Speed-up of a Knowledge-Based Clinical Diagnosis System using Reflexive Ontologies

Arkaitz Artetxe[1,4], Eider Sanchez[1,4], Carlos Toro[1], Cesar Sanin[2], Edward Szczerbicki[2], Manuel Graña[3], Jorge Posada[1]

1 Vicomtech-IK4 Research Centre, Mikeletegi Pasealekua 57, 20009 San Sebastian, Spain
{aartetxe, esanchez, ctoro}@vicomtech.org
2 Faculty of Engineering and Built Environment, University of Newcastle, Australia
3 University of the Basque Country (UPV/EHU), San Sebastian, Spain
4 Biodonostia Health Research Institute, eHealth Group, Bioengineering Area, Spain

Abstract. Most of the time expended during ontology processing is derived from query actions. Quasi-real time goals require new approaches towards time efficiency (e.g. an intensively consumed application that pulls knowledge from an ontology). Reflexive Ontologies are a recent approach that is intended to bridge some of the aforementioned time consumption issues.

In this paper we present an implementation of the Reflexive Ontologies in a knowledge-based Clinical Decision Support System for the diagnosis of Alzheimer's Disease. Our implementation is evaluated in order to show the impact of the application of reflexivity in the context described above.

We give implementation details, as well as the definition of the evaluation methodology and evaluation results. Lastly performance improvements and some highlights of the application are also discussed.

Keywords: Knowledge Engineering, Reflexive Ontologies, Evaluation, Knowledge-Based System

1 Introduction

A large number of real world applications benefit from the use of ontologies [1]. Gruber [2] defined ontologies in the computer science domain as the explicit specification of a conceptualization.

The capability of ontologies to model a specific domain is advantaged by many applications for the discovery of explicit and implicit knowledge about the domain. This is usually achieved by performing queries and processing their results using reasoners and producing rules.

Performing such queries over the knowledge base may be a time consuming process. For the aforesaid reason, the development of techniques that may help to improve the performance of traditional ontology use and processing meets a clear need.

Ontology processing could be a time consuming task that implies a high computational cost. Reasoning over ontologies is based on query actions which arguably can

be considered computationally expensive, when compared for instance, to relational databases.

Most of the time expended during this process is due to query actions which impose important delays on systems. Thus, quasi real time requirements should be addressed from new perspectives.

In this context, the Reflexive Ontologies (RO) concept was introduced by Toro *et Al.* [3] as a technique that can be used to add self contained queries to an ontology. In that paper some improvements derived from the use of this technique were presented, being one of them the *speeding of the query process*.

The acceleration of the querying process is based on the hypothesis that queries tend to recur over time. In RO the knowledge model, individuals and queries are contained in the ontology. When a new query is made, reflexivity will gather the knowledge first from the already present answers and then, if it is not contained completely or partially, a classical query will be made. This approach reduces data access latency, in a similar way to how database caching does with data models [4].

In this paper we present an implementation of a knowledge-based Clinical Decision Support System (CDSS) for the diagnosis of Alzheimer Disease, enhanced with Reflexivity. The implemented system is a benchmarking environment where the performance of the Reflexive Ontologies will be evaluated. There, the performance differences between a system based on a traditional ontology and a system based on a Reflexive Ontology are observed. The goal of this evaluation is to determine the reliability of the system and the effectiveness of the proposed optimizations, with special focus on the querying execution times.

The paper is organized as follows: in Section two we present some background concepts related to Reflexive Ontologies and Autopoiesis. In Section three we describe the implemented system. In Section four we present the evaluation methodology and the results obtained for the implemented system. In Section five we discuss conclusions and future work.

2 Related work

Previous work has been reported recently in the field of query answering performance improvement. Kollia *et Al.* introduced in [5] optimization techniques that improve query answering performance for SPARQL-OWL queries. One of the optimizations presented in their paper consists in utilizing precomputed information (e.g. the class hierarchy) in order to find the answer of a query simply with a cache lookup. This technique, along with some other optimizations such as the axiom reordering, help improve query answering performance.

Our approach is similar to the one presented by Kollia *et Al.* in the sense that both benefit from previously computed information in order to perform a cache-like access to the query answer. However, our approach goes further in the sense that RO keeps a track of all of the queries made over the ontology instead of using some precomputation made by the reasoner.

Amir *et Al.* introduced in [6] an approach known as *partition-based logical reasoning* which argues to improve the efficiency of the reasoning process. Algorithms for reasoning with partitions of related logical axioms were presented in their work. In [7] Grau *et Al.* proposed the concept of partitioning an OWL ontology in sub-domains (modelled as separate ontologies) using e-Connections to combine them. This approach was thought to reduce the Knowledge Base portion that the reasoner has to work with, by keeping irrelevant components of the ontology unloaded. These techniques are based on the idea of reducing the search-space within the knowledge base in order to improve the reasoning efficiency. Our work tackles the reasoning time issue from a different approach that is based on query caching rather than ontology partitioning.

Cobos *et Al.* proposed in [8] an architecture which uses the Reflexivity concept in order to perform a fast semantic retrieval in the Film Heritage domain. The results of the experiment showed a clear efficiency gain, with an improvement of two orders of magnitude in the execution time. Although the concept of using RO for a fast query recovery is the same for both cases, the architecture and implementation differ in some extent from our approach.

The experiment by Cobos *et Al.* was carried out using only simple queries (containing a simple condition clause) and the ontology they used within the experiment included 63 individuals. In this paper we test the Reflexive Ontologies concept in a more complex environment, since we use complex queries and our domain ontology contains more than 10,000 individuals. In addition, our system handles a non-static ontology, i.e. an ontology that grows over time, while the system by Cobos *et Al.* works with a static ontology. The implementation of the Autopoiesis concept makes possible the use of non-static Reflexive Ontologies.

2.1 Reflexive Ontologies

The Reflexive Ontologies (RO) concept was introduced by Toro *et Al.* [3] to define the capability of an abstract structure of knowledge (an ontology and its instances, in this case) of maintaining, in a persistent manner, every query performed on it, and store those queries as individuals of a class that extends the original ontology.

Formally defined "*a Reflexive Ontology is a description of the concepts, and the relations of such concepts in a specific domain, enhanced by an explicit self contained set of queries over the instances*" [3].

Figure 1 shows the logical structure of a RO, which is, basically, a traditional ontology extended with a reflexive structure (mainly composed by the query instances in the left part of the image) (see [3] for further details).

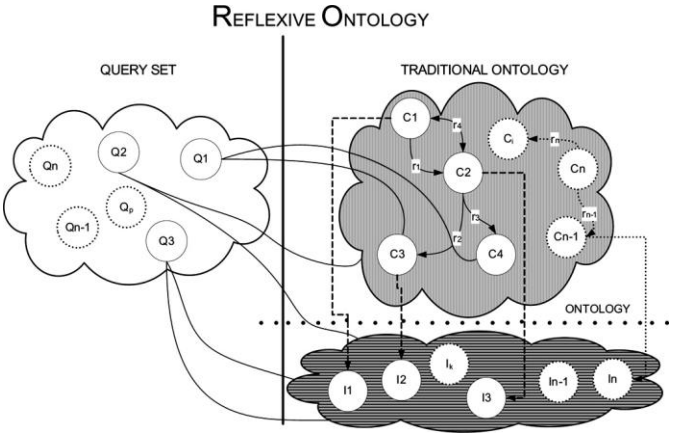


Fig. 1. Schematic representation of the structure of a RO [3]

2.2 Autopoiesis

The etymology of autopoiesis is "self-creation or self-production". The concept was originally introduced by biologists Maturana and Varela [9] to describe a system that is capable of modifying, creating and destroying components of the system itself according to external perturbations. Reflexive Ontologies have an autopoietic behaviour as long as its structure is self generated and grows with every new query launched. RO are capable of storing the history of performed queries. The autopoietic behaviour ensures the integrity of the RO. When a new individual is created, modified or removed from the ontology, the reflexive structure is updated. The updating process consists of modifying or generating new references to individuals for each query instance related to the change.

3 Testing system implementation

Our testing system is a Clinical Decision Support System (CDSS) for the diagnosis of Alzheimer's Disease (AD). It has been implemented within the scope of a large scale Spanish research project[1] aiming the early diagnosis of AD.

The CDSS implemented is a collaborative and multidisciplinary tool where physicians can *(i)* introduce patient data and the results from the clinical tests carried out to them, *(ii)* review and edit this data at any time and location, and *(iii)* get support from the system to assist them during decision making about diagnosis of AD.

This system consists of three different modules, as presented by Sanchez *et Al.* in [10]: *a)* the ontologies module, *b)* the reasoning module and *c)* the query system. Briefly, each of these modules is described below.

[1] https://www.portalmind.es/

3.1 Domain ontology

Our ontology module, presented in [11], consists of a domain ontology defined by experts for the specific domain of the AD diagnostic system. Our domain ontology provides a description of the different clinical tests carried out to patients in order to detect the AD. This ontology was implemented in OWL-DL and is mapped to SNOMED-CT and SWAN in order to provide, respectively, standardized terminology and bibliographic endorsement of the knowledge and criteria embedded.

3.2 Reasoning module

The reasoning module performs a semantic reasoning process based on a set of rules given by clinicians. The rule set of our system consists of 138 weighted rules endorsed by their corresponding bibliographic sources.

A rule is composed by at least two clauses: the clause corresponding to *if* part and the one corresponding to *then* part, while a third one, *else*, is optional. A rule is said to be simple if contains a unique clause or complex if more than one clause are present. Connectors are logical operators (AND, OR and NOT) used to build more refined clauses. Every clause is formed by four elements: *i)* Class, *ii)* Property, *iii)* Modifier and *iv)*Value. Figure 2 depicts the structure of a clause along with a simple example.

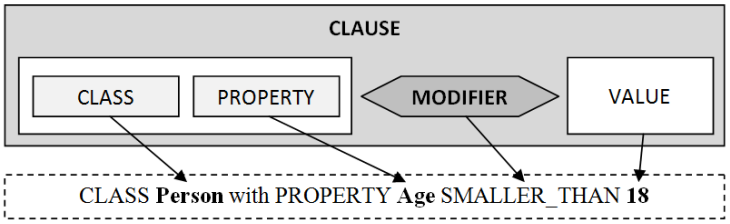


Fig. 2. Structure of a clause

The first two elements are used to identify the concept of the knowledge base which the clause refers to. The modifier is used to carry out the comparisons between the value in the knowledge base and the value specified by the rule. A modifier may be either SMALLER_THAN, GREATER_THAN or EQUALS_TO.

Our system uses the expert knowledge contained in the rules to infer a diagnosis for a certain patient. In order to do so, the system must launch the necessary queries and check whether the conditions stated by the rules are fulfilled for that patient.

3.3 Query system

The implementation of the query system supports logical operators, according to the fourth property stated by C. Toro *et Al.* in [3]. Logical operators (AND, OR, and NOT) provide a way to combine simple queries and construct complex queries using Boolean Logic.

When a complex query is made over the ontology, the system splits the query into simple queries. The answers of the simple or atomic queries are retrieved and the answer of the complex query is inferred.

Figure 3 illustrates the query process that takes place in a Reflexive Ontology-based system.

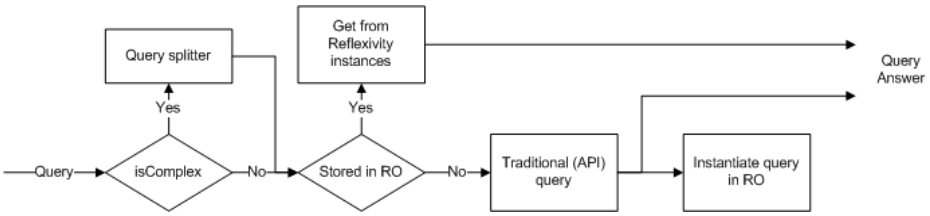


Fig. 3. Query process in Reflexive Ontologies

When a query is launched over the ontology, the system checks its complexity. If a complex query is made, the system splits the query into simple queries using a query parser. Then the system will search for simple queries along with the original (complex) query through reflexivity instances in order to check whether the queries have been made previously. If the system finds a reflexive instance for a certain query (or a similar one if syntactic similarity is used), its answer is retrieved directly from that instance. When the query is not present in the ontology, a traditional query is made via a JAVA compliant API. In addition, the query is instantiated in the reflexive structure, storing its answer within that instance.

A complex query formed by three simple queries will be:

$$Q_c = Q_1 \text{ op } Q_2 \text{ op } Q_3 \text{ , where op=\{AND, OR, NOT\}}$$

At the end of the querying process for Q_c , the reflexive structure will store four query instances: one for each simple query composing the query (Q_1, Q_2 and Q_3) and another one for the original query itself (Q_c).

4 Evaluation

4.1 Methodology and test environment

During evaluation the execution time of the diagnosis process was measured for some different patients. The objective was to observe the execution time differences between an ontology enhanced with reflexivity and a conventional one. In addition, measuring the execution time of every sub-task of the diagnosis process brings valuable information that may be useful to make the code more efficient.

The system was implemented using three different rule sets as shown in Table 1. For each of them the number of rules is shown as well as the number of queries that are instantiated within the RO.

Table 1. Size and characteristics of the rule sets

Rule set	Number of Rules	Generating query instances
RuleSet1	35	72
RuleSet2	103	246
RuleSet3	138	291

For every different rule set used, the test system is configured in two different manners: i) reflexivity-enhanced and ii) no reflexivity-enhanced.

Execution time is gathered using a process similar to the one used in Knowledge Base System benchmarking [12,13,14]. Our evaluation process differs from other approaches in the fact that our measures are not restricted to individual queries, but consider the entire diagnosis process. This fact, however, does not detract validity to our evaluation, because the diagnosis process can be assimilated to a series of queries.

Execution time is measured using built-in Java methods with every measurement being the average of 10 independent executions. For testing purposes, we have selected 10 patients (individuals) which are subjected to evaluation in each one of the configurations defined.

We have performed the experiment in a desktop computer with Intel Core 2 Quad CPU Q8300 at 2.5 GHz x 4, 2.9 GB RAM and Ubuntu 11.10 64-bit. The system was developed and evaluated in Eclipse 3.7.0 with JDK version 1.6.0. Protégé [15] API was used for ontology access and management.

4.2 Data and analysis

Table 2 is a complete list of the test results (execution time is shown in milliseconds). For every patient the system's performance is measured using both the classical approach of using a query system and Reflexive Ontologies. Both configurations are fed with the three rule sets shown in Table 1.

Table 2. Execution times in milliseconds

		Patient									
		1	2	3	4	5	6	7	8	9	10
RuleSet 1	RO	825	809	771	858	972	772	827	878	767	785
	no RO	2715	2688	2788	2752	2729	2705	2726	2720	2781	2802
RuleSet 2	RO	1813	1888	1882	1923	1932	1869	1831	1989	1976	1886
	no RO	4139	4107	4100	4165	4207	4166	4097	4130	4181	4213
RuleSet 3	RO	2541	2522	2488	2538	2553	2637	2615	2524	2436	2537
	no RO	6343	6329	6290	6298	6281	6229	6238	6241	6240	6146

Table 2 shows that, using the same rule set, the execution times vary slightly from one patient to the other. This behavior is constant with all rule sets using both conventional ontologies and RO. This is caused by the fact that the number of queries to be per-

formed at diagnosis time is determined by the complexity of each rule in the rule set. The rule set remains constant for every patient, thus the queries to perform are equal for the whole group of patients. A preliminary analysis suggests that small variations are caused due to differences in the number of clinical test instances between patients.

Aside from those differences, the reduction of the execution time needed to perform the diagnosis is evident when it comes to RO. It is better shown in Figure 4, where the average execution times are compared.

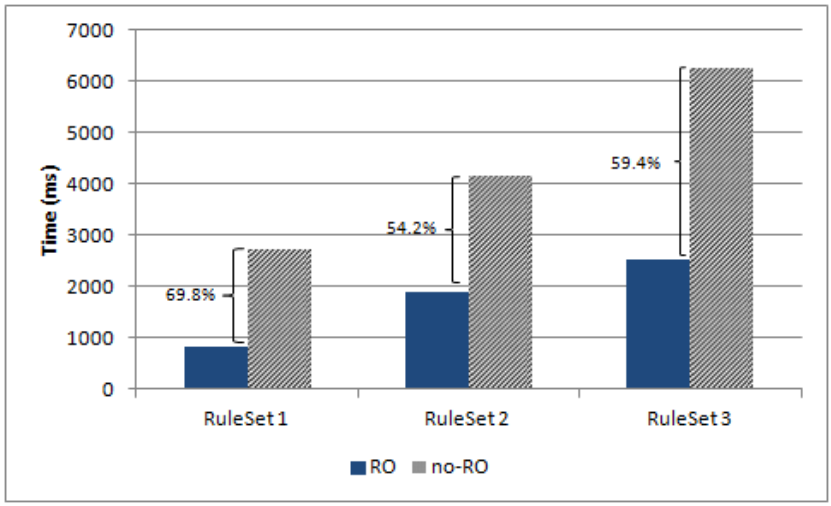
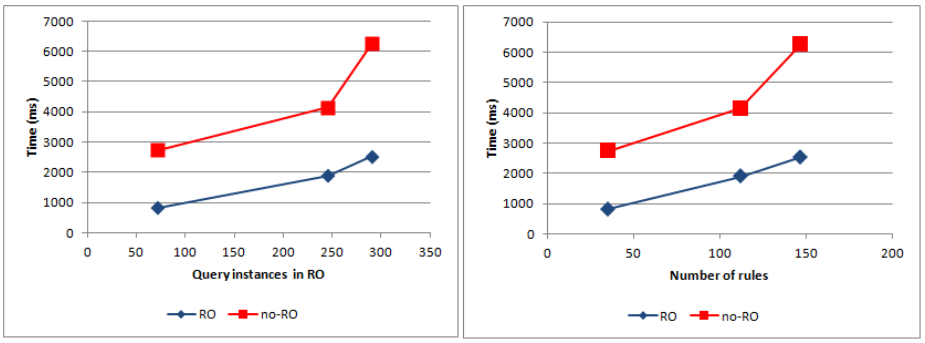


Fig. 4. Comparison of average execution times (bracketed values show RO improvement)

Figure 4 shows the execution times for the three rule sets, calculating the time from the average of the ten patients under consideration. The results show that the use of RO significantly improves execution times (69.8% for RuleSet1, 54.2% for RuleSet2 and 59.4% for RuleSet3).



(a) (b)

Fig. 5. (a) Execution time in relation to the number of query instances involved. (b) Execution time in relation to the number of rules in the rule set

Figure 5a shows the evolution of the execution times in relation to the number of query instances stored in the ontology. This factor has a major impact on the RO run-time as the number of instances in the ontology determines the time required to search the result for a given query. The chart shows that the execution times for RO grow almost linearly since it draws a slightly pronounced curve. However, we hypothesize that the curve drawn by the execution times of a conventional ontology is exponential. Nevertheless, this needs to be proven in future work. This statement means that RO is more robust in terms of scalability, since the execution time is proportional to the number of query instances stored in the ontology.

Figure 5b shows the evolution of the execution times in relation to the number of rules in the rule set. The trend is similar to that presented in Figure 5a, although in this case the evolution of the execution times using reflectivity is clearly linear. However, the number of rules may not be as significant as the number of clauses they contain, since the complexity of the rules (in terms of number of clauses) may be very varied. In other words, a rule that consists of 10 clauses counts the same as a rule of only two clauses, when the computational cost of their processing is clearly uneven.

In Figure 5a and 5b the reduction of the execution time is fairly pronounced when the use of reflexivity is on. More precisely, when reflexivity is off, Figure 5b shows an increase of more than 2000 milliseconds (that is, up to 50% more of time) when the size of the rule set grows just 35 rules. That reveals some scalability problems that may appear if larger rule sets are used. However, it is beyond the scope of this paper to determine which factors are causing this behaviour.

5 Conclusion and future work

In this paper we have presented an implementation of the Reflexive Ontologies in a knowledge-based Clinical Decision Support System for the diagnosis of Alzheimer's Disease. In the context of the implemented system, we have presented a preliminary evaluation of the RO were the impact of the reflexivity is shown.

This evaluation suggests that, in the worst case scenario, Reflexive Ontologies perform comparably to traditional ontologies. It also shows that in our diagnosis system, the use of RO significantly improves efficiency, reducing the execution time in nearly 70%.

As future work, we see the need to extrapolate the results in order to identify the theoretical efficiency roof of RO, i.e. the point where the cost of using RO equals the cost of an ontology not enhanced with reflexivity.

Additionally, we plan to extend this work in order to evaluate RO in a wider benchmarking process. In this benchmark we want to measure the impact (in terms of computational cost) of Autopoiesis [9], so that the overall cost of using RO can be shown.

References

1. McGuinness. D.L., "Ontologies Come of Age". In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press (2003)
2. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. In: International Journal of Human-Computer Studies 43(5–6), pp. 907–928 (1995)
3. Toro, C., Sanín, C., Szczerbicki, E., Posada, J.: Reflexive Ontologies: Enhancing Ontologies with self-contained queries. In: Cybernetics and Systems: An International Journal 39, 171-189 (2008)
4. Tolia, N., Satyanarayanan, M.: Consistency-preserving caching of dynamic database content. In: Proceedings of the 16th international conference on World Wide Web, pp. 311-320. ACM (2007)
5. Kollia, I., Glimm, B., Horrocks, I.: SPARQL Query Answering over OWL Ontologies. In: Proc. 8th Extended Semantics Web Conf. (ESWC' 11)(2011)
6. Amir, E., McIlraith, S.: Partition-based logical reasoning for first-order and propositional theories. In: Artificial Intelligence, vol. 162, pp. 49-88 (2005)
7. Grau, B. C., Parsia B., Sirin, E., Kalyanpur, A.: Automatic Partitioning of OWL Ontologies Using E-Connections. In: International Workshop on Description Logics (DL2005) (2005)
8. Cobos, Y., Toro, C., Sarasua C., Vaquero J. Linaza M., Posada J.: An Architecture for Fast Semantic Retrieval in the Film Heritage Domain. In: 6th International Workshop on Content-Based Multimedia Indexing (CBMI), pp. 272-279 (2008)
9. Maturana, H., Varela, F.: Autopoiesis and Cognition: The realization of the living. D. Reidel (1980)
10. Sanchez, E., Toro, C., Carrasco, E., Bueno, G., Parra, C., Bonachela, P., Graña, M., Guijarro, F.: An Architecture for the Semantic Enhancement of Clinical Decision Support Systems. In: Knowlege-Based and Intelligent Information and Engineering Systems, vol. 6882, pp. 611-620. Springer (2011)
11. Sanchez, E., Toro, C., Carrasco, E., Bonachela, P., Parra, C., Bueno, G., Guijarro, F.: A Knowledge-based Clinical Decision Support System for the diagnosis of Alzheimer Disease. In: 2011 IEEE 13th International Conference on e-Health Networking, Applications and Services (Healthcom 2011), pp. 355-361. (2011)
12. Guo, Y., Pan, Z., Heflin, J.: An evaluation of knowledge base systems for large OWL datasets. In: International Semantic Web Conference, pp 274-288. Springer (2004)
13. Bock, J., Haase, P., Ji, Q., Volz, R.: Benchmarking OWL Reasoners. Landscape, vol. 350. Citeseer (2008)
14. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., & Liu, S.: Towards a Complete OWL Ontology Benchmark. In: The Semantic Web Research and Applications, vol. 4011, pp. 125-139. Springer (2006)
15. Protégé 2007. Using the Protégé-OWL Reasoner API. Available at: http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html (accessed may 05, 2012)