Axiom Pinpointing in General Tableaux

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

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

Corresponding address:
Technische Universität Dresden
Fakultät Informatik
Institut für Theoretische Informatik
D-01062 Dresden
Germany

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. Most of 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.

Keywords. explanation, tableau methods, automated reasoning, pinpointing

1 Introduction

Description logics (DLs) [3] 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 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 [17] as standard ontology language for the semantic web. As a consequence of this standardization, several ontology editors support OWL [21, 26, 20], 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 [16, 13, 32] employ tableau-based algorithms [8], 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 [33], the concept *Amputation-of-Finger* is classified as a subconcept of *Amputation-of-Hand*. 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 [30] 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 [30] 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 $\mathcal{ALC}$ [31], 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 [28], Schlobach [29] 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

[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.

minimal way to get rid of a certain unwanted consequence.

The problem of computing minimal (maximal) subsets of a DL knowledge base that have (do not have) a given consequence was actually considered earlier in the context of extending DLs by default rules. In [5], Baader and Hollunder solve this problem by introducing a labeled extension of the tableau-based consistency algorithm for $\mathcal{ALC}$-ABoxes [14], which is very similar to the one described later in [30]. The main difference is that the algorithm described in [5] does not directly compute minimal subsets that have a consequence, but rather a monotone Boolean formula, called *clash formula* in [5], 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 [30] was extended by Parsia et al. [27] to more expressive DLs, and the one of Baader and Hollunder [5] was extended by Meyer et al. [22] 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 [5] and [30] 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 [22] follows the approach introduced in [5], 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 [5, 30] 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 [5]. 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] or the congruence closure algorithm [24], which decides the word problem for finite sets of ground identities. This notion is simpler than the tableau systems introduced in [4] in the context of translating tableaux into tree automata, and in its most general form it is not restricted to tableau-based algorithms that

[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.

generate tree-like structures. However, it turns out that, in order to ensure termination of the pinpoint extension of a tableau or to model tableaux that employ blocking, we need to restrict our general framework to tableaux that generate forest-like structures.

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 *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. Tableau-based decision procedures for DLs that can deal with general concept inclusion axioms (GCIs) or transitive roles are not "naturally" terminating; instead, termination is achieved by employing the so-called *blocking* technique, which under certain conditions blocks the application of an otherwise applicable rule. In Section 6, we introduce *blocking tableaux*, which use this technique. We show that the pinpointing extension of such tableaux is correct and always terminates.

This paper is an extended and improved version of [6]. In contrast to [6], the present paper contains detailed proofs of the results of Section 4, and the results of Sections 5 and 6 are new.

2 Basic definitions

Before we can define our general notion of a tableau algorithm, we need to define the general form of inputs to which these algorithms are applied, and the decision problems they are supposed to solve.

Definition 2.1 (Axiomatized input, c-property) *Let $\mathfrak{I}$ be a set, called the set of* inputs*, and $\mathfrak{T}$ be a set, called the set of* axioms*, and let $\mathscr{P}_{admis}(\mathfrak{T}) \subseteq \mathscr{P}_{fin}(\mathfrak{T})$ be a set of finite subsets of $\mathfrak{T}$ such that $\mathcal{T} \in \mathscr{P}_{admis}(\mathfrak{T})$ implies $\mathcal{T}' \in \mathscr{P}_{admis}(\mathfrak{T})$ for all $\mathcal{T}' \subseteq \mathcal{T}$. An* axiomatized input *for $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$ is of the form $(\mathcal{I}, \mathcal{T})$ where $\mathcal{I} \in \mathfrak{I}$ and $\mathcal{T} \in \mathscr{P}_{admis}(\mathfrak{T})$. A* consequence property (c-property) *is a set $\mathcal{P} \subseteq \mathfrak{I} \times \mathscr{P}_{admis}(\mathfrak{T})$ such that $(\mathcal{I}, \mathcal{T}) \in \mathcal{P}$ implies $(\mathcal{I}, \mathcal{T}') \in \mathcal{P}$ for every $\mathcal{T}' \in \mathscr{P}_{admis}(\mathfrak{T})$ with $\mathcal{T}' \supseteq \mathcal{T}$.*

Intuitively, c-properties on axiomatized inputs are supposed to model consequence relations in logic, i.e., the c-property $\mathcal{P}$ holds if the input $\mathcal{I}$ "follows" from the axioms in $\mathcal{T}$. Such a set of axioms might be required to satisfy a certain property, which is taken care of by the admissibility restriction. For

example, TBoxes in DLs are often not arbitrary sets of terminological axiom, but may be restricted to being unambiguous and acyclic. The monotonicity requirement on c-properties corresponds to the fact that we want to restrict the attention to consequence relations induced by monotonic logics. In fact, for non-monotonic logics, looking at minimal sets of axioms that have a given consequence does not make much sense.

To illustrate Definition 2.1, assume that $\mathfrak{I}$ is a countably infinite set of propositional variables, that $\mathfrak{T}$ consists of all Horn clauses over these variables, i.e., implications of the form $p_1 \wedge \ldots \wedge p_n \rightarrow q$ for $n \geq 0$ and $p_1, \ldots, p_n, q \in \mathfrak{I}$, and $\mathscr{P}_{admis}(\mathfrak{T}) = \mathscr{P}_{fin}(\mathfrak{T})$. Then the following is a c-property according to the above definition: $\mathcal{P} := \{(p, \mathcal{T}) \mid \mathcal{T} \models p\}$, where $\mathcal{T} \models q$ means that all valuations satisfying all implications in $\mathcal{T}$ also satisfy q. As as concrete example, consider $\Gamma := (p, \mathcal{T})$ where $\mathcal{T}$ consists of the following implications:

$$\text{ax}_1\text{:}\quad \rightarrow q, \qquad \text{ax}_2\text{:}\quad \rightarrow s, \qquad \text{ax}_3\text{:}\ \ s \rightarrow q, \qquad \text{ax}_4\text{:}\ \ q \wedge s \rightarrow p \tag{1}$$

It is easy to see that $\Gamma \in \mathcal{P}$. Note that Definition 2.1 also captures the following variation of the above example, where $\mathfrak{I}'$ consist of tuples $(p, \mathcal{T}_1) \in \mathcal{I} \times \mathscr{P}_{fin}(\mathfrak{T})$ and the c-property is defined as $\mathcal{P}' := \{((p, \mathcal{T}_1), \mathcal{T}_2) \mid \mathcal{T}_1 \cup \mathcal{T}_2 \models p\}$. For example, if we take the axiomatized input $\Gamma' := ((p, \{\text{ax}_3, \text{ax}_4\}), \{\text{ax}_1, \text{ax}_2\})$, then $\Gamma' \in \mathcal{P}'$.

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 $\mathcal{S}' \subset \mathcal{S}$. Dually, a set of axioms $\mathcal{S} \subseteq \mathcal{T}$ is called a* maximal non-axiom set (MaNA) for Γ w.r.t. $\mathcal{P}$ *if $(\mathcal{I}, \mathcal{S}) \notin \mathcal{P}$ and $(\mathcal{I}, \mathcal{S}') \in \mathcal{P}$ for every $\mathcal{S}' \supset \mathcal{S}$. The set of all MinA (MaNA) for Γ w.r.t. $\mathcal{P}$ will be denoted as $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ ($\mathsf{MAX}_{\mathcal{P}(\Gamma)}$).*

Note that the notions of MinA and MaNA are only interesting in the case where $\Gamma \in \mathcal{P}$. In fact, otherwise the monotonicity property satisfied by $\mathcal{P}$ implies that $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \emptyset$ and $\mathsf{MAX}_{\mathcal{P}(\Gamma)} = \{\mathcal{T}\}$. In our example, where we have $\Gamma \in \mathcal{P}$, it is easy to see that $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \{\{\text{ax}_1, \text{ax}_2, \text{ax}_4\},\ \{\text{ax}_2, \text{ax}_3, \text{ax}_4\}\}$. In the variant of the example where only subsets of the facts $\{\text{ax}_1, \text{ax}_2\}$ can be taken, we have $\mathsf{MIN}_{\mathcal{P}'(\Gamma')} = \{\{\text{ax}_2\}\}$.

The set $\mathsf{MAX}_{\mathcal{P}(\Gamma)}$ can be obtained from $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ by computing the minimal hitting sets of $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$, and then complementing these sets [30, 23]. A set $\mathcal{S} \subseteq \mathcal{T}$ is a *minimal hitting set* of $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ if it has a nonempty intersection with every element of $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$, and no strict subset of $\mathcal{S}$ has this property. In our example, the minimal hitting sets of $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ are $\{\text{ax}_1, \text{ax}_3\}, \{\text{ax}_2\}, \{\text{ax}_4\}$, and thus

$\mathsf{MAX}_{\mathcal{P}(\Gamma)} = \{\{\mathrm{ax}_2, \mathrm{ax}_4\},\ \{\mathrm{ax}_1, \mathrm{ax}_3, \mathrm{ax}_4\},\ \{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_3\}\}$. Intuitively, to get a set of axioms that does not have the consequence, we must remove from $\mathcal{T}$ at least one axiom for every MinA, and thus the minimal hitting sets give us the minimal sets to be removed.

The reduction we have just sketched shows that it is enough to design an algorithm for computing all MinA, since the MaNA can then be obtained by a hitting set computation. It should be noted, however, that this reduction is not polynomial: there may be exponentially many hitting sets of a given collection of sets, and even deciding whether such a collection has a hitting set of cardinality $\leq n$ is an NP-complete problem [12]. Also note that there is a similar reduction involving hitting sets for computing the MinA from all MaNA.

Instead of computing MinA or MaNA, one can also compute the pinpointing formula.[5] To define the pinpointing 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 an axiom in $\mathcal{T}$. A *monotone Boolean formula* over $\mathsf{lab}(\mathcal{T})$ is a Boolean formula using (some of) the 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 the following holds for every valuation $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{T})$: $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ iff $\mathcal{V}$ satisfies ϕ.*

In our example, we can take $\mathsf{lab}(\mathcal{T}) = \{\mathrm{ax}_1, \ldots, \mathrm{ax}_4\}$ as set of propositional variables. It is easy to see that $(\mathrm{ax}_1 \vee \mathrm{ax}_3) \wedge \mathrm{ax}_2 \wedge \mathrm{ax}_4$ is a pinpointing formula for $\mathcal{P}$ and Γ.

Valuations can be ordered by set inclusion. The following is an immediate consequence of the definition of a pinpointing formula [5].

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*

$$\begin{aligned} \mathsf{MIN}_{\mathcal{P}(\Gamma)} &= \{T_\mathcal{V} \mid \mathcal{V} \textit{ is a minimal valuation satisfying } \phi\}, \\ \mathsf{MAX}_{\mathcal{P}(\Gamma)} &= \{T_\mathcal{V} \mid \mathcal{V} \textit{ is a maximal valuation falsifying } \phi\}. \end{aligned}$$

This shows that it is enough to design an algorithm for computing a pinpointing formula to obtain all MinA and MaNA. However, like the previous reduction for computing MaNA from MinA, the reduction suggested by the lemma is not polynomial. For example, to obtain $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ from ϕ, one

[5] This corresponds to the clash formula introduced in [5]. Here, we distinguish between the pinpointing formula, which can be defined independently of a tableau algorithm, and the clash formula, which is induced by a run of a tableau algorithm.

can bring ϕ into disjunctive normal form and then remove disjuncts implying other disjuncts. It is well-known that this can cause an exponential blowup. This should, however, not be viewed as a disadvantage of approaches computing the pinpointing formula rather than $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$. If such a blowup happens, then the pinpointing formula actually yields a compact representation of all MinAs. Conversely, the set $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$ can directly be translated into the pinpointing formula

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

In our example, the set $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \{\{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_4\}, \{\mathrm{ax}_2, \mathrm{ax}_3, \mathrm{ax}_4\}\}$ yields the pinpointing formula $(\mathrm{ax}_1 \wedge \mathrm{ax}_2 \wedge \mathrm{ax}_4) \vee (\mathrm{ax}_2 \wedge \mathrm{ax}_3 \wedge \mathrm{ax}_4)$.

3 A general notion of tableaux

Before introducing our general notion of a tableau-based decision procedure, we want to motivate it by first modelling a simple decision procedure for the property $\mathcal{P}$ introduced in the Horn clause example from the previous section, and then sketching extensions to the model that are needed to treat more complex tableau-based decision procedures.

Motivating examples

To decide whether $(p, \mathcal{T}) \in \mathcal{P}$, we start with the set $A := \{\neg p\}$, and then use the rule

$$\text{If } \{p_1, \ldots, p_n\} \subseteq A \text{ and } p_1 \wedge \ldots \wedge p_n \to q \in \mathcal{T} \text{ then } A := A \cup \{q\} \tag{2}$$

to extend A until it is saturated, i.e., it can no longer be extended with the above rule. It is easy to see that $(p, \mathcal{T}) \in \mathcal{P}$ (i.e., $\mathcal{T} \models p$) iff this saturated set contains both p and $\neg p$. For example, for the axioms in (1), one can first add s using ax_2, then q using ax_3, and finally p using ax_4. This yields the saturated set $\{\neg p, p, q, s\}$.

Abstracting from particularities, we can say that we have an algorithm that works on a set of *assertions* (in the example, assertions are propositional variables and their negation), and uses rules to extend this set. A *rule* is of the form $(B_0, \mathcal{S}) \to B_1$ where B_0, B_1 are finite sets of assertions, and $\mathcal{S}$ is a finite set of *axioms* (in the example, axioms are Horn clauses). Given a set of axioms $\mathcal{T}$ and a set of assertions A, this rule is *applicable* if $B_0 \subseteq A$, $\mathcal{S} \subseteq \mathcal{T}$, and $B_1 \not\subseteq A$. Its *application* then extends

A to $A \cup B_1$.[6] Our simple Horn clause algorithm always *terminates* in the sense that any sequence of rule applications is finite (since only right-hand sides of implications in $\mathcal{T}$ can be added). After termination, we have a *saturated* set of assertions, i.e., one to which no rule applies. The algorithm *accepts* the input (i.e., says that it belongs to $\mathcal{P}$) iff this saturated set contains a *clash* (in the example, this is the presence of p and $\neg p$ in the saturated set).

The model of a tableau-based decision procedure introduced so far is too simplistic since it does not capture two important phenomena that can be found in tableau algorithms for description and modal logics: non-determinism and assertions with an internal structure. Regarding *non-determinism*, assume that instead of Horn clauses we have more general implications of the form $p_1 \wedge \ldots \wedge p_n \rightarrow q_1 \vee \ldots \vee q_m$ in $\mathcal{T}$. Then, if $\{p_1, \ldots, p_n\} \subseteq A$, we need to choose (don't know non-deterministically) with which of the propositional variables q_j to extend A. In our formal model, the right-hand side of a non-deterministic rule consists of a finite set of sets of assertions rather than a single set of assertions, i.e., non-deterministic rules are of the more general form $(B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$ where $B_0, B_1, \ldots, B_m$ are finite sets of assertions and $\mathcal{S}$ is a finite set of axioms. Instead of working on a single set of assertions, the non-deterministic algorithm thus works on a finite set $\mathcal{M}$ of sets of assertions. The non-deterministic rule $(B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$ is applicable to $A \in \mathcal{M}$ if $B_0 \subseteq A$ and $\mathcal{S} \subseteq \mathcal{T}$, and its application replaces $A \in \mathcal{M}$ by the finitely many sets $A \cup B_1, \ldots, A \cup B_m$ provided that each of these sets really extends A. For example, if we replace ax_1 and ax_2 in (1) by ax_5: $\rightarrow p \vee s$, then starting with $\{\{\neg p\}\}$, we first get $\{\{\neg p, p\}, \{\neg p, s\}\}$ using ax_5, then $\{\{\neg p, p\}, \{\neg p, s, q\}\}$ using ax_3, and finally $\{\{\neg p, p\}, \{\neg p, s, q, p\}\}$ using ax_4. Since each of these sets contains a clash, the input is accepted.

Regarding the *structure of assertions*, in general it is not enough to use propositional literals. Tableau-based decision procedures in description and modal logic try to build finite models, and thus assertions must be able to describe the relational structure of such models. For example, assertions in tableau algorithms for description logics [8] are of the form $r(a, b)$ and $C(a)$, where r is a role name, C is a concept description, and a, b are individual names. Again abstracting from particularities, a *structured assertion* is thus of the form $P(a_1, \ldots, a_k)$ where P is a k-ary predicate and $a_1, \ldots, a_k$ are constants. As an example of the kind of rules employed by tableau-based algorithms for description logics, consider the rule treating existential restrictions:

$$\text{If } \{(\exists r.C)(x)\} \subseteq A \text{ then } A := A \cup \{r(x, y), C(y)\}. \tag{3}$$

[6] The applicability condition $B_1 \not\subseteq A$ ensures that rule application really *extends* the given set of assertions.

The variables x, y in this rule are place-holders for constants, i.e., to apply the rule to a set of assertions, we must first replace the variables by appropriate constants. Note that y occurs only on the right-hand side of the rule. We will call such a variable a *fresh variable*. Fresh variables must be replaced by *new constants*, i.e., a constant not occurring in the current set of assertions. For example, let $A := \{(\exists r.C)(a), r(a,b)\}$. If we apply the substitution $\sigma := \{x \mapsto a, y \mapsto c\}$ that replaces x by a and y by the new constant c, then the above rule is applicable with σ since $(\exists r.C)(a) \in A$. Its application yields the set of assertions $A' = A \cup \{r(a,c), C(c)\}$. Of course, we do not want the rule to be still applicable to A'. However, to prevent this it is not enough to require that the right-hand side (after applying the substitution) is not contained in the current set of assertions. In fact, this would not prevent us from applying the rule to A' with another new constant, say c'. For this reason, the applicability condition for rules needs to check whether the assertions obtained from the right-hand side by replacing the fresh variables by existing constants yields assertions that are already contained in the current set of assertions.

The formal definition

In the following, $\mathcal{V}$ denotes a countably infinite set of *variables*, and $\mathcal{D}$ a countably infinite set of *constants*. 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 $a_1, \ldots, a_n \in \mathcal{D}$. Likewise, 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. 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. We say that $\sigma : V \to \mathcal{D}$ is a *substitution on* V. 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 $\mathfrak{I}$ be a set of inputs and $\mathfrak{T}$ a set of axioms. A* tableau *for $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$ is a tuple $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ where*

- *Σ is a signature;*
- *$\cdot^S$ is a 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* $B_0, \ldots, B_m$ *are finite sets 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.

An *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 defining

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

Intuitively, on input $(\mathcal{I}, \mathcal{T})$, we start with the initial set $\mathcal{M} = (\mathcal{I}, \mathcal{T})^S$ of S-states, and then use the rules in $\mathcal{R}$ to modify this set. Each rule application picks an S-state $\mathfrak{S}$ from $\mathcal{M}$ and replaces it by finitely many new S-states $\mathfrak{S}_1, \ldots, \mathfrak{S}_m$ that extend the first component of $\mathfrak{S}$. If $\mathcal{M}$ is saturated, i.e., no more rules are applicable to $\mathcal{M}$, then we check whether all the elements of $\mathcal{M}$ contain a clash. If yes, then the input is accepted; otherwise, it is rejected.

Definition 3.2 (rule application, saturated, clash) *Given an 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* 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* $B_i\rho' \not\subseteq A$*.*

Given a set of S-states $\mathcal{M}$ *and an 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 in $\mathcal{M}$ *yields the new set* $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{T}) \mid i = 1, \ldots, m\}$*, 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, 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. As usual, the* reflexive-transitive closure *of* $\to_S$ *is denoted by* $\stackrel{*}{\to}_S$*. 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 can now define under what conditions a tableau S is correct for a c-property.

Definition 3.3 (correctness) *Let* $\mathcal{P}$ *be a c-property on axiomatized inputs for* $\mathfrak{I}$ *and* $\mathscr{P}_{admis}(\mathfrak{T})$*, and S a tableau for* $\mathfrak{I}$ *and* $\mathscr{P}_{admis}(\mathfrak{T})$*. Then 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 simple decision procedure sketched in our Horn clause example is a correct tableau in the sense of this definition. More precisely, it is a tableau with unstructured assertions (i.e., the signature contains only nullary predicate symbols) and deterministic rules. It is easy to see that the polynomial-time subsumption algorithm for the DL $\mathcal{EL}$ and its extensions introduced in [1] as well as the congruence closure algorithm [24] can be viewed as correct deterministic tableaux with unstructured assertions. The standard tableau-based decision procedure for concept unsatisfiability in the DL $\mathcal{ALC}$ [31] is a correct tableau that uses structured assertions and has a non-deterministic rule.

In 2. of Definition 3.3, we require that the algorithm gives the same answer independent of what terminating chain of rule applications is considered. Thus, the choice of which rule to apply next is don't care non-deterministic in a correct tableau. This is important since a need for backtracking over these choices would render a tableau algorithm completely impractical. However, in our framework this is not really an extra requirement on *correct* tableaux: it is built into our definition of rules and clashes.

Proposition 3.4 *Let Γ be an axiomatized input and $\mathcal{M}_0 := \Gamma^S$. If $\mathcal{M}$ and $\mathcal{M}'$ are saturated sets of S-states such that $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}$ and $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}'$, then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is full of clashes.*

This proposition is a special case of Lemma 4.12, which will be proved in the next section.

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.

To motivate our pinpointing extension of general tableaux, we first describe such an extension of the simple decision procedure sketched for our Horn clause example. The main idea is that assertions are also labeled with monotone Boolean formulae. In the example, where $\mathcal{T}$ consists of the axioms of (1) and the axiomatized input is $(p, \mathcal{T})$, the initial set of assertions consists of $\neg p$. To be more precise, we have $p^S = \{\{\neg p\}\}$ and $t^S = \emptyset$ for all $t \in \mathcal{T}$. The label of the initial assertion $\neg p$ is $\top$ since its presence depends only on the input p, and not on any of the axioms. By applying the rule (2) using axiom ax_2, we can add the assertion s. Since the addition of this assertion depends on the presence of ax_2, it receives label ax_2. Then we can use ax_3 to add q. Since this addition depends on the presence of ax_3 and of the assertion s, which has label ax_2, the label of this new assertion is $\mathrm{ax}_2 \wedge \mathrm{ax}_3$. There is, however, also another possibility to generate the assertion q: apply the rule (2) using axiom ax_1. In a "normal" run of the tableau algorithm, the rule would not be applicable since it would add an assertion that is already there. However, in the pinpointing extension we need to register this alternative way of generating q. Therefore, the rule is applicable using ax_1, and its application changes the label of the assertion q from $\mathrm{ax}_2 \wedge \mathrm{ax}_3$ to $\mathrm{ax}_1 \vee (\mathrm{ax}_2 \wedge \mathrm{ax}_3)$. Finally, we can use ax_4 to add the assertion p. The label of this assertion is $\mathrm{ax}_4 \wedge \mathrm{ax}_2 \wedge (\mathrm{ax}_1 \vee (\mathrm{ax}_2 \wedge \mathrm{ax}_3))$ since the application of the rule depends on the presence of ax_4 as well as the assertions s and q. The presence of both p and $\neg p$ gives us a clash, which receives label $\top \wedge \mathrm{ax}_4 \wedge \mathrm{ax}_2 \wedge (\mathrm{ax}_1 \vee (\mathrm{ax}_2 \wedge \mathrm{ax}_3))$. This so-called clash formula is the output of the extended algorithm. Obviously, it is equivalent to the pinpointing formula $(\mathrm{ax}_1 \vee \mathrm{ax}_3) \wedge \mathrm{ax}_2 \wedge \mathrm{ax}_4$ that we have constructed by hand in Section 2.

The formal definition

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

To illustrate the definition of rule application, let us look back at the example from the beginning of this section. There, we have looked at a situation where the current set of assertions is $A := \{\neg p, s, q\}$ where $\mathsf{lab}(\neg p) = \top$, $\mathsf{lab}(s) = \mathrm{ax}_2$, and $\mathsf{lab}(q) = \mathrm{ax}_2 \wedge \mathrm{ax}_3$. In this situation, the rule (2) is pinpointing applicable using ax_1. In fact, in this case the formula ψ is simply ax_1. Since this formula does not imply $\mathsf{lab}(q) = \mathrm{ax}_2 \wedge \mathrm{ax}_3$, the assertion q is ψ-insertable into A. Its insertion changes the label of q to $\mathrm{ax}_1 \vee (\mathrm{ax}_2 \wedge \mathrm{ax}_3)$.

Consider a chain of pinpointing rule applications $\mathcal{M}_0 \to_{S^{pin}} \ldots \to_{S^{pin}} \mathcal{M}_n$ 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: $\mathcal{V}(\mathcal{M}) := \{\mathcal{V}(\mathfrak{S}) \mid \mathfrak{S} \in \mathcal{M}\}$. The following lemma is an easy consequence of the definition of the clash formula:

Lemma 4.3 *Let* $\mathcal{M}$ *be a finite set of labeled* S*-states and* $\mathcal{V}$ *a propositional valuation. Then we have that* $\mathcal{V}$ *satisfies* $\psi_\mathcal{M}$ *iff* $\mathcal{V}(\mathcal{M})$ *is full of clashes.*

Proof. Assume that $\mathcal{V}(\mathcal{M})$ is full of clashes. Then we know that, for every S-state $\mathfrak{S}_i \in \mathcal{M}$, the projection $\mathcal{V}(\mathfrak{S}_i)$ contains a clash. Thus, for all i there is a clash set A_i in $\mathfrak{S}_i$ such that $\mathsf{lab}(a)$ is satisfied by $\mathcal{V}$ for every assertion $a \in A_i$. This shows that $\mathcal{V}$ satisfies ψ_{A_i}. Hence, $\mathcal{V}$ satisfies the formula

$$\bigvee_{A' \text{ clash set in } \mathfrak{S}_i} \psi_{A'}.$$

Since this holds for every $\mathfrak{S}_i \in \mathcal{M}$, the valuation $\mathcal{V}$ satisfies the clash formula $\psi_\mathcal{M}$.

Conversely, assume that $\mathcal{V}(\mathcal{M})$ is not full of clashes, i.e., there is a $\mathfrak{S}_i \in \mathcal{M}$ such that $\mathcal{V}(\mathfrak{S}_i)$ contains no clash. Thus, for every clash set A' in $\mathfrak{S}_i$, there must be an assertion $a \in A'$ such that $\mathcal{V}$ does not satisfy $\mathsf{lab}(a)$. Consequently, $\mathcal{V}$ does not satisfy the label $\psi_{A'}$ of any of the clash sets A' in $\mathfrak{S}_i$, and thus this valuation cannot satisfy the disjunction of these labels. This shows that $\mathcal{V}$ does not satisfy the clash formula. ■

There is also a close connection between pinpointing saturatedness of a set of labeled S-states and saturatedness of its projection:

Lemma 4.4 *Let* $\mathcal{M}$ *be a finite set of labeled* S*-states and* $\mathcal{V}$ *a propositional valuation. If* $\mathcal{M}$ *is pinpointing saturated, then* $\mathcal{V}(\mathcal{M})$ *is saturated.*

Proof. Suppose that there is an S-state $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$, and a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ such that R is applicable to $\mathcal{V}(\mathfrak{S})$ with substitution ρ. This means that $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V}$, $B_0\rho \subseteq A_\mathcal{V}$, and for every $i, 1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ, we have $B_i\rho' \not\subseteq A_\mathcal{V}$.

We show that R is pinpointing applicable to $\mathfrak{S}$ with the same substitution ρ. Since $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V} \subseteq \mathcal{T}$ and $B_0\rho \subseteq A_\mathcal{V} \subseteq A$, the first two conditions of the definition of "pinpointing applicable" are satisfied. For the third condition, consider an i and a substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ. We must show that $\mathsf{ins}_\psi(B_i\rho', A) \neq \emptyset$, where $\psi := \bigwedge_{b \in B_0} \mathsf{lab}(b\rho) \wedge \bigwedge_{s \in \mathcal{S}} \mathsf{lab}(s)$. Note that $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V}$ and $B_0\rho \subseteq A_\mathcal{V}$ imply that $\mathcal{V}$ satisfies ψ. Since $B_i\rho' \not\subseteq A_\mathcal{V}$, there is a $b \in B_i$ such that $b\rho' \notin A_\mathcal{V}$. Thus $b\rho' \notin A$ or $\mathcal{V}$ does not satisfy $\mathsf{lab}(b\rho')$. In the first case, $b\rho'$ is clearly ψ-insertable into A. In the second case, $\psi \not\models \mathsf{lab}(b\rho')$ since $\mathcal{V}$ satisfies ψ, and thus $b\rho'$ is again ψ-insertable into A. ■

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

> *For every chain of rule applications $\mathcal{M}_0 \to_{S^{pin}} \ldots \to_{S^{pin}} \mathcal{M}_n$ 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 want to 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 $\mathcal{M} \to_{S^{pin}} \mathcal{M}'$ does not imply $\mathcal{V}(\mathcal{M}) \to_S \mathcal{V}(\mathcal{M}')$. First, the assertions and axioms to which the pinpointing rule was applied in $\mathcal{M}$ may not be present in the projection $\mathcal{V}(\mathcal{M})$ since $\mathcal{V}$ does not satisfy their labels. Thus, we may also have $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$. Second, a pinpointing application of a rule may change the projection (i.e., $\mathcal{V}(\mathcal{M}) \neq \mathcal{V}(\mathcal{M}')$), although this change does not correspond to a normal application of this rule to $\mathcal{V}(\mathcal{M})$. For example, consider the tableau rule (3) treating existential restrictions in description logics, and assume that we have the assertions $(\exists r.C)(a)$ with label ax_1 and $r(a,b), C(b)$ with label ax_2. Then the rule (3) is pinpointing applicable, and its application adds the new assertions $r(a,c), C(c)$ with label ax_1, where c is a new constant. If $\mathcal{V}$ is a valuation that makes ax_1 and ax_2 true, then the $\mathcal{V}$-projection of our set of assertions contains $(\exists r.C)(a), r(a,b), C(b)$. Thus rule (3) is not applicable, and no new individual c is introduced. To overcome this second problem, we define a modified version of rule application, where the applicability condition (iii) from Definition 3.2 is removed.

Definition 4.6 (modified rule application) *Given an 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 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 $\mathcal{M} \to_{S^m} \mathcal{M}'$ 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.*

The next lemma relates modified rule application with "normal" rule application, on the one hand, and pinpointing rule application on the other hand. Note that "saturated" in the formulation of the first part of the lemma means saturated w.r.t. $\to_S$, as introduced in Definition 3.2.

Lemma 4.7 *Let $\Gamma = (\mathcal{I}, \mathcal{T})$ be an axiomatized input and $\mathcal{M}_0 = \Gamma^S$.*

1. *Assume that $\mathcal{M}_0 \overset{*}{\to}_S \mathcal{M}$ and $\mathcal{M}_0 \overset{*}{\to}_{S^m} \mathcal{M}'$ and that $\mathcal{M}$ and $\mathcal{M}'$ are saturated finite sets of S-states. Then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is full of clashes.*
2. *Assume that $\mathcal{M}$ and $\mathcal{M}'$ are finite sets of labeled S-states, and that $\mathcal{V}$ is a propositional valuation. Then $\mathcal{M} \to_{S^{pin}} \mathcal{M}'$ implies $\mathcal{V}(\mathcal{M}) \to_{S^m} \mathcal{V}(\mathcal{M}')$ or $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$. In particular, this shows that $\mathcal{M}_0 \overset{*}{\to}_{S^{pin}} \mathcal{M}$ implies $\mathcal{V}(\mathcal{M}_0) \overset{*}{\to}_{S^m} \mathcal{V}(\mathcal{M})$.*

Proof. To show the second statement, assume that $\mathcal{M} \to_{S^{pin}} \mathcal{M}'$, i.e., there is an S-state $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$ and a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ such that R is pinpointing applicable to $\mathfrak{S}$ with substitution ρ, and $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \Cup_\psi B_i\sigma, \mathcal{T}) \mid 1 \leq i \leq m\}$ where σ and ψ are as in the definition of pinpointing application. Take one of the newly added S-states $\mathfrak{S}_i = (A \Cup_\psi B_i\sigma, \mathcal{T}) \in \mathcal{M}'$. By the definition of ψ-insertion, we know that every assertion $a \in A \setminus \mathsf{ins}_\psi(B_i\sigma, A)$ keeps its old label $\mathsf{lab}(a)$, each newly added assertion in $\mathsf{ins}_\psi(B_i\sigma, A) \setminus A$ gets label ψ, and each assertion $b \in A \cap \mathsf{ins}_\psi(B_i\sigma, A)$ gets the modified label $\psi \vee \mathsf{lab}(b)$.

Thus, if $\mathcal{V}$ satisfies ψ, then we have $(A \Cup_\psi B_i\sigma)_\mathcal{V} = A_\mathcal{V} \cup B_i\sigma$ since the labels ψ of the newly added assertions and the modified labels are satisfied by $\mathcal{V}$. This implies that $\mathcal{V}(\mathcal{M}) \to_{S^m} \mathcal{V}(\mathcal{M}')$ since conditions (i) and (ii) of the definition of m-applicability follow from the fact that $\mathcal{V}$ satisfies ψ.

On the other hand, if $\mathcal{V}$ does not satisfy ψ, then $(A \Cup_\psi B_i\sigma)_\mathcal{V} = A_\mathcal{V}$ since the labels ψ of the newly added assertions are not satisfied by $\mathcal{V}$, and the disjunction with ψ does not change the evaluation of the modified labels under $\mathcal{V}$. Thus, in this case $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$.

The proof of the first statement of the lemma is more involved. We defer it till we have shown that this lemma indeed implies Theorem 4.5. ∎

Given this lemma, we can easily prove Theorem 4.5. Let $\Gamma = (\mathcal{I}, \mathcal{T})$ be an axiomatized input, and assume that $\mathcal{M}_0 \to_{S^{pin}} \ldots \to_{S^{pin}} \mathcal{M}_n$ such that $\mathcal{M}_0 = \Gamma^S$ and $\mathcal{M}_n$ is pinpointing saturated. We

must show that the clash formula $\psi := \psi_{\mathcal{M}_n}$ is a pinpointing formula for the property $\mathcal{P}$. This is an immediate consequence of the next two lemmas.

Lemma 4.8 *If $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ then $\mathcal{V}$ satisfies ψ.*

Proof. Let $\mathcal{N}_0 := (\mathcal{I}, \mathcal{T}_\mathcal{V})^S$. Since S terminates on every input, there is a saturated set $\mathcal{N}$ such that $\mathcal{N}_0 \xrightarrow{*}_S \mathcal{N}$. Since S is correct for $\mathcal{P}$ and $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$, we know that $\mathcal{N}$ is full of clashes. By 2. of Lemma 4.7, $\mathcal{M}_0 \xrightarrow{*}_{S^{pin}} \mathcal{M}_n$ implies $\mathcal{V}(\mathcal{M}_0) \xrightarrow{*}_{S^m} \mathcal{V}(\mathcal{M}_n)$. In addition, we know that $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$, and Lemma 4.4 implies that $\mathcal{V}(\mathcal{M}_n)$ is saturated. Thus, 1. of Lemma 4.7, together with the fact that $\mathcal{N}$ is full of clashes, implies that $\mathcal{V}(\mathcal{M}_n)$ is full of clashes. By Lemma 4.3, this implies that $\mathcal{V}$ satisfies $\psi = \psi_{\mathcal{M}_n}$. ■

Lemma 4.9 *If $\mathcal{V}$ satisfies ψ then $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$.*

Proof. Consider again a chain of rule applications $\mathcal{N}_0 = (\mathcal{I}, \mathcal{T}_\mathcal{V})^S \xrightarrow{*}_S \mathcal{N}$ where $\mathcal{N}$ is saturated. We have $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ if we can show that $\mathcal{N}$ is full of clashes. As in the proof of the previous lemma, we have that $\mathcal{V}(\mathcal{M}_0) \xrightarrow{*}_{S^m} \mathcal{V}(\mathcal{M}_n)$, $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$, and $\mathcal{V}(\mathcal{M}_n)$ is saturated. Since $\mathcal{V}$ satisfies $\psi = \psi_{\mathcal{M}_n}$, Lemma 4.3 implies that $\mathcal{V}(\mathcal{M}_n)$ is full of clashes. By 1. of Lemma 4.7, this implies that $\mathcal{N}$ is full of clashes. ■

To complete the proof of Theorem 4.5, it remains to show the first part of Lemma 4.7. Before proving this result, we introduce the notion of a *sub-state*. An S-state $\mathfrak{S}$ is a sub-state of an S-state $\mathfrak{S}'$ if every assertion and axiom in $\mathfrak{S}$ also occurs $\mathfrak{S}'$. However, we need to make the notion more general by allowing for different constants to be used in the S-states as long as there is a "renaming" from the constants of $\mathfrak{S}$ into the ones of $\mathfrak{S}'$ such that the desired inclusion between their sets of assertions holds.

Definition 4.10 (sub-state) *The S-state $\mathfrak{S} = (A, \mathcal{T})$ is a* sub-state *of $\mathfrak{S}' = (A', \mathcal{T}')$, denoted as $\mathfrak{S} \preceq \mathfrak{S}'$, if $\mathcal{T} \subseteq \mathcal{T}'$ and there exists a renaming function $f : \mathsf{cons}(A) \to \mathsf{cons}(A')$ such that $P(a_1, \ldots, a_k) \in A$ implies $P(f(a_1), \ldots, f(a_k)) \in A'$.*

It is important to notice that, for every pair of S-states $\mathfrak{S} = (A, \mathcal{T})$ and $\mathfrak{S}' = (A', \mathcal{T}')$ such that $\mathfrak{S} \preceq \mathfrak{S}'$, the following holds: if there is a set B of patterns and a substitution ρ on $\mathsf{var}(B)$ such that $B\rho \subseteq A$, then $\rho' := \rho \circ f$ (where f is the renaming function that yields $\mathfrak{S} \preceq \mathfrak{S}'$) satisfies $B\rho' \subseteq A'$. In particular, this implies that $\mathfrak{S}'$ contains a clash whenever $\mathfrak{S}$ does.

Lemma 4.11 *Let $\mathcal{N}$ and $\mathcal{N}_0$ be sets of S-states, $\mathfrak{S} \in \mathcal{N}$, $\mathfrak{S}_0 \in \mathcal{N}_0$, with $\mathcal{N}_0$ saturated. If $\mathfrak{S} \preceq \mathfrak{S}_0$, then for every $\mathcal{N} \to_{\mathsf{R}^m} \mathcal{N}'$ there is $\mathfrak{S}' \in \mathcal{N}'$ such that $\mathfrak{S}' \preceq \mathfrak{S}_0$.*

Proof. If $\mathcal{N}'$ is obtained by the application of R to an S-state different from $\mathfrak{S}$, then $\mathfrak{S} \in \mathcal{N}'$ and thus nothing needs to be shown. Suppose now that the rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ is applied to $\mathfrak{S}$ with substitution ρ to obtain $\mathcal{N}'$, and let $\mathfrak{S} = (A, \mathcal{T})$ and $\mathfrak{S}_0 = (A_0, \mathcal{T}_0)$. Since $\mathfrak{S} \preceq \mathfrak{S}_0$, this implies that $\mathcal{S} \subseteq \mathcal{T} \subseteq \mathcal{T}_0$ and there is a substitution ρ' on $\mathsf{var}(B_0)$ such that $B_0\rho' \subseteq A_0$. Thus, conditions (i) and (ii) of the definition of rule applicability are satisfied for $\mathfrak{S}_0$, R, and ρ'. Since $\mathcal{N}$ is saturated, R actually cannot be applicable to $\mathfrak{S}_0$ with ρ'; hence, condition (iii) cannot hold. This means that there must exist an $i, 1 \leq i \leq m$ and a substitution ς on $\mathsf{var}(B_0 \cup B_i)$ extending ρ' such that $B_i\varsigma \subseteq A_0$.

On the other hand, a substitution σ extending ρ is used to construct the new set $\mathcal{N}'$ through the application of the rule R to $\mathfrak{S}$. Let $\mathfrak{S}' := (A \cup B_i\sigma, \mathcal{T})$. Since σ maps the fresh variables of R to distinct new constants, we can extend the renaming function f to $f' : \mathsf{cons}(A \cup B_i\sigma) \to \mathsf{cons}(A_0)$ by setting $f'(\sigma(x)) := \varsigma(x)$ for every fresh variable x of R appearing in B_i. This defines a complete renaming function f' for the constants in $A \cup B_i\sigma$, and by definition this function satisfies $\sigma \circ f' = \varsigma$.

We show that $\mathfrak{S}' \preceq \mathfrak{S}_0$ with the renaming function f'. Thus, assume that $P(a_1, \ldots, a_k)$ is in $A \cup B_i\sigma$. If this assertion belongs to A, then $P(f'(a_1), \ldots, f'(a_k)) = P(f(a_1), \ldots, f(a_k)) \in A_0$ since $\mathfrak{S} \preceq \mathfrak{S}_0$ with the renaming function f. If $P(a_1, \ldots, a_k) \in B_i\sigma$, then $P(a_1, \ldots, a_k) = P(\sigma(x_1), \ldots, \sigma(x_k))$ for variables $x_1, \ldots, x_k \in \mathsf{var}(B_0 \cup B_i)$. Since $\sigma \circ f' = \varsigma$, we have

$$P(f'(a_1), \ldots, f'(a_k)) = P(\varsigma(a_1), \ldots, \varsigma(a_k)) \in B_i\varsigma \subseteq A_0.$$

This completes our proof that $\mathfrak{S}' \preceq \mathfrak{S}_0$. ∎

The following lemma generalizes both Proposition 3.4 and the first part of Lemma 4.7.

Lemma 4.12 *Let Γ be an axiomatized input and $\mathcal{M}_0 := \Gamma^S$. If $\mathcal{M}$ and $\mathcal{M}'$ are saturated sets of S-states such that $\mathcal{M}_0 \xrightarrow{*}_{S^m} \mathcal{M}$ and $\mathcal{M}_0 \xrightarrow{*}_{S^m} \mathcal{M}'$, then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is full of clashes.*

Proof. Note that the application of a rule to a set of S-states removes one of these S-states, and adds a finite number of S-states that extend the removed one. Thus, for every S-state $\mathfrak{S} \in \mathcal{M}'$, there is an S-state $\mathfrak{S}_0 \in \mathcal{M}_0$ such that $\mathfrak{S}_0 \preceq \mathfrak{S}$.

Consider the chain of (modified) rule applications $\mathcal{M}_0 \to_{S^m} \mathcal{M}_1 \to_{S^m} \ldots \to_{S^m} \mathcal{M}_n = \mathcal{M}$ that leads from $\mathcal{M}_0$ to $\mathcal{M}$. Since $\mathcal{M}'$ is saturated, Lemma 4.11 yields that, for every $\mathfrak{S} \in \mathcal{M}'$, there is an S-state $\mathfrak{S}_1 \in \mathcal{M}_1$ such that $\mathfrak{S}_1 \preceq \mathfrak{S}$. By iterating this argument, we obtain that, for every $\mathfrak{S} \in \mathcal{M}'$, there is an element $\mathfrak{S}_n \in \mathcal{M}$ such that $\mathfrak{S}_n \preceq \mathfrak{S}$.

Now, assume that $\mathcal{M}$ is full of clashes, i.e., every element of $\mathcal{M}$ contains a clash. To show that $\mathcal{M}'$ is full of clashes, consider $\mathfrak{S} \in \mathcal{M}'$. Then there is an element $\mathfrak{S}_n \in \mathcal{M}$ such that $\mathfrak{S}_n \preceq \mathfrak{S}$. The fact that $\mathfrak{S}_n$ contains a clash then implies that $\mathfrak{S}$ also contains a clash. This proves the implication from left to right of the lemma. A symmetric argument can be used to prove the converse direction. ■

This completes the proof of Theorem 4.5. The theorem considers a terminating chain of pinpointing rule applications. Unfortunately, termination of a tableau S in general does not imply termination of its pinpointing extension. The reason is that a rule may be pinpointing applicable in cases where it is not applicable in the normal sense (see the discussion above Definition 4.6). This can occur even if the tableau contains purely deterministic rules, as shown in the following example.

Example 4.13 *Consider the tableau S that has the following three rules*

$$\begin{aligned}
\mathsf{R}_1 &: (\{P(x)\}, \{ax_1\}) \to \{\{r(y,y,y), Q_1(y), Q_2(y)\}\},\\
\mathsf{R}_2 &: (\{P(x)\}, \{ax_2\}) \to \{\{r(y,y,y), Q_1(y), Q_2(y)\}\},\\
\mathsf{R}_3 &: (\{Q_1(x), Q_2(y)\}, \emptyset) \to \{\{r(x,y,z), Q_1(y), Q_2(z)\}\},
\end{aligned}$$

where the function $\cdot^S$ maps every input $\mathcal{I} \in \mathfrak{I}$ to the singleton set $\{\{P(a)\}\}$, and every axiom from $\mathfrak{T} = \{ax_1, ax_2\}$ to the empty set. We assume that every subset of $\mathfrak{T}$ is admissible. For any axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, we have $\Gamma^S = \{(\{P(a)\}, \mathcal{T})\}$. Depending on which axioms are contained in $\mathcal{T}$, the rules R_1 and/or R_2 may be applicable, but R_3 is not. Notice that R_1 and R_2 have the same right-hand side, and thus application of R_1 or R_2 to Γ^S leads to the same S-state, modulo the chosen new constant introduced for the fresh variable y. Suppose we apply one of these two rules, introducing b as the new constant. Then the resulting S-state is be given by $\mathfrak{S} = (\mathcal{A}, \mathcal{T})$ where

$$A = \{P(a), Q_1(b), Q_2(b), r(b,b,b)\}.$$

No rule is applicable to $\mathfrak{S}$. In fact, in order to apply rule R_1 or R_2, the only way to satisfy Condition (ii) in the definition of rule application is to use a substitution that maps x to the constant a. Extending this substitution to map y to b violates Condition (iii) of the definition of rule application since the assertions $Q_1(b), Q_2(b)$ and $r(b,b,b)$ were already introduced by the first rule application. To satisfy Condition (ii) for rule R_3, we must choose the substitution ρ that maps x and y to b. Thus, extending ρ by mapping z to b violates Condition (iii). This shows that S indeed terminates on every axiomatized input.

However, it is possible to construct an infinite chain of pinpointing rule applications starting with $\Gamma^S = \{(\{P(a)\}, \{ax_1, ax_2\})\}$ *where* $\mathsf{lab}(P(a)) = \top$. *We can first apply rule* R_1 *leading to the S-state* $\mathfrak{S}$ *described above, where all the assertions, except for* $P(a)$, *are labeled with* ax_1. *Rule* R_2 *is pinpointing applicable to* $\mathfrak{S}$*: although there is an extension of the corresponding substitution such that all the assertions exist already in* $\mathfrak{S}$, *these assertions are labeled with the formula* ax_1, *which is not implied by* ax_2. *The pinpointing application of* R_2 *to* $\mathfrak{S}$ *adds the assertions* $Q_1(c), Q_2(c), r(c,c,c)$ *with label* ax_2. *We can now apply* R_3 *to the resulting S-state* $\mathfrak{S}'$ *with the substitution* ρ *mapping* x *and* y *to* b *and* c, *respectively. Since the S-state* $\mathfrak{S}'$ *does not contain any assertion of the form* $r(b, c, \_)$, *Condition (iii) is not violated. This rule application adds the assertions* $r(b,c,d), Q_2(d)$ *with label* $ax_1 \wedge ax_2$. *It is easy to see that the rule* R_3 *can now be applied repeatedly, thus producing an infinite chain of pinpointing rule applications.*

This example shows that the termination of a tableau S does not necessarily imply the termination of its pinpointing extension. One way to overcome this problem could be to find sufficient conditions under which termination transfers from a tableau to its pinpointing extension. However, finding such a condition that is independent of the reason why a tableau terminates has turned out to be quite hard. 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 [31, 8]. 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 description. 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 sub-description of the input concept description.

If we look at the extension of this algorithm to one that decides consistency of $\mathcal{ALC}$-ABoxes (see, e.g., [14, 8]), 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 $\widehat{\Lambda}$. An assertion from $\widehat{\Delta}$ 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 (graph structure) *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 $1, 3$. 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 in our proofs, but must be kept in mind when dealing with graph-structures and their corresponding graphs.

Recall that the tableau-based decision procedure for consistency of $\mathcal{ALC}$-ABoxes [14, 8] 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 behavior, but are 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 a*

connected subset of $\widehat{\Lambda}$*, 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 the creation of 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 5(i) 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 $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_n\}$ 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 $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_n\}$, 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 $C(a)$ (role assertions $r(a, b)$) where C is a sub-description of a concept description occurring in the input ABox $\mathcal{A}_0$ (where r is a role

occurring in the input ABox $\mathcal{A}_0$). Thus one can take the set of sub-descriptions of concept descriptions occurring in $\mathcal{A}_0$ 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 order.

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 $\mathfrak{S}_0 \to_{S^m} \mathfrak{S}_1 \to_{S^m} \cdots$ a sequence of modified rule applications starting with $\mathfrak{S}_0 \in \Gamma^S$. 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 $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_n\}$ 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 cases, 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 $Q \in B_0\sigma \cap \widehat{\Lambda}$ 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.*

6 Tableaux with blocking

The ordered forest tableaux introduced above can be used to model tableau-based algorithms that try to generate a finite tree- or forest-shaped model. In the presence of so-called general concept

inclusion axioms (GCIs) or transitive roles, DLs lose the finite tree/forest model property, and thus these algorithms need no longer terminate. Termination can be regained, however, by blocking the application of generating rules, i.e., rules that generate new nodes, in case that the node to which the rule is supposed to be applied has a predecessor node that has the same assertions. A saturated and clash-free tableau can then be unraveled into an infinite tree/forest model (see, e.g., [18]).

In order to illustrate our general model of tableaux with blocking, we consider a non-terminating forest tableau that can be made terminating by blocking. Note that the usual tableau-based algorithm for $\mathcal{ALC}$ extended with inverse roles and GCIs shows a similar behavior.

Example 6.1 *Consider a forest tableau S with the following three (deterministic) rules*

$$\begin{aligned}
\mathsf{R}_1 &: (\{C(x)\}, \emptyset) \to \{\{r(x,y), D(y)\}\},\\
\mathsf{R}_2 &: (\{D(x)\}, \emptyset) \to \{\{r(x,y), C(y)\}\},\\
\mathsf{R}_3 &: (\{C(x), r(y,x)\}, \emptyset) \to \{\{\neg D(y)\}\},
\end{aligned}$$

and the clash $\{D(x), \neg D(x)\}$. In addition, we assume that the function $\cdot^S$ maps every input $\mathcal{I} \in \mathfrak{I}$ to the singleton set $\{\{C(a_0)\}\}$ and each axiom in $\mathfrak{T}$ to the empty set. It is easy to see that S does not terminate since it can produce an infinite chain of assertions of the form $C(a_0), r(a_0,a_1), D(a_1), r(a_1,a_2), C(a_2), \ldots$. If we apply rule R_1 followed by R_2 to $\Gamma^S = \{(\{C(a_0)\}, \emptyset)\}$, then we obtain the S-state $(A, \emptyset)$ consisting of the assertions $A := \{C(a_0), r(a_0,a_1), D(a_1), r(a_1,a_2), C(a_2)\}$. At this point, blocking should prevent the application of R_1 to the node a_2:[7] it is the repeated application of R_1 that causes the generation of the above infinite chain of assertions. The reason why R_1 can be blocked is that the node a_2 contains the same assertions as its predecessor a_0: both have an assertion for C. Note, however, that the application of R_3 to a_2, which adds the assertion $\neg D(a_1)$, should still be possible. In fact, otherwise the clash could not be detected. After application of R_3, we reach the S-state $(A \cup \{\neg D(a_1)\}, \emptyset)$, where the only applicable rule is R_1, which is however blocked. Thus, the blocking variant of the tableau terminates with this blocking-saturated state.

Before we can formalize our notion of tableaux with blocking, we need to introduce some notation. In the following we always assume that we have a forest tableau S. Given an input Γ, any S-state that can be generated from Γ^S by the applications of the rules of S is called an *S-state for Γ*. In the following we assume that all the S-states that we consider are S-state for some input.

The rule $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ is called *generating* if there is an $i, 1 \leq i \leq m$, such that

[7] Since in this forest tableau the elements of Λ are all unary, nodes correspond to constants.

$B_i \cap \widehat{\Delta} \neq \emptyset$. Note that the definition of forest tableaux implies that, if such a generating rule is applicable with substitution ρ in state $\mathfrak{S}$, then $\mathfrak{S}$ contains a node N such that $B_0\rho \subseteq N$. We can thus talk about *the* node to which a generating rule is applicable or applied. Given an S-state $\mathfrak{S}$ for the input Γ, a node N in $\mathfrak{S}$ is *new* if it has been generated by the application of a generating rule. Note that this is the case iff $\mathsf{cons}(N) \cap \mathsf{cons}(\Gamma^S) = \emptyset$.

Given two nodes N, N', we say that they contain the same assertions (written $N \equiv N'$) if there is a bijection $f : \mathsf{cons}(N) \to \mathsf{cons}(N')$ such that $P(a_1, \ldots, a_n) \in N$ iff $P(f(a_1), \ldots, f(a_n)) \in N'$.

Definition 6.2 (blocking) *Given a forest tableau S, and an axiomatized input Γ, let $\mathfrak{S}$ be an S-state for Γ. The* blocking relation $\lhd$ *between nodes of $\mathfrak{S}$ is defined as follows:*

$$N_1 \lhd N_2 \quad \textit{iff} \quad N_1 \equiv N_2, N_2 \textit{ is a predecessor of } N_1, \textit{and } N_1 \textit{ is a new node.}$$

The node N is blocked *if either there is a node N' such that $N \lhd N'$, or the parent node of N is blocked. A non-generating rule is* $\lhd$-applicable *if it is applicable in the sense of Definition 3.2; a generating rule is* $\lhd$-applicable *if it is applicable and the node N to which it is applicable is not blocked. For sets of S-states $\mathcal{M}, \mathcal{M}'$ (S-states $\mathfrak{S}, \mathfrak{S}'$) we write $\mathcal{M} \to_S^{\lhd} \mathcal{M}'$ ($\mathfrak{S} \to_S^{\lhd} \mathfrak{S}'$) if $\mathcal{M} \to_S \mathcal{M}'$ ($\mathfrak{S} \to_S \mathfrak{S}'$) using a rule that is $\lhd$-applicable. The set of S-states $\mathcal{M}$ is* $\lhd$-saturated *if there is no $\mathcal{M}'$ such that $\mathcal{M} \to_S^{\lhd} \mathcal{M}'$.*

Let $\mathcal{P}$ be a c-property on axiomatized inputs for $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$, and S a forest tableau for $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$. Then S is $\lhd$-correct for $\mathcal{P}$ *if it terminates and is sound and complete with respect to $\lhd$-application, i.e., the following two conditions hold for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$:*

1. *there is no infinite chain of rule applications $\Gamma^S = \mathcal{M}_0 \to_S^{\lhd} \mathcal{M}_1 \to_S^{\lhd} \ldots$;*
2. *for every chain of rule applications $\Gamma^S = \mathcal{M}_0 \to_S^{\lhd} \ldots \to_S^{\lhd} \mathcal{M}_n$ such that $\mathcal{M}_n$ is $\lhd$-saturated we have that $\Gamma \in \mathcal{P}$ iff $\mathcal{M}_n$ is full of clashes.*

In the DL literature, different forms of blocking have been used. The variant that we model here is usually called *equality blocking* [18] since it requires that the blocked and the blocking node have the same assertions. In *subset blocking* [2], it is only required that the blocking node has all the assertions of the blocked node, but not vice versa. Our reason for using equality blocking rather than subset blocking is that it is more appropriate for DLs with inverse roles, and our notion of forest tableaux can model tableau-based algorithms for DLs with inverse roles. DLs that have both inverse roles and number restrictions require more complex notions of blocking, such as *pair-wise blocking* [19], that look not just at one node but at a node and it neighbors. Since our current notion of tableaux does not

capture rules that can identify constants, as used in tableau-based algorithms for DLs with number restrictions [15], we have decided not to model pair-wise blocking.

The notion of blocking introduced in Definition 6.2 ensures that every covered forest tableau terminates with respect to $\lhd$-application on all inputs. Instead of showing this directly, we will prove that this is the case even for its pinpointing extension. But first, we must extend the notion of blocking to the pinpointing extension. Obviously, the extended notion must take the labels of assertions into account as well.

Given an input Γ, any S-state that can be generated from Γ^S by the applications of the rules of the pinpointing extension of S is called a *labeled S-state for Γ*. Nodes of such a labeled S-state will be called *labeled nodes*. Given two such labeled nodes N, N', we say that they contain the same labeled assertions (written $N \equiv_{pin} N'$) if there is a bijection $f : \mathsf{cons}(N) \to \mathsf{cons}(N')$ such that $P(a_1, \ldots, a_n) \in N$ iff $P(f(a_1), \ldots, f(a_n)) \in N'$, and the labels of these two assertions are equivalent.

Definition 6.3 (pinpointing blocking) *Given a forest tableau S, and an axiomatized input Γ, let $\mathfrak{S}$ be a labeled S-state for Γ. The* blocking relation $\lhd_{pin}$ *between labeled nodes of $\mathfrak{S}$ is defined as follows:*

$$N_1 \lhd_{pin} N_2 \quad \textit{iff} \quad N_1 \equiv_{pin} N_2, N_2 \textit{ is a predecessor of } N_1, \textit{and } N_1 \textit{ is a new node.}$$

The node N is pinpointing blocked *if either there is a node N' such that $N \lhd_{pin} N'$, or the parent node of N is pinpointing blocked.*

The notions $\lhd_{pin}$-applicable and $\lhd_{pin}$-application as well as $\to^{\lhd}_{S^{pin}}$ and $\lhd_{pin}$-saturated are defined in the obvious way.

Our approach for proving termination of the pinpointing extension of a covered forest tableau with respect to $\lhd_{pin}$-application is similar to the one employed for showing that ordered forest tableaux always terminate. The next lemma can be proved just like Lemma 5.6.

Lemma 6.4 *Let S be a forest tableau, Γ an axiomatized input, and $\mathfrak{S}_0 \to_{S^m} \mathfrak{S}_1 \to_{S^m} \cdots$ a sequence of modified rule applications starting with $\mathfrak{S}_0 \in \Gamma^S$. 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 $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$ and $\mathsf{cons}(P) \subseteq \overrightarrow{r}$.*

Equipped with this lemma, we can now prove the desired termination result.

Theorem 6.5 *Let S be a covered forest tableau. Then the pinpointing extension of S terminates with respect to $\lhd_{pin}$-application 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 $\mathfrak{S}_0 \rightarrow_{S^{pin}} \mathfrak{S}_1 \rightarrow_{S^{pin}} \cdots$, where $\mathfrak{S}_0 \in \Gamma^S$. Since S is a covered tableau, there is a finite $\mathcal{T}$-cover Ω_Γ such that the assertions in $\mathfrak{S}_i$ use only predicate symbols from Ω_Γ, for every $i \geq 0$. As already noted, every node has a fixed finite set of constants that can appear in its assertions. By Lemma 6.4, this set is either the set of constants occurring in $\mathfrak{S}_0$ (for an old node) or the descendants in the unique edge by which the node was created (for a new node). Since the $\mathcal{T}$-cover is finite, the assertions that can occur in a given node form a finite set. Each of these assertions may repeatedly have its label modified by pinpointing rule applications; however, every pinpointing rule application produces a more general label, in the sense that the new monotone Boolean formula has more models than the previous one. Since these formulas are built over a finite set of propositional variables, this can happen only finitely often. Analogously, the label of a given edge can be changed only finitely often.

Hence, to produce a non-terminating sequence of rule applications, infinitely many new nodes must be added. Conditions 5 and 2 of Definition 5.3 ensure that every newly added node N is created as a successor of an existing node with a unique edge $r \in \widehat{\Delta}$ connecting them, and all the constants in N are new constants appearing in $\overrightarrow{r}$. If infinitely many new nodes are created, then either there is a node with infinitely many direct successors, or an infinite chain of nodes, each one being a successor of the previous, is created. The first case can be treated as in the proof of Theorem 5.8.

Thus, we concentrate on the second case. The number of constants occurring in a new node is bounded by the largest arity of a predicate name $r \in \widehat{\Delta}$. Taking into account that there are also only finitely many possible labels, this implies that there can only be finitely many different labelled nodes, up to constant renaming. Then, for every chain of nodes $N_0, N_1, \ldots, N_m$ that is sufficiently long (i.e., where m is larger than the maximal number of labelled nodes that are different up to constant renaming), there must exist $1 \leq k < \ell \leq m$ such that $N_k \equiv_{pin} N_\ell$, and thus N_ℓ is pinpointing blocked by N_k. Consequently, all the nodes N_r for $r > \ell$ are blocked, which in particular means that N_m cannot get a successor node. Thus, the second case is not possible as well, which completes the proof of the theorem. ■

As in the case of ordered tableaux, termination of the pinpointing extension also implies termination of the original tableau.

Corollary 6.6 *Let S be a covered forest tableau. Then S terminates with respect to $\lhd$-application on every input.*

We have seen that blocking can be used to regain termination of non-terminating covered forest

tableaux, and that this also the case for the pinpointing extension. However, since blocking prevents the application of rules that would be applicable in the normal sense, the proof of correctness of the pinpointing extension given in Section 4 does not apply directly to the pinpointing extension of tableaux with blocking.

Our proof of correctness will rely on the notion of the *folded version* of an S-state, which is obtained by removing all blocked nodes and adding new edges. Let S be a forest tableau and $\mathfrak{S} = (A, \mathcal{T})$ an S-state for an input Γ. Then $\mathfrak{S}$ is a *forest-structure*, i.e., it is a graph-structure consisting of a set of tree-like structures growing out of the original graph-structure induced by the input. If we remove all the blocked nodes that are descendants of other blocked nodes, we obtain a new forest-structure $\mathfrak{S}' = (A', \mathcal{T})$ in which blocked nodes appear only as leafs in the trees. For every pair of nodes N_1 and N_2 in $\mathfrak{S}'$, if N_1 is blocked by N_2, then we know that $N_1 \equiv N_2$, and hence there is a bijection $f : \mathsf{cons}(N_1) \to \mathsf{cons}(N_2)$ such that $P(a_1, \ldots, a_n) \in N_1$ iff $P(f(a_1), \ldots, f(a_n)) \in N_2$. We modify the edge with destination N_1 (i.e., the unique assertion $r(\overleftarrow{r}, \overrightarrow{r}) \in \widehat{\Delta} \cap A'$ with $\mathsf{cons}(N_1) \subseteq \overrightarrow{r}$) to $r(\overleftarrow{r}, f(\overrightarrow{r}))$ and then remove N_1. Since $f(\overrightarrow{r})$ contains only constants from N_2, this new edge points to N_2, i.e., to the node that blocks N_1. By applying this modification for all the remaining blocked nodes, we obtain the *folded version* of $\mathfrak{S}$, which we denote by $\mathfrak{S}^{\circlearrowleft}$. If $\mathcal{M}$ is a set of S-states, then its *folded version* is $\mathcal{M}^{\circlearrowleft} = \{\mathfrak{S}^{\circlearrowleft} \mid \mathfrak{S} \in \mathcal{M}\}$.

Let us illustrate folding of S-states by using the tableau of Example 6.1. We have seen there that rule application can be used to obtain the $\lhd$-saturated S-state $\mathfrak{S} = (A, \mathcal{T})$ where $A = \{C(a_0), r(a_0, a_1), D(a_1), \neg D(a_1), r(a_1, a_2), C(a_2)\}$. The folded version of this S-state does not contain the constant a_2 (since the blocked node $\{C(a_2)\}$ has been removed), but it makes up for this by an edge from a_1 to a_0, i.e., $\mathfrak{S}^{\circlearrowleft} = (A^{\circlearrowleft}, \mathcal{T})$ with $A^{\circlearrowleft} = \{C(a_0), r(a_0, a_1), D(a_1), \neg D(a_1), r(a_1, a_0)\}$.

The next lemma will allow us to reuse some of the results shown in Section 4, by relating $\lhd$-saturatedness of a state to "normal" saturatedness of the corresponding folded state.

Lemma 6.7 *If $\mathfrak{S}$ is $\lhd$-saturated, then $\mathfrak{S}^{\circlearrowleft}$ is saturated.*

Proof. Let $\mathfrak{S} = (A, \mathcal{T}), \mathfrak{S}^{\circlearrowleft} = (A^{\circlearrowleft}, \mathcal{T})$ and $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ be applicable to $\mathfrak{S}^{\circlearrowleft}$ with substitution ρ. Assume first that R is a generating rule, and let N be the node in $A^{\circlearrowleft}$ to which this rule is applied, i.e., $B_0\rho \subseteq N \subseteq A^{\circlearrowleft}$. Since folding never modifies any nodes in the graph structure, except from removing some, N is also a node in $\mathfrak{S}$, i.e., $B_0\rho \subseteq N \subseteq A$. As $\mathfrak{S}$ is $\lhd$-saturated, R is not $\lhd$-applicable to it. This means that either N is blocked, or there is a substitution σ extending ρ such that $B_i\sigma \subseteq A$ for some $i, 1 \leq i \leq m$. Since folding removes all blocked nodes and N belongs to $A^{\circlearrowleft}$, the first case cannot occur; thus, the second option must be the case. We can then construct

a substitution σ' extending ρ such that $B_i\sigma' \subseteq A^{\circlearrowleft}$ as follows: for every $x \in \bigcup_{j=0}^{m} \mathsf{var}(B_j)$, if $\sigma(x)$ is a constant in a non-blocked node of A, then we define $\sigma'(x) := \sigma(x)$; if $\sigma(x)$ belongs to a node N_1 blocked by some non-blocked node N_2, then in particular $N_1 \equiv N_2$, and thus there exists a bijection $f : \mathsf{cons}(N_1) \to \mathsf{cons}(N_2)$ such that $P(a_1, \ldots, a_n) \in N_1$ iff $P(f(a_1), \ldots, f(a_n)) \in N_2$; in this case, we define $\sigma'(x) = f(\sigma(x))$. Because these bijections are also used when defining the folded state, it is easy to see that $B_i\sigma' \subseteq A^{\circlearrowleft}$ indeed holds. This contradicts our assumption that R is applicable to $\mathfrak{S}^{\circlearrowleft}$ with substitution ρ.

Suppose now that R is a non-generating rule. If $B_0\rho \subseteq A$, since $\lhd$-applicability coincides with regular applicability for non-generating rules, the proof is analogous to the one for the previous case. Thus, we can assume w.l.o.g. that $B_0\rho \not\subseteq A$. Then, $B_0\rho$ must contain edges r that were added by the folding process; these edges are of the form $r = p(\overleftarrow{r}, f_r(\overrightarrow{r}))$ where f_r is the bijection ensuring equivalence between blocked and the blocking node, and there are corresponding edges in A that have blocked nodes as destinations. Using the bijections f_r to rename constants, we can define a substitution ρ' such that $B_0\rho' \subseteq A$. Note that this inclusion depends on our use of equality blocking. In fact, an assertion $P\rho \in B_0\rho$ may be an assertion in a blocking node N, whose constants are renamed in ρ' such that they belong to a node N' blocked by N. Thus, we need to know that all the assertions occurring in the blocking node also occur (appropriately renamed) in the blocked node. This is guaranteed by our definition of $\equiv$.

Since $\mathfrak{S}$ is $\lhd$-saturated, R is not applicable to $\mathfrak{S}$ with substitution ρ', which implies that there must exist an $i, 1 \leq i \leq m$ such that $B_i\rho' \subseteq A$. We claim that $B_i\rho \subseteq A^{\circlearrowleft}$. This is an easy consequence of the facts that (i) the assertions of non-blocked nodes in A are contained also in $A^{\circlearrowleft}$; and (ii) the assertions of blocked nodes in A are contained in a renamed variant in the blocking node (i.e., the node to which the edge leading to the blocked node has been redirected). ■

As done in Section 4, we will use projections of labeled S-states to show the correctness of the pinpointing extension. The next lemma states a close connection between pinpointing $\lhd$-saturatedness of a set of labeled S-states and $\lhd$-saturatedness of its projection.

Lemma 6.8 *Let $\mathcal{M}$ be a finite set of labeled S-states and $\mathcal{V}$ a propositional valuation. If $\mathcal{M}$ is pinpointing $\lhd$-saturated, then $\mathcal{V}(\mathcal{M})$ is $\lhd$-saturated.*

Proof. Suppose that there is an S-state $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$ and a rule $\mathsf{R} = (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ such that R is $\lhd$-applicable to $\mathcal{V}(\mathfrak{S})$ with substitution ρ. For non-generating rules, applicability and $\lhd$-applicability coincide. Consequently, if R in non-generating, then we can re-use the proof of Lemma 4.4, which shows the result for the case without blocking.

Thus, assume that R is a generating rule. We have that $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V}, B_0\rho \subseteq A_\mathcal{V}$, for every $i, 1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ, it holds that $B_i\rho' \not\subseteq A_\mathcal{V}$, and the node N' in $\mathcal{V}(\mathfrak{S})$ to which the rule is applied is not blocked.

We will show now that R is pinpointing $\lhd$-applicable to $\mathfrak{S}$ with the same substitution ρ. Since $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V} \subseteq \mathcal{T}$ and $B_0\rho \subseteq A_\mathcal{V} \subseteq A$, the first two conditions of pinpointing applicability are satisfied. For the third condition, consider an i and a substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ. We must show that $\mathsf{ins}(B_i\rho', A) \neq \emptyset$ where $\psi = \bigwedge_{b \in B_0} \mathsf{lab}(b\rho) \wedge \bigwedge_{s \in \mathcal{S}} \mathsf{lab}(s)$. Note that $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V}$ and $B_0\rho \subseteq A_\mathcal{V}$ imply that $\mathcal{V}$ satisfies ψ. Since $B_i\rho' \not\subseteq A_\mathcal{V}$, there is a $b \in B_i$ such that $b\rho' \notin A_\mathcal{V}$. Thus $b\rho' \notin A$ or $\mathcal{V}$ does not satisfy $\mathsf{lab}(b\rho')$. In the first case, $b\rho'$ is clearly ψ-insertable into A. In the second case, $\psi \not\models \mathsf{lab}(b\rho')$ since $\mathcal{V}$ satisfies ψ, and thus $b\rho'$ is again ψ-insertable into A.

We have shown up to now that R is pinpointing applicable to $\mathfrak{S}$ with the substitution ρ. It remains to show that the node $N \subseteq A$ to which this rule is applicable (i.e., the node satisfying $B_0\rho \subseteq N \subseteq A$) is not pinpointing blocked. If N is not a new node, then it cannot be blocked. Thus, we can restrict the attention to the case where N is a new node. Since $B_0\rho \subseteq A_\mathcal{V}$, we have $B_0\rho \subseteq N_\mathcal{V}$. Thus, the node N' in $\mathcal{V}(\mathfrak{S})$ to which the rule R is applied is a subset of $N_\mathcal{V}$.[8] We know that this node is not blocked. Also note that, since this node belongs to $\mathcal{V}(\mathfrak{S})$, the sequence of edges in $\mathfrak{S}$ that leads to the node N is also contained in $\mathcal{V}(\mathfrak{S})$ and leads to this node. In fact, the label of an edge is always implied by the labels of assertions occurring in nodes or as edges below this edge.

Assume that N is pinpointing blocked. We concentrate on the case where there is a predecessor node M of N such that $M \equiv_{pin} N$. (The case where the parent node of N is blocked can be reduced to this case by considering, instead of N, the (unique) predecessor node N' of N that is blocked, but whose parent node is not blocked.) The definition of the relation $\equiv_{pin}$ implies that there is a bijection f such that, for every assertion $P(a_1, \ldots, a_n) \in N' \subseteq N_\mathcal{V}$ the assertion $P(f(a_1), \ldots, f(a_n)) \in M_\mathcal{V}$. The fact that the assertions in N' are connected implies that their f-images in $M_\mathcal{V}$ are also connected, and thus they belong to a node $M' \subseteq M_\mathcal{V}$. This shows, however, that N' is blocked by M', which is a contradiction. ■

Notice that if $\mathcal{M} \rightarrow^{\lhd}_S \mathcal{M}'$, then it is also the case that $\mathcal{M} \rightarrow_S \mathcal{M}'$, and analogously for pinpointing rule application: if $\mathcal{M} \rightarrow^{\lhd}_{S^{pin}} \mathcal{M}'$, then $\mathcal{M} \rightarrow_{S^{pin}} \mathcal{M}'$. This, along with (2) of Lemma 4.7, shows that $\mathcal{M} \rightarrow^{\lhd}_{S^{pin}} \mathcal{M}'$ implies that either $\mathcal{V}(\mathcal{M}) \rightarrow_{S^m} \mathcal{V}(\mathcal{M}')$ or $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$. In particular, $\mathcal{M}_0 \overset{*}{\rightarrow}^{\lhd}_{S^{pin}} \mathcal{M}$ implies $\mathcal{V}(\mathcal{M}_0) \overset{*}{\rightarrow}_{S^m} \mathcal{V}(\mathcal{M})$.

One last observation before proceeding to the proof of correctness of the pinpointing extension is

[8] Note that connectedness of N need not imply connectedness of $N_\mathcal{V} \subseteq N$.

that the order in which rules are applied has no influence on the result of a blocking tableau.

Lemma 6.9 *Let Γ be an axiomatized input, and $\mathcal{M}_0 = \Gamma^S$. If there are $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}$ and $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}'$ and $\mathcal{M}, \mathcal{M}'$ are both $\lhd$-saturated, then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is also full of clashes.*

Proof. For every S-state $\mathfrak{S} \in \mathcal{M}'$, there is an S-state $\mathfrak{S}_0 \in \mathcal{M}_0$ such that $\mathfrak{S}_0 \preceq \mathfrak{S}$, where the corresponding constant renaming function is the identity. Recall that folding only changes assertions involving blocked nodes, and that only new nodes can be blocked. Consequently, we also have $\mathfrak{S}_0 \preceq \mathfrak{S}^{\circlearrowleft}$. Since $\mathfrak{S}^{\circlearrowleft}$ is saturated by Lemma 6.7, Lemma 4.11 thus yields an S-state $\mathfrak{S}' \in \mathcal{M}$ such that $\mathfrak{S}' \preceq \mathfrak{S}^{\circlearrowleft}$.

Now, assume that $\mathcal{M}$ is full of clashes, i.e., every element of $\mathcal{M}$ contains a clash. To show that $\mathcal{M}'$ is full of clashes, consider $\mathfrak{S} \in \mathcal{M}'$. Then there is an element $\mathfrak{S}' \in \mathcal{M}$ such that $\mathfrak{S}' \preceq \mathfrak{S}^{\circlearrowleft}$. Since $\mathcal{M}$ is full of clashes, $\mathfrak{S}'$ contains a clash, and thus $\mathfrak{S}^{\circlearrowleft}$ also contains a clash. Since $\mathfrak{S}^{\circlearrowleft}$ is obtained from $\mathfrak{S}$ by removing blocked nodes and changing some edges, and since clashes consider only single nodes, this implies that $\mathfrak{S}$ also contains a clash.

The other direction can be shown analogously. ∎

Theorem 6.10 (correctness of pinpointing with blocking) *Let S be a forest tableau for $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$ that is $\lhd$-correct for the c-property $\mathcal{P}$. Then the following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$:*

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

Proof. Let $\Gamma = (\mathcal{I}, \mathcal{T})$ be an axiomatized input, and assume that $\Gamma^S = \mathcal{M}_0 \xrightarrow{*}^{\lhd}_{S^{pin}} \mathcal{M}_n$ with $\mathcal{M}_n$ pinpointing $\lhd$-saturated. To show that $\psi_{\mathcal{M}_n}$ is a pinpointing formula for $\mathcal{P}$, we have to show that, for every propositional valuation $\mathcal{V}$, it holds that $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ iff $\mathcal{V}$ satisfies $\psi_{\mathcal{M}_n}$.

Let $\mathcal{N}_0 = (\mathcal{I}, \mathcal{T}_\mathcal{V})^S$. Since S terminates w.r.t. $\lhd$-application, there is a $\lhd$-saturated set $\mathcal{N}$ such that $\mathcal{N}_0 \xrightarrow{*}^{\lhd}_S \mathcal{N}$. Also, as $\mathcal{M}_0 \xrightarrow{*}^{\lhd}_{S^{pin}} \mathcal{M}_n$, we have that $\mathcal{V}(\mathcal{M}_0) \xrightarrow{*}_{S^m} \mathcal{V}(\mathcal{M}_n)$. Additionally, $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$ and also $\mathcal{V}(\mathcal{M}_n)$ is $\lhd$-saturated. Thus, Lemma 6.9 yields that $\mathcal{N}$ is full of clashes iff $\mathcal{V}(\mathcal{M}_n)$ is full of clashes. By the $\lhd$-correctness of S for $\mathcal{P}$, we have then that $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ iff $\mathcal{N}$ is full of clashes iff $\mathcal{V}(\mathcal{M}_n)$ is full of clashes iff $\mathcal{V}$ satisfies $\psi_{\mathcal{M}_n}$ (Lemma 4.3). ∎

Our notion of $\lhd$-correctness explicitly requires termination w.r.t. $\lhd$-application. For covered forest tableaux we have seen that this condition is always satisfied.

Corollary 6.11 *Let S be a covered forest tableau for $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$ that is sound and complete w.r.t. $\lhd$-application, i.e., for every chain of rule applications $\Gamma^S = \mathcal{M}_0 \rightarrow_S^{\lhd} \ldots \rightarrow_S^{\lhd} \mathcal{M}_n$ such that $\mathcal{M}_n$ is $\lhd$-saturated we have that $\Gamma \in \mathcal{P}$ iff $\mathcal{M}_n$ is full of clashes. Then the following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathscr{P}_{admis}(\mathfrak{T})$:*

1. *There is no infinite chain of rule applications $\Gamma^S = \mathcal{M}_0 \rightarrow_{S^{pin}}^{\lhd} \mathcal{M}_1 \rightarrow_{S^{pin}}^{\lhd} \ldots$;*
2. *For every chain of rule applications $\Gamma^S = \mathcal{M}_0 \rightarrow_{S^{pin}}^{\lhd} \ldots \rightarrow_{S^{pin}}^{\lhd} \mathcal{M}_n$ such that $\mathcal{M}_n$ is pinpointing $\lhd$-saturated, the clash formula $\psi_{\mathcal{M}_n}$ induced by $\mathcal{M}_n$ is a pinpointing formula for $\mathcal{P}$ and Γ.*

7 Discussion

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 concept of an ordered forest tableau, and have shown that ordered forest tableaux and their pinpointing extensions always terminate. Tableau algorithms for DLs that allow for general concept inclusion axioms (GCIs) or transitive roles use blocking to achieve termination. We have shown that forest tableaux also provide us with a framework that can be used to model tableau algorithms that employ blocking. More precisely, we have shown that covered forest tableaux that use equality blocking always terminate, and that the same is true for their pinpointing extension. In addition, correctness transfers from forest tableaux with equality blocking to their pinpointing extension.

One of the surprising lessons that we have learned through the development of this general approach is that termination of pinpointing extensions of terminating tableau-based algorithms is far from trivial. In fact, our results show that, in general, termination does not transfer. Instead, the specific reason for the termination of standard tableau-based algorithms for DLs (namely, that they build finitely branching trees of bounded depth) is strong enough to ensure also termination of the pinpointing extension. Another lesson learned is that, for tableaux with blocking, the proof of correctness of the pinpointing extension is much more problematic than for the case without blocking. This also has consequences for the development of pinpointing extensions of specific tableau algorithms since it shows where one needs to be particularly careful in the proofs.

The results for pinpointing extensions of general tableaux presented in this paper subsume several results on pinpointing extensions of specific tableau algorithms published in the DL literature. For example, the tableau algorithms and their pinpointing extensions presented in [5, 30, 22] are instances of our approach, and thus termination and correctness of these pinpointing extensions follow from our general results. More precisely, correctness of the pinpointing algorithms for the DL $\mathcal{ALC}$ presented in [5, 30] follows from the results of Section 4, and termination from the results of Section 5. Termination and correctness of the pinpointing algorithm for $\mathcal{ALC}$ with GCIs presented in [22] follow from the results of Section 6. In [22], the extension of their approach to the DL $\mathcal{SI}$, which extends $\mathcal{ALC}$ with transitive and inverse roles, is mentioned as an open problem. The known tableau algorithms for $\mathcal{SI}$ with GCIs and its extension $\mathcal{SHI}$ by role hierarchies [18] are covered forest tableaux that employ equality blocking. Consequently, the results of Section 6 yield correct and terminating pinpointing algorithms for these DLs.

In addition to standard tableau algorithms for DLs like the ones mentioned above, also other decision procedures that are usually not considered to be tableau-based are instances of our general notion. As already mention in Section 3, the polynomial-time subsumption algorithms for the DL $\mathcal{EL}$ and its extensions introduced in [1] as well as the congruence closure algorithm [24] can be viewed as correct deterministic tableaux with unstructured assertions. Since termination is trivial in the unstructured case, the results of Section 4 yield correct and terminating pinpointing extensions of these algorithms. In particular, correctness of the pinpointing algorithm for $\mathcal{EL}^+$ described in [7] follows from these results.[9]

Axiom pinpointing has also been investigated 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 [23, 10], but also algorithms that extend a resolution-based SAT solver directly [11, 34]. 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. Since the standard tableau algorithm for propositional (un)satisfiability is a correct tableau with unstructured assertions (in the sense introduced in this paper), the results of Section 4 yield a correct and terminating pinpointing algorithm for propositional (un)satisfiability, which can be used to compute maximally satisfiable and minimally unsatisfiable subsets of a set of propositional formulae.

[9] Note that in [7] we already refer the reader to these general results for the proof of correctness of the pinpointing algorithm.

It should be noted, however, that tableau algorithms usually do not yield efficient algorithms for propositional (un)satisfiability, which is probably the reason why the SAT community has concentrated on other approaches.

In the area of satisfiability modulo theories (SMT) [25], one is interested in computing small sets of axioms that have a given consequence w.r.t. a specific theory. One of the theories considered in this context is the theory of equality with uninterpreted functions (EUF), for which consequences can be computed using congruence closure [24]. As mentioned above, the congruence closure algorithm can be viewed as a deterministic tableau with unstructured assertions, and thus the results of Section 4 can be used to obtain an algorithm that computes all minimal sets of axioms that have a given consequence w.r.t. EUF. It should be noted, however, that in the context of SMT it is usually sufficient to have one such set, which should be "small," but need not be minimal [24].

In first-order theorem proving, work that tries to explain a given consequence usually does not focus on pinpointing. The reason is that one often has relatively few axioms, but very long proofs. Thus, the main problem is to present these proofs in a compact and comprehensible way. For example, the proof of a theorem in group theory usually involves all the three axioms of the standard equational theory defining groups, and thus axiom pinpointing would not be of any help here. In contrast, DL-based ontologies often consist of a huge number of axioms, but a given consequence usually depends only on a few axioms, and the proof of the consequence from these axioms is usually quite short [9]. This explains why the DL community spends considerable effort on developing pinpointing algorithms.

Our current framework cannot deal with the standard tableau-based algorithms for DLs with number restrictions [15, 8]. In fact, 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 standard rule treating at-most number restrictions (see, e.g., [8]), cannot be modelled. A similar problem occurs for the tableau systems introduced in [4]. 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. In addition, number restrictions require a more complex notion of blocking (called pair-wise blocking) if added to a DL with GCIs and inverse roles [19, 8]. We believe that the approach used in Section 6 can be extended such that it can treat pair-wise blocking, but this would make the definitions and proofs more complicated.

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, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[3] 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.

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

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

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

[7] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of KI2007*, LNAI 4667, pages 52–67. Springer-Verlag, 2007.

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

[9] F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of KR-MED'08*, Poenix, Arizona, 2008.

[10] 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.

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

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (CA, USA), 1979.

[13] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, LNCS 2083, pages 701–706. Springer-Verlag, 2001.

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

[15] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR'91*, pages 335–346, 1991.

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

[17] 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.

[18] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[19] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.

[20] 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.

[21] 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.

[22] K. Lee, Th. Meyer, and J. Pan. Computing maximally satisable terminologies for the description logic $\mathcal{ALC}$ with GCIs. In *Proc. of DL 2006*, CEUR Electronic Workshop Proceedings, 2006.

[23] 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.

[24] R. Nieuwenhuis and A. Oliveras. Fast congruence closure and extensions. *Information and Computation*, 205(4), 2007.

[25] R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Challenges in satisfiability modulo theories. In *Proc. of RTA'07*, LNCS 4533, pages 2–18, 2007.

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

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

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

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

[30] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI 2003*, pages 355–362, 2003.

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

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

[33] 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.

[34] 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.