TECHNISCHE UNIVERSITÄT DRESDEN

Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Pinpointing in Terminating Forest Tableaux

Franz Baader Rafael Peñaloza

LTCS-Report 08-03

Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Nöthnitzer Str. 46
01187 Dresden
Germany

Pinpointing in Terminating Forest Tableaux

Franz Baader
Theoretical Computer Science, TU Dresden, Germany
baader@tcs.inf.tu-dresden.de

Rafael Peñaloza*
Intelligent Systems, University of Leipzig, Germany
penaloza@informatik.uni-leipzig.de

Abstract

Axiom pinpointing has been introduced in description logics (DLs) to help the user to understand the reasons why consequences hold and to remove unwanted consequences by computing minimal (maximal) subsets of the knowledge base that have (do not have) the consequence in question. The pinpointing algorithms described in the DL literature are obtained as extensions of the standard tableau-based reasoning algorithms for computing consequences from DL knowledge bases. Although these extensions are based on similar ideas, they are all introduced for a particular tableau-based algorithm for a particular DL.

The purpose of this paper is to develop a general approach for extending a tableau-based algorithm to a pinpointing algorithm. This approach is based on a general definition of "tableau algorithms," which captures many of the known tableau-based algorithms employed in DLs, but also other kinds of reasoning procedures.

1 Introduction

Description logics (DLs) [2] are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as

*Funded by the German Research Foundation (DFG) under grant GRK 446.

natural language processing, configuration, databases, and bio-medical ontologies, but their most notable success so far is the adoption of the DL-based language OWL [14] as standard ontology language for the semantic web. As a consequence of this standardization, several ontology editors support OWL [16, 19, 15], and ontologies written in OWL are employed in more and more applications. As the size of such ontologies grows, tools that support improving the quality of large DL-based ontologies become more important. Standard DL reasoners [13, 10, 25] employ tableau-based algorithms [7], which can be used to detect inconsistencies and to infer other implicit consequences, such as subsumption relationships between concepts or instance relationships between individuals and concepts.

For a developer or user of a DL-based ontology, it is often quite hard to understand why a certain consequence holds,[1] and even harder to decide how to change the ontology in case the consequence is unwanted. For example, in the current version of the medical ontology SNOMED [26], the concept *Amputation-of-Finger* is classified as a subconcept of *Amputation-of-Arm*. Finding the six axioms that are responsible for this among the more than 350,000 terminological axioms of SNOMED without support by an automated reasoning tool is not easy.

As a first step towards providing such support, Schlobach and Cornet [23] describe an algorithm for computing all the *minimal subsets* of a given knowledge base that *have* a given *consequence*. To be more precise, the knowledge bases considered in [23] are so-called unfoldable $\mathcal{ALC}$-terminologies, and the unwanted consequences are the unsatisfiability of concepts. The algorithm is an extension of the known tableau-based satisfiability algorithm for $\mathcal{ALC}$ [24], where labels keep track of which axioms are responsible for an assertion to be generated during the run of the algorithm. The authors also coin the name "axiom pinpointing" for the task of computing these minimal subsets. Following Reiter's approach for model-based diagnosis [21], Schlobach [22] uses the minimal subsets that have a given consequence together with the computation of Hitting Sets to compute *maximal subsets* of a given knowledge base that *do not have* a given (unwanted) *consequence*.[2] Whereas the minimal subsets that have the consequence help the user to comprehend why a certain consequence holds, the maximal subsets that do not have the consequence suggest how to change the knowledge base in a minimal way to get rid of a certain unwanted consequence.

The problem of computing minimal (maximal) subsets of a DL knowledge

[1] Note that this consequence may also be the inconsistency of the knowledge base or the unsatisfiability of a concept w.r.t. the knowledge base.

[2] Actually, he considers the complements of these sets, which he calls minimal diagnoses.

base that have (do not have) a given consequence was actually considered earlier in the context of extending DLs by default rules. In [4], Baader and Hollunder solve this problem by introducing a labeled extension of the tableau-based consistency algorithm for $\mathcal{ALC}$-ABoxes [11], which is very similar to the one described later in [23]. The main difference is that the algorithm described in [4] does not directly compute minimal subsets that have a consequence, but rather a monotone Boolean formula, called *clash formula* in [4], whose variables correspond to the axioms of the knowledge bases and whose minimal satisfying (maximal unsatisfying) valuations correspond to the minimal (maximal) subsets that have (do not have) a given consequence.[3]

The approach of Schlobach and Cornet [23] was extended by Parsia et al. [20] to more expressive DLs, and the one of Baader and Hollunder [4] was extended by Meyer et al. [18] to the case of $\mathcal{ALC}$-terminologies with general concept inclusions (GCIs), which are no longer unfoldable. The choice of the DL $\mathcal{ALC}$ in [4] and [23] was meant to be prototypical, i.e., in both cases the authors assumed that their approach could be easily extended to other DLs and tableau-based algorithms for them. However, the algorithms and proofs are given for $\mathcal{ALC}$ only, and it is not clear to which of the known tableau-based algorithms the approaches really generalize. For example, the pinpointing extension described in [18] follows the approach introduced in [4], but since GCIs require the introduction of so-called blocking conditions into the tableau-based algorithm to ensure termination, there are some new problems to be solved.

Thus, one can ask to which DLs and tableau-based algorithms the approaches described in [4, 23] apply basically without significant changes, and with no need for a new proof of correctness. This paper is a first step towards answering this question. We develop a general approach for extending a tableau-based algorithm to a pinpointing algorithm, which is based on the ideas underlying the pinpointing algorithm described in [4]. To this purpose, we define a general notion of "tableau algorithm," which captures many of the known tableau-based algorithms for DLs and Modal Logics,[4] but also other kinds of decision procedures, like the polynomial-time subsumption algorithm for the DL $\mathcal{EL}$ [1]. This notion is simpler than the tableau systems introduced in [3] in the context of translating tableaux into tree automata, and it is not restricted to tableau-based algorithms that generate tree-like structures.

[3]In this paper, we call a formula with these properties a *pinpointing formula*; the term clash formula is used for the formula computed by our pinpointing algorithm.

[4]Note that these algorithms are decision procedures, i.e., always terminate. Currently, our approach does not cover semi-decision procedures like tableau procedures for first-order logic.

Axiom pinpointing has also been considered in other research areas, though usually not under this name. For example, in the SAT community, people have considered the problem of computing maximally satisfiable and minimally unsatisfiable subsets of a set of propositional formulae. The approaches for computing these sets developed there include special purpose algorithms that call a SAT solver as a black box [17, 8], but also algorithms that extend a resolution-based SAT solver directly [9, 27]. To the best of our knowledge, extensions of tableau-based algorithms have not been considered in this context, and there are no general schemes for extending resolution-based solvers.

In the next section, we define the notions of minimal (maximal) sets having (not having) a given consequence in a general setting, and show some interesting connections between these two notions. In Section 3 we introduce our general notion of a tableau, and in Section 4 we show how to obtain a pinpointing extension of such a tableau. The main result shown in Section 4 is that this pinpointing extension is correct in the sense that the clash formula computed by a terminating run of it is indeed a pinpointing formula. Unfortunately, however, termination need not transfer from a given tableau to its pinpointing extension. To overcome this problem, Section 5 introduces so-called *ordered forest tableaux*, which are guaranteed to terminate. The main result of Section 5 is that the pinpointing extension of an ordered forest tableau also terminates on all inputs, and thus always computes a pinpointing formula.

2 Basic Definitions

We will begin by defining a general notion of inputs in which our decision algorithms are applied and the decision problems that they are supposed to solve.

Definition 2.1 (Axiomatized input, c-property) *Let $\mathfrak{I}$ and $\mathfrak{T}$ be the sets of* inputs *and* axioms*, respectively. An* axiomatized input *over these sets is of the form $(\mathcal{I}, \mathcal{T})$ where $\mathcal{I} \in \mathfrak{I}$ and $\mathcal{T} \in \mathscr{P}_{fin}(\mathfrak{T})$ is a finite subset of $\mathfrak{T}$. A* consequence property *(or* c-property *for short) is a set $\mathcal{P} \subseteq \mathfrak{I} \times \mathscr{P}_{fin}(\mathfrak{T})$ such that $(\mathcal{I}, \mathcal{T}) \in \mathcal{P}$ implies $(\mathcal{I}, \mathcal{T}') \in \mathcal{P}$ for every $\mathcal{T}' \supseteq \mathcal{T}$.*

Intuitively, a c-property $\mathcal{P}$ holds if an input $\mathcal{I}$ "follows" from the axioms in $\mathcal{T}$. Due to the monotonicity requirement, there can be some superfluous axioms; that is, axioms that are not necessary for the property to hold. We are interested in distinguishing these from the axioms that are responsible for the property.

Definition 2.2 *Given an axiomatized input* $\Gamma = (\mathcal{I}, \mathcal{T})$ *and a c-property* $\mathcal{P}$*, a set of axioms* $\mathcal{S} \subseteq \mathcal{T}$ *is called a* minimal axiom set (MinA) *for* Γ *w.r.t.* $\mathcal{P}$ *if* $(\mathcal{I}, \mathcal{S}) \in \mathcal{P}$ *and* $(\mathcal{I}, \mathcal{S}') \notin \mathcal{P}$ *for every* $\mathcal{S}' \subset \mathcal{S}$*. The set of all MinA for* Γ *w.r.t.* $\mathcal{P}$ *will be denoted as* $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$*.*

Note that the notions of MinA and MaNA are only interesting if $\Gamma \in \mathcal{P}$. Otherwise, the monotonicity requirement in $\mathcal{P}$ would entail that $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \emptyset$.

Instead of directly trying to compute $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$, one can also try to compute a pinpointing formula. In order to define this formula, we assume that every axiom $t \in \mathcal{T}$ is labeled with a unique propositional variable $\mathsf{lab}(t)$. Let $\mathsf{lab}(\mathcal{T})$ be the set of all propositional variables labeling axioms in $\mathcal{T}$. A *monotone Boolean formula* over $\mathsf{lab}(\mathcal{T})$ is a Boolean formula using variables in $\mathsf{lab}(\mathcal{T})$ and only the connectives conjunction and disjunction. As usual, we identify a propositional *valuation* with the set of propositional variables it makes true. For a valuation $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{T})$, let $\mathcal{T}_\mathcal{V} = \{t \in \mathcal{T} \mid \mathsf{lab}(t) \in \mathcal{V}\}$.

Definition 2.3 (pinpointing formula) *Given a c-property* $\mathcal{P}$ *and an axiomatized input* $\Gamma = (\mathcal{I}, \mathcal{T})$*, a monotone Boolean formula* ϕ *over* $\mathsf{lab}(\mathcal{T})$ *is called a* pinpointing formula *for* $\mathcal{P}$ *and* Γ *if for every valuation* $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{T})$ *it holds that* $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ *iff* $\mathcal{V}$ *satisfies* ϕ*.*

The next lemma follows directly from the definition of a pinpointing formula.

Lemma 2.4 *Let* $\mathcal{P}$ *be a c-property,* $\Gamma = (\mathcal{I}, \mathcal{T})$ *an axiomatized input, and* ϕ *a pinpointing formula for* $\mathcal{P}$ *and* Γ*. Then*

$$\mathsf{MIN}_{\mathcal{P}(\Gamma)} \quad = \quad \{\mathcal{T}_\mathcal{V} \mid \mathcal{V} \textit{ is a minimal valuation satisfying } \phi\}$$

This lemma shows that if we want to obtain all MinA, it is enough to design an algorithm that computes a pinpointing formula. Conversely, the set $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ can be translated into the pinpointing formula

$$\bigvee_{\mathcal{S} \in \mathsf{MIN}_{\mathcal{P}(\Gamma)}} \bigwedge_{s \in \mathcal{S}} \mathsf{lab}(s)$$

3 General Tableaux

General tableaux and their pinpointing extension were first introduced in [5]. For the rest of this paper we use $\mathcal{V}$ and $\mathcal{D}$ to denote countably infinite sets of *variables* and *constants*, respectively. A *signature* Σ is a set of predicate

symbols, where each predicate $P \in \Sigma$ is equipped with an arity. A Σ-*assertion* is of the form $P(a_1, \ldots, a_n)$ where $P \in \Sigma$ is an n-ary predicate and $a_1, \ldots, a_n \in \mathcal{D}$. Analogously, a Σ-*pattern* is of the form $P(x_1, \ldots, x_n)$ where $P \in \Sigma$ is an n-ary predicate and $x_1, \ldots, x_n \in \mathcal{V}$. If the signature is clear from the context, we will often just say pattern (assertion). For a set of assertions A (patterns B), $\mathsf{cons}(A)$ ($\mathsf{var}(B)$) denotes the set of constants (variables) occurring in A (B).

A *substitution* is a mapping $\sigma : V \to \mathcal{D}$, where V is a finite set of variables. In this case, we say that σ is a *substitution on* V. If B is a set of patterns such that $\mathsf{var}(B) \subseteq V$, then $B\sigma$ denotes the set of assertions obtained from B by replacing each variable by its σ-image. The substitution θ on V' *extends* σ on V if $V \subseteq V'$ and $\theta(x) = \sigma(x)$ for all $x \in V$.

Definition 3.1 (Tableau) *Let $\mathfrak{I}$ and $\mathfrak{T}$ be sets of inputs and axioms, respectively. A* tableau *for $\mathfrak{I}$ and $\mathfrak{T}$ is a tuple $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ where:*

- *Σ is a signature;*
- *$\cdot^S$ is function that maps every $\mathcal{I} \in \mathfrak{I}$ to a finite set of finite sets of Σ-assertions and every $t \in \mathfrak{T}$ to a finite set of Σ-assertions;*
- *$\mathcal{R}$ is a set of* rules *of the form $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$, where for every $i, 0 \leq i \leq m, B_i$ is a finite set of Σ-patterns and $\mathcal{S}$ is a finite set of axioms;*
- *$\mathcal{C}$ is a set of finite sets of Σ-patterns, called* clashes.

Given a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$, the variable y is a fresh variable *in R if it occurs in one of the sets $B_1, \ldots, B_m$, but not in B_0.*

A S-state is a pair $\mathfrak{S} = (A, \mathcal{T})$ where A is a finite set of assertions and $\mathcal{T}$ a finite set of axioms. We extend the function $\cdot^S$ to axiomatized inputs by setting

$$(\mathcal{I}, \mathcal{T})^S = \{(A \cup \bigcup_{t \in \mathcal{T}} t^S, \mathcal{T}) \mid A \in \mathcal{I}^S\}.$$

A tableau works in the following way. Given an input $(\mathcal{I}, \mathcal{T})$, we begin with the initial set of states $\mathcal{M} = (\mathcal{I}, \mathcal{T})^S$, and then use the rules in $\mathcal{R}$ to modify this set. Each rule application picks a S-state $\mathfrak{S}$ from $\mathcal{M}$ and replaces it by finitely many new S-states $\mathfrak{S}_1, \ldots, \mathfrak{S}_m$, each of them extending the first component of $\mathfrak{S}$. When no more rules are applicable to $\mathcal{M}$ we check whether all the elements of $\mathcal{M}$ contain a clash. If it is the case, then the input is accepted; otherwise, it is rejected.

Definition 3.2 (rule application, saturated, clash) *Given a S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$, and a substitution ρ on $\mathsf{var}(B_0)$,* R *is* applicable to $\mathfrak{S}$ with ρ *if (i) $\mathcal{S} \subseteq \mathcal{T}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $i, 1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ we have that $B_i\rho' \not\subseteq A$.*

Given a set of S-states $\mathcal{M}$ and a S-state $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$ to which the rule R is applicable with substitution ρ, the application of R to $\mathfrak{S}$ with ρ *yields the new set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{T}) \mid 1 \leq i \leq m\}$, where σ is a substitution on the variables occurring in R that extends ρ and maps the fresh variables of R to distinct new constants; i.e., constants not occurring in A.*

If $\mathcal{M}'$ is obtained from $\mathcal{M}$ by the application of R, then we write $\mathcal{M} \to_{\mathsf{R}} \mathcal{M}'$, or simply $\mathcal{M} \to_S \mathcal{M}'$ if it is not relevant which of the rules of the tableau S was applied. The reflexive-transitive closure *of $\to_S$ is denoted by $\xrightarrow{*}_S$. A set of S-states $\mathcal{M}$ is called* saturated *if there is no $\mathcal{M}'$ such that $\mathcal{M} \to_S \mathcal{M}'$.*

The S-state $\mathfrak{S} = (A, \mathcal{T})$ contains a clash *if there is a $C \in \mathcal{C}$ and a substitution ρ on $\mathsf{var}(C)$ such that $C\rho \subseteq A$, and the set of S-states $\mathcal{M}$ is* full of clashes *if all its elements contain a clash.*

We will proceed now to relate tableaux and c-properties, by describing the conditions under which a tableau is considered correct for a c-property.

Definition 3.3 (correctness) *Let $\mathcal{P}$ be a c-property on axiomatized inputs over $\mathfrak{I}$ and $\mathfrak{T}$, and S a tableau for $\mathfrak{I}$ and $\mathfrak{T}$. We say that S is* correct for $\mathcal{P}$ *if the following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathfrak{T}$:*

1. *S terminates on Γ; i.e., there is no infinite chain of rule applications $\mathcal{M}_0 \to_S \mathcal{M}_1 \to_S \mathcal{M}_2 \to_S \ldots$ starting with $\mathcal{M}_0 = \Gamma^S$.*

2. *For every chain of rule applications $\mathcal{M}_0 \to_S \ldots \to_S \mathcal{M}_n$ such that $\mathcal{M}_0 = \Gamma^S$ and $\mathcal{M}_n$ is saturated, we have $\Gamma \in \mathcal{P}$ iff $\mathcal{M}_n$ is full of clashes.*

The second condition in this definition requires that the algorithm gives the same answer independent of the chain of rule applications considered. Thus, if several rules are applicable simultaneously, the choice of which of them to apply next has no influence on the final result. However, this requirement is in fact built into our definition of rules and clashes. For every tableau S, if there are two terminating chains of rule applications $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}$ and $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}'$, then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is also full of clashes.

As shown in [6], the first condition need not hold for every tableau, and it is even undecidable in general whether a given tableau satisfies it or not. In Section 5, we show a class of tableau for which termination is guaranteed.

4 Pinpointing extensions of general tableaux

Given a correct tableau, we show how it can be extended to an algorithm that computes a pinpointing formula. As shown in Section 2, all minimal axiom sets (maximal non-axiom sets) can be derived from the pinpointing formula ϕ by computing all minimal (maximal) valuations satisfying (falsifying) ϕ. Recall that, in the definition of the pinpointing formula, we assume that every axiom $t \in \mathcal{T}$ is labeled with a unique propositional variable, $\mathsf{lab}(t)$. The set of all propositional variables labeling an axiom in $\mathcal{T}$ is denoted by $\mathsf{lab}(\mathcal{T})$. In the following, we assume that the symbol $\top$, which always evaluates to true, also belongs to $\mathsf{lab}(\mathcal{T})$. The pinpointing formula is a monotone Boolean formula over $\mathsf{lab}(\mathcal{T})$, i.e., a Boolean formula built from $\mathsf{lab}(\mathcal{T})$ using conjunction and disjunction only.

Given a tableau $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ that is correct for the c-property $\mathcal{P}$, we show how the algorithm for deciding $\mathcal{P}$ induced by S can be modified to an algorithm that computes a pinpointing formula for $\mathcal{P}$. Given an axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, the modified algorithm also works on sets of S-states, but now every assertion a occurring in the assertion component of an S-state is equipped with a label $\mathsf{lab}(a)$, which is a monotone Boolean formula over $\mathsf{lab}(\mathcal{T})$. We call such S-states *labeled S-states*. In an initial S-state $(A, \mathcal{T}) \in (\mathcal{I}, \mathcal{T})^S$, an assertion $a \in A$ is labeled with $\top$ if $a \in \mathcal{I}^S$ and with $\bigvee_{\{t \in \mathcal{T} \mid a \in t^S\}} \mathsf{lab}(t)$ otherwise.

The definition of rule application must also take the labels of assertions and axioms into account. Let A be a set of labeled assertions and ψ a monotone Boolean formula. We say that the assertion a is *ψ-insertable into* A if (i) either $a \notin A$, or (ii) $a \in A$, but $\psi \not\models \mathsf{lab}(a)$. Given a set B of assertions and a set A of labeled assertions, the set of *ψ-insertable elements of B into A* is defined as $\mathsf{ins}_\psi(B, A) := \{b \in B \mid b \text{ is } \psi\text{-insertable into } A\}$. By ψ-inserting these insertable elements into A, we obtain the following new set of labeled assertions: $A \uplus_\psi B := A \cup \mathsf{ins}_\psi(B, A)$, where each assertion $a \in A \setminus \mathsf{ins}_\psi(B, A)$ keeps its old label $\mathsf{lab}(a)$, each assertion in $\mathsf{ins}_\psi(B, A) \setminus A$ gets label ψ, and each assertion $b \in A \cap \mathsf{ins}_\psi(B, A)$ gets the new label $\psi \vee \mathsf{lab}(b)$.

Definition 4.1 (pinpointing rule application) *Given a labeled S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$, and a substitution ρ on $\mathsf{var}(B_0)$, this rule is* pinpointing applicable to $\mathfrak{S}$ with ρ *if (i) $\mathcal{S} \subseteq \mathcal{T}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $i, 1 \leq i \leq m$, and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ we have $\mathsf{ins}_\psi(B_i\rho', A) \neq \emptyset$, where $\psi := \bigwedge_{b \in B_0} \mathsf{lab}(b\rho) \wedge \bigwedge_{s \in \mathcal{S}} \mathsf{lab}(s)$.*

Given a set of labeled S-states $\mathcal{M}$ and a labeled S-state $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$ to which the rule R is pinpointing applicable with substitution ρ, the pin-

pointing application of R to $\mathfrak{S}$ with ρ in $\mathcal{M}$ *yields the new set* $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \uplus_\psi B_i\sigma, \mathcal{T}) \mid i = 1, \ldots, m\}$, *where the formula* ψ *is defined as above and* σ *is a substitution on the variables occurring in* R *that extends* ρ *and maps the fresh variables of* R *to distinct new constants.*

If $\mathcal{M}'$ *is obtained from* $\mathcal{M}$ *by the pinpointing application of* R, *then we write* $\mathcal{M} \rightarrow_{\mathsf{R}^{pin}} \mathcal{M}'$, *or simply* $\mathcal{M} \rightarrow_{S^{pin}} \mathcal{M}'$ *if it is not relevant which of the rules of the tableau* S *was applied. As before, the* reflexive-transitive closure *of* $\rightarrow_{S^{pin}}$ *is denoted by* $\xrightarrow{*}_{S^{pin}}$. *A set of labeled* S*-states* $\mathcal{M}$ *is called* pinpointing saturated *if there is no* $\mathcal{M}'$ *such that* $\mathcal{M} \rightarrow_{S^{pin}} \mathcal{M}'$.

Consider a chain of pinpointing rule applications $\mathcal{M}_0 \rightarrow_{S^{pin}} \ldots \rightarrow_{S^{pin}} \mathcal{M}_n$ such that $\mathcal{M}_0 = \Gamma^S$ for an axiomatized input Γ and $\mathcal{M}_n$ is pinpointing saturated. The label of an assertion in $\mathcal{M}_n$ expresses which axioms are needed to obtain this assertion. A clash in an S-state of $\mathcal{M}_n$ depends on the joint presence of certain assertions. Thus, we define the label of the clash as the conjunction of the labels of these assertions. Since it is enough to have just one clash per S-state $\mathfrak{S}$, the labels of different clashes in $\mathfrak{S}$ are combined disjunctively. Finally, since we need a clash in every S-state of $\mathcal{M}_n$, the formulae obtained from the single S-states are again conjoined.

Definition 4.2 (clash set, clash formula) *Let* $\mathfrak{S} = (A, \mathcal{T})$ *be a labeled* S*-state and* $A' \subseteq A$. *Then* A' *is a* clash set in $\mathfrak{S}$ *if there is a clash* $C \in \mathcal{C}$ *and a substitution* ρ *on* $\mathsf{var}(C)$ *such that* $A' = C\rho$. *The* label *of this clash set is* $\psi_{A'} := \bigwedge_{a \in A'} \mathsf{lab}(a)$.

Let $\mathcal{M} = \{\mathfrak{S}_1, \ldots, \mathfrak{S}_n\}$ *be a set of labeled* S*-states. The* clash formula induced by $\mathcal{M}$ *is defined as*

$$\psi_{\mathcal{M}} := \bigwedge_{i=1}^{n} \bigvee_{A' \text{ clash set in } \mathfrak{S}_i} \psi_{A'}.$$

Recall that, given a set $\mathcal{T}$ of labeled axioms, a propositional valuation $\mathcal{V}$ induces the subset $\mathcal{T}_\mathcal{V} := \{t \in \mathcal{T} \mid \mathsf{lab}(t) \in \mathcal{V}\}$ of $\mathcal{T}$. Similarly, for a set A of labeled assertions, the valuation $\mathcal{V}$ induces the subset $A_\mathcal{V} := \{a \in A \mid \mathcal{V} \text{ satisfies } \mathsf{lab}(a)\}$. Given a labeled S-state $\mathfrak{S} = (A, \mathcal{T})$ we define its $\mathcal{V}$*-projection* as $\mathcal{V}(\mathfrak{S}) := (A_\mathcal{V}, \mathcal{T}_\mathcal{V})$. The notion of a projection is extended to sets of S-states $\mathcal{M}$ in the obvious way: $\mathcal{V}(\mathcal{M}) := \{\mathcal{V}(\mathfrak{S}) \mid \mathfrak{S} \in \mathcal{M}\}$.

Given a tableau that is correct for a property $\mathcal{P}$, its pinpointing extension is correct in the sense that the clash formula induced by the pinpointing saturated set computed by a terminating chain of pinpointing rule applications is indeed a pinpointing formula for $\mathcal{P}$ and the input.

Theorem 4.3 (correctness of pinpointing) *Let $\mathcal{P}$ be a c-property on axiomatized inputs over $\mathfrak{I}$ and $\mathfrak{T}$, and S a correct tableau for $\mathcal{P}$. Then the following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathfrak{T}$:*

> *For every chain of rule applications $\mathcal{M}_0 \rightarrow_{S^{pin}} \ldots \rightarrow_{S^{pin}} \mathcal{M}_n$ such that $\mathcal{M}_0 = \Gamma^S$ and $\mathcal{M}_n$ is pinpointing saturated, the clash formula $\psi_{\mathcal{M}_n}$ induced by $\mathcal{M}_n$ is a pinpointing formula for $\mathcal{P}$ and Γ.*

To prove this theorem, we consider projections of chains of pinpointing rule applications to chains of "normal" rule applications. Unfortunately, things are not as simple as one might hope for since in general $\mathcal{M} \rightarrow_{S^{pin}} \mathcal{M}'$ does not imply $\mathcal{V}(\mathcal{M}) \rightarrow_S \mathcal{V}(\mathcal{M}')$. To overcome this problem, we define a modified version of rule application, where the applicability condition (iii) from Definition 3.2 is removed. This concept will also be useful in the following section for showing termination of forest tableaux.

Definition 4.4 (modified rule application) *Given an S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$, and a substitution ρ on $\mathsf{var}(B_0)$, this rule is m-*applicable to $\mathfrak{S}$ with ρ *if (i) $\mathcal{S} \subseteq \mathcal{T}$ and (ii) $B_0\rho \subseteq A$. In this case, we write $\mathcal{M} \rightarrow_{S^m} \mathcal{M}'$ if $\mathfrak{S} \in \mathcal{M}$ and $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{T}) \mid i = 1, \ldots, m\}$, where σ is a substitution on the variables occurring in R that extends ρ and maps the fresh variables of R to distinct new constants.*

Theorem 4.3 considers a terminating chain of pinpointing rule applications. Unfortunately, termination of a tableau S in general does not imply termination of its pinpointing extension. See [5] for an example. One way to overcome this problem could be to find sufficient conditions under which termination transfers from a tableau to its pinpointing extension. However, it is undecidable even for terminating tableaux whether their pinpointing extensions also terminate (see [6]). For this reason, the next section follows a different approach: we introduce a class of terminating tableaux whose pinpointing extensions terminate as well.

5 A class of terminating tableaux

One of the reasons why tableau algorithms for certain DLs terminate is that they create a tree structure for which the out-degree and the depth of the tree are bounded by a function of the size of the input formula. The nodes of these trees are labeled, but the input determines a finite number of possible labels. A typical example is the tableau-based decision procedure for satisfiability of

$\mathcal{ALC}$-concepts [24, 7]. This algorithm generates sets of assertions of the form $r(a, b)$ where r is a so-called role and $C(a)$ where C is an $\mathcal{ALC}$-concept. The tree structure is induced by role assertions, and the nodes are labeled by sets of concepts, i.e., node a is labeled with $\{C_1, \ldots, C_n\}$ if $C_1(a), \ldots, C_n(a)$ are all the concept assertions involving a. The main reasons why the algorithm terminates are:

- the depth of the tree structure is bounded by the size n of the input, i.e., the maximal length m of chains $r_1(a_0, a_1), r_2(a_1, a_2), \ldots, r_m(a_{m-1}, a_m)$ in a set of assertions generated by the algorithm is bounded by n;
- the out-degree of the tree structure is bounded by n , i.e., the maximal number m of assertions $r_1(a_0, a_1), r_2(a_0, a_2), \ldots, r_m(a_0, a_m)$ in a set of assertions generated by the algorithm is bounded by n;
- for every assertion $C(a)$ occurring in a set of assertions generated by the algorithm, C is a subconcept of the input concept.

If we look at the extension of this algorithm to one that decides consistency of $\mathcal{ALC}$-ABoxes (see, e.g., [12, 7]), then things are a bit more complicated: rather than a single tree one obtains a forest, more precisely, several trees growing out of the input ABox. But these trees satisfy the restrictions mentioned above, which is enough to show termination.

Basically, we want to formalize this reason for termination within the general framework of tableaux introduced in this paper. However, to be as general as possible, we do not want to restrict assertions to be built from unary predicates (concepts) and binary predicates (roles) only. For this reason, we allow for predicates of arbitrary arity, but restrict our assertions such that states (i.e., sets of assertions) induce graph-like structures.

In order to have a graph-like structure, we must be able to distinguish between nodes and edges. For this reason, we now assume that the signature Σ is partitioned into the sets Λ and Δ, where each predicate name $P \in \Lambda$ is equipped with an arity n, while every predicate name $r \in \Delta$ is equipped with a double arity $0 < m < n$. Strictly speaking, the arity of $r \in \Delta$ is n; however, the first m argument positions are grouped together, as are the last $n - m$. Intuitively, the elements of Λ correspond to DL concepts and form the nodes of the graph-like structure, whereas the elements of Δ correspond to DL roles and induce the edges.

If a pattern/assertion p starts with a predicate from Δ (Λ), we say that p is a *Δ-pattern/assertion (Λ-pattern/assertion)*, and write $p \in \widehat{\Delta}$ ($p \in \widehat{\Lambda}$). In our $\mathcal{ALC}$ example, the set Λ consists of all $\mathcal{ALC}$-concepts, which have arity 1, and Δ consists of all role names, which have double arity $1, 2$. For the rest of

this paper, assertions and patterns in $\widehat{\Lambda}$ will be denoted using capital letters $(P, Q, R, \ldots)$, and those in $\widehat{\Delta}$ using lower-case letters $(r, s, t, \ldots)$. Given a predicate $p \in \Delta$ with double arity m, n, the sets of *parents* and *descendants* of the pattern $r = p(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$ are given by $\overleftarrow{r} = \{x_1, \ldots, x_m\}$ and $\overrightarrow{r} = \{x_{m+1}, \ldots, x_n\}$, respectively.

In our $\mathcal{ALC}$ example, the nodes of the tree are the constants occurring in the set of assertions, and the concept assertions give rise to the labels of these nodes. In the general case, nodes are not single constants, but rather sets of assertions built over a connected set of constants.

Definition 5.1 (connected) *Let B be a set of Σ-patterns (Σ-assertions), and $x, y \in \mathsf{var}(B)$ ($a, b \in \mathsf{cons}(B)$). We say that x and y (a and b) are B-*connected*, denoted as $x \sim_B y$ ($a \sim_B b$), if there are variables $x_0, x_1, \ldots, x_n \in \mathsf{var}(B)$ (constants $a_0, a_1, \ldots, a_n \in \mathsf{cons}(B)$) and patterns $P_1, \ldots, P_n \in B \cap \widehat{\Lambda}$ (assertions $P_1, \ldots, P_n \in B \cap \widehat{\Lambda}$) such that $x = x_0, y = x_n$ ($a = a_0, b = a_n$) and for every $1 \leq i \leq n$ it holds that $\{x_{i-1}, x_i\} \subseteq \mathsf{var}(P_i)$ ($\{a_{i-1}, a_i\} \subseteq \mathsf{cons}(P_i)$).*

We say that B is connected *if, for every $x, y \in \mathsf{var}(B)$ ($a, b \in \mathsf{cons}(B)$), we have $x \sim_B y$ ($a \sim_B b$). If B is clear from the context, we will simply write $x \sim y$ to represent $x \sim_B y$.*

Connected sets of assertions can be viewed as *bundles* that join the constants contained in them. Nodes will be formed by maximal sets of assertions from $\widehat{\Delta}$. An assertion from $\widehat{\Lambda}$ will be treated as a (directed) edge that connects a node containing its parent constants with a node containing its descendant constants.

Definition 5.2 (B-graph) *Let B be a set of assertions. A maximal connected subset $N \subseteq B \cap \widehat{\Lambda}$ is called a* node *in B. An assertion $r \in B \cap \widehat{\Delta}$ is called an* edge *in B if there are two nodes N_1 and N_2 in B such that $\overleftarrow{r} \subseteq \mathsf{cons}(N_1)$ and $\mathsf{cons}(N_2) \subseteq \overrightarrow{r}$. In this case, we say that r* connects *N_1 to N_2. The set B is a* graph structure *if every $r \in B \cap \widehat{\Delta}$ is an edge. If B is a graph structure, the corresponding B-*graph *$\mathcal{G}_B$ contains one vertex v_N for every node N, and an edge (v_N, v_M) if there is an edge connecting N to M. The notion of a graph structure and of the corresponding graph can be extended to states $\mathfrak{S} = (B, \mathcal{T})$ in the obvious way: $\mathfrak{S}$ is a* graph structure *if B is one, and in this case $\mathcal{G}_{\mathfrak{S}} := \mathcal{G}_B$.*

If a set of assertions B is a graph structure, then the set of nodes forms a partition of $B \cap \widehat{\Lambda}$, and each of its elements either belongs to a node or is a (directed) edge. Observe, however, that an edge $r \in \widehat{\Delta}$ may connect a node with more than one successor node. For example, consider the set

of assertions $B = \{P(a), Q(b), R(c), r(a,b,c)\}$ where $P, Q, R \in \Lambda$ are unary, and $r \in \Delta$ has double arity arity $1, 2$. This set forms a graph structure consisting of the nodes $N_1 := P(a), N_2 := Q(b), N_3 := R(c)$ and the edge $r(a,b,c)$. This single edge connects N_1 to both N_2 and N_3. $\mathcal{G}_B$ is then the graph $(\{v_1, v_2, v_3\}, \{(v_1, v_2), (v_1, v_3)\})$. This will create no problem for termination, but must be kept in mind when dealing with graph-structures and their corresponding graphs.

Recall that the tableau-based decision procedures for consistency of $\mathcal{ALC}$-ABoxes [12, 7] starts with an ABox, which can be viewed as a graph, but then extends this ABox by trees that grow out of the nodes of this graph. The following definition introduces *forest tableaux*, which show a similar behaviour, but is based on the more general notion of a graph structure introduced above.

Definition 5.3 (forest tableau) *The tableau $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ is called a* forest tableau *if for every axiomatized input Γ and every $\mathfrak{S} \in \Gamma^S$, the state $\mathfrak{S}$ is a graph structure, every clash $C \in \mathcal{C}$ is connected, and the following conditions hold for every rule $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ and every $1 \leq i \leq m$:*

1. *for every Σ-pattern $r \in B_0 \cap \widehat{\Delta}$, there exists a Σ-pattern $P \in B_0 \cap \widehat{\Lambda}$ such that $\overleftarrow{r} \subseteq \mathsf{var}(P)$ or $\overrightarrow{r} \subseteq \mathsf{var}(P)$.*
2. *for every Σ-pattern $r \in B_i \cap \widehat{\Delta}$, there exists a Σ-pattern $P \in B_0 \cap \widehat{\Lambda}$ such that $\overleftarrow{r} \subseteq \mathsf{var}(P)$.*
3. *for every Σ-pattern $r \in B_i \cap \widehat{\Delta}$, we have $\overrightarrow{r} \cap \mathsf{var}(B_0) = \emptyset$.*
4. *if $r, s \in B_i \cap \widehat{\Delta}$ are distinct patterns, then $\overrightarrow{r} \cap \overrightarrow{s} = \emptyset$.*
5. *for every Σ-pattern $P \in B_i \cap \widehat{\Lambda}$, either*
   (i) there is a Σ-pattern $r \in (B_0 \cup B_i) \cap \widehat{\Delta}$ such that $\mathsf{var}(P) \subseteq \overrightarrow{r}$ or $\mathsf{var}(P) \subseteq \overleftarrow{r}$, or
   (ii) there is a $Q \in B_0 \cap \widehat{\Lambda}$ with $\mathsf{var}(P) \subseteq \mathsf{var}(Q)$.
6. *if $B_0 \cap \widehat{\Delta} \neq \emptyset$, then $B_i \cap \widehat{\Delta} = \emptyset$.*
7. *$B_0 \cap \widehat{\Lambda}$ is connected.*

A few intuitive explanations for these conditions are in order. *Condition 1* ensures that every edge triggering a rule application is connected to a node, which may be either a parent or a descendant node of this edge. *Condition 2* makes sure that for every newly introduced edge, a parent node was present before the rule is applied. This implies that rule application

cannot add a new parent for a node, and that newly introduced nodes are not disconnected from the rest of the graph structure. Both of these properties are vital for obtaining forest structures. *Condition 3* states that every newly generated edge has only new constants in its descendant set. In other words, new edges cannot connect old nodes, but only generate new nodes as descendant. *Condition 4* ensures that, even if several edges are added by a single rule application, these edges connect different nodes with the parent node, avoiding this way that a node is connected by multiple edges to a parent node. *Condition 5* makes sure that we alway have a connected graph. It states that, whenever a non-edge assertion is added, it must either belong to an old node, or belong to a descendant node added by a new edge within the same rule application. *Condition 6* states that the addition of new edges must only depend on the assertions belonging to the parent nodes, but never on the presence of other edges. In particular, this ensures that each descendant is created independently from its siblings, as long this is done in distinct rule applications. Finally, *Condition 7* ensures that the non-edge assertions triggering a rule application all belong to the same node.

The different (disjunctive) options stated in Conditions 1 and 3 require an additional explanation. They allow the tableau rules to propagate information not just to successor nodes, but also to predecessor nodes in the trees. The main reason for including this possibility in our framework is that it makes it general enough to deal with constructors such as *inverse roles* in DLs. The price to pay for this is that more cases must be analyzed in the proofs.

Clearly, just ensuring that all states generated by a tableau have forest structure is not sufficient to yield termination. We must also ensure that the trees in the forest cannot grow indefinitely (i.e., that the overall number of nodes that can be generated is bounded), and that the same is true for the nodes (i.e., that the number of assertions making up a single node is bounded). The next definition deals with the second condition.

Definition 5.4 (cover) *Let $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ be a tableau and $\mathcal{T}$ a set of axioms. A set $\Omega \subseteq \Sigma$ is called a $\mathcal{T}$-*cover *if, for every rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_n\}$ such that $\mathcal{S} \subseteq \mathcal{T}$ and B_0 contains only predicates from Ω, the sets B_i for $i = 1, ..., n$ also contain only predicates from Ω. The tableau S is* covered *if, for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, there is a finite $\mathcal{T}$-cover Ω_Γ such that every S-state in Γ^S contains only predicates from Ω_Γ.*

Given such a covered tableau, every state that can be reached from an initial state in Γ^S by applying rules from S contains only predicates from Ω_Γ. We will see that this ensures that nodes cannot grow indefinitely. To prevent

the trees from growing indefinitely (i.e., to bound the number of nodes), it is enough to enforce finite branching and finite paths in the trees. Finite branching actually already follows from the conditions we have stated so far. To bound the length of paths, we additionally require the predicates occurring in rules to be decreasing. Given a strict partial order $<$ on predicates, we extend it to patterns (assertions) by defining $P < Q$ if the predicate of the pattern (assertion) P is smaller than the predicate of the pattern (assertion) Q.

Definition 5.5 (ordered tableaux) *A covered tableau S is called an* ordered tableau *if, for every axiomatized input Γ, there is a strict partial ordering $<_\Gamma$ on the predicate names in $\Omega_\Gamma \cap \Lambda$ such that, for every rule $(B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_n\}$, every $1 \leq i \leq n$, and every $P \in B_0 \cap \widehat{\Lambda}$ and $Q \in B_i \cap \widehat{\Lambda}$, we have $Q <_\Gamma P$.*

For example, the tableau-based decision procedure for consistency of $\mathcal{ALC}$-ABoxes is an ordered tableau. It is covered since rule application only adds concept assertions $C(a)$ where C is a sub-description of a concept description occurring in the input ABox $\mathcal{A}_0$, and thus one can take the set of these sub-descriptions together with the roles occurring in $\mathcal{A}_0$ as a cover. In addition, rule application only adds concept assertions of a smaller role-depth (i.e., nesting of existential and value-restrictions) than the one that triggered it. Thus, ordering concept descriptions by their role-depth yields the desired partial ordering.

Ordered tableaux have the property that, if applied to an axiomatized input Γ, none of the trees in the generated forest can have a depth greater than the cardinality of the cover Ω_Γ. This easily follows from the next lemma. We will actually look at modified rule application rather than normal one since we want to show not only termination of the tableau itself, but also of its pinpointing extension.

Lemma 5.6 *Let S be an ordered forest tableau, Γ an axiomatized input, and $\mathfrak{S}_0 \rightarrow_{S^m} \mathfrak{S}_1 \rightarrow_{S^m} \cdots$ a sequence of modified rule applications. Then, for every $\mathfrak{S}_i = (A_i, \mathcal{T})$ and $P \in A_i \cap \widehat{\Lambda}$, either $\mathsf{cons}(P) \subseteq \mathsf{cons}(A_0)$ or there are $r \in A_i \cap \widehat{\Delta}$ and $Q \in A_i \cap \widehat{\Lambda}$ such that $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$, $\mathsf{cons}(P) \subseteq \overrightarrow{r}$, and $P <_\Gamma Q$.*

Proof. The proof is by induction on i. For $\mathfrak{S}_0$ the result is trivial. Suppose now that it holds for $\mathfrak{S}_i$, and that the rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_n\}$ is applied to $\mathfrak{S}_i$ to obtain $\mathfrak{S}_{i+1} = (A_{i+1}, \mathcal{T})$, where $A_{i+1} = A_i \cup B_j\sigma$ for some substitution σ and some $j, 1 \leq j \leq n$. Let $P \in A_{i+1} \cap \widehat{\Lambda}$. If $P \in A_i$,

then by the induction hypothesis and the fact that $A_i \subseteq A_{i+1}$, the result holds. Otherwise, P was added by the application of R. By Condition 5 of Definition 5.3, we have either (i) an $r \in (B_0 \cup B_j)\sigma \cap \widehat{\Delta}$ with $\mathsf{cons}(P) \subseteq \overrightarrow{r}$ or $\mathsf{cons}(P) \subseteq \overleftarrow{r}$, or (ii) there is a $Q \in B_0\sigma \cap \widehat{\Lambda}$ with $\mathsf{cons}(P) \subseteq \mathsf{cons}(Q)$.

We will analyze Case (ii) first. Since the rule was applied with substitution σ, we have $B_0\sigma \subseteq A_i$, and thus $Q \in A_i \cap \widehat{\Lambda}$. Since S is ordered, we also know that $P <_\Gamma Q$. By the induction hypothesis, either $\mathsf{cons}(Q) \subseteq \mathsf{cons}(A_0)$, or $\overleftarrow{r} \subseteq \mathsf{cons}(Q')$, $\mathsf{cons}(Q) \subseteq \overrightarrow{r}$, and $Q <_\Gamma Q'$ for assertions $r, Q' \in A_i$. In both case, transitivity of $<_\Gamma$ and of $\subseteq$ yield the desired result.

We focus now on Case (i). Suppose first that $\mathsf{cons}(P) \subseteq \overrightarrow{r}$. If $r \in B_j\sigma$, then Condition 2 of Definition 5.3 ensures that there is a $Q \in B_0\sigma \subseteq A_i$ such that $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$. Since S is ordered, we also have $P <_\Gamma Q$, which completes the proof for the case where $\mathsf{cons}(P) \subseteq \overrightarrow{r}$ and $r \in B_j\sigma$.

Next, we consider the case where $\mathsf{cons}(P) \subseteq \overrightarrow{r}$ and $r \in B_0\sigma$. Then, by Condition 1 of Definition 5.3, there must exist a $Q \in B_0\sigma$ such that $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$ or $\overrightarrow{r} \subseteq \mathsf{cons}(Q)$. In the former case, the proof is analogous to the one for the first part of this case. In the latter case, we have $\mathsf{cons}(P) \subseteq \overrightarrow{r} \subseteq \mathsf{cons}(Q)$, which is an instance of Case (ii).

Finally, suppose that $\mathsf{cons}(P) \subseteq \overleftarrow{r}$. We can assume without loss of generality that there is no $Q \in B_0 \cap \widehat{\Lambda}$ such that $\mathsf{cons}(P) \subseteq \mathsf{cons}(Q)$. In fact, if it existed, we would be in Case (ii) analyzed above. Consequently, r cannot belong to $B_i\sigma$ since this would violate Condition 2 of Definition 5.3. Hence, $r \in B_0\sigma$ and there must exist a $Q \in B_0\sigma \cap \widehat{\Lambda}$ such that $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$ or $\overrightarrow{r} \subseteq \mathsf{cons}(Q)$.

In the first case, we have $\mathsf{cons}(P) \subseteq \overleftarrow{r} \subseteq \mathsf{cons}(Q)$, which brings us back to Case (ii) analyzed above. In the other case, we know that $P <_\Gamma Q$ and $Q \in A_i$. Thus, by the induction hypothesis, the statement of the lemma holds for Q.

If $\mathsf{cons}(Q) \subseteq \mathsf{cons}(A_0)$, then—since we have assumed for this case that $\overrightarrow{r} \subseteq \mathsf{cons}(Q)$—we also have $\overrightarrow{r} \subseteq \mathsf{cons}(A_0)$. This means that r was not added by any previous rule application as otherwise this would violate Condition 3 of Definition 5.3. Thus, r must have been already present in A_0, which implies $\overleftarrow{r} \subseteq \mathsf{cons}(A_0)$. Since $\mathsf{cons}(P) \subseteq \overleftarrow{r}$, it also holds that $\mathsf{cons}(P) \subseteq \mathsf{cons}(A_0)$.

Now, assume that $\mathsf{cons}(Q) \not\subseteq \mathsf{cons}(A_0)$. By the induction hypothesis, there exist $s \in A_i \cap \widehat{\Delta}$ and $R \in A_i \cap \widehat{\Lambda}$ such that $\overleftarrow{s} \subseteq \mathsf{cons}(R), \mathsf{cons}(Q) \subseteq \overrightarrow{s}$, and $Q <_\Gamma R$. Since $\mathsf{cons}(Q) \not\subseteq \mathsf{cons}(A_0)$, we know that Q and s were added by a (previous) rule application. We claim that $r = s$. In fact, we have $\emptyset \neq \overrightarrow{r} \subseteq \mathsf{cons}(Q) \subseteq \overrightarrow{s}$. If we had $r \neq s$, then this would violate Condition 3 or 4 of Definition 5.3, where Condition 3 covers the case where r and s are introduced by different rule applications, and Condition 4 covers the case

where these two assertions are added by the same rule application.

Overall, we thus know that $\mathsf{cons}(P) \subseteq \overleftarrow{r} \subseteq \mathsf{cons}(R)$ and $P <_\Gamma R$. Since $R \in A_i$, by the induction hypothesis, we have once again that either $\mathsf{cons}(R) \subseteq \mathsf{cons}(A_0)$ or there exist $r' \in A_i \cap \widehat{\Delta}$ and $Q' \in A_i \cap \widehat{\Lambda}$ such that $\overleftarrow{r'} \subseteq \mathsf{cons}(Q'), \mathsf{cons}(R) \subseteq \overrightarrow{r'}$, and $R <_\Gamma Q'$. In both cases, the fact that $\mathsf{cons}(P) \subseteq \mathsf{cons}(R)$ and $P <_\Gamma R$, together with the transitivity of $\subseteq$ and $<_\Gamma$, yields the desired result. ■

An easy consequence of this lemma is that a path consisting of m new edges in a state generated by rule applications from a state in Γ^S implies a decreasing sequence w.r.t. $<_\Gamma$ of the same same length. Consequently, the length of such paths is bounded by the number of predicate symbols occurring in the finite cover Ω_Γ.

Proposition 5.7 *Let $\mathfrak{S}_0 \xrightarrow{*}_{S^m} \mathfrak{S}$ where $\mathfrak{S}_0 = (A_0, \mathcal{T}) \in \Gamma^S$ and $\mathfrak{S} = (A, \mathcal{T})$. Suppose that A contains edges $r_1, \ldots, r_m$ and nodes $N_0, \ldots, N_m$ such that for all $i, 1 \leq i \leq m$, $r_i \notin A_0$ and r_i connects N_{i-1} with N_i. Then, there exist assertions $Q_1, \ldots, Q_m \in A$ such that $Q_1 >_\Gamma Q_2 >_\Gamma \ldots >_\Gamma Q_m$.*

Proof. Since r_i connects N_{i-1} with N_i for $i = 1, \ldots, m$, we know by Definition 5.2 that $\overleftarrow{r_i} \subseteq \mathsf{cons}(N_{i-1})$ and $\mathsf{cons}(N_i) \subseteq \overrightarrow{r_i}$. This implies that $\overleftarrow{r_i} \subseteq \overrightarrow{r_{i-1}}$ for all $i, 1 < i \leq m$.

For each of the edges r_i we have assumed that it is new, i.e., $r_i \notin A_0$. Thus, r_i must have been added by some rule application. Condition 3 of Definition 5.3 entails then that, for every $1 \leq i \leq m$, $\overrightarrow{r_i} \cap \mathsf{cons}(A_0) = \emptyset$, and thus, for every $1 < i \leq m$ it also holds that $\overleftarrow{r_i} \cap \mathsf{cons}(A_0) = \emptyset$, as $\overleftarrow{r_i} \subseteq \overrightarrow{r_{i-1}}$.

Since r_m was added by a rule application, by Condition 2 of Definition 5.3, there must be an assertion $Q_m \in A \cap \widehat{\Lambda}$ such that $\overleftarrow{r_m} \subseteq \mathsf{cons}(Q_m)$. Hence, $\mathsf{cons}(Q_m) \not\subseteq \mathsf{cons}(A_0)$. By Lemma 5.6, there exist $r \in A \cap \widehat{\Delta}$ and $Q_{m-1} \in A \cap \widehat{\Lambda}$ such that $\overleftarrow{r} \subseteq \mathsf{cons}(Q_{m-1}), \mathsf{cons}(Q_m) \subseteq \overrightarrow{r}$, and $Q_m <_\Gamma Q_{m-1}$. We have $\overleftarrow{r_m} \subseteq \overrightarrow{r_{m-1}}$ and $\overleftarrow{r_m} \subseteq \mathsf{cons}(Q_m) \subseteq \overrightarrow{r}$, which implies that $\overrightarrow{r_{m-1}} \cap \overrightarrow{r} \neq \emptyset$. However, Conditions 3 and 4 of Definition 5.3 ensure that distinct assertions in $\widehat{\Delta} \setminus A_0$ must have disjoint sets of descendants. Thus, we know that $r = r_{m-1}$.

We can now apply the same argument as above to r_{m-1} and Q_{m-1} to obtain an assertion Q_{m-2} such that $\overleftarrow{r_{m-2}} \subseteq \mathsf{cons}(Q_{m-2}), \mathsf{cons}(Q_{m-1}) \subseteq \overrightarrow{r_{m-2}}$, and $Q_{m-1} <_\Gamma Q_{m-2}$. By iterating this argument, we thus obtain the desired descending chain $Q_1 >_\Gamma Q_2 >_\Gamma \ldots >_\Gamma Q_m$. ■

The following two remarks will be useful in the proof of the main theorem of this section. First, note that Condition 7 of Definition 5.3 ensures that the assertions from $\widehat{\Lambda}$ triggering a rule application all belong to the same node.

Second, given a new node N (i.e., one that was not present in the initial state) and an assertion $P \in N$, Lemma 5.6 yields an edge r such that $\mathsf{cons}(P) \subseteq \overrightarrow{r}$. Since distinct edges have disjoint sets of descendants (Condition 4 of Definition 5.3) any other assertion in $Q \in N$ also satisfies $\mathsf{cons}(Q) \subseteq \overrightarrow{r}$. This shows that the constants occurring in a node all belong to the descendant set of the edge whose introduction created the node.

We are now ready to show termination of the pinpointing extension of any ordered forest tableaux.

Theorem 5.8 *If S is an ordered forest tableau, then its pinpointing extension terminates on every input.*

Proof. Suppose that there is an input $\Gamma = (\mathcal{I}, \mathcal{T})$ for which there is an infinite sequence of pinpointing rule applications $\mathfrak{S}_0 \rightarrow_{S^{pin}} \mathfrak{S}_1 \rightarrow_{S^{pin}} \ldots$, with $\mathfrak{S}_0 \in \Gamma^S$. Since S is a covered tableau, there is a finite $\mathcal{T}$-cover Ω_Γ such that, for all $i \geq 0$, the assertions in $\mathfrak{S}_i$ use only predicate symbols from Ω_Γ. As noted above, for every node there is a fixed finite set of constants that can occur in the assertions of this node. This set is either the set of constants occurring in $\mathfrak{S}_0$ (for an old node) or it consists of the descendants in the unique edge whose introduction created the node (for a new node). Together with the fact that the $\mathcal{T}$-cover Ω_Γ is finite, this restricts the assertions that can occur in the node to a fixed finite set. Each of these assertion may repeatedly have its label modified by applications of the pinpointing rules. However, every application of a rule makes the label more general in the sense that the new monotone Boolean formula has more models than the previous one. Since these formulae are built over a finite set of propositional variables, this can happen only finitely often. The same argument shows that the label of a given edge can be changed only finitely often.

Hence, to get a non-terminating sequence of rule applications, infinitely many new nodes must be added. By Conditions 5 and 2 of Definition 5.3, each newly added node N is created as successor of an existing node w.r.t. a unique edge $r \in \widehat{\Delta}$ such that the constants in N are new constants contained in $\overrightarrow{r}$. If infinitely many new nodes are created, then either there is a node that obtains infinitely many direct successors, or an infinite chain of nodes is created, where each is a successor of the previous one.

Proposition 5.7 implies that the latter case cannot occur. In fact, given nodes $N_0, N_1, \ldots, N_m$ and edges $r_1, \ldots, r_m$ such that, for all $i, 1 \leq i \leq m$, r_i connects N_{i-1} to N_i, Proposition 5.7 yields a sequence of assertions $Q_1, \ldots, Q_m \in \widehat{\Lambda}$ such that $Q_1 >_\Gamma Q_2 >_\Gamma \ldots >_\Gamma Q_m$. However, the length of such a descending sequence is bounded by the cardinality of the finite $\mathcal{T}$-cover Ω_Γ. Thus, it is not possible that an infinite path is created by rule application.

Now, consider the first case, i.e., assume that there is a node N for which infinitely many successors are created. However, the constants in N are from a fixed finite set of constants C, and the predicate symbols that can occur in the applied rules must all belong to the finite $\mathcal{T}$-cover Ω_Γ. Thus, up to variable renaming, there are only finitely many rules that can be applied to N, and there are only finitely many ways of replacing the variables in the left-hand side of rules by constants from C. The fresh variables in the right-hand side are always replaced by distinct new constants. Thus, for a fixed rule and a fixed substitution σ replacing the variables in the left-hand side of this rules by constants from C, the assertions introduced by two different applications of this rule using σ only differ by a renaming of these new constants. By the way pinpointing rule applicability is defined, such renamed variants can only be added as long as their labels are not equivalent. But there are only finitely many labels up to equivalence. Thus, N can in fact obtain only a finite number of successors. This finishes the proof that the pinpointing extension of an ordered forest tableau always terminates. ■

Note that termination of the pinpointing extension implies termination of the original tableau. In fact, a non-terminating sequence of rule applications for the original tableau can easily be transformed into a non-terminating sequence of rule applications for its pinpointing extension.

Corollary 5.9 *An ordered forest tableau terminates on every input.*

The definition of forest tableaux imposes quite a number of restrictions on such tableaux. Thus, it is natural to ask whether all these restrictions are indeed necessary. The answer is yes: if any of these restrictions is removed, then Theorem 5.8 no longer holds. In fact, it is possible to construct terminating tableaux satisfying all other properties whose pinpointing extensions do not terminate. Here, we illustrate this by one example, where we remove Condition 6 of Definition 5.3.

Example 5.10 *Consider the tableau S that has the following four rules:*

$$\begin{array}{lcl}
\mathsf{R}_1 & : & (\{P(x)\}, \{ax_1\}) \rightarrow \{\{R(x), Q_1(x)\}\}, \\
\mathsf{R}_2 & : & (\{P(x)\}, \{ax_2\}) \rightarrow \{\{R(x), Q_2(x)\}\}, \\
\mathsf{R}_3 & : & (\{R(x)\}, \emptyset) \rightarrow \{\{r(x,y)\}, \{Q_1(x)\}, \{Q_2(x)\}\}, \\
\mathsf{R}_4 & : & (\{P(x), r(x,y)\}, \emptyset) \rightarrow \{\{T(y), r(x,z)\}\},
\end{array}$$

and where the function $\cdot^S$ maps every input $\mathcal{I} \in \mathfrak{I}$ to the singleton set $\{\{P(a)\}\}$, and each axiom in $\mathfrak{T} = \{ax_1, ax_2\}$ to the empty set.

It is easy to verify that S with the ordering $T < Q_2 < Q_1 < R < P$ satisfies all the conditions of an ordered forest tableau, except for Condition 6 of Definition 5.3.

For any axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, we have $\Gamma^S = \{(\{P(a)\}, \mathcal{T})\}$, and thus neither R_3 nor R_4 is applicable to Γ^S. Depending on which axioms are contained in $\mathcal{T}$, the rules R_1 and/or R_2 may be applicable. However, their application introduces $Q_1(a)$ or $Q_2(a)$ into the set of assertions, and thus the non-deterministic rule R_3 is not applicable. Obviously, R_4 becomes applicable only after R_3 has been applied. Consequently, S terminates on every axiomatized input Γ.

It is possible, however, to construct an infinite chain of pinpointing rule applications starting with $\Gamma^S = \{(\{P(a)\}, \{ax_1, ax_2\})\}$ where $\mathsf{lab}(P(a)) = \top$. In fact, we can first apply the rule R_1. This adds the assertions $R(a)$ and $Q_1(a)$, both with label ax_1. An application of the rule R_2 adds the assertion $Q_2(a)$ with label ax_2, and modifies the label of the assertion $R(a)$ to $\mathsf{lab}(R(a)) = ax_1 \vee ax_2$. At this point, we have reached an S-state $\mathfrak{S}$ containing the assertions $P(a)$, $R(a)$, $Q_1(a)$, $Q_2(a)$ with labels $\mathsf{lab}(P(a)) = \top$, $\mathsf{lab}(R(a)) = ax_1 \vee ax_2$, $\mathsf{lab}(Q_1(a)) = ax_1$, and $\mathsf{lab}(Q_2(a)) = ax_2$. The rule R_3 is pinpointing applicable to this S-state. Indeed, although both $Q_1(a)$ and $Q_2(a)$ are contained in the assertion set of $\mathfrak{S}$, their labels are not implied by $\mathsf{lab}(R(a))$. The application of R_3 to $\mathfrak{S}$ replaces $\mathfrak{S}$ by three new S-states. One of these new S-states contains the assertion $r(a, b)$ for a new constant b. At this point, rule R_4 becomes applicable. Its application adds the assertions $T(b)$ and $r(a, c)$ for a new constant c. Since there is no assertion of the form $T(c)$, R_4 becomes again applicable, and its application adds a new constant d within an assertion $r(a, d)$. It is easy to see that we can now continue applying rule R_4 indefinitely.

If we consider the tableau that has only the rule R_4 and where every input $\mathcal{I} \in \mathfrak{I}$ is mapped to $\{\{P(a), r(a, b)\}\}$, then this yields an example of a non-terminating tableau that satisfies all the conditions of an ordered forest tableau, except for Condition 6.

6 Conclusions

We have introduced a general notion of tableaux, and have shown that tableaux that are correct for a consequence property can be extended such that a terminating run of the extended procedure computes a pinpointing formula. This formula can then be used to derive minimal axiom sets and maximal non-axiom sets from it.

We have also shown that, in general, termination of a tableau does not imply termination of its pinpointing extension, even if all tableau rules are deterministic. To overcome this problem, we have then introduced the concepts of ordered forest tableaux, and have shown that the pinpointing extensions of ordered forest tableaux always terminate.

Our current framework has two restrictions that we will try to overcome in future work. First, our tableau rules always extend the current set of assertions. We do not allow for rules that can modify existing assertions. Thus, tableau-based algorithms that identify constants, like the rule treating at-most number restrictions in description logics (see, e.g., [7]), cannot be modelled. A similar problem occurs for the tableau systems introduced in [3]. There, it was solved by modifying the definition of rule application by allowing rules that introduce new individuals (in our notation: rules with fresh variables) to reuse existing individuals. However, this makes such rules intrinsically non-deterministic. In our setting, we believe that we can solve this problem more elegantly by introducing equality and inequality predicates.

Second, our approach currently assumes that a correct tableau always terminates, without considering additional blocking conditions. As shown in [18], extending a tableau with blocking to a pinpointing algorithm requires some additional effort. Solving this for the case of general tableaux will be a second important direction for future research.

References

[1] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI 2005*, pages 364–369. Morgan Kaufmann, Los Altos, 2005.

[2] 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, 2003.

[3] F. Baader, J. Hladik, C. Lutz, and F. Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57(2–4):247–279, 2003.

[4] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. of Automated Reasoning*, 14:149–180, 1995.

[5] F. Baader and R. Penaloza. Axiom pinpointing in general tableaux. In *Proc. Tableaux 07*, LNAI 4548, pages 11–27. Springer-Verlag, 2007.

[6] F. Baader and R. Peñaloza. Blocking and pinpointing in forest tableaux. LTCS-Report LTCS-08-02, Dresden University of Technology, Germany, 2008. See http://lat.inf.tu-dresden.de/research/reports.html.

[7] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[8] J. Bailey and P. J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *Proc. of PADL'05*, LNCS 3350, pages 174–186. Springer-Verlag, 2005.

[9] G. Davydov, I. Davydova, and H. Kleine Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Ann. of Mathematics and Artificial Intelligence*, 23(3–4):229–245, 1998.

[10] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, 2001.

[11] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of German Workshop on AI*, pages 38–47. Springer-Verlag, 1990.

[12] Bernhard Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Ann. of Mathematics and Artificial Intelligence*, 18(2–4):133–157, 1996.

[13] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR'98*, pages 636–647, 1998.

[14] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[15] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, and J. Hendler. Swoop: A Web ontology editing browser. *J. of Web Semantics*, 4(2), 2005.

[16] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In *Proceedings of the Third Int. Semantic Web Conf.*, Hiroshima, Japan, 2004.

[17] M. H. Liffiton and K. A. Sakallah. On finding all minimally unsatisfiable subformulas. In *Proc. of SAT'05*, LNCS 3569, pages 173–186. Springer-Verlag, 2005.

[18] T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of AAAI 2006*. AAAI Press/The MIT Press, 2006.

[19] D. Oberle, R. Volz, B. Motik, and S. Staab. An extensible ontology software environment. In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 311–333. Springer-Verlag, 2004.

[20] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW'05*, pages 633–640. ACM, 2005.

[21] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[22] S. Schlobach. Diagnosing terminologies. In *Proc. of AAAI 2005*, pages 670–675. AAAI Press/The MIT Press, 2005.

[23] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI 2003*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.

[24] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[25] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of DL 2004*, pages 212–213, 2004.

[26] K.A. Spackman, K.E. Campbell, and R.A. Cote. SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement.

[27] L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Proc. of DATE'03*, pages 10880–10885. IEEE Computer Society Press, 2003.