Using and improving bio-ontologies stored in relational databases*

C. Maria Keet
KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano
Piazza Domenicani 3, 39100 Bozen-Bolzano, Italy
keet@inf.unibz.it

Abstract

Current persistent storage implementations of large bio-ontologies fall short in meeting usability and reusability requirements. Main limitations are demonstrated with two representative ontologies-stored-in-databases: the Gene Ontology and the Foundational Model of Anatomy. We discuss and propose several improvements for ontology development. Advantages of a better implementation are illustrated with granularity based querying. This granular information retrieval enables both retrieving more information quicker and can be used for improving and enlarging an ontology.

1. Introduction

The added-value of ontologies has been reiterated many times over, but at present, its effective usage falls short of the promises. Ontologies are not intended just for storing knowledge about the subject domain, browsing and (manual) annotation of data, but can be used for data integration, inferencing, Semantic Web agents mediation, ought/will be more than just an online encyclopaedia and so forth. While, of course, it is first a challenge to develop a good ontology, and a second to use it effectively, but *how* the captured knowledge is stored persistently affects what one can do with the ontology: OWL files can be used for reasoning purposes to e.g. classify a taxonomy and check consistency, whereas this is not possible with ontologies stored in a (relational) database. On the other hand, relational databases have about 30 years of established research and implementations behind them, where one can take advantage of e.g. sophisticated querying. Taking into account ontology as an engineering artifact, then implementation decisions for ontology development are not trivial. In addition, over the years there always is a 'requirements slip': desiring to use an engineering artifact for other purposes it was originally designed for and users moving the goal posts. For instance, data/information retrieval from ontologies – getting out what has been put in the ontology – with as a sub-goal granularity, i.e. querying an ontology at one's desired different levels of detail, which goes further than the "retrieve ancestors"-type of query. Querying an ontology stored in a database at the desired level of detail can be done 'on the fly' with database views that return e.g. all subtypes of tissue that reside in the *Tissue*-level, or with a full formal domain granularity framework respecting ontological differences in types of granularity [9] that contains different perspectives on the data or ontology, has its granular levels for each perspective predefined, and may be used for inferencing. A conceptually relatively simple method is to position a taxonomy and partonomy orthogonally and to use the *partOf* relation (or spatial *containedIn*, *involvedIn* for processes) to distinguish between levels of granularity. With this combination, one can structure and retrieve more information from the same resource that is otherwise available in the ontology but inaccessible.

With ontologies stored in databases, such change in content and structure of the queries, is, *in theory*, not a problem. However, with current *engineering practices* this is not possible, or only after elaborate reengineering. Two well-known large ontologies stored in relational databases are the Gene Ontology (GO) with more than 19,000 entities [4] and the Foundational Model of Anatomy (FMA) with about 72,000 entities and 1.9 million relations [11], which we analyze and discuss in the next section. We have chosen these two ontologies-stored-in-databases, because the GO is a *de facto* trendsetter for the Open Biomedical Ontologies (OBO) family of ontologies and the FMA is the frontrunner for expressive ontologies created with the Protégé ontology development environment. These ontologies are widely used for annotation but rarely used for ontology-driven information systems [5]. The issues that will be brought afore in the next section, however, are not unique, but representative of the challenges ahead and design considerations to take into account to bring ontologies to the next level to facilitate effective usage, which also has the advantage of

*This is an extended abstract of the talk held at *SBIOLBD 2006*, 13-2-2006, EPFL, Lausanne, Switzerland.

being able to improve the ontologies themselves, discussed in section 3. We conclude in section 4.

2 Case studies: GO and FMA

2.1. The GO database

Common types of queries of the GO in the DAG-Edit tool [21], AmiGO web page [20], or directly to its database are "find string *x*" in the tree, definition, comment, id, etc., or the supertype or sibling of *x*; querying for "descendants only" is problematic. Utilising the taxonomy and partonomy separately or orthogonally is as of yet not possible without losing crucial information. The GO (MySQL) database itself does not support recursive queries (required for traversing trees), but offers a workaround with the additional `graph_path` table. In `graph_path`, "Paths between any two terms (traversing down only) this table states whether there exists a path between a parent and a child, and the distance between them. multiple paths mean multiple entries in this table" [24]. Querying a section of the biological_process GO [23] with the sample query of [24] using `ancestor.name = 'blood coagulation'` and adding `ORDER BY graph_path.distance ASC`:

```
SELECT rchild.name, rchild.acc, graph_path.distance
FROM term AS rchild, term AS ancestor, graph_path
   WHERE graph_path.term2_id = rchild.id
      AND graph_path.term1_id = ancestor.id
      AND ancestor.name = 'blood coagulation'
   ORDER BY graph_path.distance ASC;
```

gives the output as in Table 1 (data from v3-8-2005 of the geneontology.obo (BP)), having fetched every descendant of *BloodCoagulation* by *listing* the descendants with distance to the assigned root instead of maintaining the hierarchical structure. A DAG-Edit "descendants only" can be retrieved by selecting "`has` *ancestor* `that equals` *blood coagulation*", which returns an unordered set without the distances retrieved with the SQL query. Problematic is that `distance` puts together *any* type of relation: the in-/extrinsic pathways of blood coagulation are taxonomic subtypes whereas the other entities are *part of* blood coagulation. Hence, although the knowledge is stored in the database, the query answer is ontologically meaningless. To allocate entities correctly in the appropriate level of granularity, one requires a (depth-first) algorithm that during the search uses the *type* of relation that is traversed. The translations of the GO database to make it browsable through Protégé or convert it into OWL-format does not automatically solve these issues, as will be described in the next sections.

Table 1. Output for fetching descendants of *Blood coagulation* in GO, using the database query.

Name	acc	distance
blood coagulation	GO:0007596	0
platelet activation	GO:0030168	1
regulation of blood coagulation	GO:0030193	1
blood coagulation, intrinsic pathway	GO:0007597	1
blood coagulation, extrinsic pathway	GO:0007598	1
positive regulation of blood coagulation	GO:0030194	2
negative regulation of blood coagulation	GO:0030195	2
fibrinolysis	GO:0042730	3

2.2. The FMA database

One can browse the FMA online [22] and search for a single entity, or get the database dump and install MySQL and Protégé to browse its contents locally through the Protégé software, that does not allow SQL database queries. The FMA database is tightly integrated with, and entirely dependent on, the Protégé application software, which is revealed not only upon inspecting the database dump, but even clearer after reverse engineering the conceptual model (with VisioModeler 3.1), shown in Fig. 1, that is devoid of semantics. For instance the `'short_value' varchar(255) default NULL` in the definition of the `FMA` table may record values, among others, like `'English'`, `'31227'`, and `'Thu Aug 12 11:30:30 PDT 1999'`. Browsing the FMA in Protégé, one has to click each individual relation and navigate through the myriad of pop-up screens to find information. Finding the answer to a simple query like "how many organs does a human have?" is impossible, assuming one does not want to browse and count manually thousands of organ types in Protégé. This combination has been re-engineered into a format usable for querying: the OQAFMA query agent [11]. Pre-processing steps involve optimizations such as the creation of a new table for each edge (relation) type, adding views and storing it in a PostgreSQL database [11], and can be queried through an online interface [25] using the STRUQL query language that returns the answer in XML for further processing (e.g. visualization). Before testing OQAFMA features to support granular queries, several queries were formulated in natural language, bearing in mind how domain granularity for human anatomy may be defined (e.g. [10]) with organ-tissue-cell-organelle and so forth based on the FMA partonomy as shown in Table 2 [1],

[1] It is outside the scope of this extended abstract to go into details of a) the ontological (un)soundness of the presented level definition, naming, and allocation of the entities in their respective level, and b) the granularity

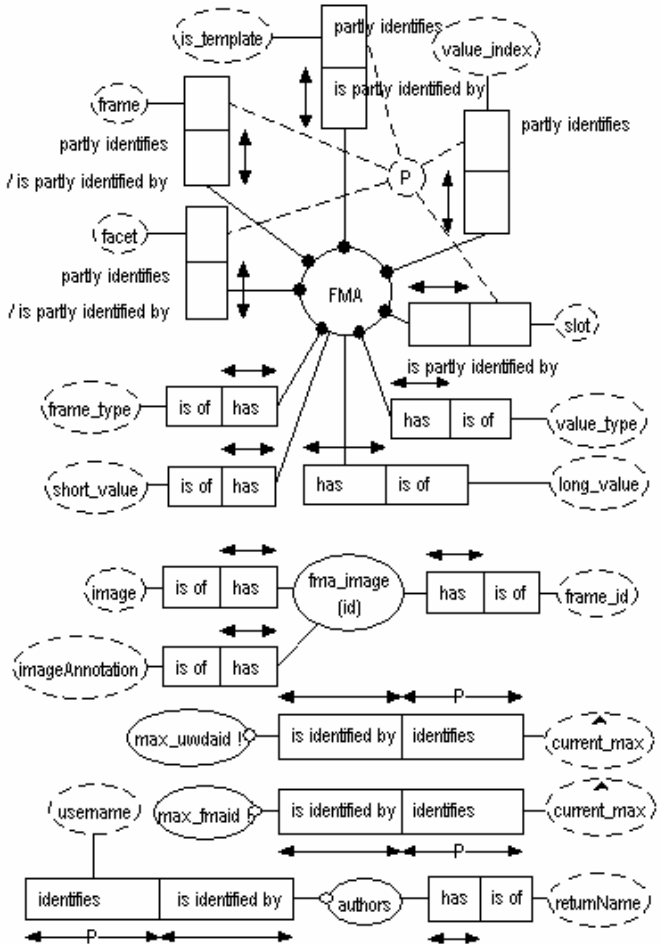

Figure 1. Reverse engineered ORM conceptual model of the FMA database.

but not declared and hard coded in the database. Two of the questions are:

I. "what are the cellular components of blood?", where *Blood* resides in another level (be it *BodySubstance* or *Tissue*) than the cells it has as parts.

II. "In which organs are macrophages located?", i.e. for each macrophage and its subtypes, in which organ (subtype) is it located?

These questions were translated into STRUQL and the database queried through the OQAFMA interface to retrieve the result in XML format.

Query I can be reformulated into STRUQL syntax as shown in Fg. 2: the `WHERE` clause contains mappings and

in a granularity hierarchy to 'skip' levels where one may identify in reality more levels than used in the software system, like $gp_{hsa}gl_2$ collapses three levels into one. Both [16] and [19] achieved better performance with software implementation using wider and shallower levels of granularity than with fewer perspectives that had more detailed levels; this is a topic of future research.

Table 2. Granular perspective gp_{hsa} human structural anatomy and nine defined levels with some examples.

Level	Name	Examples
$gp_{hsa}gl_1$	Body	*MaleBody*
$gp_{hsa}gl_2$	Principal body part Subdiv. of pr. body part Organ system	*Head, Limb* *LimbGirdle, Face* *RespiratorySystem*
$gp_{hsa}gl_3$	Organ	*SalivaryGland, Pancreas*
$gp_{hsa}gl_4$	OrganPart	*Cortex, LymphNode*
$gp_{hsa}gl_5$	Tissue	*Epithelium, SmoothMuscle*
$gp_{hsa}gl_6$	Tissue part	*HairFollicle, Nail*
$gp_{hsa}gl_7$	Cell	*Melanocyte*
$gp_{hsa}gl_8$	Cell part	*Chromosome, Cytoskeleton*
$gp_{hsa}gl_9$	Molecule	*Protein, Melanin*

which relations have to be traversed, "+" is the closure operator following the specified type of relation an arbitrary number of times, and a "*" combines the "+" with the optional operator "?" for path alternation, the `CREATE` clause says what to return as the query answer [11], hence OQAFMA supports recursive queries. The query answer is graphically depicted in Fig. 3 as the 40 not greyed-out names in the tree. The 12 checked entities are listed in the FMA/Protégé interface's "Part" scroll-down box. Obviously, the FMA/Protégé parts list is incomplete and positioning the taxonomy orthogonally *automatically* does reveal the lacunas in the list. This also indicates the difficulty of constructing a parts list *de novo* in contradistinction to querying the extant modelled knowledge. Thus, through combining existing encoded information differently by using granularity of the subject domain in formulation of the query, one can retrieve or discover 'new' information that otherwise would have remained inaccessible and hidden in the database. Put differently: there is more information captured in the FMA than meets the eye when only browsed with Protégé. Considering the query answer has 40 entities as opposed to the incomplete 12 listed in FMA/Protégé, one can use database querying for verification of, and finding gaps in, the FMA contents as a first step, and subsequently investigate the possibilities to use this approach to aid (semi-)automatic creation of new relations in the FMA.

The above brings afore some design considerations: should one include in the parts list all 'intermediate' entities, only the leaves, or maybe only the shared common ancestor? For instance, only *Reticulocyte*, *Echinocyte* and *MatureErythrocyte*, or also their common subsumer *Erythrocyte*, or only *Erythrocyte* but not its subtypes? The answer depends on the goal(s) one wants to achieve: leaves only, minimal amount of coarse-grained (top-level) parts, or re-

construct the relevant section of the taxonomic tree. Deciding either way will improve reliable and consistent querying of the FMA. A second design consideration is querying versus browsing only: while current browsing options are visually more appealing, querying the FMA allows much more flexibility for different groups of domain experts to retrieve desired information in a wider range of scenarios, hence increasing usefulness of the FMA (as will be illustrated with the next query). Furthermore, the XML output obtained from OQAFMA easily can be ported to a GUI tool to make an appealing graphical visualisation such that one can have the best of both: powerful querying and an appealing interface for the domain expert.

```
WHERE
Blood->":NAME"->"Blood",
Blood->"part"+->BloodCellPart,
BloodCellPart->":DIRECT-SUPERCLASSES"+->Cell,
Cell->":NAME"->"Cell",
BloodCellPart->
   ":DIRECT-SUBCLASSES"*->BloodCellPartSubclasses,
BloodCellPartSubclasses->":NAME"->Result
CREATE
BloodShouldContain(Result)
```

Figure 2. STRUQL code for *Query I* to retrieve the parts of *Blood* that are cells.

Query II, retrieving the types of organ residing in the *Organ*-level that has as part a type of macrophage in the *Cell*-level, faces several problems. Fig. 4 depicts the STRUQL query and Fig. 5 the query answer. Note that it is not required to include the label with the type of macrophage in the result (the correct CREATE argument is `CREATE OrganWithMacrophages(OrganSubclassesName)`), which returns *Liver* only, but the additional information is included for illustrative purpose. Except for *Hepatic-Macrophage* in the *Liver*, discovered during querying with OQAFMA and *not* when browsing the FMA in Protégé, the FMA lacks *any* relation from other types of macrophages to other entities, even though in reality macrophages are located in/part of organs; we set aside a discussion on monocyte/macrophage [18, 14, 7] as a phased sortal. Regardless the avoidance of phased sortals in the FMA, *Query II* realises more advanced information retrieval compared to *Query I*: for each iteration, the query's evaluation strategy is to traverse the taxonomic structure for macrophages as well as the isA for organs, then it checks if there is an orthogonally positioned $partOf$ path between any of the entities in the two sub-trees.

It is worth noting that the FMA is at present one of the most comprehensive bio-ontologies, but it is incomplete both regarding the relations between entities and particularly concerning the *Cell*-level and below, although it is arguable if molecules belong in the subject domain (the present FMA cut-off 'level' is *BiologicalMacromolecule*). Positioning orthogonally the structural and functional anatomy granular perspectives is an outstanding task in its entirety. With this in place, one can query for e.g. the types of hormones residing in the kidney.

Figure 3. Difference Protégé/FMA browsing (check-boxes) and OQAFMA answer (in black) of *Query I*.

```
WHERE
Macrophage->":NAME"->"Macrophage",
Macrophage->":DIRECT-SUBCLASSES"+->MacrophageSubclasses,
MacrophageSubclasses->":NAME"->MacrophageSubclassesName,
Organ->":NAME"->"Organ",
Organ->":DIRECT-SUBCLASSES"+->OrganSubclasses,
OrganSubclasses->":NAME"->OrganSubclassesName,
OrganSubclasses->"part"+->MacrophageSubclasses
CREATE
OrganWithMacrophages(OrganSubclassesName,
                     MacrophageSubclassesName)
```

Figure 4. *Query II* **to retrieve organs that have as part a (subtype of)** *Macrophage***.**

```
<results>
<OrganWithMacrophages>
<MacrophageSubclassesName>Hepatic macrophage
    </MacrophageSubclassesName>
<OrganSubclassesName>Liver</OrganSubclassesName>
</OrganWithMacrophages>
<results>
```

Figure 5. OQAFMA result for *Query II***.**

3. Discussion

Two main issues are ontology development and diversification of usage of ontologies. In the past few years, much emphasis in (bio-)ontologies has been, and is being, put on ontological correctness of the modeled subject domain (e.g. [6, 15]), which is certainly still an important issue. However, *how* an ontology is to be made persistent for both offline and online use has received much less attention, possibly because it ought not matter, although in practice it does.

3.1. Engineering

Looking at alternatives for storing ontologies in databases, adoption of OWL and RDF as standard ontology languages has its advantages concerning data and ontology interoperability, and ontology integration is facilitated when they are written in the same language. Also, its reasoning support is a welcome addition to improve ontology development. On the other hand, expressivity of query languages for both OWL and RDF lags behind relational database query languages. For instance, OWL-QL 'merges' network protocol approaches with querying distributed OWL-based knowledge bases [3], which is useful for the Semantic Web, but this is not necessarily a main query aim of a bio-ontology. RDF query languages include SPARQL [28] (a W3C draft), RQL [26, 8] and SeRQL [27]. SPARQL focuses on sub-graph isomorphisms for query answering of RDF graphs in databases, but for recursive queries it relies on SQL (and e.g. STRUQL) and the DBMSs that support it. RQL supports "recursive traversal of class and property hierarchies (for advanced pattern-matching)" of trees with arbitrary depth [26]. SeRQL is a variation on RQL, and both return a bag as query answer [8], although with an additional `construct` one can rebuild the tree [27]. However, paths have to be specified with query formulation and RQL thus cannot 'discover' the path(s) between e.g. some macrophage and an organ. This considered, storing large ontologies in (relational) databases is a sensible option. It has an additional advantage when one desires to develop interoperable software and integrate databases, because querying over ontologies-stored-in-databases with data-databases is easier. But for this to realise, it is essential to have the ontology stored in a *good* database. A good database, as one can learn already in undergraduate courses in database development [13], has a meaningful conceptual model and ensures there is a separation of the database layer from the application layer in order to achieve relative implementation independence, thereby increasing reusability and simplifying maintenance of both. Unfortunately, FMA/Protégé meets neither. A workaround may be to generalise Mork et al's pre-processing operations [11], but this has to be carried out for each Protégé-database, therefore it will be more efficient and user-friendly if the Protégé software already takes care of this by modifying the development environment to map frames and slots to meaningful conceptual models. For instance, to create separate entities/relations in an ER or ORM model for each type of relation in the ontology and giving meaningful names to widely used attributes that end up in separate table columns, such as 'comment', 'language', 'user' and so forth. Aside from desirable optimization, an advantage is that then one can define database views for each granular level to simplify granular querying. For instance, the *Organ*-level will contain all taxonomic subtypes of *Organ*, and the tuples and values for the leaves in the taxonomic tree can be counted automatically and we would finally know how many organs we have.

A separate issue is the use of RDBMS. A common characteristic of ontologies are their hierarchical structures, be it taxonomies, partonomies, or e.g. a hierarchy of *developsFrom*. In this setting, recursive queries are necessary for useful information retrieval. SQL:1999 supports recursive queries, which has been gradually implemented in extant RDBMSs like Oracle, DB2, MS SQL, and in deductive databases. While MySQL is freely available, it lacks support for recursive queries, which may not be an issue when focusing on ontology creation and enlargement only, but is a serious limitation if one actually wants to use the ontology to go beyond finding an entity or retrieve an ambiguous

distance to other entities.

3.2. Diversification of use

The demand for recursive queries brings afore the second point: diversification of uses of ontologies. Ontologies are not just an inert repository, even if the ontology is a reference ontology: if it offers nothing more than an online encyclopaedia, then why put in the effort to build the ontology? An encyclopaedia cannot answer questions like which macrophages are part of some organ, or be able to anticipate other *ad hoc* peculiar questions like hormones located in the kidney or molecules in the mitochondrion. An ontology can, provided it is implemented in a usable format. Ontologies can – and should – have a variety of uses (e.g. [1, 2, 12, 17]), which has to be taken into account during the development stage. Top-level ontologies may not need computational support as they are intended to guide modelling decisions off-line, but developing ontology-mediated query formulation and information retrieval, such as utilising the encoded knowledge better through granular queries, requires run-time access to the ontology. At present, this is not possible with the GO and FMA/Protégé implementations, but is with the reengineered-FMA OQAFMA. With the current engineering limitations, offering an ontology in several storage formats (RDF, OWL, XML, etc.) to simplify its use and reuse to match the desired task can be a feasible approach, with the preconditions to have a clear versioning system in place and informing the user what is lost or gained with one storage format compared to the other. Applying this to the FMA case study, the gain one obtains with OQAFMA is that it has 'added' the semantics of the application layer to the database such that one can examine the information contained in the FMA more easily and comprehensively, experiment with e.g. applied domain granularity to explore its contents in a novel way, and enables wider use and reuse of the FMA to, for instance, link it with other ontologies at the subcellular levels or query medical data through this ontology. Aside from this, now having the opportunity to position orthogonally the taxonomy with the partonomy is useful for discovering lacunas and inconsistencies shown in the browsers and can be used for semi-automatic addition of the newly suggested relations.

Aside from information retrieval scenarios and sophisticated querying, the desire for and offers of 'reasoning support' for ontologies is underspecified: what exactly is available where and what type of reasoning does the user want/need? Automation for classification of a taxonomy, automated instance categorization, tree comparisons, test a hypothesised theory or relation for satisfiability, finding missing relationships, more? A *finer-grained goal specification* as part of the requirements analysis process will not only help deciding how one best can store the ontology with the presently available technology, but also provide an impetus for logicians and software developers to address and solve the most pressing users' needs first.

4. Conclusions

Current persistent storage implementations of large bio-ontologies fall short in meeting usability and reusability requirements. Some main limitations were demonstrated with two representative ontologies-stored-in-databases: the Gene Ontology and the Foundational Model of Anatomy. Improvements are possible with regards to ontology development environment software, taking into account general database development good practice, using RDBMSs that support expressive and recursive queries, and finer-grained specification of goals of what the ontology and system it may be part of are and may be intended for. We demonstrated the added-value of a variation of querying, granularity, which, by availing of extant database technology and implementation (OQAFMA), enables retrieval of more information quicker and can be used for improving and enlarging an ontology compared to the more common bio-ontology usage and implementations stored in relational databases. Current and future work involves improving ontology usage and developing granularity further to provide a reliable and reusable framework for granular information retrieval.

Acknowledgments
I thank Peter Mork, Sergio Tessaris, and Scott Marshall for their helpful comments on querying.

References

[1] Catarci, T., Dongilli, P., Di Mascio, T., Franconi, E., Santucci, G., Tessaris, S. An Ontology Based Visual Tool for Query Formulation Support. In: *Proceedings of ECAI 2004*, Valencia, Spain, August 2004.

[2] Daraselia, N., Egorov, S., Yazhuk, A., Novichkova, S., Yuryev, A., Mazo, I. Extracting Protein Function Information from MEDLINE Using a Full-Sentence Parser. *Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics*. 2004. pp11-18.

[3] Fikes, R., Hayes, P., Horrocks, I. OWL-QL – a language for deductive query answering on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2004, 2(1):19-29.

[4] Gene Ontology Consortium. The Gene Ontology (GO) project in 2006. *Nucleic Acids Research*, 2006, 34:D322-D326.

[5] Guarino, N. Formal Ontology and Information Systems. *Proceedings of Formal Ontology in Information Systems (FOIS'98)*. Amsterdam: IOS Press, 1998.

[6] Guarino, N., Welty, C.A. An overview of OntoClean. In: *Handbook on ontologies*. Stab, S., Studer, R. (eds.). Springer Verlag, 2004. pp151-159.

[7] Hoffbrand, A.V., Pettit, J.E. *Atlas of Clinical Heamatology*. 3rd ed. Elsevier/Mosby, 2000.

[8] Karvounarakis, G., Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M., Tolle, K. RQL: A Functional Query Language for RDF. In: *The Functional Approach to Data Management: Modelling, Analyzing and Integrating Heterogeneous Data*, Gray, P.M.D., Kerschberg, L., King, P.J.H., Poulovassilis, A. (eds.), LNCS, Springer-Verlag, 2004.

[9] Keet, C.M. A taxonomy of types of granularity. *IEEE Conference on Granular Computing (GrC2006)*, 10-12 May 2006, Atlanta, USA.

[10] Kumar, A., Smith, B., Novotny, D.D. Biomedical Informatics and Granularity. *Comparative and Functional Genomics*, 2005, 5(6-7): 501-508.

[11] Mork, P., Brinkley, J.F., Rosse, C. OQAFMA Query Agent for the Foundational Model of Anatomy: a prototype for providing flexible and efficient access to large semantic networks. *Journal of Biomedical Informatics*, 2003, 36: 501-517.

[12] Mungall, C.J. Obol: integrating language and meaning in bio-ontologies. *Comparative and Functional Genomics*, 2004, 5(6-7):509-520.

[13] Newton, M., Armstrong, S., Kwee Brown, J., Cooke, I., Field, M., Findlay, J., Lewis, S., Taylor, J., Yazdanjoo, K. *M358: Relational Databases*. Open University UK. 2001.

[14] Scheuerer, B., Ernst, M., Drrbaum-Landmann, I., Fleischer, J., Grage-Griebenow, E., Brandt, E., Flad H.-D., Petersen, F. The CXC-chemokine platelet factor 4 promotes monocyte survival and induces monocyte differentiation into macrophages. *Blood*, 2000, 95(4):1158-1166.

[15] Smith, B., Ceusters, W., Klagges, B., Kohler, J., Kumar, A., Lomax, J., Mungall, C.J., Neuhaus, F., Rector, A., Rosse, C. Relations in Biomedical Ontologies. *Genome Biology*, 2005, 6:R46.

[16] Tange, H.J., Schouten, H.C, Kester, A.D.M. and Hasman, A. The Granularity of Medical Narratives and Its Effect on the Speed and Completeness of Information Retrieval. *Journal of the American Medical Informatics Association*, 1998, 5(6): 571-582.

[17] Wolstencroft, K., Lord, P., Tabernero, L., Brass, A., Stevens, R. Using ontology reasoning to classify protein phosphatases [abstract]. *8th Annual Bio-Ontologies Meeting 2005*, 24 June 2005, Detroit, USA.

[18] Yang, J.-H., Sakamoto, H., Xu E.C., Lee, R.T. Biomechanical Regulation of Human Monocyte/Macrophage Molecular Function. *Am. J. of Pathology*, 2000, 156:1797-1804.

[19] Zukerman, I., Albrecht, D., Nicholson, A., Doktor, K. Trading off granularity against complexity in predictive models for complex domains. In: *Proceedings of the 6th International Pacific Rim Conference on Artificial Intelligence (PRCAI 2000)*.

[20] AmiGO. http://www.godatabase.org/cgi-bin/amigo/go.cgi.

[21] DAG-Edit. http://www.godatabase.org/dev/.

[22] Foundational Model of Anatomy (FMA). 2003. http://fme.biostr.washington.edu:8089/FME/index.html.

[23] Gene Ontology Consortium. http://www.geneontology.org.

[24] Gene Ontology – optimisations. http://www.godatabase.org/dev/sql/modules/go-optimisations.sql.

[25] Ontology Query Agent for the Foundational Model of Anatomy. http://sig.biostr.washington.edu/projects/oqafma/.

[26] RQL. http://139.91.183.30:9090/RDF/RQL/.

[27] Sesame RQL. http://www.openrdf.org/doc/rql-tutorial.html.

[28] SPARQL. http://www.w3.org/TR/rdf-sparql-query/.