Blocking and Pinpointing in Forest Tableaux

Franz Baader Rafael Peñaloza

LTCS-Report 08-02

Blocking and Pinpointing in Forest Tableaux

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

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

Abstract

Axiom pinpointing has been introduced in description logics (DLs) to help the used understand the reasons why consequences hold by computing minimal subsets of the knowledge base that have the consequence in consideration. Several pinpointing algorithms have been described as extensions of the standard tableau-based reasoning algorithms for deciding 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, using specific traits of them. In the past, we have developed a general approach for extending tableau-based algorithms into pinpointing algorithms.

In this paper we explore some issues of termination of general tableaux and their pinpointing extensions. We also define a subclass of tableaux that allows the use of so-called blocking conditions, which stop the execution of the algorithm once a pattern is found, and adapt the pinpointing extensions accordingly, guaranteeing its correctness and termination.

1 Introduction

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

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

but perhaps their most notable success so far is the adoption of the DL-based language OWL [9] as standard ontology language for the semantic web. As a consequence of this, ontologies written in OWL are more usual. With the size of such ontologies, the importance of developing tools that support improving the quality of large DL-based ontologies grows. A common approach for detecting inconsistencies and infering other implicit consequences consists in applying tableau-based algorithms [6].

Whenever a consequence holds, it is usually desirable to understand the reasons for it, or even to decide how to change the ontology if it is an unwanted consequence. This is a hard task, and would be unrealistic to try to perform it without the help of an automated reasoning tool in ontologies with hundreds of thousands of axioms.

Axiom pinpointing has been introduced to provide that support. Most of the pinpointing algorithms described in the DL literature (see, e.g. [3, 13, 12, 11, 10]) are obtained as extensions of tableau-based reasoning algorithms [6] for deciding consequences of DL knowledge bases. These papers describe the pinpointing algorithms and prove their correctness only for a specific DL using a specific type of knowledge base, and it is in general not clear on how this approach can be generalized, nor which of the known tableau-based algorithms for DLs can be extended in a similar fashion. For example, the pinpointing extension described in [10], which can deal with general concept inclusions (GCIs) in the DL $\mathcal{ALC}$, follows the approach introduced in [3], but since GCIs require the introduction of so-called blocking conditions into the tableau-based algorithm to ensure termination [6], there are some new non-trivial problems to be solved.

To avoid the need of designing a new pinpointing extension for every possible tableau-based algorithm, and needing to prove its correctness every time, we introduced in [4] a general approach for extending tableau-based algorithms to pinpointing algorithms. This approach, however, is limited only to tableau-based algorithms that terminate without any cylce-checking mechanisms such as blocking. Furthermore, even if the tableau-based algorithm terminates, it is not necessarily the case that the pinpointing extension does that too. In fact, we will show that the problem of verifying whether a tableau-based algorithm (or its pinpointing extension) terminates on every input is undecidable.

A subclass of tableau-based algorithm that ensure termination of both the original algorithm and its pinpointing extension was presented in [5]. The algorithms in this class produce forest-like models with a bounded depth. In this paper we use a similar approach to define the notion of blocking, and extend the pinpointing methods to apply also for the case when the tableau execution is stoped by means of blocking.

The paper is organized as follows. The next section introduces the main basic notions regarding the problem we are trying to solve. Afterwards, in Section 3, we briefly recall the notions of general tableaux and their pinpointing extension, and show that it is in general undecidable whether their execution stops after a finite number of steps. Finally, we introduce in Section 4 the notion of forest tableaux and show how blocking techniques can be applied to them. We also adapt the pinpointing extension accordingly, and show its correctness and termination for a specific kind of blocking called subset blocking. All this is followed by some conclusions and thoughts on future work.

2 Basic Definitions

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

Definition 1 (Axiomatized input, c-property) *Let $\mathfrak{I}$ and $\mathfrak{T}$ be the sets of* inputs *and* axioms*, respectively. An* axiomatized input *over these sets is of the form $(\mathcal{I}, \mathcal{T})$ where $\mathcal{I} \in \mathfrak{I}$ and $\mathcal{T} \in \mathscr{P}_{fin}(\mathfrak{T})$ is a finite subset of $\mathfrak{T}$. A* consequence property *(or* c-property *for short) is a set $\mathcal{P} \subseteq \mathfrak{I} \times \mathscr{P}_{fin}(\mathfrak{T})$ such that $(\mathcal{I}, \mathcal{T}) \in \mathcal{P}$ implies $(\mathcal{I}, \mathcal{T}') \in \mathcal{P}$ for every $\mathcal{T}' \supseteq \mathcal{T}$.*

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

Definition 2 *Given an axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ and a c-property $\mathcal{P}$, a set of axioms $\mathcal{S} \subseteq \mathcal{T}$ is called a* minimal axiom set (MinA) *for Γ w.r.t. $\mathcal{P}$ if $(\mathcal{I}, \mathcal{S}) \in \mathcal{P}$ and $(\mathcal{I}, \mathcal{S}') \notin \mathcal{P}$ for every $\mathcal{S}' \subset \mathcal{S}$. The set of all MinA for Γ w.r.t. $\mathcal{P}$ will be denoted as $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$.*

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

Instead of directly trying to compute $\mathsf{MIN}_{\mathcal{P}(\Gamma)}$, one can also try to compute a pinpointing formula. In order to define this formula, we assume that every axiom $t \in \mathcal{T}$ is labeled with a unique propositional variable $\mathsf{lab}(t)$. Let $\mathsf{lab}(\mathcal{T})$ be the set of all propositional variables labeling axioms in $\mathcal{T}$. A *monotone Boolean formula* over $\mathsf{lab}(\mathcal{T})$ is a Boolean formula using variables in $\mathsf{lab}(\mathcal{T})$

and only the connectives conjunction and disjunction. As usual, we identify a propositional *valuation* with the set of propositional variables it makes true. For a valuation $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{T})$, let $\mathcal{T}_\mathcal{V} = \{t \in \mathcal{T} \mid \mathsf{lab}(t) \in \mathcal{V}\}$.

Definition 3 (pinpointing formula) *Given a c-property $\mathcal{P}$ and an axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, a monotone Boolean formula ϕ over $\mathsf{lab}(\mathcal{T})$ is called a* pinpointing formula *for $\mathcal{P}$ and Γ if for every valuation $\mathcal{V} \subseteq \mathsf{lab}(\mathcal{T})$ it holds that $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ iff $\mathcal{V}$ satisfies ϕ.*

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

Lemma 4 *Let $\mathcal{P}$ be a c-property, $\Gamma = (\mathcal{I}, \mathcal{T})$ an axiomatized input, and ϕ a pinpointing formula for $\mathcal{P}$ and Γ. Then*

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

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

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

3 General Tableaux

General tableaux and their pinpointing extension were first introduced in [4]. For the rest of this paper we use $\mathcal{V}$ and $\mathcal{D}$ to denote countably infinite sets of *variables* and *constants*, respectively. A *signature* Σ is a set of predicate symbols, where each predicate $P \in \Sigma$ is equipped with an arity. A Σ-*assertion* is of the form $P(a_1, \ldots, a_n)$ where $P \in \Sigma$ is an n-ary predicate and $a_1, \ldots, a_n \in \mathcal{D}$. Analogously, a Σ-*pattern* is of the form $P(x_1, \ldots, x_n)$ where $P \in \Sigma$ is an n-ary predicate and $x_1, \ldots, x_n \in \mathcal{V}$. If the signature is clear from the context, we will often just say pattern (assertion). For a set of assertions A (patterns B), $\mathsf{cons}(A)$ ($\mathsf{var}(B)$) denotes the set of constants (variables) occurring in A (B).

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

Definition 5 (Tableau) *Let $\mathfrak{I}$ and $\mathfrak{T}$ be sets of inputs and axioms, respectively. A* tableau *for $\mathfrak{I}$ and $\mathfrak{T}$ is a tuple $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ where:*

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

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

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

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

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

Definition 6 (rule application, saturated, clash) *Given a S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \dots, B_m\}$, and a substitution ρ on $\mathsf{var}(B_0)$, R is* applicable to $\mathfrak{S}$ with ρ *if (i) $\mathcal{S} \subseteq \mathcal{T}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $i, 1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ we have that $B_i\rho' \not\subseteq A$.*

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

If $\mathcal{M}'$ is obtained from $\mathcal{M}$ by the application of R, then we write $\mathcal{M} \to_{\mathsf{R}} \mathcal{M}'$, or simply $\mathcal{M} \to_S \mathcal{M}'$ if it is not relevant which of the rules of the tableau

S was applied. The reflexive-transitive closure *of $\to_S$ is denoted by $\xrightarrow{*}_S$. A set of S-states $\mathcal{M}$ is called* saturated *if there is no $\mathcal{M}'$ such that $\mathcal{M} \to_S \mathcal{M}'$.*

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

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

Definition 7 (correctness) *Let $\mathcal{P}$ be a c-property on axiomatized inputs over $\mathfrak{I}$ and $\mathfrak{T}$, and S a tableau for $\mathfrak{I}$ and $\mathfrak{T}$. We say that S is* correct for $\mathcal{P}$ *if the following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathfrak{T}$:*

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

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

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

The first condition turns out to be more problematic. It requires that every chain of rule applications leads to a saturated state after a finite number of steps, regardless of the input given to the tableau. In general, it is undecidable whether a given tableau satisfies this requirement. To prove this, we will show that it is possible to simulate the execution of a Turing machine (TM) by means of tableau rule applications. A *Turing machine* is a quadruple $M = (Q, \Omega, \delta, q_0)$ where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, Ω is a finite set of *symbols* containing the *blank symbol* $\sqcup$, and $\delta : Q \times \Omega \to (Q \cup \{h, \text{“}yes\text{”}, \text{“}no\text{”}\}) \times \Omega \times \{\leftarrow, \to\}$ is the *transition relation*. For a given TM M, we will construct a tableau whose states simulate the configuration of the tape at each execution step of M.

To simulate a TM $M = (Q, \Omega, \delta, q_0)$, we will use a predicate symbol for each symbol in Ω; that is, for every $g \in \Omega$, the signature of our tableau will contain the unary predicate symbol T_g. Intuitively, an assertion of the form $T_g(a)$ expresses that the symbol g appears in the current configuration of the tape. To simulate the position of the symbols, we add the unary predicate symbols F_z for $z \in \mathbb{Z}$ to the signature, where $F_z(a)$ says that a is allocated in tape position z. The extra assertion $H_q(a)$ indicates the internal state of the machine. The constants in this setting work as gluing elements to express which symbols appear in which tape positions; they can also be used to identify the position of the head. The initial tape configuration can easily be expressed with the help of these assertions.

When the TM executes a transition step, it changes the symbol on the tape cell where the head is pointing, changes the internal state of the machine, and moves the head either to the left or to the right, according to the transition relation. To reflect these changes in the states of a tableau, we would have to remove the assertions expressing the position of the head, the state of the cell position and the internal state, and replace them accordingly. Our definition of tableau does not allow for assertions to be removed by a rule application, and hence those changes cannot be performed directly. We will instead add the unary predicate symbol $\perp$ to our signature. Intuitively, the assertion $\perp(a)$ expresses that no assertion using the constant a should be taken into consideration any more.

A TM can in write over an arbitrary number of tape cells, although the initial word written on the tape is finite. All the rest of the tape cells are assumed to be written with the blank symbol $\sqcup$. Since tableaux do not allow infinite sets of assertions as states, we cannot simply represent the infinite tape as such, but need to be able to add assertions of the form $F_z(a)$ as they are needed, along with the corresponding assertion $T_\sqcup(a)$ stating the symbol in the new tape cell. These cells should only be added if the cell is not yet in use. Since rule applicability cannot check the non-existence of an assertion before triggering the rule, we will need to use non-deterministic rules for this.

Definition 8 (simulating tableau) *Let $M = (Q, \Omega, \delta, q_0)$ be a TM and let the set of inputs $\mathfrak{I} \subseteq \Omega^*$ and the set of axioms $\mathfrak{T} = \emptyset$. The* tableau simulating *M is the tableau for $\mathfrak{I}$ and $\mathfrak{T}$ given by $S_M = (\Sigma, \cdot^S, \mathcal{R}, \emptyset)$ where*

- $\Sigma = \{F_z \mid z \in \mathbb{Z}\} \cup \{T_g \mid g \in \Omega\} \cup \{H_q \mid q \in Q\} \cup \{\perp\}$*, all with arity* 1.
- *for every $w = g_1 \dots g_k \in \mathfrak{I}$ we have*
$$w^S = \{T_{g_i}(a_i), F_i(a_i) \mid 1 \leq i \leq k\} \cup \{H_{q_0}(a_1)\}$$
- *for every pair $(q, g) \in Q \times \Omega$, let $B(q, g) = \{F_k(x), T_g(x), H_q(x)\}$; then if $\delta(q, g) = (q', g', \rightarrow)$, the rules*
$$\begin{aligned}(B(q,g) \cup \{F_{k+1}(y), S_{g''}(y)\}, \emptyset) &\rightarrow \{\{F_k(z), T_{g'}(z), H_{q'}(y), \perp(x)\}\} \\ (\{F_k(x), T_g(x), H_q(x)\}, \emptyset) &\rightarrow \{\{F_{k+1}(z)\}, \{F_{k+1}(z), T_\perp(z)\}\}\end{aligned}$$

are in $\mathcal{R}$, and if $\delta(q,g) = (q', g', \leftarrow)$, the rules

$$\begin{aligned}(B(q,g) \cup \{F_{k-1}(y), S_{g''}(y)\}, \emptyset) &\rightarrow \{\{F_k(z), T_{g'}(z), H_{q'}(y), \bot(x)\}\}\\ (\{F_k(x), T_g(x), H_q(x)\}, \emptyset) &\rightarrow \{\{F_{k-1}(z)\}, \{F_{k-1}(z), T_\bot(z)\}\}\end{aligned}$$

are in $\mathcal{R}$, for all $k \in \mathbb{Z}$.

Proposition 9 *Let M be a TM. The tableau S_M terminates on input $(w, \emptyset)$ iff M halts on input w.*

Since the halting problem for Turing machines is undecidable [7], then the problem of knowing whether a tableau terminates on every input or not must also be undecidable.

A correct tableau S can be extended to an algorithm that computes a pinpointing formula. Recall that for the definition of this formula we assume that every axiom $t \in \mathcal{T}$ is labeled with a unique propositional variable, $\mathsf{lab}(t)$ and the set of all propositional variables labeling an axiom in $\mathcal{T}$ is denoted by $\mathsf{lab}(\mathcal{T})$. In the following, we further 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 $\mathsf{lab}(\mathcal{T})$.

For an axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, the modified algorithm works also on sets of S-states, but now every assertion a occurring in the assertion component of a S-state is equipped with a label $\mathsf{lab}(a)$ consisting of a monotone Boolean formula over $\mathsf{lab}(\mathcal{T})$. We call such S-states *labeled S-states*. In the inintial set of S-state $\mathcal{M} = (\mathcal{I}, \mathcal{T})^S$, every assertion is labeled with $\top$.

We must also modify the definition of rule application, so that it takes the labels of the assertions into account. Let A be a set of labeled assertions and ψ a monotone Boolean formula. The assertion a is *ψ-insertable into A* if either (i) $a \notin A$, 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 10 (pinpointing rule application) *Let S be a tableau. Given a labeled S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$, and a substitution ρ on $\mathsf{var}(B_0)$, this rule is* pinpointing applicable to $\mathfrak{S}$ with ρ *if (i) $\mathcal{S} \subseteq \mathcal{T}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $1 \leq i \leq m$ 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 $\mathcal{S} \in \mathcal{M}$ 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 $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \Cup_\psi B_i\sigma, \mathcal{T}) \mid 1 \leq i \leq m\}$, where ψ is defined as above and σ is a substitution on the variables occurring in* R *that extends ρ and maps the fresh variables of* R *to distinct new constants.*

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

Consider a chain of pinpointing rule applications $\mathcal{M}_0 \rightarrow_{S^{\mathsf{pin}}} \ldots \rightarrow_{S^{\mathsf{pin}}} \mathcal{M}_n$ 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 a 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 11 (clash formula) *Let $\mathfrak{S} = (A, \mathcal{T})$ be a labeled S-state and $A' \subseteq A$. 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*

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

Theorem 12 (correctness of pinpointing) *Let $\mathcal{P}$ be a c-property on axiomatized inputs over $\mathfrak{I}$ and $\mathfrak{T}$, and S a correct tableau for $\mathcal{P}$. The following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathfrak{T}$:*

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

As with the original tableau methods, the correctness of a pinpointing extension assumes that this pinpointing algorithm terminates. Perhaps a little more surprising than Proposition 9 is the fact that, even if we restrict our attention to terminating tableaux, it is undecidable whether the pinpointing extension terminates. We can show this by a reduction similar to the one made for tableaux, but in this case constructing, for a TM M, a *terminating* tableau whose pinpointing extension simulates M.

Notice that none of the rules of the tableau S_M in Definition 8 is applicable if there is no assertion of the form $H_q(x)$ describing the internal state of the machine. We can then create a tableau that starts with a state describing the whole input, but leaving aside the internal state of the machine, which we know that must be q_0 at the beginning of the execution of the TM. If this tableau never adds the assertion $H_{q_0}(a_1)$ to the states, then the rules for simulating the execution of the TM are never triggered. Thus, we want to construct a tableau such that, when executed in the normal way, it always terminates but whose pinpointing extension adds the assertion $H_{q_0}(a_1)$ if the correct axioms are used, starting this way the simulation of the TM. This tableau would then be terminating, but the termination of its pinpointing extension will depend on whether the TM it is simulating halts or not on every input.

We construct the tableau S_M^{pin} by allowing the set of axioms to be $\mathfrak{T} = \{\mathsf{ax}_1, \mathsf{ax}_2\}$ and modifying S_M from Definition 8 as follows. Add to the signature the unary predicate names P, P', Q_1, Q_2, and add to R the rules

$$(\{P(x)\}, \{\mathsf{ax}_1\}) \rightarrow \{\{P'(x), Q_1(x)\}\} \tag{1}$$

$$(\{P(x)\}, \{\mathsf{ax}_2\}) \rightarrow \{\{P'(x), Q_2(x)\}\} \tag{2}$$

$$(\{P'(x)\}, \emptyset) \rightarrow \{\{H_{q_0}(x)\}, \{P_1(x)\}, \{P_2(x)\}\}. \tag{3}$$

We also modify the definition of $\cdot^S$ to repalce $H_{q_0(a_1)}$ by $P(a_1)$ in w^S.

This tableau is terminating. At the initial state, none of the rules from S_M can be triggered since $H_{q_0}(a_1)$ is not present. The only way to add this assertion is to apply Rule (3) above, which in turn can only be applied once an assertion of the form $P'(x)$ is present; that is, only after applying either Rule (1) or Rule (2). But the application of any of these rules immediately dissallows application of Rule (3). Hence, after at most two rule applications, S_M^{pin} reaches a saturated state.

If instead of executing this tableau we apply its pinpointing extension with both axioms ax_1 and ax_2 as input, after applying the first two rules, Rule (3) is pinpointing applicable. After its pinpointing application, we have three S_M^{pin}-states. Two of them are already saturated, but not the third one, which now contains the assertion $H_{q_0}(a_1)$, the only missing piece for starting the simulation of M over the given input. This shows the following undecidability result.

Proposition 13 *Let M be a TM. The pinpointing extension of S_M^{pin} terminates on input $(w, \{\mathsf{ax}_1, \mathsf{ax}_2\})$ iff M halts on input w.*

As shown by Propositions 9 and 13, one of the main issues of this approach to pinpointing corresponds to dealing with termination. We can turn now our attention to see how this issues are dealt with in practice. One are where tableau-based decision procedures are widely used is in description logics. One common approach for ensuring the termination of the decision procedure consists on building a (non-terminating) tableau, and then extending it with appropriate an cycle-checking technique, also called *blocking*, that stops the execution of the algorithm. This method constructs partial tree models that can be completed into infinite tree models, or *raveled* into finite models.

A relaxed notion of rule application, called *modified rule application* was introduced in [4, 5] as an aid to prove the correctness of the pinpointing extension. For this, the applicability condition (iii) from Definition 6 is removed. The same notion will be helpful in the following section, and is therefore here recalled.

Definition 14 (modified rule application) *Given a S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ and a substitution ρ on $\mathsf{var}(B_0)$, R is 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 $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{T}) \mid 1 \leq i \leq m\}$, where σ is a substitution on the variables occurring in R that extends ρ and maps the fresh variables of R to distinct new constants.*

We will proceed now to show how this approach can be adapted to the pinpointing framework. In the next section, we will define a sub-class of tableau for which a blocking condition can be used to ensure termination, and also termination of their pinpointing extension.

4 Tableaux with Blocking

One of the reasons for termination of the tableau algorithms for certain DLs is that they create a tree structure with an out-degree bounded by a function of the size of the input formula. The nodes of these trees have labels that must be selected from a finite set. After a depth bounded also by the size of the input formula, one can reuse the information presented by the finite tree in order to obtain an infinite tree, simply by concatenating finite portions in the correct way. An example of this method is the tableau-based decision procedure for satisfiability of $\mathcal{ALC}$-concepts w.r.t. GCIs [1]. The algorithm can be seen as generating a set of assertions of the form $r(a, b)$ where r is a role and $C(a)$ where C is an $\mathcal{ALC}$-concept. The tree structure is induced by role assertions, and the nodes are labeled by sets of concepts; that is, a 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. When a node has a predecessor whose label is a superset of its own label, then that node stops being expanded, since the predecessor can be used to complete the tree. We say then that the node is *blocked*.

Things in general can be a little more complicated. For example, the algorithm that decides consistency of $\mathcal{ALC}$-ABoxes (see [8, 6]) does not construct a tree, but rather several trees growing out from the input ABox. Each of these trees can be treated as in the previous case, ensuring termination.

We want to formalize the notion of the tree structures built by the tableau algorithm, as well as the notion of blocking, within the general framework of tableaux. In order to be as general as possible, we do not want to restrict assertions to be built from unary and binary predicates only. For this reason, we allow predicates to have arbitrary arity, but restrict our assertions such that the states induce graph-like structures.

We must be able to distinguish between nodes and edges in the graph. Thus, 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 $r \in \Delta$ has 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 will form the nodes of the graph-like structure, while the elements of Δ will 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 $p \in \widehat{\Delta}$ ($p \in \widehat{\Lambda}$). 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.

Definition 15 (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 joining the constants that appear in them. Nodes will be formed by maximal sets of assertions from $\widehat{\Delta}$. An assertion from $\widehat{\Lambda}$ will be treated as an edge that connects a node containing its parent constants with a node containing the descendant ones.

Definition 16 (B-graph) *Let B be a set of assertions. A maximal connected subset $N \subseteq B \cap \widehat{\Lambda}$ is called a* node *in B. An assertion $r \in B \cap \widehat{\Delta}$ is called an* edge *in B if there are two nodes N_1 and N_2 in B such that $\overleftarrow{r} \subseteq \mathsf{cons}(N_1)$ and $\mathsf{cons}(N_2) \subseteq \overrightarrow{r}$. In this case, we say that r* connects *N_1 to N_2. The set B is a* graph structure *if every $r \in B \cap \widehat{\Delta}$ is an edge. If B is a graph structure, the corresponding B-*graph *$\mathcal{G}_B$ contains one vertex v_N for every node N, and an edge (v_N, v_M) if there is an edge connecting N to M. The notion of a graph structure and of the corresponding graph can be extended to states $\mathfrak{S} = (B, \mathcal{T})$ in the obvious way: $\mathfrak{S}$ is a* graph structure *if B is one, and in this case $\mathcal{G}_{\mathfrak{S}} := \mathcal{G}_B$.*

Recall that the tableau-based decision procedure for consistency of $\mathcal{ALC}$-ABoxes starts with an ABox, that is, a graph, and extends it by trees that grow out of the nodes of this graph. We introduce now *forest tableaux*, which are meant to show a similar behaviour, based in the more general notion of graph structure introduced above.

Definition 17 (forest tableau) *The tableau $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ is called a* forest tableau *if for every axiomatized input Γ and every $\mathfrak{S} \in \Gamma^S$, the state $\mathfrak{S}$ is a graph structure, every clash $C \in \mathcal{C}$ is connected, and the following conditions hold for every rule $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ and every $1 \leq i \leq m$:*

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

A rule where there is a $1 \leq i \leq m$ *such that* $B_i \cap \widehat{\Delta} \neq \emptyset$ *is called* generating.

We will briefly state the intuitions of these conditions. Condition 1 ensures that every edge, either old or newly introduced by a rule application, is connected to a parent node. In particular, this implies that rule application cannot add new parents for a node, and new nodes are connected to the rest of the graph structure. Condition 2 states that every new edge has only new constants as descendants; i.e., new edges never connect old nodes, but only generate new nodes. Condition 3 ensures that, even if several edges are added by a single rule application, these edges connect different nodes with the parent node, and thus every node has only one edge connecting it to its parent. Condition 4 states that whenever a non-edge assertion is added, it must belong to either an old node, or to the descendant node added by a new edge. Condition 5 states that addition of new edges must depend only on the assertions appearing in the parent nodes, and never in the presence of other edges. Finally, Condition 6 endures that the non-edge assertions triggering a rule application belong to the same node.

Given a forest tableau S and an S-state $\mathfrak{S}$, we can define a partial ordering $<_{\mathfrak{S}}$ on the nodes of $\mathfrak{S}$. Given an axiomatized input Γ, the family of strict partial orderings $<_S = \{<_{\mathfrak{S}} \mid \Gamma^S \xrightarrow{*}_S \mathcal{M}, \mathfrak{S} \in \mathcal{M}\}$ is called a *compatible order for S and* Γ if for all S-states $\mathfrak{S}$ and $\mathfrak{S}'$ it holds that $\mathfrak{S} \subseteq \mathfrak{S}'$ implies that $<_{\mathfrak{S}} \subseteq <_{\mathfrak{S}'}$ and $<_{\mathfrak{S}}$ contains the parent relation on $\mathfrak{S} \setminus \Gamma^S$.

In order to ensure termination of a tableau, we will implement a cycle-checking technique called blocking that stops the application of rules. For this technique to be applicable, we must ensure that nodes cannot grow indefinitely; that is, that the number of assertions that can appear in a single node is bounded.

Definition 18 (cover) *Let $S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$ be a tableau and $\mathcal{T}$ a set of axioms. A set $\Omega \subseteq \Sigma$ is called a $\mathcal{T}$-*cover *if, for every rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_n\}$ such that $\mathcal{S} \subseteq \mathcal{T}$ and B_0 contains only predicates from Ω, the sets B_i for $i = 1, ..., n$ also contain only predicates from Ω. The tableau S is* covered *if, for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, there is a finite $\mathcal{T}$-cover Ω_Γ such that every S-state in Γ^S contains only predicates from Ω_Γ.*

Given such a covered tableau, every state that can be reached from an initial state in Γ^S by applying rules from S contains only predicates from Ω_Γ. We will show that this ensures that nodes cannot grow indefinitely. We proceed now to define what is a blocking condition and how it affects the execution of a tableau. For this we need the notion of substate. For two sets of assertions A and A' we denote by $A \preceq A'$ the existance of a renaming function $f : \mathsf{cons}(A) \to \mathsf{cons}(A')$ such that if the assertion $P(a_1, \dots, a_k)$ is in A, then $P(f(a_1), \dots, f(a_k))$ is in A'. A state $\mathfrak{S} = (A, \mathcal{T})$ is a *substate* of $\mathfrak{S}' = (A', \mathcal{T}')$, denoted as $\mathfrak{S} \preceq \mathfrak{S}'$, if it holds that $\mathcal{T} \subseteq \mathcal{T}'$ and $A \preceq A'$.

Definition 19 (blocking) *Given a forest tableau S, an axiomatized input Γ, and $<_S$ a compatible order for S and Γ, let $\mathfrak{S}$ be a S-state forming a graph structure A binary relation $\lhd$ between nodes is a* blocking relation w.r.t. $<_S$ *if for every $N_1 \lhd N_2$ it holds that $N_1 \preceq N_2$, $N_2 <_{\mathfrak{S}} N_1$, and* $\mathsf{cons}(N_1) \cap \mathsf{cons}(\Gamma^S) = \emptyset$*. A node N is* blocked *if either there is a node N' such that $N \lhd N'$, or the predecessor node of N is blocked.*

A non-generating rule is $\lhd$-applicable *if it is applicable; a generating rule is* $\lhd$-applicable *if it is applicable with substitution ρ and the node containing all the constants in the range of ρ is not blocked.*

Let $\mathcal{M}$ and $\mathcal{M}'$ be finite sets of S-states. If $\mathcal{M}'$ is obtained from $\mathcal{M}$ by the $\lhd$-application of R*, then we write $\mathcal{M} \to_{\mathsf{R}}^{\lhd} \mathcal{M}'$ or simply $\mathcal{M} \to_S^{\lhd} \mathcal{M}'$ if it is not relevant which of the rules of the tableau S was applied. A S-state is* $\lhd$-saturated *if no rule is $\lhd$-applicable to it. A set of S-states $\mathcal{M}$ is* $\lhd$-saturated *if every $\mathfrak{S} \in \mathcal{M}$ is $\lhd$-saturated.*

A forest tableau is $\lhd$-correct *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 $\Gamma^S = \mathcal{M}_0 \to_S^{\lhd} \mathcal{M}_1 \to_S^{\lhd} \dots$;*
2. *for every chain of rule applications $\Gamma^S = \mathcal{M}_0 \to_S^{\lhd} \dots \to_S^{\lhd} \mathcal{M}_n$ such that $\mathcal{M}_n$ is $\lhd$-saturated we have that $\Gamma \in \mathcal{P}$ iff $\mathcal{M}_n$ is full of clashes.*

According to this definition, for a node to be blocked either it or any of its predecessors should satisfy the blocking relation. We will show now that in fact, for $\lhd$-saturated states, being blocked entails that there is another node in the tree containing the same assertions, modulo constant renaming.

The intuition behind blocking is that we can reuse a node that was previously generated instead of generating a new one, if the old node contains the "same" assertions that the new one will have. This way, we can avoid repeating unnecessary work. Notice that there are several blocking conditions, depending on the ordering $<_S$ and binary relation $\lhd$ chosen. We will use now the weakest notion of blocking possible, where $<_S$ is the family containing exclusively the parent-relation on the nodes in every S-state, and $\lhd$ requires no additional conditions than those stated in Definition 19. This will be called *subset blocking* and denoted as $\subseteq$-blocking. We will call a forest tableau using $\subseteq$-blocking a $\subseteq$*-blocking tableau.* It is possible to show that, under subset blocking, a forest tableau always terminates; nonetheless, we will prove a stronger result: that under subset blocking, the pinpointing extension of a forest tableau terminates in every case.

Since we have changed conditions under which a rule is applicable in a forest tableau with blocking to dissallow rule applications in nodes that are already blocked, we need to change the notion of pinpointing rule application accordingly. When executing the pinpointing algorithm, the blocking condition must also consider the labels on the states before stoping the execution. This is done in a straightforward manner. For two sets of labeled assertions A and A' we denote by $A \preceq_{\mathsf{pin}} A'$ the existance of a renaming function $f : \mathsf{cons}(A) \to \mathsf{cons}(A')$ such that if the assertion $P(a_1, \ldots, a_k)$ is in A with label ϕ, then $P(f(a_1), \ldots, f(a_k))$ is in A' with label ψ such that $\phi \models \psi$. A state $\mathfrak{S} = (A, \mathcal{T})$ is a *pinpointing substate* of $\mathfrak{S}' = (A', \mathcal{T}')$, denoted as $\mathfrak{S} \preceq_{\mathsf{pin}} \mathfrak{S}'$, if it holds that $\mathcal{T} \subseteq \mathcal{T}'$ and $A \preceq_{\mathsf{pin}} A'$.

Definition 20 (pinpointing blocking) *Let S be a forest tableau, Γ an axiomatized input and Γ and $\mathfrak{S}$ a S-state for Γ. Given two nodes N, N', we will use the notation $N \lhd_{\mathsf{pin}}^{\subseteq} N'$ if it holds that $N' < N$, $N \preceq_{\mathsf{pin}} N'$ and $\mathsf{cons}(N) \cap \mathsf{cons}(Ga^S) = \emptyset$, where $<$ denotes the predecessor order in $\mathfrak{S} \setminus \Gamma^S$. A node N is* pinpointing $\subseteq$-blocked *if either there is a node N' such that $N \lhd_{\mathsf{pin}}^{\subseteq} N'$, or the predecessor node of N is pinpointing $\subseteq$-blocked.*

A non-generating rule is pinpointing $\lhd$-applicable *if it is pinpointing applicable; a generating rule is* pinpointing $\lhd$-applicable *if it is applicable with substitution ρ and the node containing all the constants in the range of ρ is not blocked.*

Let $\mathcal{M}$ and $\mathcal{M}'$ be finite sets of S-states. If $\mathcal{M}'$ is obtained from $\mathcal{M}$ by the pinpointing $\lhd$-application of R, then we write $\mathcal{M} \to_{\mathsf{R}_{\mathsf{pin}}}^{\lhd} \mathcal{M}'$ or simply $\mathcal{M} \to_{S_{\mathsf{pin}}}^{\lhd} \mathcal{M}'$ if it is not relevant which of the rules of the tableau S was applied. A S-state is pinpointing $\lhd$-saturated *if no rule is pinpointing $\lhd$-applicable to it. A set of S-states $\mathcal{M}$ is* pinpointing $\lhd$-saturated *if every $\mathfrak{S} \in \mathcal{M}$ is pinpointing $\lhd$-saturated.*

We will begin by proving that the pinpointing extenstion of a $\subseteq$-blocking tableau terminates on every input. As a first step towards this goal, we will show that if a tree in the forest generated by rule applications reaches a depth big enough, then we will be able to find a blocked node in every branch. This follows easily from the next lemma. The lemma is in fact shown for a more general case, using the modified rule applications defined in the previous section.

Lemma 21 *Let S be a $\subseteq$-blocking tableau and $\mathfrak{S}_0 \to_{S^m} \mathfrak{S}_1 \to_{S^m} \cdots$ a sequence of modified rule applications. Then, for every $\mathfrak{S}_i = (A_i, \mathcal{T})$ and*

$P \in A_i \cap \widehat{\Lambda}$, *either* $\mathsf{cons}(P) \subseteq \mathsf{cons}(A_0)$ *or there are* $r \in A_i \cap \widehat{\Delta}$ *and* $Q \in A_i \cap \widehat{\Lambda}$ *such that* $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$ *and* $\mathsf{cons}(P) \subseteq \overrightarrow{r}$*.*

Proof. The proof is by induction on i. For $\mathfrak{S}_0$ the result is trivial. Suppose now that it holds for $\mathfrak{S}_i$ and that the rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$ 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 $j, 1 \leq j \leq m$. 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, $\mathcal{P}$ was added by the application of R. By Condition 4 of Definition 17, we have that either (i) there is $r \in (B_0 \cup B_j)\sigma \cap \widehat{\Delta}$ with $\mathsf{cons}(P) \subseteq \overrightarrow{r}$, or (ii) there is a $Q \in B_0\sigma \cap \widehat{\Lambda}$ with $\mathsf{cons}(P) \subseteq \mathsf{cons}(Q)$.

We analyze Case (ii) first. Since R was applied with substitution σ, we have that $B_0\sigma \subseteq A_i$, and thus $Q \in A_i \cap \widehat{\Lambda}$. By the induction hypothesis, either $\mathsf{cons}(Q) \subseteq \mathsf{cons}(A_0)$ or $\overleftarrow{r} \subseteq \mathsf{cons}(Q'), cons(Q) \subseteq \overrightarrow{r}$ for $r, Q' \in A_i$. In both cases the transitivity of $\subseteq$ yields the desired result.

We will analyze now Case (i). We have here that $\mathsf{cons}(P) \subseteq \overrightarrow{r}$ and $r \in (B_0 \cup B_j)\sigma$. By Condition 1 of Definition 17 there must exist a $Q \in B_0\sigma \subseteq A_i$ such that $\overleftarrow{r} \subseteq \mathsf{cons}(Q)$, which finishes the proof. ■

A couple of remarks come in place before proving termination of the pinpointing extension of $\subseteq$-blocking tableaux. Notice first that Condition 6 of Definition 17 ensures that all the assertions in $\widehat{\Lambda}$ triggering a rule applications must all belong to the same node. Second, by Lemma 21, for every new node N (i.e., a node that was not present in the initial state) and assertion $P \in N$ there is an edge r such that $\mathsf{cons}(P) \subseteq \overrightarrow{r}$. Since distinct edges have disjoint sets of descendants (Condition 3 of Definition 17, any other assertion $Q \in N$ also satisfies $\mathsf{cons}(Q) \subseteq \overrightarrow{r}$. Thus, all the constants occurring in a node all belong to the descendant set of the edge by which the node was created.

Theorem 22 *Let S be a $\subseteq$-blocking tableau, then its pinpointing extension terminates on every input.*

Proof. Suppose that there is an input $\Gamma = (\mathcal{I}, \mathcal{T})$ for which ther is an infinite sequence of pinpointing rule applications $\mathfrak{S}_0 \rightarrow_{S^{\mathsf{pin}}} \mathfrak{S}_1 \rightarrow_{S^{\mathsf{pin}}} \cdots$, 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. 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 my 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 4 and 1 of Definition 17 ensure that every newly added node N is created as a successor for an existing node with a unique edge $r \in \widehat{\Delta}$ joining 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 number of constants appearing in a new node is bounded by the largest cardinality of a predicate name $r \in \Delta$; hence, there can only be finitely many different labelled nodes, up to constant renaming. Then, for every chain of nodes $N_0, N_1, \ldots, N_m$ which is sufficiently long (i.e., where m is larger than the number of distinct labelled nodes possible) there must exist $1 \leq n < n' \leq m$ such that $N_{n'} \preceq_{\mathsf{pin}} N_n$, and thus $N_{n'}$ is pinpointing $\subseteq$-blocked by N_n. Thus, the second case above is not possible.

Consider now the first case; that is, that there is a node N for which infinitely many successors are created. As stated before, 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. Moreover, 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 longs as their labels are not equivalent. But there are only finitely many labels up to propositional equivalence. Thus, N can in fact obtain only a finite number of successors. Hence, the pinpointing extension of the tableau S must always terminate. ■

Notice that if a rule is applicable in the normal tableau sense, then it is also pinpointing applicable. Thus, termination of the pinpointing extension implies termination of the original tableau.

Corollary 23 *A $\subseteq$-blocking forest tableau terminates on every input.*

We have now shown that the problems relating termination do not apply to forest tableaux with subset blocking. Unfortunately, since blocking changes the applicability conditions for rules, and we had to adapt the pinpointing extension accordingly, the previous proofs of correctness of these extensions are not valid in this framework. Hence, we need to prove the correctness of blocking pinpointing extensions; that is, that they still compute a pinpointing formula for the property. We will show this with the help of the raveled versions of S-states. Suppose that $\mathcal{M}$ is a set of states such that $\Gamma^S \rightarrow^{\lhd}_S \mathcal{M}$. Then $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$ is a graph-structure that consists on a set of tree-like structures growing out from a graph structure appearing in Γ^S, we call this a *forest-like structure.* For every blocked node N_1, we remove all its descendants. This way, we obtain another forest-like structure, where blocked nodes only appear as leafs of the trees. For every pair of nodes nodes N_1 and N_2 in $\mathfrak{S}$ such that $N_1 \lhd N_2$ we know that $N_1 \preceq N_2$ and hence there is a renaming function $f : \mathsf{cons}(N_1) \rightarrow \mathsf{cons}(N_2)$. We modify the (only) assertion $r(\overleftarrow{r}, \overrightarrow{r}) \in \widehat{\Delta} \cap A$ with $\overrightarrow{r} = \mathsf{cons}(N_1)$ to $r(\overleftarrow{r}, f(\overrightarrow{r}))$ and then remove N_1. The graph-structure obtained this way is called the *raveled version* of $\mathfrak{S}$, and denoted as $\mathfrak{S}^{\circlearrowleft}$. If $\mathcal{M}$ is a set of S-states, then its *raveled version* is $\mathcal{M}^{\circlearrowleft} = \{\mathfrak{S}^{\circlearrowleft} \mid \mathfrak{S} \in \mathcal{M}\}$.

As the raveling process removes all the blocked nodes, one could think that this may lead to a loss of some information contained in the original S-state. The next lemma shows that this is not the case, since all the assertions appearing in a blocked node also appear in a node of smaller depth.

Proposition 24 *Let $\mathfrak{S}$ be $\lhd$-saturated and N a node in $\mathfrak{S}$. If N is blocked, then there is another node N' of $\mathfrak{S}$ such that $N \preceq N'$ and the depth of N' is smaller than the depth of N.*

Proof. The proof goes by induction on the structure of $\mathfrak{S}$. If N is blocked, then either there is a N' such that $N \lhd N'$ or the parent node of N is blocked. In the first case the result holds trivially. We will focus now on the second case, assuming that the result holds for the parent node M of N; thus, there is a node M' such that $M \preceq M'$. Since we are dealing with a blocking tableau, N could only be created by a rule application using the node M; as $M \preceq M'$, the same rule could be applied to M'; thus, there is a descendant N' of M' that is joined to M' by an edge equivalent (modulo renaming) to the edge joining M to N. Analogously, any rule application that could be applied to M or N to add assertions in N would also be applicable in M' or N', respectively, to add equivalent assertions in N'. Thus, $N \preceq N'$. ∎

A trivial consequence of this proposition is that raveling has no influence in the presence of clashes in $\lhd$-saturated states. Since clashes are always connected sets of patterns, any valuation that maps them to a state has to actually map them to a node; as we have seen, if we have a node in a $\lhd$-saturated state $\mathfrak{S}$, then an equivalent node appears in $\mathfrak{S}^{\circlearrowleft}$.

Corollary 25 *Let $\mathfrak{S}$ be $\lhd$-saturated. $\mathfrak{S}$ is clash-free iff $\mathfrak{S}^{\circlearrowleft}$ is clash-free.*

We can relate $\lhd$-saturatedness to saturatedness in the general sense by means of the raveled versions of states. This way, we can reuse some of the results shown for the general tableau framework that will be helpful for showing the correctness of the pinpointing extensions of blocking tableau.

Lemma 26 *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. Then $B_0\rho$ is connected, and hence belongs to a node in $\mathfrak{S}^{\circlearrowleft}$, and since raveling never modifies any nodes in the graph structure, except from removing some, $B_0\rho$ must also be a node in $\mathfrak{S}$. In particular, $B_0\rho \subseteq A$. Since $\mathfrak{S}$ in $\lhd$-saturated, R is not $\lhd$-applicable to it. This means that either the node containing $B_0\rho$ is blocked, or there is a substitution σ extending ρ such that $B_i\sigma \subseteq A$ for some $1 \leq i \leq m$. Since raveling removes all blocked nodes and $B_0\rho \subseteq A^{\circlearrowleft}$, 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 $\sigma'(x) = \sigma(x)$; if $\sigma(x)$ belongs to a node N_1 blocked by N_2, then in particular $N_1 \preceq N_2$ and thus there exists a function $f : \mathsf{cons}(N_1) \to \mathsf{cons}(N_2)$; in this case define $\sigma'(x) = f(\sigma(x))$. This violates the condition for R to be applicable to $\mathfrak{S}^{\circlearrowleft}$ with substitution ρ, contradicting this way our initial assumption.

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 that were added by the raveling process; these edges are of the form $p(\overleftarrow{r}, f_r(\overrightarrow{r}))$, where $N \subseteq \overrightarrow{r}$ for a blocked node N, and f_r is the function given by the definition of blocking. Since S is a forest tableau, the sets $\overrightarrow{r}$ are pairwise disjoint. For each constant $B_0\rho \backslash \widehat{\Lambda}$ there is a constant a_r such that $f_r(a_r) = a$. We construct the valuation

ρ' as follows: $\rho'(x) = \rho(x)$ if $x \in \mathsf{var}(B_0 \cap \widehat{\Lambda})$; $\rho'(x) = a_r$ if $x \in \mathsf{var}(B_0 \setminus \widehat{\Lambda})$ and $\rho(x) = a$. In particular, $B_0\rho' \subseteq A$.

Let N be the node of $\mathfrak{S}$ containing $B_0\rho' \cap \widehat{\Lambda}$. By the way raveling was defined, we know that if N' is the node of $\mathfrak{S}^{\circlearrowleft}$ containing $B_0\rho \cap \widehat{\Lambda}$, then $N = N'$. Since R is non-generating, by Conditions 4 and 6 of Definition 17 it must be the case that $\mathsf{var}(B_i) \subseteq \mathsf{var}(B_0 \cap \widehat{\Lambda})$ for all $1 \leq i \leq m$. Since $\mathfrak{S}$ is $\lhd$-saturated, then R is not applicable to $\mathfrak{S}$ with substitution ρ'; this implies that there must exist a $1 \leq i \leq m$ such that $B_i\rho' \subseteq A$. In particular, $B_i\rho'$ is contained in the node N, and since ρ' coincides with ρ on every variable of B_i, $B_i\rho \subseteq A^{\circlearrowleft}$. Thus, R is not applicable to $\mathfrak{S}^{\circlearrowleft}$. ■

As in the general case, the proof of correctness of pinpointing extensions will use projections of S-states. Recall that given a set $\mathcal{T}$ of labeled axioms, a propositional valuation $\mathcal{V}$ induces the subset $\mathcal{T}_{\mathcal{V}} = \{t \in \mathcal{T} \mid \mathsf{lab}(t) \in \mathcal{V}\}$. Analogously, for a set A of labeled assertions, the valuation $\mathcal{V}$ induces the subset $A_{\mathcal{V}} = \{a \in A \mid \mathcal{V}$ satisfies $\mathsf{lab}(a)\}$. The *$\mathcal{V}$-projection* of $\mathfrak{S} = (A, \mathcal{T})$ is $\mathcal{V}(\mathfrak{S}) = (A_{\mathcal{V}}, \mathcal{T}_{\mathcal{V}})$. For a set of S-states $\mathcal{M}$, $\mathcal{V}(\mathcal{M}) = \{\mathcal{V}(\mathfrak{S}) \mid \mathfrak{S} \in \mathcal{M}\}$. The following lemma is a direct consequence of the definition of clash formula (see [4]).

Lemma 27 *Let $\mathcal{M}$ be a finite set of labeled S-states and $\mathcal{V}$ a propositional valuation. Then $\mathcal{V}$ satisfies $\psi_{\mathcal{M}}$ iff $\mathcal{V}(\mathcal{M})$ is full of clashes.*

There is also a close connection between pinpointing $\lhd$-saturatedness of a set of labeled S-states and $\lhd$-saturatedness of its projection.

Lemma 28 *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 a 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. If R is a non-generating rule, then it would also be applicable (in the normal sense) to $\mathcal{V}(\mathfrak{S})$ with substitution ρ, and hence the proof given for the general case [5] applies here too; thus we assume that R is a generating rule. This means 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 $B_i\rho' \nsubseteq A_{\mathcal{V}}$, and the node containing all the constants in the range of ρ 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 $\mathsf{ins}_{\psi}(B_i\rho', A) \neq \emptyset$ where $\psi = \bigwedge_{b \in B_0} \mathsf{lab}(b\rho) \wedge \bigwedge_{s \in \mathcal{S}} \mathsf{lab}(s)$. Note that

$S \subseteq \mathcal{T}_\mathcal{V}$ and $B_0\rho \subseteq A_\mathcal{V}$ imply that $\mathcal{V}$ satisfies ψ. Since $B_i\rho' \nsubseteq A_\mathcal{V}$, 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 valuation ρ. It remains to show that the node containing all the constants in the range of ρ is not pinpointing $\subseteq$-blocked. Call this node N. Since $N_\mathcal{V}$ is not blocked, for every predecessor node M of N it holds that $N_\mathcal{V} \not\preceq M_\mathcal{V}$; thus for every renaming function $f : \mathsf{cons}(N_\mathcal{V}) \to \mathsf{cons}(M_\mathcal{V})$ there is an assertion $a \in N_\mathcal{V}$ such that $f(a) \notin M_\mathcal{V}$. Then, either $f(a) \notin M$ or $f(a) \in M$ but $\psi \not\models \mathsf{lab}(f(a))$. In both cases it holds that $N \not\preceq_{\mathsf{pin}} M$, and hence N is not pinpointing $\subseteq$-blocked. ■

The idea of the projections of states is that they simulate the behaviour of the tableau in case that the input $(\mathcal{I}, \mathcal{T}_\mathcal{V})$ is given. Unfortunately, this does not work so easily, since there are rules that could be pinpointing applied to a S-state but not to its projection. We can overcome this problem by using modified rule applications, analogously to the way they are used in [5]. The following proposition was shown in [5].

Proposition 29 *Let $\mathcal{M}, \mathcal{M}'$ be two sets of S-states. If $\mathcal{M} \to_{S^{\mathsf{pin}}} \mathcal{M}'$, then either $\mathcal{V}(\mathcal{M}) \to_{S^m} \mathcal{V}(\mathcal{M}')$ or $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$.*

Notice that if $\mathcal{M} \to_S^{\lhd} \mathcal{M}'$, then it is also the case that $\mathcal{M} \to_S \mathcal{M}'$; analogously for pinpointing rule application, if $\mathcal{M} \to_{S^{\mathsf{pin}}}^{\lhd} \mathcal{M}'$, then $\mathcal{M} \to_{S^{\mathsf{pin}}} \mathcal{M}'$. This, along with the previous proposition, shows that $\mathcal{M} \to_{S^{\mathsf{pin}}}^{\lhd} \mathcal{M}'$ implies that either $\mathcal{V}(\mathcal{M}) \to_{S^m} \mathcal{V}(\mathcal{M}')$ or $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$. In particular, it also holds that $\mathcal{M}_0 \overset{*}{\to}{}_{S^{\mathsf{pin}}}^{\lhd} \mathcal{M}$ implies $\mathcal{V}(\mathcal{M}_0) \overset{*}{\to}_{S^m} \mathcal{V}(\mathcal{M})$.

One consequence of the lemmas shown so far is that if there are $\Gamma^S = \mathcal{M}_0, \mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M}_0 \overset{*}{\to}_S \mathcal{M}, \mathcal{M}_0 \overset{*}{\to}_S \mathcal{M}'$, and $\mathcal{M}, \mathcal{M}'$ are both $\lhd$-saturated, then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is full of clashes. To see this, recall that for sets of states $\mathcal{N}$ and $\mathcal{N}_0$, if $\mathcal{N}_0$ is saturated and there are $\mathfrak{S} \in \mathcal{N}$ and $\mathfrak{S}_0 \in \mathcal{N}_0$ with $\mathfrak{S} \preceq \mathfrak{S}_0$, then for every $\mathcal{N} \to_{S^m} \mathcal{N}'$ there is a $\mathfrak{S}' \in \mathcal{N}'$ such that $\mathfrak{S}' \preceq \mathfrak{S}_0$ [4]. Since for every $\mathfrak{S} \in \mathcal{M}$ there is a $\mathfrak{S}_0 \in \mathcal{M}_0$ with $\mathfrak{S}_0 \preceq \mathfrak{S}$ and $\mathcal{M}^{\circlearrowleft}$ is saturated (Lemma 26), we know that for every $\mathfrak{S} \in \mathcal{M}$ there is a $\mathfrak{S}' \in \mathcal{M}'$ such that $\mathfrak{S}' \preceq \mathfrak{S}^{\circlearrowleft}$. Thus, if $\mathcal{M}'$ is full of clashes, so is $\mathcal{M}$ (Corollary 25). The same argument can be used analogously for proving the other direction.

With all this, we can now proceed to proving the correctness of the pinpointing extension of a blocking tableau.

Theorem 30 (correctness of pinpointing) *Let $\mathcal{P}$ be a c-property on axiomatized inputs over $\mathfrak{I}$ and $\mathfrak{T}$, and S a correct $\subseteq$-blocking tableau for $\mathcal{P}$. Then the following holds for every axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$ over $\mathfrak{I}$ and $\mathfrak{T}$:*

> *For every chain of rule applications $\mathcal{M}_0 \rightarrow^{\lhd}_{S^{\mathsf{pin}}} \ldots \rightarrow^{\lhd}_{S^{\mathsf{pin}}} \mathcal{M}_n$ 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 $\Gamma^S = \mathcal{M}_0 \overset{*}{\rightarrow}{}^{\lhd}_{S^{\mathsf{pin}}} \mathcal{M}_n$ with $\mathcal{M}$ pinpointing $\lhd$-saturated. To show that $\psi_{\mathcal{M}}$ 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 $\psi_{\mathcal{M}}$.

Let $\mathcal{N}_0 = (\mathcal{I}, \mathcal{T}_{\mathcal{V}})^S$. Since S terminates, there is a $\lhd$-saturated set $\mathcal{N}$ such that $\mathcal{N}_0 \overset{*}{\rightarrow}{}^{\lhd}_{S} \mathcal{N}$. Also, as $\mathcal{M}_0 \overset{*}{\rightarrow}{}^{\lhd}_{S^{\mathsf{pin}}} \mathcal{M}$, we have that $\mathcal{V}(\mathcal{M}_0) \overset{*}{\rightarrow}_{S^m} \mathcal{V}(\mathcal{M})$. Additionally, $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$ and also $\mathcal{V}(\mathcal{M}_0)$ is $\lhd$-saturated; thus, $\mathcal{N}$ is full of clashes iff $\mathcal{V}(\mathcal{M})$ is full of clashes. By the 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 $\mathcal{V}(\mathcal{M})$ is full of clashes iff $\mathcal{V}$ satisfies $\psi_{\mathcal{M}}$ (Lemma 27) ■

We have thus shown how one can correctly adapt pinpointing to the well-known subset blocking condition. Unfortunately, our framework cannot deal with the DL constructors regarding *inverses* of roles, as the forest tableaux defined in [5] do. If we weaken the restrictions of forest tableaux to allow such constructors, then Lemma 26 does not hold anymore unless we use a stronger notion of blocking (equality blocking) as used in the DL literature.

5 Conclusions

In this paper we have shown that the problem of deciding termination of general tableaux is undecidable. Furthermore, even deciding termination of the pinpointing extension of terminating tableaux is an undecidable problem. To overcome this drawback, we introduced the notion of forest tableaux and how they can be used to specify blocking conditions that ensure termination of both the tableaux and its pinpointing extension.

The framework presented here can only deal with the weakest notion of blocking, also known as subset blocking. Future research will try to extend this framework to stronger versions of blocking such as equality blocking, which is widely used in DLs for dealing with inverse roles.

References

[1] Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, and Jörg H. Siekmann. Concept logics. In John W. Lloyd, editor, *Computational Logics, Symposium Proceedings*, pages 177–201. Springer-Verlag, 1990.

[2] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

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

[4] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. In *Proceedings of the 16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods TABLEAUX 2007*, LNAI, Aix-en-Provence, France, 2007. Springer.

[5] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 2008. Submitted.

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

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

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

[9] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[10] Kevin Lee, Thomas Meyer, Jeff Z. Pan, and Richard Booth. Computing maximally satisfiable terminologies for the description logic *lc* with cyclic definitions. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *Description Logics*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[11] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In Allan Ellis and Tatsuya Hagino, editors, *Proc. of the 14th International Conference on World Wide Web (WWW'05)*, pages 633–640. ACM, 2005.

[12] Stefan Schlobach. Diagnosing terminologies. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 670–675. AAAI Press/The MIT Press, 2005.

[13] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.