Tableaus and Automata for Description Logics

Jan Hladík
SAP Research CEC Dresden

Abstract. In this paper, we briefly present three different approaches at examining the relation between tableaus and automata, and possibly combining their advantages in the context of testing satisfiability of description logic concepts. The first approach tries to transfer the efficiency of tableaus to automata by performing the automata emptiness test with a tableau algorithm. The second one examines conditions which allow for obtaining \textsc{PSpace} complexity results from automata algorithms. The third one defines an abstract framework for tableau algorithms that makes it possible to derive from an algorithm formalised within this framework both an \textsc{ExpTime} automata algorithm and a terminating and practically efficient tableau algorithm.

1 Introduction

Description Logics (DLs) are a family of knowledge representation languages with a well-defined logic-based semantics and practically usable inference algorithms, which enable DL systems to deduce implicit information about the domain of interest from the explicitly provided facts [1]. The syntax of DLs is based on the notion of concepts, which represent classes of individuals, and roles, which stand for relations between individuals.

The pivotal DL inference problem, to which many other inference problems can be reduced, is the satisfiability problem, i.e. the question if a given concept expression is satisfiable. Two important classes of algorithms for deciding this problem are tableau and automata algorithms. Tableau algorithms (TAs) test the satisfiability of an input by trying to construct a tableau, i.e. a (pre-)model for the input according to a set of rules, thus going non-deterministically “top-down” from complex to simple concepts. Automata algorithms (AAs) construct an automaton accepting all models for the input and subsequently test the language accepted by that automaton for emptiness, going deterministically “bottom-up” from simple to complex concepts. Although the rules of a tableau algorithm look similar to the transitions of the corresponding automaton, TAs and AAs behave quite differently in practice and have complementary advantages and disadvantages: TAs are well-suited to show \textsc{PSpace} upper bounds, but require considerable effort to run in \textsc{ExpTime} (see e.g. [5]). The main reason for this is that, since they are going top-down, they have to make non-deterministic choices to find a model. Moreover, proving termination for expressive logics requires a cycle detection mechanism, which significantly complicates the correctness proofs. In practice however, implementations of tableau algorithms perform surprisingly
well, even for logics with “intractable” satisfiability problems. The reason for this is again the top-down direction, which makes it possible to restrict the attention to concepts that are relevant for the satisfiability of the initial concept.

Conversely, AAs employ a deterministic emptiness test of the (finite) automaton in order to decide the satisfiability problem. Thus, they handle non-determinism and infinite structures implicitly, which makes them very elegant from a theoretical point of view and well-suited for proving \( \text{ExpTime} \) upper bounds, but not for lower complexity classes. The main disadvantage of AAs is that they require exponential time even in the best case; hence, an unoptimised automata algorithm cannot be used as basis for an efficient implementation. The reason for this is the bottom-up direction, which avoids making non-deterministic choices, but requires examining an exponential number of states.

Our aim in this paper\(^1\) is to examine possibilities for reconciling the advantages of the two approaches in spite of these differences. In particular, we are trying to answer the following questions:

1. What are the precise relations between the data structures used in the different approaches? Can a tableau constructed by a TA serve as an input for the corresponding automaton?
2. Is it possible to achieve acceptable performance in practice with an automata algorithm using techniques stemming from tableau algorithms?
3. Is it possible to transfer the complexity results in either direction, i.e. can one obtain a \( \text{PSPACE} \) result from an AA or an \( \text{ExpTime} \) result from a TA?

## 2 Translation of Alternating Automata into DLs

In [7], we develop a way to translate two-way alternating automata into the rather inexpressive DL \( \text{FLEUI}_f \) such that the emptiness test of the automaton can be reduced to \( \text{FLEUI}_f \) satisfiability. Using this translation on automata resulting from an AA for the logic \( \text{ALCIO} \) [9] enables us to employ existing DL reasoners to decide \( \text{ALCIO} \) satisfiability although they cannot handle that logic natively.

We evaluate the performance of these reasoners on the output from the translation sketched above to find out if the optimisations implemented in tableau-based DL reasoners are able to remedy the exponential blow-up induced by the automata algorithm. Empirical evaluation shows that this is not the case: the steep increase in processing time in relation to the size of the input indicates that results of automatic translation procedures are much harder to process than knowledge bases occurring in real-life applications. On the positive side, an examination of the data structures used reveals that a tableau generated during the satisfiability test looks almost exactly like a strategy tree for the corresponding automaton, i.e. a witness for nonemptiness of the automaton. This again demonstrates the close relationship between alternating automata and tableau algorithms.

---

\(^1\) This paper presents the main results of the author’s PhD thesis [6], which also contains additional explanations and proofs for the claims made here.
3 Blocking Automata

The results from Section 2 show that the efficiency of tableau algorithms cannot easily be transferred to automata by translating an automaton into a DL for which satisfiability can be tested by a tableau algorithm. In [3], we use a different approach to improve the efficiency of automata algorithms by employing methods originating from tableaus: instead of first constructing the automaton and then testing its emptiness with a TA, we modify the automata construction and the emptiness test themselves using methods known from TAs. Our focus is also not on performance in practice, as in Section 2, but on the complexity class that can be obtained from an AA.

Concretely, we define blocking automata, a class of automata for which emptiness can be tested by constructing a run a) interleaved with the automaton itself, i.e. the transitions of the automaton are computed on-the-fly; b) in a non-deterministic, top-down, depth-first manner, keeping only one path in memory at a time; and c) ensuring that the maximum depth that has to be checked is bounded polynomially by the size of the input. This last condition is met by ensuring that every path of a certain length contains two “equivalent” states. It then remains to establish a polynomial bound for this length. This idea is an adaptation of the blocking technique [4] known from TAs, which allows for stopping the construction of a tableau after a blocked node has been reached.

For blocking automata, it is possible to obtain a PSPACE complexity result instead of the EXPTIME result that can be achieved with the “standard” deterministic bottom-up emptiness test. As an example for an application of this method, we show how blocking automata can be used to decide satisfiability of SI concepts [8] w. r. t. acyclic TBoxes in PSPACE.

4 Tableau Systems

After establishing the possibility of transferring complexity results from tableau to automata algorithms in Section 3, we search for possibilities to use automata in order to remedy the two main drawbacks of tableau algorithms: firstly, the need to find a blocking condition suitable for the corresponding logic and to prove soundness in presence of this blocking condition, and secondly the problem with obtaining EXPTIME upper bounds, i.e. a deterministic complexity class. Since the determinism of automata algorithms results from the bottom-up emptiness test, which in turn is the main reason for the poor performance of AAs in practice, modifying the TA itself in such a way that it runs in deterministic exponential time is not a promising approach: it would sacrifice the main advantage of tableaus (see also [5]).

Instead, our aim in this section is to define a general framework for EXPTIME logics, called Tableau Systems [2], from which both a practically usable tableau algorithm and a worst-case optimal automata algorithm can be derived, which avoids the need to construct the AA by hand. In order to achieve this, it is necessary to begin with a formalisation of the key properties of TAs such as tableau
rules and the trees they operate on. Afterwards, we show how tableau systems yield automata- and tableau-based algorithms. For the TA, an appropriate blocking condition can be obtained directly from the properties of the tableau system. It is thus possible to prove the correctness of blocking in general, which avoids the need to deal with it in the case of specific tableau systems. However, this feature makes it necessary to impose additional restrictions on the tableau rules that are allowed, in particular to require that rule applications must always add information (which renders it impossible to e.g. merge or remove nodes), and they must do so in such a way that they approach saturation, i.e. a situation in that no rule is applicable any more (for details, see [2]).

The usefulness of tableau systems is illustrated by two examples, one involving the well-known logic $SHIQ$ [8] and the other one proving a new ExpTime complexity result for the logic $SHIO$. Our decision to disallow removal of nodes necessitates formalising some rules that are usually defined as deterministic in a non-deterministic way. In an implementation, these rules should be modified to the standard, deterministic versions for efficiency reasons.

The correctness proofs for tableau systems are significantly simplified by the fact that they do not need to take a blocking condition into consideration. We therefore believe that the simplicity of the proofs justifies the additional overhead resulting from the formalisation of the algorithm within the tableau systems framework.

5 Conclusion

We conclude by summarising how the questions raised in Section 1 can be answered using the results presented in this paper.

1. Relations between data structures. In Section 2, we use a tableau algorithm to perform the emptiness test of alternating automata, and it turns out that a tableau generated by the TA looks almost exactly like a strategy tree for the corresponding automaton. This similarity of the structures is also exploited by the tableau systems framework in Section 4, where an essential step in the proof is to show that a tree that is generated by exhaustive application of the tableau rules can also serve as an input that is accepted by the corresponding automaton.

2. Transfer of optimisations from tableaus to automata. The translation of automata into DLs in Section 2 allows us to perform the emptiness test for an automaton with a tableau algorithm. However, our hope that the optimisations of the tableau-based reasoners could compensate for the overhead introduced by the automata construction have not materialised: empirical results show that the computation time increases exponentially in the size of the input. Thus, our translation approach does not lead towards a practically usable decision procedure for automata algorithms.

3. Transfer of complexity results. The framework of blocking automata we described in Section 3 gives rise to AAs that can show PSpace results. This framework avoids the inefficiency of automata algorithms by interleaving the automata construction with the emptiness test and thus restricting the con-
sidered transitions to those that are relevant for the emptiness problem. Using


techniques known from TAs, we establish a polynomial bound on the number


of transitions that have to be kept in memory at a time and thus obtain a


non-deterministic algorithm requiring space polynomial in the size of the input,


which yields a PSPACE upper bound for the emptiness test.


For the opposite direction, we have found a method to obtain an EXPTime


result from a tableau algorithm. Instead of modifying the algorithm itself, how-


ever, we developed a formalisation of tableau algorithms such that an EXPTime


automata algorithm can be derived from a formalised tableau algorithm. From


the characteristics of the rules, we can obtain an appropriate blocking condition


ensuring that the tableau algorithm induced by the tableau system terminates.


In summary, we have established that the structural similarity of tableau


and automatata algorithms is close enough to allow for the transfer of worst-case


complexity results in both directions. The question whether it is possible to


combine the desirable properties of both paradigms in such a way that a single


algorithm is both worst-case optimal and efficient in practice remains a subject


for research.


References


1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F.


Patel-Schneider, editors. The Description Logic Handbook: Theory, Implementation,


2. Franz Baader, Jan Hladik, Carsten Lutz, and Frank Wolter. From tableaux to


3. Franz Baader, Jan Hladik, and Rafael Peñaloza. Automata can show PSPACE


results for description logics. Information and Computation, Special Issue: First


International Conference on Language and Automata Theory and Applications


4. Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning


in terminological knowledge representation systems. In Proceedings of IJCAI-93,


5. Francesco Donini, Giuseppe De Giacomo, and Fabio Massacci. EXPTIME tableaux


for ALC. In Lin Padgham, Enrico Franconi, Manfred Gehrke, Deborah L. McGuin-


ness, and Peter F. Patel-Schneider, editors, Proceedings of DL‘96, number WS-96-05


7. Jan Hladik and Ulrike Sattler. A translation of looping alternating automata into


description logics. In Franz Baader, editor, Proceedings of the 19th Conference on


8. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive


description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov,


9. Ulrike Sattler and Moshe Y. Vardi. The hybrid µ-calculus. In Bernhard Nebel,


