16. Wikström, G. LTE latency reduction: preparing for 5G [Electronic resource] / G. Wikström // ERICSSON. – 2016. – Available at: https://www.ericsson.com/research-blog/lte-latency-reductions-preparing-5g/
17. Teyeb, O. Evolving LTE to fit the 5G future [Electronic resource] / O. Teyeb, G. Wikström, M. Stattin, T. Cheng, S. Faxér, H. Do // ERICSSON Technology Review. – 2017. – Available at: https://www.ericsson.com/assets/local/publications/ericsson-technology-review/docs/2017/etr_evolving_lte_to_fit_the_5g_future.pdf
18. Tikhonov, V. I. Conveyor module resource scheduling in packet based communication channel [Text] / V. I. Tikhonov, A. Taher, O. V. Tykhonova // Bulletin of the National Technical University «KhPI». A series of «Information and Modeling». – 2016. – Issue 21. – P. 152–161. doi: 10.20998/2411-0558.2016.21.17

Проведено аналіз різонерів для роботи з онтологічними базами знань з метою збільшення швидкодії цих баз. Запропоновано варіант рішення, в якому комбінуються переваги «tableau»-based і «hypertableau»-based різонерів. Проведено аналіз можливості застосування такого рішення на сервері онтологічних баз знань «Virtuoso». В результаті дослідження був розроблений метод комбінації різонерів для оптимізації роботи онтологічних баз знань. В результаті застосування даного методу було отримано збільшення продуктивності роботи баз знань при роботі з різнотипними онтологіями

Ключові слова: комбінація різонерів, Jena, Virtuoso, hypertableau, tableau, HermiT, Pellet, FaCT++, ABox, TBox, RBox

Проведен анализ ризонеров для работы с онтологическими базами знаний с целью увеличения быстродействия этих баз. Предложен вариант решения, в котором комбинируются преимущества «tableau»-based и «hypertableau»-based ризонеров. Проведен анализ возможности применения такого решения на сервере онтологических баз знаний «Virtuoso». В результате исследования был разработан метод комбинации ризонеров для оптимизации работы онтологических баз знаний. В результате применения данного метода было получено увеличение производительности работы баз знаний при работе с разнотипными онтологиями

Ключевые слова: комбинация ризонеров, Jena, Virtuoso, hypertableau, tableau, HermiT, Pellet, FaCT++, ABox, TBox, RBox

UDC 004.8
DOI: 10.15587/1729-4061.2017.112347

ONTOLOGICAL KNOWLEDGE BASES PRODUCTIVITY OPTIMIZATION THROUGH THE USE OF REASONER COMBINATION

I. Bibichkov
Postgraduate student*
E-mail: bibi4kov@gmail.com
V. Sokol
Postgraduate student*
E-mail: sokol@sw-expert.com
O. Shevchenko
PhD, Associate Professor*
E-mail: shevchenko@sw-expert.com
*Department of Artificial Intelligence
Kharkiv National University of Radio Electronics
Nauky ave., 14, Kharkiv, Ukraine, 61166

1. Introduction

Currently, there is an increased use of the ontological model of domain description. This is the result of the versatility and flexibility of this model. Reasoners, which accounted for most of the time spent on processing ontologies, play an important role in it [1].

The standard solution for ontologies with unchanged structure of the classes and properties is a preliminary run of a reasoner during the load process of an ontology. This approach is effective on the condition that the result of the reasoning is cached. However, if you need to change the structure of an ontology, it becomes rather time-consuming, which is a problem for intelligent systems working in real time.

The possibility of combining different reasoning models depending on the type and structure of the ontology is being researched as an alternative solution to this problem.

The study has compared the characteristics of the most popular reasoners: FaCT++ Pellet, HermiT [1, 2]. The possibility of the combined use of these reasoners and ontological information store Virtuoso Server [3] has also been evaluated.

The particular attention has been paid to the description of optimization techniques based on the use of the HermiT reasoner. This particular reasoner is of great interest because unlike its analogues, such as FaCT++, Pellet, RacerPro, whose work is based on the standard tableau algorithm, it uses the hypertableau algorithm as an alternative. The application of the hypertableau algorithm is extremely useful and

even indispensable when working with the nondeterministic ontologies, which currently prevail.

The main problem, which you are bound to face when working with ontology knowledge bases is the significant time cost of both obtaining a list of the properties and objects and checking the knowledge base for consistency. Therefore, only those researches should be recognized relevant, which are aimed at increasing the performance of ontological knowledge bases, containing a great volume of stored information.

2. Literature review and problem statement

The study has analyzed the works on the optimization of ontological knowledge base performance [4–10].

Reasoners are one of the main components of the ontological systems and the performance of reasoners is the most resource-intensive task in ontology processing. Ontology performance largely depends on the performance of a reasoner, which is used in it. The application of the HermiT reasoners reduced ontology processing time 4–15 times, and in some cases up to 25 times or more [4] compared to other reasoners (Pellet, FaCT++).

At the moment, the main methodologies implemented by the reasoners are: tableau and hypertableau. For example, hypertableau reasoners are the most effective when processing medical ontologies [5]. However, [5] does not provide examples of the application of reasoners for ontologies that contain the abundance of role axioms and their hierarchies.

The modification of the standard tableau algorithm designed to increase productivity when working with ontologies has been suggested in [6]. Algorithm modification methods, proposed by the authors, were implemented in the developed LIGHT reasoner. The disadvantage of the concept suggested in [6] is that the method is exclusively limited by ontologies based on the descriptive logic ALC.

Tableau algorithm modification, presented in [7] is based on the application of hypertableau and hyperresolution algorithm components. In addition, [7] contains the presentation of "Anywhere Pairwise Blocking" blocking system, which provides an additional optimization. The method proposed in [7] is effective for GALLEN ontologies. The disadvantage of the concept suggested in [7] is the limited application of this method as it is effective only for the GALLEN and NCI ontologies.

[8] looks through ontology optimization method, developed on a version of a model that was built by the Pellet reasoner. The drawback of this research is the limited application of this method by ontologies containing only complex ABox.

Reasoner performance enhancing optimization in ontological knowledge bases can be obtained through input preprocessing. The study [9] is devoted to this type of optimization. The drawback of this study is the lack of practical results of applying the method for different reasoners. Furthermore, the peculiarities of the ontology structure, for which this approach should be applied, have not been taken into account.

[10] introduces the ontological system optimization methods by applying an alternative classification algorithm. The disadvantage of this study is the lack of study validation for tableau-based reasoners.

On the basis of these studies, it can be seen that the various reasoners show good results on speed processing of certain ontologies whereas with other types of ontologies they show rather poor performance. The studies mentioned above neither include the detailed comparison of reasoners nor investigate the possibility of combining them in order to enhance the performance of ontological systems.

3. The aim and objectives of the study

The aim of this research is to develop a generic method of application of combinations of reasoners in ontological knowledge bases to increase their performance. This will make it possible to extend the application area of ontologies and move on to the concept of a single repository for heterogeneous systems.

In order to achieve the aim, we set the following objectives:

– to investigate the peculiar features of reasoners;

– to justify the choice of performance enhancing optimization method for ontological knowledge bases;

– to carry out testing of the developed method.

4. The research of the characteristics of reasoners

The important point for enhancing the performance of ontological knowledge base is the selection of a reasoner. The proper selection of a reasoner requires an understanding of its structure, internal structure and the specificities of the application, as well as the understanding of the internal structure of the ontology, with which it will interact.

The study reviewed the reasoners, whose work is based on the two most popular methodologies tableau and hypertableau, such as:

– FaCT++;

– Pellet (tableau);

– HermiT (hypertableau).

Virtuoso server has been used as a knowledge base storage server. We also used Jena Framework to retrieve the ontology from Virtuoso server, and create a model ontology for the work of a reasoner. The justification in favour of such a selection can be found in [11].

4. 1. The application of the tableau and hypertableau methodologies

Below is a comparison of two major methodologies, tableau and hypertableau, used in the HermiT, FaCT++ and Pellet reasoners and reviewed in this study.

The Pellet and FaCT++ reasoners are written using the tableau methodology, and the HermiT reasoner is based on hypertableau one.

The standard knowledge base consists of the following components: the properties of axioms (RBox), classes of axioms (TBox) and facts (ABox), within the Description Logic knowledge base.

Rbox is the axioms containing roles and the role hierarchies. The R in the RBox represents a final set of the transitivity axioms of Trans type (R) and inclusion role axioms. TBox (from Eng. terminology) is a set of general concept inclusions. Abox (from Eng. assertions) – a set of concept assertions – is individuals.

The advantages of application of the hypertableau algorithm in practice in comparison with its analogues, such as tableau, can be seen when performing standard operations. For example, to check the ontological knowledge base ($K=(R, T, A)$) for satisfiability, the tableau algorithm builds the output which is a sequence of ABoxes $A_0, A_1,..., A_n$ where $A_0=a$ and every $(A)_{(i)}$ is obtained from $(A)_{(i)-1}$ on the application of the rule of inference. Inference rules make the implicit information in the axioms R and T explicit and, thus, the development of ABox (A) in the direction of the model K. The algorithm stops operating in two cases:

1) if there are no applicable inference rules for some $(A)_n$. In this case, A_n represents the model K;

2) if $(A)_n$ contains an obvious contradiction.

The hypertableau algorithm, which is used, for example, in the HermiT reasoner, performs the optimization in knowledge bases, including through absorption. This is due to the alignment of the axioms of descriptive logic to a specific form. This form allows you to run standard, role or binary absorption at the same time, which is a distinctive feature of the hypertableau algorithm in comparison with tableau one. In the latter, the application of additional absorption algorithms is impossible.

Tableau algorithm and the corresponding reasoners, written on its base, use a standard data blocking mechanism for the integrity of the created model. In such case, each specific individual may be blocked only by its ancestors; the algorithm is called ancestor blocking, or the blocking by the ancestors. In contrast to this approach, hypertableau extends the algorithm through an advanced algorithm of blocking, anywhere blocking (i. e. blocking from anywhere). Within this algorithm, the individual may be blocked almost by any other individual.

Although the anywhere blocking can often hamper the creation of multiple copies of identical individuals, it is often possible to create models for tableau procedures that can contain similar individuals.

Standard reasoners based on tableau algorithms apply the nearest neighbour search algorithm to retrieve the root individuals. Neighbours are identified recursively, starting with individuals located in the initial ABox. Tableau algorithm implies the exact number of neighbours that will be present in the final model. Then, follows the creation of the appropriate number of root elements (individuals). In contrast to this algorithm, hypertableau uses an alternative method that prevents the creation of unnecessary root individuals. The essence of this method is that instead of having to add new root individuals, the existing ones are simply marked as root. If you make sure the root individuals remain unique, you can ensure the correctness of the algorithm, without increasing the size of the constructed models.

4. 2. The application of the HermiT reasoner

HermiT is an open source reasoner, based on the hypertableau methodology [4].

HermiT uses the direct semantics and meets all the requirements of the OWL2 semantic data reasoning.

Starting with version 1.1, HermiT can process DL Safe rules. In addition, rules can be directly added into incoming ontology in a functional style (for data preprocessing) or in other OWL syntaxes, supported by OWL API. It should be noted that reasoning through DL Safe rules will be incomplete if the ontology contains the property chains, transitivity axioms, or complex properties used within the rules.

The reasoner is based on the principle of hypertableau calculus. The specific features of the reasoner eliminate the performance problems associated with nondeterminism and large amounts of data, which are currently the primary sources of complexity in modern OWL models. HermiT has an improved locking strategy that provides additional benefits when processing large-sized ontologies compared to other reasoners. HermiT also includes some other new improvements such as a more efficient approach to the processing of nominals and various optimization methods for the classification of ontologies. HermiT, in some cases, contributes to a significant performance enhancement as opposed to other reasoners, for instance, in the classification of complex ontologies [4]. This is due to the application of the classification mechanism for a number of ontologies. At this point, such a mechanism does not exist in the analogues.

The HermiT reasoner implements the methods that provide additional calculation optimization and are used to build the data model. These include:

1. Checking the satisfiability of a concept. The aim of this method is for the HermiT reasoner to create some knowledge base K' for checking the feasibility of a concept A.

2. Caching Blocking Labels. This is the locking method, which prevents the creation of multiple identical submodels during the check for consistency. Conceptually, instead of performing different tests, while generating n different models, one test that builds one model containing n independent fragments is performed.

4. 3. The use of FaCT++ and Pellet reasoners

FaCT++ (Fast Classification of Terminologies) is a reasoner based on the descriptive logic [12]. FaCT++ 1.0 is a relatively new OWL DL reasoner designed as a separate platform. The FaCT++ reasoner contains various algorithms and optimization techniques. It includes most of the standard optimization techniques available within ontological systems. The FaCT++ architecture is better suited for more complex algorithms and is open to a wider range of heuristic operations.

The distinctive feature of this reasoner is the support of the performance optimization through a series of optimization methods within preprocessing of an ontology. The most effective methods are absorption and simplification.

One of the optimization techniques used in the FaCT++ reasoner is the subsystem of simplifying complex logical operations and bringing them to a certain simplified normal form (SNF). The aim of SNF is to bring the complex logical operations to the simplest four: negation "¬", conjunction "∩", universal restriction "∀", at-most restriction "≤".

Absorption is the exclusion of the so-called "General Concept Inclusion", which is the most costly.

The Pellet reasoner is an open source reasoner, written in the Java language. It is based on the principles of the description logic, which comply with the OWL-DL standard.

The Pellet reasoner implements most modern optimization techniques, such as: simplification, absorption, semantic branching, dependency-directed backjumping [13].

The difference between the Pellet reasoner and the other ones represented in this study is that the Pellet reasoner is more flexible when working with different data types, even with user-defined data types (based on numeric and date/time data types). It gives a reasoner an advantage over other tableau-based reasoners, in particular, FaCT++.

5. The development of the reasoner combination method in order to enhance the performance of ontological systems

Development of the reasoner combination method includes two phases. Firstly, there is the load of the ontology from the knowledge base server and the formation of the model, with which the reasoner will later interact. Then, on the basis of the specific features of the loaded ontology, there is the selection of a reasoner and connection it to the ontology model.

5. 1. The creation of the ontology model for the performance of a reasoner

Jena Framework is used when designing a data model for the further work of a reasoner. This framework is a set of Java libraries for working with a variety of ontological systems. In particular, Jena Framework provides the ability to work with a Virtuoso.

Virtuoso server is a versatile server to work with different data types. The study investigates the possibility of using Virtuoso as a triplet store.

Jena is a free open source Java platform for Semantic Web applications (SW). Its latest version is Jena 3.1.0, which uses Java 8 [14].

Currently, Jena is one of the most popular Java tools that is used in developing SW applications [14].

The use of Jena Framework when developing SW applications is extremely effective because it contains the support for the following components:

– RDF API;

– OWL API (can be used as RDFS API);

– Reading and writing of RDF in RDF/XML, N3 and N-triple format;

– Temporary and permanent store of RDF models;

– SPARQL request handler;

– Request handler based on inference rules.

Jena Framework supports most of the functions of RDFS standard.

5. 2. The description of a reasoner selection method

The development of the reasoner selection method has been based on this study. Fig. 1 displays how this method works.

The application loads the ontology into the system.

Jena ontology model. After loading the ontology, Jena Framework builds an appropriate model, with which a reasoner will later interact.

The JDBC driver manager and DBC driver are the specific libraries, which are used for Jena Framework and triplet store.

Triplet store is a place which stores an ontology in the form of triplets. Virtuoso server is used as triplet store.

The selection of the reasoner depending on the ontology type. This phase includes the identification of the most effective reasoner for a particular ontology. At first, the system makes a choice between tableau-based and hypertableau-based reasoners. If it has been identified that it is optimal to use tableau-based reasoners, then there is a choice between the FaCT++ and Pellet reasoners. In another case, the hypertableau-based HermiT reasoner is selected.

The choice between tableau-based and hypertableau-based reasoners is based on the identification of the existence of many complex RBox [15]. If these components make up more than 8 % of RBox, TBox and ABox in the ontology, the system selects the tableau-based reasoner. If such items are missing or their number is negligible (less than 8 % of RBox, TBox and ABox), the system selects the hypertableau-based reasoner. Thus, this is the criterion for choosing between tableau-based and hypertableau-based reasoners.

If the system has identified that the tableau-based reasoners are optimal, then there is a choice between the Pellet and FaCT++ reasoners. If the ontology contains multiple complex ABox, as well as complex data types, the system selects the Pellet reasoner as the main one.

The activation is conducted after selecting a suitable reasoner.

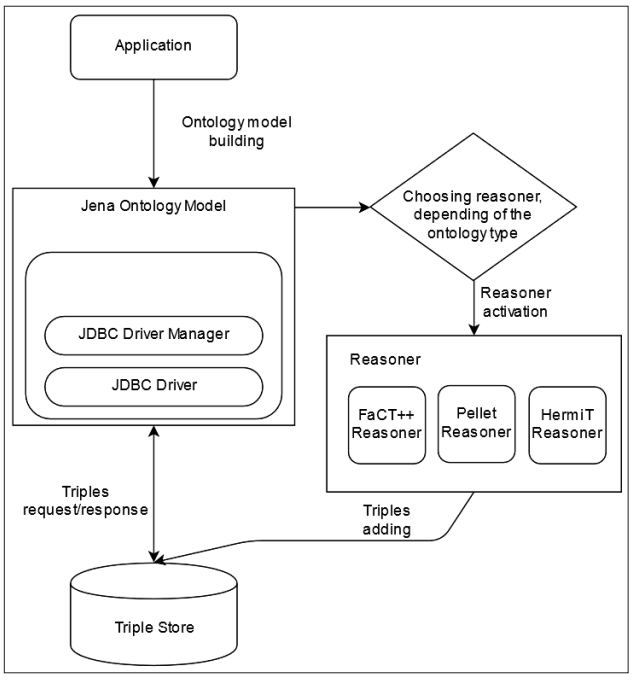

Fig. 1. Scheme for choosing a reasoner

The functionality of reasoners implies the identification of an ontology for consistency (coherence), as well as its expansion by adding triplets.

6. The results of the application of reasoner combination method

To test the effectiveness of the developed reasoner combination method, 8 different ontologies have been selected in this study (Table 1). The selected ontologies are characterised by expressivity (in terms of Descriptive logic), the number of TBox, ABox, RBox, individuals and classes that determine the structure of the ontology.

The structure of the ontology is crucial when choosing a reasoner. Table 1 lists the different types of ontologies and their characteristics, the most important for the performance of a reasoner.

The advantage of the HermiT reasoner is the ability to process ontologies that cannot be processed by other reasoners, or their work is insufficiently productive.

When working with ontologies that contain a branched structure of description graphs, the HermiT reasoner works faster in comparison with the analogues (Table 2). This type of ontologies prevails in biomedical systems (e. g. FMA Lite).

The enhancement of the HermiT reasoner performance is mainly a consequence of the application of the hypertableau algorithm, instead of tableau, which is the basis for the Pellet and FaCT++ reasoners.

Table 1

The quantitative structure of ontologies

Ontology name	Number of classes	Number of individuals	TBox	RBox	Abox	Expressivity
BP XP OBOL	10,295	43,446	6,980	0	43,446	ALE
FMA Lite	75,141	46,225	119,558	3	46,225	ALEI+
Fly Taxonomy	6,602	5,350	6,603	0	5,350	AL
Biological Process	14,955	73,901	27,051	1	73,901	EL++
DLP ExtDnS	96	0	606	384	0	SHIN
MGED	216	579	430	240	642	ALCOF (D)
DOLCE-Plans	118	27	265	948	68	SHOIN (D)
SWEET Numerics	1,506	113	2,784	305	340	SHOIN (D)

The application of the hypertableau algorithm in the HermiT reasoner when working with ontologies involves the use of the developed locking strategy. In addition to the locking strategy, the performance efficiency is enhanced by the use of inference rules, different from those used in Pellet and FaCT++.

The use of the HermiT reasoner implies some limitations. For example, in the tests using such ontologies as ExtDnS, DLPMGED, the HermiT reasoner proved to be worse than FaCT++ and Pellet [13] (Table 2). This is due, in particular, to the fact that DLP ExtDnS ontology includes a much more complex RBox structure than most other ontologies.

There are some disadvantages of the HermiT reasoner, which subsequently have a negative impact on the overall performance. These include limited support for data types and the peculiarities of realization of transitivity, which is done by overwriting the axioms.

Table 2

Comparative table of reasoners work time

Ontology	Pellet (ms)	FaCT++ (ms)	Hermit (ms)	The combination of reasoners (ms)
BP XP OBOL	505, 100	1, 742, 300	8, 700	10, 440
FMA Lite	Error	Error	43, 800	50, 370
Fly Taxonomy	1, 200	5, 300	1, 100	1, 280
Biological Process	10, 700	79, 200	2, 400	2, 830
DLP ExtDnS	7, 100	100	95, 800	120
MGED	800	249	5, 700	291
DOLCE-Plans	105, 150	Error	1, 080, 950	120, 175
SWEET Numerics	3, 800	210	76, 520	245

The results of the studies cited above (Table 2) clearly show that the application of the reasoner combination method outstrips the performance of any reasoner, considering that a reasoner will process ontologies of different types.

7. Discussion of the results of application of the combination method

The method of reasoner combination, proposed in the study showed the enhancement of reasoning performance when working with different types of ontologies. This is due to the fact that in each case a reasoner with the optimal performance is selected in accordance with the structure and type of ontologies. The method of reasoner combination selects a reasoner with the optimal performance for each ontology. This method gives an advantage over any of the reasoners when compared to the entire set of ontologies used in this study. The results in Table 2 show the effectiveness of the proposed method. The differences between the results of the work of a reasoner and the combination method are due to the fact that the method of reasoner combination spends time analyzing ontologies. The development of the ontology model through using Jena and Virtuoso server has been reviewed. This model is used to analyze the ontology structure, which, in turn, allows us to select the most effective reasoner for this particular ontology.

The disadvantages include a small number of ontologies which were tested. This is due to the large amount of time required for the preparation of each ontology. It is worth noting that even such a number of ontologies is sufficient for the research. It also should be noted that the study does not contain a description of the interaction between the HermiT, Pellet and FaCT++ reasoners and Virtuoso server since the study of this interaction was not part of this work. But such interaction could increase the performance of the ontological knowledge base, which can be a subject for a separate study.

The enhancement of a reasoner performance opens up new possibilities for the application of ontological knowledge bases, because the main problem of such systems is a poor performance, and their slowest component is a reasoner. At a high enough speed when the reasoner performance allows working in real time (delay will be indistinguishable to the user), ontological systems will be able to replace databases and provide a great intellectual toolkit for processing the stored knowledge. Principles of optimization described above can be used when working with large amounts of information which require the use of reasoning.

The advantage of this study is an attempt to combine different methodologies (tableau and hypertableau) in one method, which allows obtaining higher performance compared with a single reasoner.

The limitation of this method is the need for ontology to meet the OWL DL specification, as the Pellet and FaCT++ reasoners may cause the hangup of reasoning on the ontology relevant to the OWL Full specification.

The study echoes the paper [16], but is not a direct sequel. The paper [16] contains outstanding optimization techniques aimed at building optimal SPARQL requests, which were not reviewed in this study.

In the future, it is planned to continue the research on the improvement of ontological knowledge repositories and development of productive samples of ontological knowledge bases.

8. Conclusions

1. The generic method of reasoner combination for enhancing the performance of ontological systems has been developed. Its essence is the selection of the appropriate reasoner depending on the type of ontology. The distinctive feature is the combination of the advantages of tableau and hypertableau methodologies. Such an approach provides the maximum performance of the proposed combination method among the HermiT, Pellet and FaCT++ reasoners.

2. The HermiT, Pellet and FaCT++ reasoners were used for carrying out the experiments. The application of the HermiT reasoner for BP XP OBOL ontology was 6 times more effective (compared to the Pellet reasoner) and 20 times (compared to the FaCT++ reasoner). When working with Fly Taxonomy, the HermiT reasoner proved to be 5 times more effective in comparison with the FaCT++ reasoner and 100 ms faster than the Pellet reasoner. When processing Biological Process, the HermiT reasoner was 4 times as good as Pellet and 25 times faster than FaCT++.

3. For ontologies that contain description graphs, the application of the HermiT reasoner is much more beneficial in comparison with the analogues (such ontologies are often found in biomedical ontologies). For instance, the application of the HermiT reasoner for processing FMA Lite ontology took 43 800 ms. For comparison, the Pellet and FaCT++ reasoners did not cope with this task at all.

4. It has been found that the HermiT reasoner is not always the best choice. For example, it took the HermiT reasoner considerably more time than the Pellet and FaCT++ reasoners to process the DLP ExtDnS and MGED ontologies. It took the HermiT reasoner 950 and 13 times more time than the FaCT++ and Pellet reasoners, respectively, to process DLP ExtDnS ontology. It is similar for MGED ontology, where the HermiT reasoner spent 7 and 22 times as much time as the Pellet and FaCT++ reasoners, respectively.

5. The testing of the proposed method provided the results proving its effectiveness. The reasoner combination method selects a reasoner with the optimal performance for each ontology. This method gives an advantage over any reasoner when compared to the entire set of ontologies used in this study.

References

1. List of Reasoners [Electronic resource]. – OWL. – Available at: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/
2. Reasoners and rule engines: Jena inference support [Electronic resource]. – Apache Jena. – Available at: https://jena.apache.org/documentation/inference/
3. OpenLink Virtuoso Universal Server Documentation [Electronic resource]. – OpenLink Software. – 2017. – Available at: http://docs.openlinksw.com/virtuoso/
4. Sheare, R. HermiT: A Highly-Efficient OWL Reasoner [Electronic resource] / R. Sheare, B. Motik, I. Horrocks // Available at: http://www.cs.ox.ac.uk/boris.motik/pubs/smh08HermiT.pdf
5. Motik, B. Hypertableau Reasoning for Description Logics [Text] / B. Motik, R. Shearer, I. Horrocks // Journal of Artificial Intelligence Research. – 2009. – Issue 36. – P. 165–228.
6. Zuo, M. Intelligent Tableau Algorithm for DL Reasoning [Text] / M. Zuo, V. Haarslev // Lecture Notes in Computer Science. – 2013. – P. 273–287. doi: 10.1007/978-3-642-40537-2_23
7. Motik, B. Optimized Reasoning in Description Logics Using Hypertableaux [Text] / B. Motik, R. Shearer, I. Horrocks // Lecture Notes in Computer Science. – 2007. – P. 67–83. doi: 10.1007/978-3-540-73595-3_6
8. Pukancová, J. Tableau-Based ABox Abduction for Description Logics: Preliminary Report [Electronic resource] / J. Pukancová, M. Homola // Available at: http://ceur-ws.org/Vol-1577/paper_23.pdf
9. Surianarayanan, C. A survey on optimization approaches to semantic service discovery towards an integrated solution [Text] / C. Surianarayanan, G. Ganapathy // ICTACT Journal on Soft Computing. – 2012. – Vol. 02, Issue 04. – P. 377–383. doi: 10.21917/ijsc.2012.0059
10. Glimm, B. Optimising Ontology Classification [Text] / B. Glimm, I. Horrocks, B. Motik, G. Stoilos // Lecture Notes in Computer Science. – 2010. – P. 225–240. doi: 10.1007/978-3-642-17746-0_15
11. Shevchenko, A. Modern ontological knowledge base management systems comparison [Text] / A. Shevchenko, E. Shevchenko // Visnyk SevNTU. – 2012. – Issue 131. – P. 82–86.
12. Dentler, K. Comparison of Reasoners for large Ontologies in the OWL 2 EL Profile [Text] / K. Dentler, R. Cornet, A. Teije, N. Keizer // IOS Press. – 2011. – Issue 1. – P. 1–17.
13. Sirin, E. Pellet: A practical OWL-DL reasoned [Text] / E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz // Web Semantics: Science, Services and Agents on the World Wide Web. – 2007. – Vol. 5, Issue 2. – P. 51–53. doi: 10.1016/j.websem.2007.03.004
14. Apache Jena Overview [Electronic resource]. – Apache. – Available at: http://jena.apache.org/documentation/javadoc/jena/
15. Fokoue, A. The Summary Abox: Cutting Ontologies Down to Size [Text] / A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, K. Srinivas // Lecture Notes in Computer Science. – 2006. – P. 343–356. doi: 10.1007/11926078_25
16. Bibchkov, I. Optimizing the performance of ontological knowledge bases built on the basis of «VIRTUOSO» [Text] / I. Bibchkov, V. Sokol, A. Shevchenko // Eastern-European Journal of Enterprise Technologies. – 2014. – Vol. 5, Issue 2 (71). – P. 4–8. doi: 10.15587/1729-4061.2014.28553