancias que queremos recuperar, razonar sobre sus instancias y aprovechar este nuevo conocimiento para mejorar el resultado de las consultas, y todo esto, de forma eficiente y escalable. Por tanto, queda claro que es necesario introducir técnicas de bases de datos que completen los resultados aportados por la comunidad de Inteligencia Artificial. Las principales aportaciones que se pueden hacer desde el punto de vista de las bases de datos son:

- Definir modelos de almacenamiento persistente para las ontologías.
- Acoplar los razonadores existentes como RACER para que realicen los razonamientos estructurales sobre un modelo de almacenamiento persistente.
- Definir un lenguaje de consultas que permita recuperar instancias, que cumplan determinadas restricciones, que estén almacenadas en la Abox, al estilo de los lenguajes de consulta de las bases de datos.
- Implementar los mecanismos de razonamiento sobre la Abox sobre estos modelos persistentes, y usarlos para resolver de forma más completa las consultas.

- Definir diseños físicos de las bases de datos que almacenen las ontologías (camino de acceso, etc.) que mejoren el rendimiento de las consultas y los razonamientos.

Referencias

- [1] A. Borgida, M. Lenzerini, and R. Rosati. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.
- [2] Richard Fikes, Pat Hayes, Ian Horrocks. OWL-QL: A Language for Deductive Query Answering on the Semantic Web. KSL Technical Report 03-14.
- [3] Eric Prud'hommeaux, Andy Seaborne. SPARQL Query Language for RDF. W3C Candidate Recommendation 6 April 2006. <http://www.w3.org/TR/rdf-sparql-query/>
- [4] Haarslev, V., Möller, R. RACER System Description. Proceedings of International Joint Conference on Automated Reasoning, IJCAR'2001, Springer-Verlag, 2001.
- [5] V. Haarslev, R. Möller, R. van der Straeten, and M. Wessel. Extended Query Facilities for Racer and an Application to Software-Engineering Problems. In Proceedings of the 2004 International Workshop on Description Logics (DL-2004), Whistler, BC, Canada, June 6-8, pages 148-157, 2004.
- [6] G. Karvounarakis, A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, K. Tolle, RQL: A Functional Query Language for RDF. At The Functional Approach to Data Management: Modelling, Analyzing and Integrating Heterogeneous Data, P.M.D.Gray, L.Kerschberg, P.J.H.King, A.Poulovassilis (eds.), LNCS Series, Springer-Verlag, 2004.
- [7] RDQL - A Query Language for RDF W3C Member Submission 9 January 2004. <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
- [8] The SeRQL query language (revision 1.2). <http://www.openrdf.org/doc/sesame/users/ch06.html>
- [9] Libby Miller. RDF Squish query language and Java implementation. Latest version: <http://ilrt.org/discovery/2001/02/squish/>
- [10] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. J. of Web Semantics, 1(4):345-357, 2004.
- [11] María del Mar Roldán García, José F. Aldana Montes. Lenguaje de Consultas Avanzado. Informe Técnico Proyecto ICARO, IT-4.2-1. Enero 2006.
- [12] Maurizio Lenzerini and Andrea Schaerf. Concept languages as query languages. Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI'91).
- [13] Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In National conference on artificial intelligence (AAAI 2000), pages 399-404, 2000.