Evaluating OWL 2 Reasoners in the Context Of Checking Entity-Relationship Diagrams During Software Development

Alexander A. Kropotin

Department of Economic Informatics, Leuphana University of Lüneburg, Lüneburg, Germany

Abstract. This paper evaluates the performances of the OWL 2 reasoners HermiT, FaCT++ and TReasoner in the context of an ontological decision support system in designing entity-relationship diagrams during software development. First, I described a developed ontology which is the knowledge base of the developed application for designing databases. In the first set of experiments I compared how the classification and realization time of the DBOM ontology varied when increasing the ABox with ERD elements individuals. In the second set of experiments the consistency checking time of the DBOM ontology was estimated by increasing the ABox with ERD elements individuals.

Keywords: Benchmark, Description Logics, FaCT++, HermiT, Ontology, OWL 2 reasoners, TReasoner

1 Introduction

In the software development process special attention is paid to the databases designing stage. Working capacity and extensibility of the developed system depend on the operations performed at this stage. This process consists of creation and updating of an information model according to the levels and rules of the designing process [5]. Nonobservance of the designing process rules or making mistakes can lead to the situation when the developed software won't allow upgrade and extension, won't support all enterprise business logic etc.

As a tool for the design of database diagrams, the entity-relationship model (ERM) can be considered as generalization and extension of existing data models (network, relational, information, etc.), which allow to describe all completeness of the relations between database diagrams elements at different designing levels [3,4]. Besides it is quite probable to express some knowledge bases (KB) of an ontology described in Web Ontology Language (OWL) in this model [2].

That expressive likenesses of description logics (DL) and ERM are the main arguments in favor of the hypothesis of that it is probable to express some information domain model. The model was created in the form of semantic network, in the form of DL formalism and to check for creation rules consistency of not only ERM, but also of databases designing rules in different notations.

2 The DBOM ontology

The *Database OWL Model* (DBOM) is developed as a knowledge base of the developed application for the automatization of the design process of databases diagrams. It is planned that this application will allow to detect semantic and syntax mistakes in databases diagrams, which were designed in different notations and data models, and also to convert databases diagrams from one notation and/or a data model into another.

At present, this ontology describes a meta model of P. Chen ERM [3] in terminological box (TBox), including the assertions about the notation elements and rules within a frame of the relational data model. In other words, the DBOM ontology is the pattern for the ERD formalization. It means that each new ERD should be described in assertional box (ABox) of DBOM. Table 1 provides a DBOM in terms of number of entities, individuals, axioms and expressivity.

Table 1. Summary of the DBOM ontology metrics

Metric	Value
DL Expressivity	SROIQ (D)
# Classes	25
# Object properties	19
# Data properties	11
#Individuals	3
#Axioms	295
#Logical axioms	239

It is worth noting that the metrics were taken from ontology DBOM in a general view, without describing of ERD elements individuals in ABox.

The main idea of ERD validation by reasoner consists in ontology consistency checking [6,7,8]. Thus, if ERD was designed with mistakes, its interpretation in DBOM will have consistencies of ABox assertions to TBox assertions. However, it works only for detection of semantic and syntax of ERD designing mistakes.

Therefore, for detection of *live lock* mistakes, which can be detected by methods of simulation modeling, I use the DL transitive property. The idea is in describing in ABox the transitive object property *hasCycle* which will be as the super property for object property *one-to-many* in ERD [10]. During the process of describing ERD elements individuals in ABox of DBOM it is necessary to describe the negative object property *hasCycle* for each individual. This negative object property will assert that object property *hasCycle* can't be reflexive for this individual. Thus, an ABox will be completed by transitive object property *hasCycle* during the process of the *tableau algorithm* [1] running. And if in the ABox there is an individual, from a set of individuals which form the live lock mistake, then the reflexive object property *hasCycle* surely will be described in this individual. And that will be the identifier of existence of *live lock* mistake.

3 Evaluation

I evaluated the scalability of the DBOM ontology by OWL 2 reasoners: HermiT 1.3.8 [12], FaCT++ 1.6.2 [14] and TReasoner [11]. These OWL 2 reasoners are winners of the OWL reasoner evaluation workshop ORE 2013 [9]. The tests were performed on a Windows 7 64-bit desktop computer with 8 GB of RAM and an Intel Core i7-3770S 3.10 GHz CPU. The following JVM arguments were used*: java -Xms500M -Xmx4400M -DentityExpansionLimit=100000000*.

As the developed the DBOM ontology has to provide the possibility of a logical output about consistency ERD to design rules and classification of each ERD element, the evaluation of two inference services of the "standard" set of DL inference services was estimated: realization and consistency checking [13].

As test data, I designed ERD consisting of 20 elements: 5 entities, 4 relationships and 11 attributes first. Then, during making evaluation experiments, I added 419 ERD elements to the DBOM ontology incrementally. On each grown increment I formalized created ERD in DBOM. After that, I created a relationship between one entity type individual of formalized ERD and other entity type individual of the DBOM ontology, which were selected in a random way. Based on the assumption that is in case of database extension it can be concluded that there is at least one new entity which will be connected with at least one entity of an expanded database. I also supposed that 419 ERD elements is enough for the simulation of a statistically average ERD diagram.

All in all, I represent two sets of experimental results that are given below. Note that all results reported in this paper were acquired as averages of at least 10 repetitions of the described experimental setup.

3.1 Evaluating realization of the DBOM ontology with ERD elements increasing

In the first set of experiments I compared how the classification and realization time of the DBOM ontology varied when increasing the ABox with ERD elements individuals. That end, I recorded the time taken by each reasoner to perform classification first (i.e. execution of the method *precomputeInferences(CLASS_HIERARCHY)*) and after realization of each ERD elements individual (i.e. execution of the method *getTypes()*). Realization can be performed only after classification since direct types are defined with respect to a class hierarchy [13]. Figure 1 summarizes the realization times of the DBOM ontology that were obtained for HermiT and FaCT++. As TReasoner does not provide realization methods yet and realization can be executed only after classification, I also recorded the time taken by each reasoner to perform classification. Figure 2 summarizes the classification times of the DBOM ontology obtained for all three reasoners.

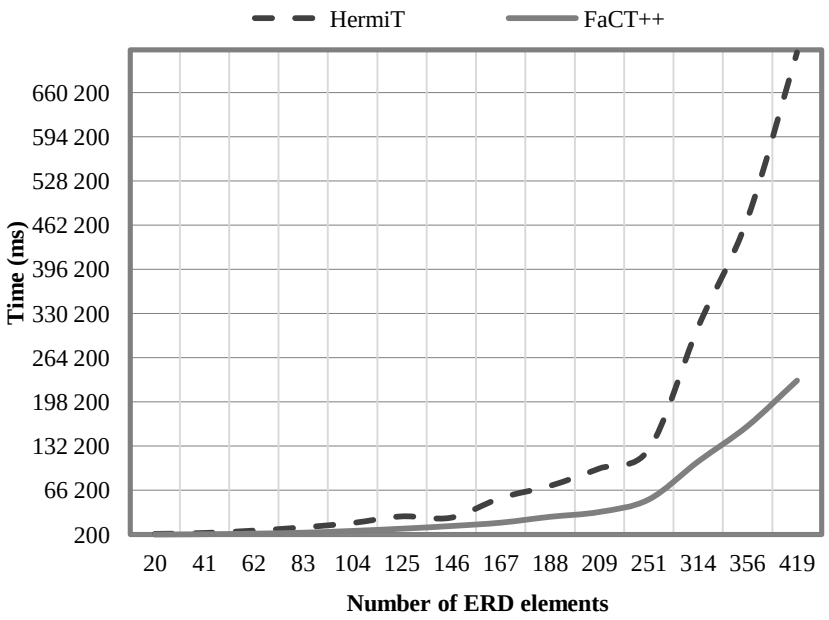

Fig. 1. Realization times of the DBOM with 20 to 419 ERD elements

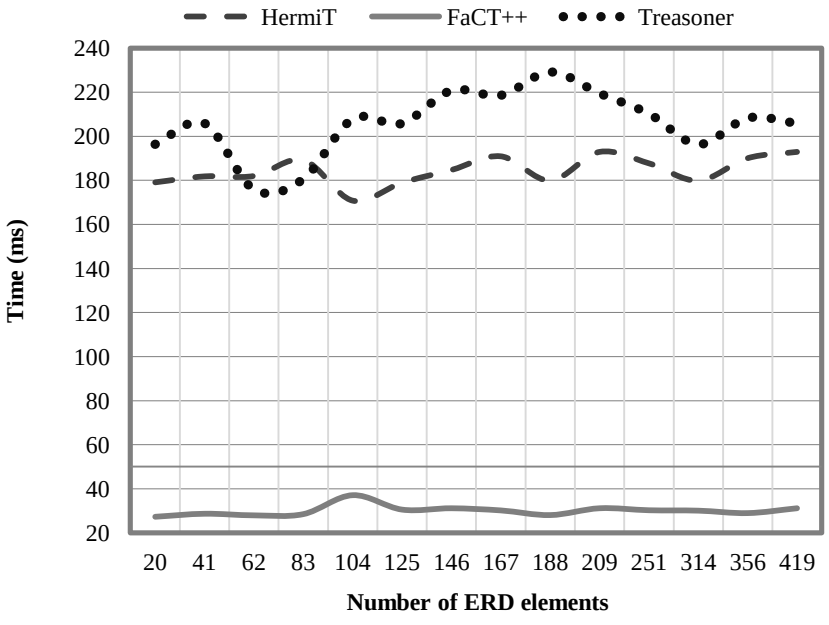

Fig. 2. Classification times of the DBOM with 20 to 419 ERD elements

As expected, the realization time increases as ERD elements individuals are added to the ontology. It's noticeable that HermiT and FaCT++ have similar behaviors. At first, time of realization with respect to the number of individuals in the ontology increases rather gradually, but then it increases very quickly and clearly with a non-linear fashion for both reasoners (see Figure 1). This behavior seems like an exponential sequence. And the HermiT realization time is considerably slower, it increases more quickly in comparison with FaCT++.

As can be seen, HermiT and TReasoner have similar behavior and insignificant time difference. Arithmetic average value of TReasoner classify runtime is 22 milliseconds less, than HermiT has (see Figure 2). FaCT++ has the smallest time again and it is 143 milliseconds faster than the others. In addition, the classification time of the DBOM ontology by all three reasoners does not depend on the increase in the number of ERD elements individuals in ABox to 419.

3.2 Evaluating consistency checking of the DBOM ontology with ERD elements increasing

In the second set of experiments the consistency checking time of the DBOM ontology was estimated by increasing the ABox with ERD elements individuals. That end, I recorded the time taken by each reasoner to perform consistency checking (i.e. execution of the method *isConsistent()*) of the DBOM ontology. Figure 3 summarizes the consistency checking times of the DBOM ontology obtained for all four reasoners.

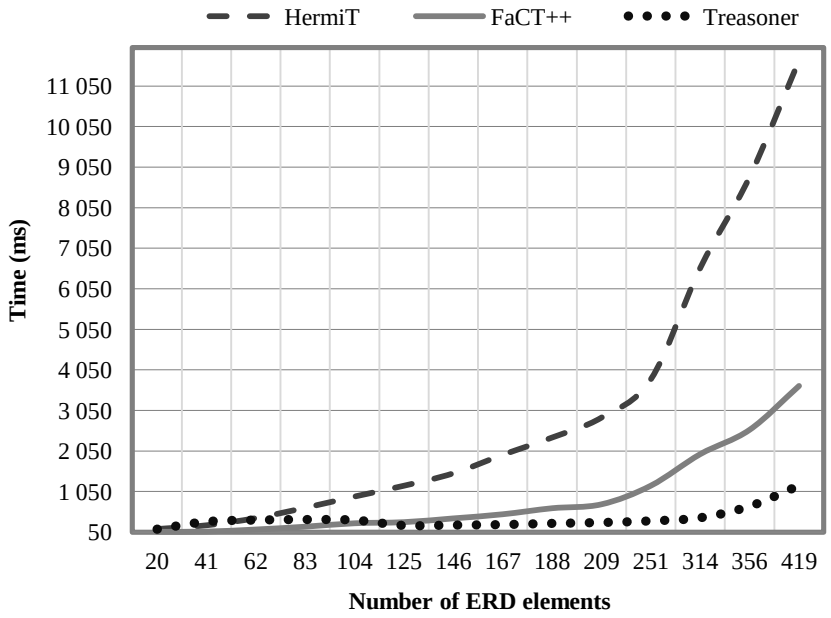

Fig. 3. Consistency checking times of the DBOM with 20 to 419 ERD elements

In this case TReasoner has the smallest time of consistency checking (see Figure 3). And it's noticeable that HermiT and FaCT++ have similar behaviors again, but it differs from TReasoner very much. While the increasing time of both reasoners is gradual at first then it is very quick, the TReasoner increasing time is more gradual in comparison with HermiT and FaCT++, but also non-linearly. Although the consistency checking time achieved by HermiT is the longest, the consistency checking time by FaCT++ is also considerably slower compared with TReasoner.

4 Conclusions

In this paper I evaluated empirically the realization and consistency checking performances of the DBOM ontology by three OWL 2 reasoners: HermiT, FaCT++ and TReasoner. They are also winners of the OWL reasoner evaluation workshop ORE 2013 [9].

I found out that FaCT++ is the best choice for my application since it provides very fast inference time for realization and middle interface time for consistency checking. Though TReasoner provides very fast inference time for consistency checking the DBOM ontology, it also does not provide realization methods. The best solution is to combine both reasoners : TReasoner for consistency checking and FaCT++ for realization. HermiT has the last position in realization and consistency checking, but it is faster than TReasoner in the classification of the DBOM ontology.

It is noteworthy that HermiT and FaCT++ have similar behaviors in realization and consistency checking of the DBOM ontology, but they both have very disparate behaviors in comparison with TReasoner. HermiT and TReasoner have similar behavior and insignificant time difference in classification of the DBOM ontology. And classification times of the DBOM ontology by all three reasoners does not depend on the increase in the number of ERD elements individuals in ABox to 419.

Also excellent productivity of HermiT, TReasoner and FaCT++ showed that the DBOM ontology is a good decision for application oriented tasks for verification of ERD since it is capable to provide acceptable speed of consistency checking operation of the DBOM ontology.

References

1. Baader, F., Sattler, U.: An Overview of Tableau Algorithms for Description Logics. Studia Logica, Volume 69, Issue 1, pp 5-40 (2001)
2. Bucella, A., Penabad, M.R., Rodriguez, F.J., Farina, A., Cechich, A.: From relational databases to OWL ontologies, Digital Libraries: Advanced Methods and Technologies. Digital Collection, Pushchino, Russia
3. Chen, P. P-S.: The Entity-Relationship Model-Toward a Unified View of Data, ACM Transactions on Database Systems, pp. 9-36, (March 1976)
4. Chen, P. P-S.: The ER Designer: Reference Manual. Chen & Associates (1987)
5. Connolly, T. : Database Systems: A Practical Approach to Design, Implementation and Management. Addison Wesley; 5 edition (24 Feb 2009)

6. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: A framework for the collaborative specification of semantically annotated business processes in Journal of Software Maintenance and Evolution: Research and Practice, vol. 23, pp. 261-295 (2011)
7. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Reasoning on Semantically Annotated Processes, in International Conference on Service- Oriented Computing, Springer, vol. 5364, pp. 132 - 146 (2008)
8. Ghidini, C., Rospocher, M., Serani, L.: A formalisation of BPMN in description logics. Technical Report TR 2008-06-004, FBK (2008)
9. Gonçalves, R., Others.: OWL Reasoner Evaluation (ORE) Workshop 2013 Results: Short Report. ORE 2013. (2013)
10. Grigoriev, A.V., Kropotin, A.A., Ovsyannikova, E.O.: The Problem of Detecting Inconsistencies on UML Class Diagrams. Scientific log. Collection of scientific works SWorld. The modern problems and ways of their decision in science, transport, production and education' 2012, Odessa, ISSN 2224-0187. Volume 14, pp 3-10 (2012)
11. Grigoryev, A.V., Ivashko, A.G.: TReasoner: System Description. Proceedings of the 2nd OWL Reasoner Evaluation Workshop (ORE 2013), pp. 26-31 (2013)
12. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. Artif. Intell. Res. 36, 165–228 (2009)
13. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2), pp. 51–53 (2007)
14. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Third International Joint Conference on Automated Reasoning, IJCAR., pp. 292–297 (2006)