Kalpa Publications in Computing

Volume 10, 2019, Pages 15–19

Selected Student Contributions and
Workshop Papers of LuxLogAI 2018

A Roadmap to Gradually Compare and Benchmark Description Logic Calculi

Fred Freitas
[1] Centro de Informática, Universidade Federal de Pernambuco (CIn/UFPE)
fred@cin.ufpe.br

Abstract

DL reasoners were developed with cutting-edge performance, implementing plenty of specific optimization techniques over tableaux-based methods, which took over the field. However, promising methods may have been neglected in such a scenario, in which the tough competition is often focused on gains through optimizations. Therefore, perhaps there is still room available for "basic research" on DL reasoning. The purpose of this work is to stimulate research on trying out DL calculi other than tableaux. Such endeavors should be carried out by making a careful, detailed comparison between tableaux and other inference methods in a systematic way: first starting with simpler languages (like $\mathcal{ALC}$) without any optimizations. Then gradually including optimizations and comparing them; and continuing these interactive steps: enhancing language expressivity, including optimizations, and testing until reaching the most expressive DL fragments such as $\mathcal{SROIQ}$. The comparison can also be done by in terms memory usage and algorithm asymptotic analysis, with worst and average cases, etc. The rationale is identifying whether there are fragments which are more suitable to certain inference methods, as well as which aspects or constructs (e.g., the costliest combinations, which usually involve inverses, nominals, equalities, etc) are sensitive to which calculus.

1 Introduction

The most expressive layer of the Semantic Web is based on Description Logic (DL) (Baader, 2003). This fact naturally brought about a larger interest in this family of formalisms and their languages. The Semantic Web choice for DL was surely based in two main pros from these formalisms: its rich expressiveness and its inherent reasoning possibilities.

Concerning this last aspect, DL reasoners were developed with cutting-edge performance, implementing plenty of specific optimization techniques over tableaux-based methods, which took over the field for years. On the other hand, promising methods may have been neglected in such a scenario, in which the tough competition is often focused on gains through optimizations. Therefore, perhaps

C. Benzmüller, X. Parent and A. Steen (eds.), LuxLogAI 2018 (Kalpa Publications in Computing, vol. 10), pp. 15–19

there is still room available for "basic research" on DL reasoning, in the sense that other efficient calculi need to be adapted to DL, tuned and tested.

The purpose of this work is to stimulate research on trying out DL calculi (other than tableaux) by making a careful, detailed comparison between tableaux and other inference methods in a systematic way: first starting with simpler languages (like $\mathcal{ALC}$) without any optimizations. Then gradually including optimizations and comparing them; and continuing these interactive steps: enhancing language expressivity, including optimizations, and testing until reaching the last advances on optimizations and the most expressive DL fragments such as $\mathcal{SROIQ}$ (Kroetsch, 2010).

The comparison can also be done by in terms of two other additional aspects: memory usage and algorithm asymptotic analysis, with worst and average cases, etc. The rationale is identifying whether there are fragments which are more suitable to certain inference methods, as well as which aspects or constructs (e.g., the costliest combinations, which usually involve inverses, nominals, equalities, etc) are sensitive to which calculus.

This article is organized as follows. Section 2 discusses adapting first-order logic (FOL) calculi to DL in its aspects. Section 3 describes an iterative process to compare and benchmark DL calculi and their respective reasoners with tableaux solutions (and even among themselves). Section 4 presents conclusions and future work.

2 Adapting First-Order Calculi to DL

Conceiving DL systems by adapting of first-order logic (FOL) ones is, unfortunately, not a trivial task. The traps and pitfalls along the way, together with the subtleties involved, like the risk of non-termination, and the consequent need for proofs of completeness, soundness and termination demand plenty of investigation and care in their design. The aim of this section is therefore to provide a non-exhaustive discussion based on experience to facilitate the job for researchers who are knowledgeable in FOL reasoning and are willing to adapt a FOL inference system to DL.

The adaptation process is different when clausal inference systems are being dealt with. For such systems, the steps involved in this transformation are usually representation (without variables, when possible, like the usual DL representation), and normalization (in which a normal form needs to be defined, so as to facilitate reasoning and/or save memory). For any system, being it clausal or non-clausal, reasoning, formalization and benchmarking are the remaining steps.

Reasoning requires solutions for at least two main issues:

- Taking into account the substitution of Skolem functions by some mechanism which can guarantee soundness and completeness, i.e., assuring that the system will display the same result (true, false) for a same query entered for a FOL reasoner
- Dealing with cycles constitutes a key aspect, which have to be looked into carefully. The lack of the binomial Skolem functions-unification demand a blocking mechanism to ensure termination, as shown in (Freitas & Otten, 2016) and (Freitas, 2017). Blocking for simple languages such as $\mathcal{ALCN}$ usually does not constitute a barrier. It suffices checking if the set of concepts (τ) of the last two instances created in the cycle is not changing (Schmidt, 2007). For more expressive DLs, e.g., the ones with inverse roles, more sophisticated forms of blocking are called for, like dynamic and double blocking.

The author started an effort to use connection calculi (Bibel, 1993) for DL reasoning, which arrived at the formalization of a DL connection calculus for $\mathcal{ALC}$. Thus, an example on how blocking was introduced in the calculus is presented below in a quick manner (due to text space restrictions). Detailed explanations can be found at (Freitas & Otten, 2016).

Example 1 ($\mathcal{ALC}$ θ-CM blocking). The treatment of cycles in the original connection method (CM) is hidden behind a multiplicity function μ (Bibel, 1993), and, thus, not shown in the formalized calculus. Therefore, with Skolem functions ruled out and unification replaced by a new type of substitutions (θ-substitutions, which play a similar role than unification), the new *Copy rule (Cop),* shown in Figure 1, explicits the cycle treatment by copying a required column in the matrix (as CM represents formulae as columns in a matrix). More important, it brings the blocking condition, hence, regulating the generation of new individuals.

$$Copy\ Rule\ (Cop)\ \frac{C \cup \{L_1\}, M \cup \{C_2^{\mu}\}, Path \cup \{L_2\}}{C \cup \{L_1\}, M, Path \cup \{L_2\}}$$
$$with\ L_2 \in C_2, \mu \leftarrow \mu + 1, and$$
$$\left(x_{\mu}^{\theta} \notin N_O\ and\ \tau(x_{\mu}^{\theta}) \nsubseteq \tau(x_{\mu-1}^{\theta})\right)(blocking\ condition), \theta(L_1) = \theta(\overline{L_2})$$

Figure 1. The *Copy Rule* for the $\mathcal{ALC}$ θ-CM connection calculus (Freitas F. O., 2016)

The new calculus naturally led to the production of a reasoner for the DL $\mathcal{ALC}$, RACCOON (ReAsoner based on the Connection Calculus Over ONtologies) (Freitas, 2017). RACCOON was benchmarked against the most well-known DL reasoners Hermit (Glimm, 2014), FaCT++ (Tsarkov, 2006) and Konclude (Steigmiller, 2014)

In our performance comparisons, the sensation was indeed to compare apples and oranges. For instance, Konclude (Steigmiller, 2014) uses all CPU cores and multi-threading, while the other reasoners do not. RACCOON runs only over $\mathcal{ALC}$, while the others were conceived and are prepared to deal with "larger"and more expressive DLs, like $\mathcal{SROIQ}$. Consequently, they may have lost time on applying techniques which are suitable for more expressive fragments, but only waste processing time in $\mathcal{ALC}$. Other relevant aspect in this discussion is parsing: RACCOON seems to have the best parsing in our experiments and could process the smaller ontologies much faster than the others. But we wanted to assess reasoning, in the experiments, not parsing. Such aspects hamper a fair comparison among DL calculi. In the next section, a discussion on how to effectively compare than in a reasonable manner is proposed.

3 Proposal: A Process to Compare and Benchmark DL Calculi and their Respective Reasoners

A clear iterative process for comparing and benchmarking other calculi and their respective reasoners with tableaux (and even among themselves) is envisaged and is depicted in Figure 2. An initial remark about parsing is needed here, As mentioned earlier, to make a fair comparison between two inference systems may lead to either use the same parser or, in case this alternative is not available, to compute parsing time separately.

The comparison should start with the basic DL language $\mathcal{ALC}$ (with cycles), which presents two interesting features as a starting point for the comparison: it combines a good expressiveness with tractable reasoning[*].

The comparison begins with a version of the reasoner for the new calculus and a similar based on tableaux (or even other calculus), both without any implemented optimizations. Then, they are benchmarked and compared. The next step is to include *one* optimization at a time, and then benchmark again. After iteratively analyzing all possible optimizations for the current DL fragment (which may

[*] The simpler language $\mathcal{EL}$ can be avoided as a starting point since it is solvable even with a finite automata approach, thus falling into a smallest complexity category (Baader, 2003).

encompass many or all combinations of optimizations), the next step, is experimenting the next DL fragment (in terms of expressiveness), by commencing with the reasoners for this fragment again without any optimizations.

However, sometimes it is not easy to devise how to implement in a calculus a particular optimization, which is a crucial step for the comparison. Indeed, here lies an advantage of a calculus which consists of a set of rules, like tableaux and – more recently – the connection calculus (Bibel, 1993) – instead of an algorithm: when a new optimization or technique needs to be added to the calculus, it is possible to just augment the rule set with one more rule or constraint in a single rule or a rule subset for getting the job done.

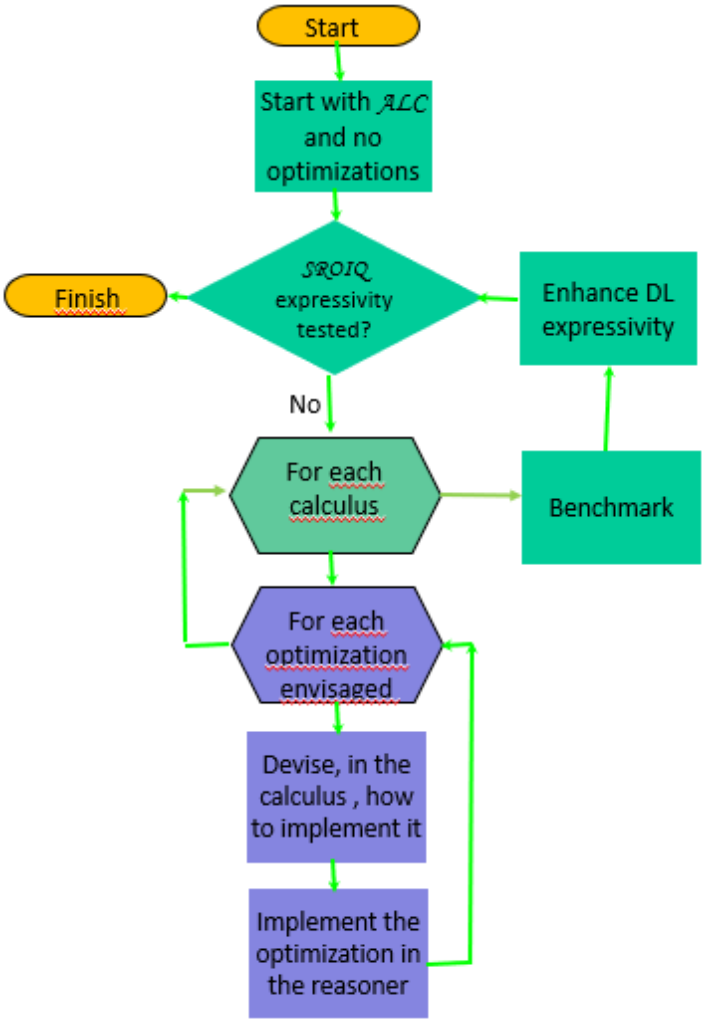

Figure 2: A proposed iterative process to compare and benchmark DL calculi and their respective reasoners. A crucial step to accomplish such an analysis is to devise how to implement optimizations and techniques in a particular calculus. Calculus described as a rule set are easier to be modified than algorithmic ones.

4 Conclusions and Future Work

In this work, an iterative proposal for comparing and benchmarking DL calculi and reasoners has been sketched, which can be a solution for checking whether other calculus than tableaux could constitute effective competitors in terms of efficiency.

As for future work, concretely the idea is to start out with our DL connection calculus (Freitas & Otten, 2016) and reasoner, the RACCOON, which for now takes on $\mathcal{ALC}$, the basic DL fragment, and displayed a surprisingly effective performance in comparison with other well-established reasoners. A solution for dealing with equality and, therefore, processing instance in/equalities and cardinality restrictions has already been presented (Freitas & Varzinczac, 2018) and is currently under implementation. It will enable RACCOON to infer over $\mathcal{ALCHQ}=$ ontologies.

References

Baader, F. C.-S. (2003). *The Description Logic Handbook.* Cambridge: Canbridge University Press.

Bibel, W. (1993). *Deduction: Automated Logic.* London: Academic Press.

Freitas, F. M. (2017). RACCOON: A Connection Reaosner for ALC. *LPAR-21 - 21st International Conference on Logic for Programming Artificial Intelligence and Reasoning* . Maun, Botswana: Epic Series, EasyChair.

Freitas, F. O. (2016). A Connection Calculus over the Description Logic ALC. *Canadian Conference on Artificial Intelligence.* Victoria, CA: Springer Verlag.

Freitas, F. V. (2018). Cardinality Restrictions for Description Logic Connection Calculi. *International Joint Conference on Rules and Reasoning (RuleML+RR).* Luxembourg: Springer verlag.

Glimm, B. H. (2014, 3 3). HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoner, 3*, pp. 245-269.

J., O. (2017). nanoCoP: Natural Non-clausal Theorem Proving. *Interntional Joint Conference on Artificial Intelligence* (pp. 4924-4928). Buenos Aires: IJCAI.

Kroetsch, M. (2010). *Description Logic Rules. Studies on the Semantic Web.* (Vol. 008). Amsterdam: IOS Press.

Schmidt, R. T. (2007). Analysis of Blocking Mechanisms for Description Logics. *Workshop on Automated Reasoning.*

Steigmiller, A. L. (2014, 3 3). Konclude: System Description. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 1*, pp. 78-85.

Tsarkov, D. H. (2006). FaCT++ Description Logic Reasoner: System Description. *International Joint Conference on Automated Reasoning* (pp. 292-297). Berlin: Springer.