Two Phase Description Logic Reasoning for Efficient Information Retrieval

Zsolt Zombori[1]

1 **Department of Computer Science and Information Theory**
Budapest University of Technology and Economics
Budapest, Magyar tudósok körútja 2. H-1117, Hungary
zombori@cs.bme.hu

1 Overview

Description Logics (DLs) [1] is family of logic languages designed to be a convenient means of knowledge representation. They can be embedded into FOL, but – contrary to the latter – they are decidable which gives them a great practical applicability. A DL knowledge base consists of two parts: the TBox (terminology box) and the ABox (assertion box). The TBox contains general background knowledge in the form of rules that hold in a specific domain. The ABox stores knowledge about individuals. For example, let us imagine an ontology about the structure of a university. The TBox might contain statements like "Every department has exactly one chair", "Departments are responsible for at least 4 courses and for each course there is a department responsible for it". In contrast, the ABox might state that "The Department of Computer Science is responsible for the course Information Theory" or that "Andrew is the chair of the the Department of Music".

As DL languages are being used more and more frequently, there is an increasing demand for efficent automated reasoning services. Some reasoning tasks involve the TBox only. This is the case, for example, when we want to know what rules follow from the ones that we already know, or we want to verify that the model of a certain domain does not contain obvious mistakes in the form of contradictions and unsatisfiable concepts. We might want to make sure that there are not so many restrictions on the chair that it is impossible to be one (which is the case if he has to spend 70 percent of his time on research an another 70 percent on teaching). Other reasoning problems use both the ABox and the TBox: in such cases we might ask if a certain property holds for a certain individual (*instance check* – Is Andrew a chair?) or we might want to collect all individuals satisfying a given property (*instance retrieval* – What are the courses taught by the Department of Music?).

The Tableau Method [1] has long provided the theoretical background for DL reasoning and most existing DL reasoners implement some of its variants. Typical DL reasoning tasks can be reduced to concept consistency checking and this is exactly what the Tableau Method provides. While the Tableau itself has proven to be very efficient, the reduction to consistency check is rather costly for some ABox reasoning tasks. In particular, instance retrieval (i.e., to enumerate those individuals that belong to a given concept) requires running the Tableau Method for every single individual that appears in the knowledge base. Several techniques have been developed to make tableau-based reasoning more efficient on large data sets, (see e.g. [3]), that are used by the state-of-the-art DL reasoners, such as RacerPro [4] or Pellet [9].

Other approaches use first-order resolution for reasoning. A resolution-based inference algorithm is described in [5] which is not as sensitive to the increase of the ABox size as the tableau-based methods. The system KAON2 [8] is an implementation of this approach, providing reasoning services over the description logic language $\mathcal{SHIQ}$. The algorithm used in KAON2 in itself is not any more efficient for instance retrieval than the Tableau, but

several steps that involve only the TBox can be performed before accessing the ABox, after which some axioms can be eliminated because they play no further role in the reasoning. This yields a qualitatively simpler set of axioms which then can be used for an efficient, query driven data reasoning. For the second phase of reasoning KAON2 uses a disjunctive datalog engine and not the original calculus. Thanks to the preprocessing, query answering is very focused, i.e., it accesses as little part of the ABox as possible. However, in order for this to work, KAON2 still needs to go through the whole ABox once at the end of the first phase.

2 Research Direction

In my PhD work I try to develop algorithms that can be used for reasoning over large ABoxes while the TBox is relatively small. These assumptions do not hold for all ontologies, but there are some very important examples when this is the case: one can, for instance, think of searching the WEB in the context of a specific, well characterized domain.

It seems that the complexity comes from two sources: on one hand the TBox contains complex background knowledge that requires sophisticated reasoning, and on the other the size of the ABox makes the sophisticated algorithm too slow in practice. An important lesson to be learned from KAON2 is that we might be able to cope with these two sources separately: let us perform the complex reasoning on the TBox – which we assume to be small – and turn it into a syntactically simpler set of rules before accessing the ABox. Afterwards, the simpler rules can be used for a focused, query driven ABox reasoning.

It is not clear how to separate the reasoning for the Tableau. This algorithm tries to build a model of the knowledge base, but a model of a small part of the knowledge base is not necessarily useful for constructing a model of the whole. Resolution approaches are more suitable: we can deduce implicit consequences of the axioms in one way at the beginning and then deduce further consequences in another way. In particular, we will be interested in solutions where we start with a bottom-up strategy and finish with a focused top-down strategy.

To perform first-order resolution, we need to transform the initial axioms to first-order clauses. While the initial knowledge base does not contain function symbols (there are no functions in DLs), existential restrictions and minimum restrictions in the TBox translate to existential quanifiers, which are eliminated by introducing new function symbols, called *skolem functions*. This is problematic, because termination is hard to guarantee if we can obtain terms of ever increasing depth. Furthermore, some top-down reasoning algorithms (top-down reasoning is a must if the ABox is really large), such as datalog only work if there are no function symbols. For this reason, it is very important to find some way to eliminate function symbols before performing the data reasoning. Note that this is intuitively very possible: the ABox does not contain any knowledge about functions since they were introduced by us during clausifying the axioms from the TBox. Hence, everything that is to know about function symbols is in the clauses derived from the TBox and whatever role they play, they should be able to play it at the beginning of the reasoning.

3 Two Phase Reasoning

The above considerations motivate a two phase reasoning algorithm. In the first phase we only work with the clauses derived from the TBox. We use a bottom-up algorithm, deduce lots of consequences of the TBox, in particular all the important consequences of the clauses

containing function symbols. By the end of the first phase, function symbols can play no further role and hence the clauses containing them can be eliminated. The second phase now begins and the reduced clause set can be used for a focused, top-down reasoning on the ABox.

This separation of TBox and ABox reasoning is only partially achieved in [8]. By the end of the first phase, we can only eliminate clauses with term depth greater than one. So, while function symbols persist, there is no more nesting of functions into each other. In order for the second phase to work, all function symbols are eliminated using a *syntactic* transformation: for every function symbol and every constant in the ABox a new constant is introduced. Note that this step involves scanning through the ABox and results in adding new constants whose number is linear in the size of the ABox.

Reading the whole ABox even once is not a feasible option in case the ABox contains billions of assertions or the content of the ABox changes so frequently that on-the-fly ABox access is an utmost necessity. Such scenarios include reasoning on web-scale or using description logic ontologies directly on top of existing information sources, such as in a DL based information integration system.

4 Results

I started my PhD at Budapest University of Technology in September 2009. I work as member of a team developing the DLog DL data reasoner [7], available to download at `http://www.dlog-reasoner.org`. This is a resolution based reasoner, built on principles similar to KAON2. One difference is that instead of a datalog engine, we use the reasoning mechanism of the Prolog language [2] to perform the second phase [6]. Reasoning with function symbols using Prolog is possible, unlike the datalog engine, but for considerations about termination it is equally important to eliminate function symbols during the first phase.

I work to provide DLog with a purely two phase reasoning algorithm. In [10] I presented a modified resolution calculus for the $\mathcal{SHIQ}$ language that allows us to perform more inferences in the first phase (compared with KAON2), yielding a simpler TBox to work with in the second phase. Namely, the new calculus ensures that no function symbols remain at all, without the need to go through the ABox. The modification makes the first phase somewhat slower, however, the speed of the second phase becomes independent of the amount of data that is irrelevant to the query. The greater the ABox the better DLog performs compared to its peers. Another great advantage of DLog is that its architecture allows for storing the ABox in an external database that is assessed through direct database queries.

Afterwards, I worked on a new DL calculus ([12] and [11]) where we move resolution from first-order clauses to DL axioms, saving many intermediary transformation steps. Even if the speed of the first phase is not as critical as that of the second, this optimisation is important. With the increase of the TBox the first phase can become hopelessly slow, such that DLog is impossible to use. Making the first phase faster slightly increases the critical TBox size within which it is still worth reasoning with DLog. On the other hand, the DL calculus is a complete algorithm for TBox reasoning. It is novel in that the metodology is still resolution, but the inference rules are given directly for DL expressions. It is not as fast for TBox reasoning as the Tableau, but it provides an alternative and I hope that it will motivate research in the area. I tried to extend the DL calculus to ABox reasoning, but I have not yet been successful in doing that.

I managed to make DLog support the $\mathcal{RIQ}$ language which extends $\mathcal{SHIQ}$ with complex

role hierarchies. I also tried to incorporate nominals into the reasoner, i.e., move from $\mathcal{RIQ}$ to $\mathcal{ROIQ}$, however, this work yielded only negative results. It turned out that the presence of nominals blurs the distinction between TBox and ABox (the whole content of the ABox can be rephrased using TBox axioms), hence there is no possibility of separating terminology and data reasoning into two separate phases. These last two results have not yet been published.

5 Current Work

I would like to fully explore the relation between DLog and the $\mathcal{SROIQ}$ language on which the new web ontology language OWL 2 is based. This exploration involves partly to extend DLog towards more expressive language constructs and partly to understand its limitations.

I also am working to better explore the complexity of our algorithms. Bottom up reasoning in the first phase is very costly: it is at most triply exponential in the size of the TBox, although our experiments indicate that there could be a better upper bound. We also need to better explore the clauses that are deduced from the TBox. While our main interest is to eliminate function symbols, we deduce other consequences as well. Some of them make the data reasoning faster, some of them do not, and we cannot yet well characterize them.

6 Concluding Remarks

With the proliferation of knowledge intensive applications, there is a vivid research in the domain of knowledge representation. Description Logics are designed to be a convenient means for such representation task. One of the main advantages over other formalisms is a clearly defined semantics. This opens the possibility to provide reasoning services with mathematical rigorousness.

My PhD work is concerned with Description Logic reasoning. I am particularly interested in ABox reasoning when the available data is really large. This domain is much less explored than TBox reasoning. Nevertheless, reasoning over large ABoxes is useful for problems like web-based reasoning.

I am one of the developers of the DLog data reasoner which implements a two phase reasoning: the first phase uses complex reasoning to turn the TBox into simple rules, while the second phase is geared towards fast query answering over large ABoxes. DLog currently supports the $\mathcal{SHIQ}$ DL language, but we plan to extend it as far as $\mathcal{SROIQ}$, the logic behind OWL 2.

References

1 F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2004.

2 Alain Colmerauer and Philippe Roussel. *The birth of Prolog.* ACM, New York, NY, USA, 1996.

3 V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004, Whistler, BC, Canada, June 2-5*, pages 163–173, 2004.

4 V. Haarslev, R. Möller, R. van der Straeten, and M. Wessel. Extended Query Facilities for Racer and an Application to Software-Engineering Problems. In *Proceedings of the 2004*

International Workshop on Description Logics (DL-2004), Whistler, BC, Canada, June 6-8, pages 148–157, 2004.

5 Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning for Description Logics around SHIQ in a resolution framework. Technical report, FZI, Karlsruhe, 2004.

6 Gergely Lukácsy and Péter Szeredi. Efficient description logic reasoning in Prolog: the DLog system. *Theory and Practice of Logic Programming*, 09(03):343–414, May 2009.

7 Gergely Lukácsy, Péter Szeredi, and Balázs Kádár. Prolog based description logic reasoning. In Maria Garcia de la Banda and Enrico Pontelli, editors, *Proceedings of 24th International Conference on Logic Programming (ICLP'08),Udine, Italy*, pages 455–469, December 2008.

8 Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany, January 2006.

9 Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Web Semant.*, 5(2):51–53, 2007.

10 Zsolt Zombori. Efficient two-phase data reasoning for description logics. In Max Bramer, editor, *IFIP AI*, volume 276 of *IFIP*, pages 393–402. Springer, 2008.

11 Zsolt Zombori. A resolution based description logic calculus. *Submitted to Acta Cybernetica*, 2009.

12 Zsolt Zombori and Gergely Lukácsy. A resolution based description logic calculus. In Boris Motik Bernardo Cuenca Grau, Ian Horrocks and Ulrike Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford*, volume 477 of *CEUR Workshop Proceedings*, Oxford, UK, July 27-30 2009.