EXPTIME Tableaux for $\mathcal{ALC}$

(Extended Abstract)

Giuseppe De Giacomo[1] and **Francesco M. Donini**[1] and **Fabio Massacci**[1,2]

[1] Dip. Di Informatica e Sistemistica, Univ. di Roma I "La Sapienza", Italy

{degiacomo,donini,massacci}@dis.uniroma1.it

[2] Computer Laboratory, Cambridge University, England

Fabio.Massacci@cl.cam.ac.uk

1 Motivations

We propose a tableaux calculus requiring simple exponential time for satisfiability of an $\mathcal{ALC}$ concept C wrt a TBox $\mathcal{T}$ containing general axioms of the form $C \sqsubseteq D$.

From correspondences with Propositional Dynamic Logic (PDL) it is known that this problem is in EXPTIME [Pratt, 1978; Vardi and Wolper, 1986]. However, an algorithm directly derived from the methods used to prove such a result would always require exponential time and space even in simple cases, e.g. when a simple model satisfying both $\mathcal{T}$ and C can be easily found.

On the other hand, proposed tableaux methods [Buchheit *et al.*, 1993], which explore a space of candidate models for $\mathcal{T}$ and C starting from simple ones, can take advantage of such cases. However, there can be an exponential number of possibly exponential-size candidate models. Hence, an algorithm based on tableaux methods requires doubly exponential time in the worst case.

We devise a refined tableaux calculus that integrates the techniques used in PDL with tableaux, thus achieving a tableaux-based procedure working in simple exponential time. In a nutshell, traditional tableaux methods close a branch only by "first principles" (atomic clashes), whereas our enhanced tableau exploits previously proved inconsistencies as additional lemmata to decide that a branch can be closed without having to find the same atomic clashes again.

For sake of brevity, we assume, wlog, that all concepts are expressed in negation normal form, i.e. we form concepts in $\mathcal{ALC}$ by means of the following syntax (A denotes a concept name, C and D arbitrary concepts, and R a role name):

$$C, D \quad ::= \quad \top \mid \bot \mid A \mid \neg A \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

We express a concept inclusion $C \sqsubseteq D$ in the TBox $\mathcal{T}$ as $\neg C \sqcup D = \top$ where $\neg C \sqcup D$ has also been expressed in negation normal form.

2 The tableaux method

The search for a model corresponds to a search in an AND-OR-tree, where OR nodes correspond to branching points in the tableau (i.e. alternative models) while AND-nodes correspond to individuals and the links between them.

This structure is reflected by our notation in which both branching points and links are explicit. Indeed we use *prefixed formulae*, that are triples of the form $\langle b \mid p : C \rangle$, where:

- the *segment* b is a binary string representing the choices made at the branching points;
- the *element* p is a string alternating integers (names for individuals) and role names, which represents a linked individual – e.g., the element $1R_1 2R_4 6$ represents the individual 6, which is an R_4-filler of the individual 2, which in turn is an R_1-filler of 1;
- C is an $\mathcal{ALC}$ concept.

Given two strings σ_1 and σ_2 (either segments or elements) $\sigma_1 \preceq \sigma_2$ means that σ_1 is a prefix of σ_2, and $\sigma_1 \prec \sigma_2$ means that $\sigma_1 \preceq \sigma_2$ and $\sigma_1 \neq \sigma_2$.

A *tableau* T is a set of prefixed formulae.

Given a tableau T and a segment b, we say that an *element p is present in b* if there is a prefixed formula $\langle b' \mid p : C \rangle$ in T such that $b' \preceq b$, and *p is new in b* if it is not present. We also say that a *segment b is present in* T if there is a prefixed formula $\langle b \mid p : C \rangle$ in the tableau for some p and some C. A *segment b_M is maximal for another segment b* in a tableau T if both b_M and b are present in T and b_M is the longest segment in T of which b is a prefix. Formally $b \preceq b_M$ and for all b' it is $b_M \not\prec b'$.

The rules we use for our tableau are shown in Fig. 1. We assume that rules are applied in the obvious way without unnecessary repetitions. For instance rule AND and SOME are never applied twice to the same prefixed formula, and rule KB never adds twice the same prefixed formula. Rule ALL can be applied again for every pRn that is present but of course not twice for the same b and pRn. In equal fashion we can apply rule OR again

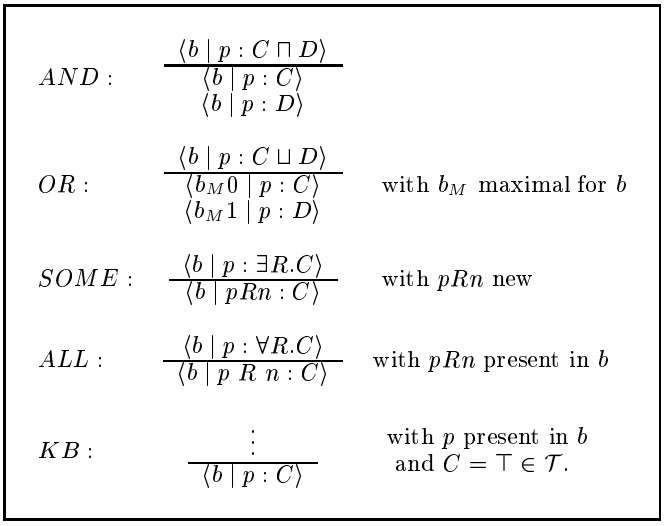

Figure 1: Rules for $\mathcal{ALC}$ Tableaux

for all maximal b_M that are present but never twice for the same b_M. Moreover, once we have applied it for a certain b_M, we do not apply the OR rule to the same formula with any other segment b', introduced at some subsequent stages, such that $b_M \preceq b'$.

The traditional tree-like shape of a tableau can be easily reconstructed by using segments. Branches can be reconstructed by collecting all formulae in all segments sharing one given maximal segment. Branching points are created by rule OR, where $b0$ corresponds to the left branch and $b1$ to the right one. In this case, the condition for the OR rule simply becomes "apply the OR rule only at the leaves of a subtree, and just once for each formula and each subtree".

A tableau T for a TBox $\mathcal{T}$ and a concept C is a set of prefixed formulae obtained by means of the rules above starting from $\langle 0 \mid 1 : C\rangle$ and using $\mathcal{T}$ in the rule KB.

Given a tableau T, we call *concepts of an element p along the segment b* – denoted by $\texttt{concept}\,(b \mid p)$ – the set of concepts C such that $\langle b' \mid p : C\rangle$ with $b' \preceq b$ is in the tableau. Formally:

$$\texttt{concept}\,(b \mid p) \stackrel{\text{def}}{=} \{C \mid \langle b' \mid p : C\rangle \in T \text{ and } b' \preceq b\}$$

We say that an element p is a *copy* of an element q in a segment b if $\texttt{concept}\,(b \mid p) = \texttt{concept}\,(b' \mid q)$ for some b'. An element is *reduced* if no rule with the exception of rule SOME can be applied to one of the formulae in which it appears, if also rule SOME cannot be applied the element is *fully reduced*.

A segment b of a tableau T is *completed* if all elements present in b are reduced, and for each element p which is not fully reduced a fully reduced copy of p is present in the tableau.

Definition 2.1 (Inconsistent set) *Let T be a tableau. For every segment b and every element p, we say that the set* $\texttt{concept}\,(b \mid p)$ *is an* inconsistent set *($\bot$-set) for T if – inductively – one of the following conditions holds:*

- *either* $\bot \in \texttt{concept}\,(b \mid p)$ *or* $A, \neg A \in \texttt{concept}\,(b \mid p)$ *for some concept name A (atomic clash);*
- *$S \subseteq \texttt{concept}\,(b \mid p)$ for some $\bot$-set S;*
- *both* $\texttt{concept}\,(b0 \mid p)$ *and* $\texttt{concept}\,(b1 \mid p)$ *are $\bot$-sets;*
- *for some element q, with $p \prec q$,* $\texttt{concept}\,(b \mid q)$ *is a $\bot$-set and* $\texttt{concept}\,(b \mid p) \neq \texttt{concept}\,(b' \mid p)$ *for all $b' \prec b$.*

What characterises a $\bot$-set are indeed the concepts that compose it and not the way it is constructed. Thus, if $\texttt{concept}\,(b \mid p)$ is a $\bot$-set then, obviously, all other $\texttt{concept}\,(b' \mid p') = \texttt{concept}\,(b \mid p)$ are $\bot$-sets too – this is a particular case of the second condition in the previous definition.

We envisage for the implementation an auxiliary data structure in which we collect all the $\bot$-sets found at each stage of the construction of the tableau.

Definition 2.2 (Closed segment) *A segment b in a tableau T is* closed *if there is an element p such that* $\texttt{concept}\,(b \mid p)$ *is a $\bot$-set for T.*

Intuitively, a concept C is satisfiable in a TBox $\mathcal{T}$ iff there is a tableau T for $\mathcal{T}$ and C, containing a segment b which is completed and not closed.

When more rules are applicable we follow the preference criteria below, which are useful both to simplify the proof of termination and to prove that only simple exponential time is needed in the worst case to terminate the search for a complete, non-closed segment.

1. Apply rules to a formula $\langle b \mid p : C\rangle$ only if all $q \prec p$ present in b are fully reduced.
2. Apply rule KB before other rules.
3. Apply rule AND before rule ALL.
4. Apply rule ALL before rule OR.
5. Apply rule OR before rule SOME.

With this criteria we can prove the following lemma, in which we leave implicit the reference to tableaux.

Lemma 2.1 (Stability) *Let p be an element present in a segment b. If for all formulae $\langle b \mid p : C\rangle$ the rule AND has been applied, then further rule applications do not change* $\texttt{concept}\,(b \mid p)$ *any more.*

Note that by applying OR to a formula $\langle b \mid p : C\rangle$, we may add formulae involving the same element p but different (longer) segments $b_M 0$, $b_M 1$. Since we apply rules to an element p only after its predecessors q with $q \prec p$

have been fully reduced, it is never the case that rule ALL applied to q introduces a formula involving element p. Rule ALL may only introduce elements of the form pR_in.

To complete the construction of our tableau we only need the conditions for termination: we *do not reduce* anymore the prefixed formulae of the segment b and element p iff one of the following two conditions holds:

1. p is a copy in segment b and $\texttt{concept}\,(b \mid p)$ will not change;
2. $\texttt{concept}\,(b \mid p)$ is a $\bot$-set.

Note that the preference between rules we propose is not essential; other proof strategies may also be devised. However there are some minimal requirements to guarantee EXPTIME behaviour and termination.

To guarantee termination it is important to check whether $\texttt{concept}\,(b \mid p)$ is not a copy before reducing prefixed formulae of the form $\langle b \mid p : \exists R.C\rangle$ using rule SOME. Our preference criteria guarantee that elements obey the property "once a copy, always a copy". However, any other preference criteria satisfying the same property would do. If one wants to use strategies selecting rules and formulae by some heuristics, then one can introduce the notion of "temporary copy": check whether $\texttt{concept}\,(b \mid p)$ is a copy and freeze the reduction of these formulae; temporary copies can be reactivated when new relevant formulae are introduced.

To achieve the exponential-time upper bound, the strategy must guarantee the condition for termination plus two other ones. First, it must check whether $\texttt{concept}\,(b \mid p)$ is an already found $\bot$-set before applying rule OR (branching) or applying rule SOME (generating new elements) to formulae with b and p. Second, if a set $\texttt{concept}\,(b \mid p)$ is found to be a $\bot$-set, then Def. 2.1 must be applied and all new $\bot$-sets thus generated must be stored for future checking.

3 Correctness and Complexity

The correctness and completeness of the method is a simple adaptation of the proofs of [Buchheit *et al.*, 1993]. A little care is necessary for closure of a branch due to the presence of a $\bot$-set, since we discard a branch before actually finding an atomic clash.

Theorem 3.1 *Let b be a segment in a tableau T. If an element p is present in b such that $\texttt{concept}\,(b \mid p)$ is a $\bot$-set for T, then there is no model for b.*

This can be proved by a simple induction or a "cut-and-paste" argument: each $\bot$-set is eventually generated by the presence of some atomic clash so when we meet a $\bot$-set we paste down – renaming elements – the previously found segments with the atomic clash responsible for the inconsistency of the set. This padding argument is then applied in the induction step of the proof. For instance if $\texttt{concept}\,(b \mid p)$ is a $\bot$-set because both $\texttt{concept}\,(b0 \mid p)$ and $\texttt{concept}\,(b1 \mid p)$ are $\bot$-sets, then we can paste below b both segments $b0$ and $b1$ (and recursively for $b0$ and $b1$ if they do not contain an atomic clash) thus obtaining a corresponding "traditional" closed tableau.

The proof that the worst-case complexity is indeed optimal is somehow more intricate; we sketch it briefly.

In the following we denote by n the size of $\mathcal{T}$ plus C, which we assume coincides with the number of sub-concepts occurring in $\mathcal{T}$ and C. We use c as an arbitrary constant (> 1).

Proposition 3.1 *The size of every element present in a segment is $O(2^{cn})$.*

Indeed, there are no more that 2^n sets of sub-concepts of $\mathcal{T}$ and C. If an element is longer that 2^n then it is a copy of one of its prefixes. See [Buchheit *et al.*, 1993].

Proposition 3.2 *The size of every segment is $O(2^{cn})$.*

In fact, the size of the longest segments is increased by the application of rule OR. The number of sub-concepts of the form $C \sqcup D$ is at most n. Hence for each element there are at most n possible applications of rule OR. Since there are $O(2^{cn})$ elements, the size of a segment is $O(n \cdot 2^{cn})$.

We say that $\texttt{concept}\,(b \mid p)$ is an *internal $\bot$-set* for T if both the segments $b0$ and $b1$ appear in T.

Proposition 3.3
Let T be a tableau. Every $\texttt{concept}\,(b \mid p)$ which is an internal $\bot$-set for T is different from any other internal $\bot$-set $\texttt{concept}\,(b' \mid q)$.

Indeed if this was not the case, the "second detected" would not have been an internal $\bot$-set. – as a result of the second condition for termination.

Proposition 3.4 *The number of internal $\bot$-sets in a tableau T is at most 2^n.*

This follows from the previous proposition and the fact that there are at most 2^n possible $\bot$-sets (since there are 2^n sets of sub-concepts of $\mathcal{T}$ and C).

Proposition 3.5 *The number of segments that are present in a tableau T is $O(2^{cn})$.*

To prove this proposition define the internal segments as the segments b such that $b0$ and $b1$ occur in T. Then the number of internal segments is at most the number of internal $\bot$-set, while the number of non-internal segments is the same as the one of internal segments.

Lemma 3.2 *The size of a tableau T is at most $O(2^{cn})$.*

Indeed, the number (size) of formulae $\langle b \mid p : C\rangle$ in T is bounded by the number (size) of segments ($O(2^{cn})$), the number (size) of elements ($O(2^{cn})$) an the number (size) of sub-concepts ($O(n)$).

As a consequence of the above lemma, we get our main result:

Theorem 3.3 *The proposed tableaux method terminates and returns an answer in simple exponential time.*

4 Discussion

In this paper we have presented a tableaux calculus for satisfiability of a concept wrt a TBox (and hence also for subsumption in a TBox) which works in worst-case exponential time. In fact we do not need to change substantially the "normal" construction used by tableaux which has proven to be reasonably effective in practice [Bresciani *et al.*, 1995]. The key point is to make use of an auxiliary data structure which is used store sets of concepts whose conjunction was already proved to be inconsistent.

Our main ideas behind our procedure can be used to device EXPTIME tableaux procedures for various extension of $\mathcal{ALC}$. In particular, our calculus can be easily modified to deal with an ABox as well.

References

[Bresciani *et al.*, 1995] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: Preliminary report. In Alexander Borgida, Maurizio Lenzerini, Daniele Nardi, and Bernhard Nebel, editors, *Working Notes of the 1995 Description Logics Workshop*, Technical Report, RAP 07.95, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", pages 131–139, Rome (Italy), 1995.

[Buchheit *et al.*, 1993] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[Pratt, 1978] V. R. Pratt. A practical decision method for propositional dynamic logic. In *Proceedings of the Tenth ACM Symposium on Theory of Computing (STOC-78)*, pages 326–337, 1978.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC-84)*.