TReasoner: System Description

Andrey V. Grigorev and Alexander G. Ivashko

Tyumen State University,
Semakova. 18, 625003 Tyumen, Russian Federation
{ivashco,107th}@mail.ru

Abstract. TReasoner is a reasoning system supporting the $SHOIQ(D)$ logic expressiveness, which forms the basis of the OWL DL language. The TReasoner was developed for using in the enterprise architecture verification expert systems, but the OWL API package allows to use the system for performing ontology operations. The reasoner implements a tableau algorithm and optimization techniques, some of them were developed and were used for the first time. This description also contains an assessment of the developed system efficiency.

Keywords: Description Logic, OWL, Tableau Algorithm, Reasoner, Classification

1 Introduction

Ontologies are a powerful tool of knowledge representation, which became very popular for using by expert systems [8]. First of all because of the fact that they are based on the description logic formalism, which has a formally defined semantics allowing to develop tableau algorithm for a logic inference. OWL [13] is the basic ontology representation language recommended by the W3C consortium. Nowadays the OWL 2 standard is valid. The OWL DL language uses the $SHOIN$ [1] description logic with support of data values.

To date many OWL reasoning systems such as FaCT++ [15], HermiT [7] (for OWL DL), jcel [11], ELK [10] (for OWL 2 EL) were developed, they implement different algorithms for a logic inference.

The article introduces a new OWL Reasoner. The TReasoner is $SHOIQ(D)$ reasoner implementing tableau algorithm with some novel optimization techniques. TReasoner is free distributed by GNU General Public License v2. Source code of the TReasoner, compiled class library and wrapper for system usage are available at `http://treasoner.googlecode.com`.

This system description has the following structure. Section 2 provides a supported language and an implemented algorithm. Section 3 contains information about architecture, implementation and optimization techniques that are used by the TReasoner. Results of the system testing are described in section 4. Section 5 concludes this work.

2 Supported Language Subset and Implemented Reasoning Algorithm

The TReasoner allows to perform a concept satisfiability checking, a consistency checking and a classification on OWL ontologies that use the $SHOIQ(D)$ description logic. It means that the system works correctly with concepts described by the disjunction, the conjunction, the existential quantifier and the universal quantifier. Besides, the $SHOIQ$ allows roles to be transitive and inverse to other roles. There may be concepts consisting of one individual (nominal), at the same time the $SHOIQ$ is extended by number restrictions ($n \leq R.C$ or $n \geq R.C$). D letter at $SHOIQ(D)$ logics allows to describe knowledge with support of datatypes (strings, numbers, dates and etc.).

The TReasoner implements the tableau algorithm [6]. The concept satisfiability checking is carried out through graph-model existence checking.

The tableau algorithm for $SHOIQ$ has NExpTime complexity, but the developed system implements different new and old optimization techniques, which allow to significantly reduce worktime in practice.

3 Architecture and Implementation

The TReasoner was developed using the Java language, because of the cross-platform portability. The system consists of 6 packages. The RuleGraph package implements data structures for the inner representation of concepts. Also this package implements algorithms for the concepts simplification and the axiom simplification. TBox, ABox and RBox axioms are contained in KnowledgeBase package classes. The OWL API package is used for loading the OWL ontologies and transforming them to inner system representation. Main package is Checker. It contains classes that implement tableau algorithm and optimization techniques for it. Checker package classes use Interpretation package classes, which implement data structures for the interpretation building. All packages use classes of the Help package, which implements different supporting algorithms and data structures such as binary heap, hash-table, etc. The UML package diagram is presented on the Fig. 1.

The TReasoner implements optimization techniques, which can be divided into 3 groups:

1. Preprocessing optimizations;
2. Tableau algorithm optimizations;
3. Classification optimizations.

The system uses both time-tested and newly developed optimizations.

3.1 Optimization Techniques

Preprocessing optimizations are used by the ontologies transformation to inner structures, which are understandable by the TReasoner. Also they are used for

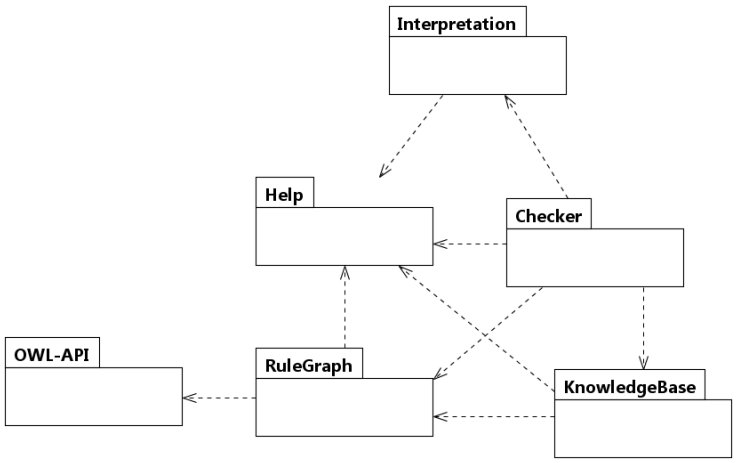

Fig. 1. The structure of packages of the TReasoner system

the transformation of GCIs and equivalence axioms. For the concept representation the system uses direct acyclic graph (DAG), each vertex of the graph corresponds to some operation or quantifier, and neighbours of this vertex are operands of the operation, in addition each vertex in the DAG has a number that uniquely identifies it. To reduce memory usage, same concepts are represented by only one subgraph. For each vertex (in order of height increasing), a hash-function value is calculated, this function consider unique numbers of all neighbours, operation type of the vertex, unique number of a role and number restriction (for existential and universal quantifiers, and for number restriction operations). If this function value doesn't exists in hash-table, the vertex with its hash-function value will be added to hash-table. If function value is found then all edges which enter to this vertex will change its direction to vertex with corresponding value of hash-function that contained in hash-table.

To reduce memory usage, removal of brackets technique was developed. The algorithm is performed in two runs. In first run, for each vertex v (in order of height increasing) that represent a concept, set of concepts $H(v)$ is calculated. Concepts of this set defined as follows:

1. If current graph vertex v represents atomic cocnept C, then $H(v)$ set contains two elements: C and $\top$;
2. If current graph vertex v represents $\sqcap$-cocnept then $H(v)$ set contains elements of $H(u_1) \sqcup H(u_2) \sqcup ... \sqcup H(u_k)$ for all u_i which are neighbours of v;
3. If current graph vertex v represents $\sqcup$-cocnept then $H(v)$ set contains elements of $H(u_1) \sqcap H(u_2) \sqcap ... \sqcap H(u_k)$ for all u_i which are neighbours of v;

In second run, vertexes are considered in order of height decreasing, each vertex is transformed to $\sqcap$-vertex with neighbours of all concepts from $H(v)$ and itself

vertex, so each concept of $H(v)$ will be deleted from $H(u_i)$ for all u_i which are neighbours of v. For example, concept $(((C \sqcap B) \sqcup (D \sqcap B)) \sqcap A) \sqcup (B \sqcap ((C \sqcap A) \sqcup (E \sqcap C)))$ will be transformed to $B \sqcap ((C \sqcap (A \sqcup E)) \sqcup (A \sqcap (C \sqcup D)))$

After loading and preprocessing of concepts, a processing of axioms will be performed. Absorption technique [5] is used for this task.

Tableau algorithm optimizations that are implemented in the TReasoner, include such optimizations as backJumping [14], caching [3, 5] and global caching [12]. The system implements novel optimization techniques. The main of such techniques is the SS-branching [9], which determine disjointness of concepts without using of tableau algorithm. It is applicable not in every cases, but in wide range of concepts. The SS-branching procedure determine disjointness of two concepts by analyzing of structures of DAGs that represent its. For example, if concepts C and D are conjunctions of other concepts ($C \equiv C_1 \sqcap C_2 \sqcap ... \sqcap C_n, D \equiv D_1 \sqcap D_2 \sqcap ... \sqcap D_n$) and some of the concepts C_i and D_j are disjoint, then C and D are disjoint. Conditions of disjointness for cases when C and D are disjunctions, disjunction and conjunction, existential and unversal quantifiers were identified. However, SS-branching can not to determine disjointness of concepts. To cover wider class of concepts Bron-Kerbosch algorithm was used. For disjointness checking of the concepts like $E_1 \sqcap E_2 \sqcap ... \sqcap E_n$, where every concept E_i is a disjunction ($E_I \equiv F_1 \sqcup F_2 \sqcup ... \sqcup F_k$). Such concepts will be presented of n-partite form, where every vertex of the part presents F_j concept, so vertexes form different parts will be connected, if corresponding concepts are disjointness. Model existing checking of such concepts performed by using of Bron-Kerbosch algorithm, which used for n-clique finding in n-partite graph.

Classification optimizations allow to reduce system worktime to perform the classification operation. Enhanced traversal method [2] is used for the classification, information about disjointness is extracted not only from subsumption test, but during the concept satisfiability testing. During the construction of model by tableau algorithm, labels of all individuals are checked in the presence of concepts A and $\neg B$, though A and B are concept names. If those individuals exist, then $A \not\sqsubseteq B$, without performing $A \sqsubseteq B$ subsumption test.

4 System Performance Evaluation

The TReasoner system performance testing uses ontologies classification tests that were used on the OWL Reasoner Evaluation Workshop 2012 and compares the results received by HermiT (ver. 1.3.6) and FaCT++ (1.6.2) reasoners. They implement hypertableau and tableau algorithms and support the $SROIQ(D)$ logic. Information about used ontologies is shown in table 1.

System testing results in comparison with other reasoners are shown in table 2. First column of the table contains name of used ontology, and every subsequent column shows time spent for ontology classification by the relevant system. Testing was carried out on ASUS Notebook VX7SX Series Intel Core i7-2630QM CPU@2.00 GHz 2.00 GHz; 6.00 GB RAM running under Windows 7.

Table 1. Used ontologies

Ontology	Logic	Axioms	Concepts	Roles	Individuals
obi	SHOIN(D)	8530	1161	60	140
plant_trait	ALC	4317	976	1	3177
po_temporal	ALC	2839	284	1	2559
DLPOnts_Information_397	SHOIN	1037	120	198	12
DLPOnts_DOLCE-Lite_397	SHIF	351	37	70	0
DLPOnts_Plans_397	SHOIN	1281	117	264	27
pathway	ALC	1927	646	1	1160
protein	ALCS	5821	1055	2	4768
quality	ALCSH	4815	1980	13	2653
rex	ALC	1725	555	2	991

Table 2. Testing results

Ontology	TReasoner	FaCT++	HermiT
obi	17,065	1,313	130,359
plant_trait	1,035	0,099	0,228
po_temporal	0,098	0,071	0,064
DLPOnts_Information_397.owl.txt	5,177	0,94	11,443
DLPOnts_DOLCE-Lite_397.owl.txt	0,045	0,13	0,4
DLPOnts_Plans_397.owl.txt	5,515	0,167	217,667
pathway.owl	0,458	0,094	0,519
protein.owl	0,851	0,179	0,376
quality.owl	3,337	0,101	0,411
rex.owl	0,504	0,53	0,124

The resulting classification coincides to the reference classification provided together with chosen ontologies.

5 Conclusion

This system description presents the new reasoning system, implemented algorithms and implemented optimization techniques, which contribute to reduce worktime of different ontologies operations (classification, concept satisfiability checking, consistency checking). The developed system allows to perform logical analysis for expressive $SHOIN(D)$ logic that is used in OWL DL. This fact allows to use TReasoner to perform operations on a wide class of ontologies.

The presented testing results show that TReasoner may not compete yet with most popular systems such as HermiT and FaCT++ reasoners, but in future implementation of tableau algorithm will be improved and reduce of system worktime is expected.

In further researches improving of the TReasoner is expected in order to use it not only as module of the enterprise architecture verification expert system, but as self-dependent OWL reasoning system.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* CUP, 2003.
2. F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, E. Franconi. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or Making KRIS get a move on* *KR-92*, pages 270-281, 1992
3. Y. Ding and V. Haarslev. Tableau caching for description logics with inverse and transitive roles. In *Proc. DL-2006: International Workshop on Description Logics*, pages 143-149, 2006.
4. Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. Semantic Web 2(1): 11-21, 2011.
5. I. Horrocks. *Optimising Tableaux Decision Procedures for DescriptionLogics.* PhD thesis, University of Manchester, 1997.
6. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ. In D. MacAllester, editor, *Proc. of the CADE 2000*, number 1831, pages 482496. Springer-Verlag, 2000.
7. Ian Horrocks, Boris Motik, and Zhe Wang. The HermiT OWL Reasoner. OWL Reasoner Evaluation Workshop. 2012.
8. A. Ivashko, E. Ivanova, E. Ovsyannikova, S. Kolomiyets. Applying DL for information system architecture description. *Vestnik of Tyumen State University*, 98(4): 137-142, 2012.
9. A. Ivashko, A. Grigorjev, M. Grigorjev. Modification of tableau algorithm based on checking disjointness of complex concepts. *Vestnik of Tyumen State University*, 98(4): 143-150, 2012.
10. Yevgeny Kazakov, Markus Krotzsch and Frantisek Simancik. ELK Reasoner: Architecture and Evaluation. OWL Reasoner Evaluation Workshop. 2012.
11. Julian Mendez. jcel: A Modular Rule-based Reasooner. OWL Reasoner Evaluation Workshop. 2012.
12. Linh Anh Nguyen. An Efficient Tableau Prover using Global Caching for the Description Logic ALC. *Artificial Intelligence*, 93(1):273-288, 2009.
13. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL WebOntology Language: Semantics and Abstract Syntax, W3C Recommendation, February 10 2004. http://www.w3.org/TR/owl-semantics/.
14. P. Prosser. Hybrid Algorithms for the Constraint Satisfaction Problem. *Computational Intelligence*. 9(3): 268-299, 1993.
15. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. IJCAR 2006*, pages 292-297, 2006.