The $\mathcal{EL}$epH$\mathcal{H}$ant Reasoner System Description

Barış Sertkaya

sertkaya.baris@googlemail.com

Abstract. We intoduce the $\mathcal{EL}$epH$\mathcal{H}$ant reasoner, a consequence-based reasoner for the $\mathcal{EL}^+$ fragment of DLs. We present optimizations, implementation details and experimental results for classification of several large bio-medical knowledge bases.

1 Introduction

In [6, 5] Brandt has shown that the tractability result in [1] for subsumption w.r.t. cyclic $\mathcal{EL}$ TBoxes can be extended to the DL $\mathcal{ELH}$, which in addition to $\mathcal{EL}$ allows for general concept inclusion axioms and role hierarchies. Later in [2], Baader et. al. have shown that the tractability result can even be further extended to the DL $\mathcal{EL}^{++}$ which in addition to $\mathcal{ELH}$ allows for the bottom concept, nominals, role inclusion axioms, and a restricted form of concrete domains. In addition to these promising theoretical results, it turned out that despite their relatively low expressivity, these fragments are still expressive enough for the well-known bio-medical knowledge bases SNOMED [8] and (large parts of) Galen [19], and the Gene Ontology GO [7]. In [3, 4, 21] the practical usability of these fragments on large knowledge bases has been investigated. The CEL Reasoner [18] was as a result of these studies the first reasoner that could classify the mentioned knowledge bases from the life sciences domain in reasonable times.

Successful applications of the $\mathcal{EL}$ family increased investment in further work in this direction. The $\mathcal{EL}$ family now provides the basis for the profile OWL2 EL[1]. Moreover, there are now several other reasoners specifically tailored for the $\mathcal{EL}$ family, like Snorocket [16], TrOWL [22], CB [11] (which extends the $\mathcal{EL}^{++}$ algorithm to Horn $\mathcal{SHIQ}$), JCEL [17] (which is a Java implementation of CEL) and ELK [12, 14, 13, 15] (which is currently the only reasoner that can classify large ontologies from real-life applications within only a few seconds). A comprehensive study comparing the performace of several reasoners on large bio-medical knowledge bases has been presented in [9]. A more recent comparison can be found in the experimental results section of [15].

In the present paper we introduce the $\mathcal{EL}$epH$\mathcal{H}$ant reasoner,[2] a consequence-based reasoner for the $\mathcal{EL}^+$ fragment of DLs. It is the successor of our prototype reasoner CHEETAH [20]. Our motivation to develop $\mathcal{EL}$epH$\mathcal{H}$ant is on the one hand

[1] http://www.w3.org/TR/owl2-profiles/#OWL_2_EL

[2] http://code.google.com/p/elephant-reasoner

push the performance of OWL EL reasoning further by investigating different optimizations, and on the other hand provide a reasoner with a small footprint that can be used on platforms with limited memory and computing capabilities for applications like embedded reasoning [10].

The paper is organized as follows: After describing the implementation details, we present the results of our experiments for classification of large ontologies from the biomedical domain. Although there is still room for improvements like multithreading, the experimental results show that the performance is still promising.

2 Implementation Details

The $\mathcal{EL}$ep$\mathcal{H}$ant reasoner is the successor of the CHEETAH prototype [20]. The motivation for the CHEETAH prototype was to improve the worst-case complexity of the $\mathcal{EL}^+$ classification algorithm by using the linear-closure algorithm from databases. There we used a modified version of the linear-closure algorithm for computing the closure of atomic concepts under the axioms of the knowledge base, i.e., for saturating the knowledge base. For each concept name, we kept a counter that is used to check whether it already satisfies the left-hand side of an axiom. The experimental results there showed that the overhead of this method was too big compared to the performance gain.

The $\mathcal{EL}$ep$\mathcal{H}$ant reasoner does not use this method. Instead, it implements the consequence-based algorithm used in ELK [12, 15] with some small modifications. It differs from ELK in the implementation of the inference rules, and in scheduling of the input and derived axioms.

As also pointed out in [15], the most time consuming phase of consequence-based classificiation is the phase where the inference rules are applied for saturating the knowledge base. Therefore it is important to optimize this phase for getting a good performance. The original saturation algorithm uses a queue for keeping the scheduled axioms. Our experiments showed that the queue operations take a considerable amount of time since these operations are executed millions of times for classifying large knowledge bases like SNOMED CT. Removal from the front and addition at the back are indeed costly operations compared to adding and removing on only one side since in the former case the links to the next queue element have to be maintained properly. In order to avoid this overhead, in $\mathcal{EL}$ep$\mathcal{H}$ant we keep the scheduled axioms in a stack. Our experiments show that for some of the ontologies this results in a larger number of derivations, but the overall performance becomes better. The performance difference between queue and stack processing is shown in Table 2.

One other optimization that $\mathcal{EL}$ep$\mathcal{H}$ant implements is that it uses the told subsumer information as input axioms for initializing the stack. For each concept name A, instead of using axioms of type $A \sqsubseteq A$ as input, it uses $A \sqsubseteq B$, where B is a told subsumer of A.

Apart from these basic optimizations, $\mathcal{EL}$ep$\mathcal{H}$ant also implements some of the optimization techniques introduced in [15]. It implements the optimization

of rules for decomposing conjunctions and existential restrictions. More precisely, it does not decompose a conjunction if it has been previously derived by conjunction introduction. Similarly, if an existential restriction has previously been derived via the existential introduction rule, it does not decompose this existential restriction. Unlike ELK, when an existential restriction is decomposed, $\mathcal{EL}$ep$\mathcal{H}$ant does not schedule a so-called *init* axiom, it directly schedules an axiom with the filler of the existential on both sides. $\mathcal{EL}$ep$\mathcal{H}$ant does not yet support concurrent reasoning, but it is planned for future versions.

During saturation, we often need to do a lookup to check if a concept is subsumed by another, or if an axiom has already been processed before, etc. For such operations, we need an efficient data structure. Besides doing a lookup, we also often need to insert new elements to these data structures, like adding a new subsumer to the subsumers list, marking an axiom as processed, etc. But we never delete elements from these data structures. We also sometimes need to iterate over the elements of these data structures. For these operations, the most appropriate data structure is a an associative array. As associative array implementation, we used the Judy array library[3] for `C`. The library is optimized to avoid CPU cache misses as often as possible. Its memory consumption scales smoothly with number of entries, even when the keys are sparsely distributed.

For some of the data structures that are traversed often during saturation, $\mathcal{EL}$ep$\mathcal{H}$ant keeps double indexes. For instance, the list of subsumers of a concept is stored once as a conventional array and once as Judy array. If during saturation we need to check whether a concept is subsumed by another, we do a lookup in the Judy array. But if we need to traverse the subsumer list, for instance in existential introduction rule, we use the conventional array.

Just like its predecessor, the $\mathcal{EL}$ep$\mathcal{H}$ant reasoner is implemented in the `C` programming language. The reason why CHEETAH was implemented in `C` is the large amount of memory required by the algorithm that it implements. Due to the large number of concept names and axioms in real-life ontologies, this algorithm requires a large amount of memory and an efficient memory management. This is why we chose `C` as the implementation language. Although the $\mathcal{EL}$ep$\mathcal{H}$ant reasoner does not use this algorithm, large part of its code is based on the code of the CHEETAH prototype.

3 Experimental Results

In order to test the performance of $\mathcal{EL}$ep$\mathcal{H}$ant, we performed a series of experiments on large biomedical knowledge bases from real-life applications. We used the January 2013 international release of SNOMED CT by converting it to OWL functional syntax by the converter provided. Additionally, we used 6 ontologies, namely GO1, FMA, ChEBI, EMAP, Molecule Role and Galen 7 provided in the test ontology suite on the ELK web page.[4] We did not use the Galen8, GO2 and Fly Anatomy ontologies provided there since they contain disjointness axioms,

[3] http://judy.sourceforge.net
[4] http://code.google.com/p/elk-reasoner

	A	r	$C \sqsubseteq D$	$C \equiv D$	$r \sqsubseteq s$	$r_1 \circ r_2 \sqsubseteq s$	Trans(r)
GO1	19468	1	28869	0	0	0	1
FMA	80469	14	126544	0	3	0	1
SNOMED CT	296518	57	228954	67563	12	1	0
ChEBI	31160	9	67182	0	0	0	2
EMAP	13731	1	13730	0	0	0	0
Molecule Role	9217	2	9627	0	0	0	2
Galen7	28482	964	27820	19326	1357	385	0

Table 1. Number of concepts, roles and different types of axioms.

which is not yet supported by $\mathcal{EL}$ep$\mathcal{H}$ant. The metrics of the used ontologies are shown in Table 1.

In order to measure the effects of optimizations described in Section 2, we have run a series of experiments. The experiments were run on a computer with Intel Core i3 processor with 2.1 GHz clock speed, 8 GB of main memory and Linux operating system with 3.2.0 kernel. The results were obtained as average of 5 runs per setting per ontology. We tested the performance gain obtained by using a stack instead of a queue and performance gain obtained by initializing the stack with told subsumer information. Runtimes in miliseconds, and also total and unique number of derivations obtained from these experiments are presented in Table 2. Test results for the setting where a queue is used are marked with 'queue', results for the setting where a stack is used with 'stack' and the results for setting where a stack is used and the stack is initialized with told subsumer information is marked with 'stack+told'. The results show that except for the EMAP and Molecule Role ontologies, using a stack improves the runtime performance even if the number of total or unique derivations does not change. This is due to the overhead of enqueue and dequeue operations compared to the push and pop operations. It is also seen in the table that using the told subsumer information for preparing the input axioms always improves the runtime performance and reduces both the number of total and unique derivations.

We have also run a series of tests for comparing the loading and classification performances of $\mathcal{EL}$ep$\mathcal{H}$ant to that of ELK. We have run ELK 5 times for each ontology with the `-XX:+AggressiveHeap` parameter and taken the average of these runtimes. The results presented in Table 3 show that ELK classifies the SNOMED CT ontology faster, but needs more time to load it compared to $\mathcal{EL}$ep$\mathcal{H}$ant. For all smaller ontologies, both classification and loading times of $\mathcal{EL}$ep$\mathcal{H}$ant are slightly shorter. We conjecture that this is due to the overhead of starting the Java virtual machine. In terms of memory usage the performance of $\mathcal{EL}$ep$\mathcal{H}$ant is quite good as well. For classifying SNOMED CT, the maximum memory usage is around 420MB as the Linux `top` command shows. For ELK, it is around 1.6GB with the aggressive heap option and around 1GB without this option.

	classification time	total derivations	unique derivations
GO1			
queue	94	261302	206205
stack	65	261302	206205
stack+told	61	241834	186737
FMA			
queue	637	1314973	1312745
stack	373	1314973	1312745
stack+told	359	1234504	1232276
SNOMED CT			
queue	24013	23299190	12576268
stack	16396	21373957	12576268
stack+told	16170	20956134	12346881
ChEBI			
queue	661	1250852	1022277
stack	482	1250852	1022277
stack+told	471	1219692	991117
EMAP			
queue	8	27461	27461
stack	40	27461	27461
stack+told	6	13730	13730
Molecule Role			
queue	10	32857	32391
stack	23	32857	32391
stack+told	11	23640	23174
Galen7			
queue	1598	1658475	1068115
stack	1246	1715446	1068115
stack+told	1197	1658677	1055492

Table 2. Performance gain obtained by optimizations. Runtimes are in miliseconds.

	GO1	FMA	Molecule Role	ChEBI	EMAP	SNOMED CT	Galen7
	classification						
ELK	989	1564	657	1275	693	10674	2676
$\mathcal{EL}$ep$\mathcal{H}$ant	61	359	11	471	6	16170	1197
	loading + classification						
ELK	3338	7522	2695	3215	2368	23169	4883
$\mathcal{EL}$ep$\mathcal{H}$ant	284	1144	222	746	106	20032	1554

Table 3. Loading and classification times in miliseconds.

4 Concluding Remarks and Future Work

We have introduced the consequence-based $\mathcal{EL}^+$ reasoner $\mathcal{EL}$ep$\mathcal{H}$ant, described implementation details and presented experimental results.

Currently the $\mathcal{EL}$ep$\mathcal{H}$ant reasoner is still under heavy construction, and there is a number of improvements that we plan to do as future work. First of all, we are going to investigate the role of ordering derived axioms for reducing the number of derivations. We are going to check whether this can be implemented with a feasible overhead. We are going to implement concurrent reasoning in order to further improve the performance. We are going to extend the supported expressivity by allowing disjointness axioms. Last but not least, we are going to implement an OWL API wrapper using the Java native interface in order to make it compatible with ontology editors and other practical applications.

References

1. F. Baader. Terminological cycles in a description logic with existential restrictions. In G. Gottlob and T. Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 325–330. Morgan Kaufmann, 2003.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In L. P. Kaelbling and A. Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, (IJCAI 05)*, pages 364–369. Professional Book Center, 2005.
3. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proceedings of the Methods for Modalities Workshop (M4M-05)*, 2005.
4. F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Journal of Logic, Language and Information, Special Issue on Method for Modality (M4M)*, 2007. To appear.
5. S. Brandt. On subsumption and instance problem in $\mathcal{ELH}$ w.r.t. general tboxes. In V. Haarslev and R. Möller, editors, *Proceedings of the 2004 International Workshop on Description Logics, (DL2004)*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
6. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In R. L. de Mántaras and L. Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, (ECAI 2004)*, pages 298–302. IOS Press, 2004.
7. T. G. O. Consortium. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
8. R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, International, Northfield, IL: College of American Pathologists, 1993.
9. K. Dentler, R. Cornet, A. ten Teije, and N. de Keizer. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web Journal*, 2011. To appear.
10. S. Grimm, M. Watzke, T. Hubauer, and F. Cescolini. Embedded $\mathcal{EL}+$ reasoning on programmable logic controllers. In *Proceedings of the 11th International Semantic*

Web Conference (ISWC 2012), volume 7650 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, 2012.
11. Y. Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In C. Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence, (IJCAI 2009)*, pages 2040–2045, 2009.
12. Y. Kazakov, M. Krötzsch, and F. Simančík. Concurrent classification of $\mathcal{EL}$ ontologies. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *Proceedings of the 10th International Semantic Web Conference (ISWC'11)*, volume 7032 of *Lecture Notes in Computer Science*. Springer-Verlag, 2011.
13. Y. Kazakov, M. Krötzsch, and F. Simančík. ELK: a reasoner for OWL EL ontologies. System description, University of Oxford, 2012. available from http://code.google.com/p/elk-reasoner/wiki/Publications.
14. Y. Kazakov, M. Krötzsch, and F. Simančík. ELK reasoner: Architecture and evaluation. In *Proceedings of the OWL Reasoner Evaluation Workshop 2012 (ORE'12)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
15. Y. Kazakov, M. Krötzsch, and F. Simančík. The incredible ELK. 2013. available from http://code.google.com/p/elk-reasoner/wiki/Publications.
16. M. Lawley and C. Bousque. Fast classification in Protege: Snorocket as an OWL2 EL reasoner. In *Proceedings of Australasian Ontology Workshop*, 2010.
17. J. Mendex. jcel: A modular rule-based reasoner. In *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*, volume 858 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
18. J. Mendez and B. Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. In B. C. Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
19. A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, 1997.
20. B. Sertkaya. In the search of improvements to the $\mathcal{EL}+$ classification algorithm. In *Proceedings of the 24th International Workshop on Description Logics (DL 2011)*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
21. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. Ph.D. dissertation, Institute for Theoretical Computer Science, TU Dresden, Germany, 2009.
22. E. Thomas, J. Z. Pan, and Y. Ren. TrOWL: Tractable owl 2 reasoning infrastructure. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, (ESWC 2010)*, volume 6089 of *Lecture Notes in Computer Science*, pages 431–435. Springer-Verlag, 2010.