Axiom Pinpointing in General Tableaux

Franz Baader[1] and Rafael Peñaloza[2⋆]

[1] Theoretical Computer Science, TU Dresden, Germany
baader@inf.tu-dresden.de
[2] Intelligent Systems, University of Leipzig, Germany
penaloza@informatik.uni-leipzig.de

Abstract. Axiom pinpointing has been introduced in description logics (DLs) to help the user to understand the reasons why consequences hold and to remove unwanted consequences by computing minimal (maximal) subsets of the knowledge base that have (do not have) the consequence in question. The pinpointing algorithms described in the DL literature are obtained as extensions of the standard tableau-based reasoning algorithms for computing consequences from DL knowledge bases. Although these extensions are based on similar ideas, they are all introduced for a particular tableau-based algorithm for a particular DL.
The purpose of this paper is to develop a general approach for extending a tableau-based algorithm to a pinpointing algorithm. This approach is based on a general definition of "tableaux algorithms," which captures many of the known tableau-based algorithms employed in DLs, but also other kinds of reasoning procedures.

1 Introduction

Description logics (DLs) [2] are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as 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 [13] as standard ontology language for the semantic web. As a consequence of this standardization, several ontology editors support OWL [15, 18, 14], 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 [12, 10, 24] employ tableau-based algorithms [6], 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

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

[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.

change the ontology in case the consequence is unwanted. For example, in the current version of the medical ontology SNOMED [25], the concept *Amputation-of-Finger* is classified as a subconcept of *Amputation-of-Arm*. Finding the 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 [22] 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 [22] 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 $\mathcal{ALC}$ [23], 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 [20], Schlobach [21] uses the minimal subsets that have a given consequence together with the computation of Hitting Sets to compute *maximal subsets* of a given knowledge base that *do not have* a given (unwanted) *consequence*.[2] Whereas the minimal subsets that have the consequence help the user to comprehend why a certain consequence holds, the maximal subsets that do not have the consequence suggest how to change the knowledge base in a minimal way to get rid of a certain unwanted consequence.

The problem of computing minimal (maximal) subsets of a DL knowledge base that have (do not have) a given consequence was actually considered earlier in the context of extending DLs by default rules. In [4], Baader and Hollunder solve this problem by introducing a labeled extension of the tableau-based consistency algorithm for $\mathcal{ALC}$-ABoxes [11], which is very similar to the one described later in [22]. The main difference is that the algorithm described in [4] does not directly compute minimal subsets that have a consequence, but rather a monotone Boolean formula, called *clash formula* in [4], whose variables correspond to the axioms of the knowledge bases and whose minimal satisfying (maximal unsatisfying) valuations correspond to the minimal (maximal) subsets that have (do not have) a given consequence.

The approach of Schlobach and Cornet [22] was extended by Parsia et al. [19] to more expressive DLs, and the one of Baader and Hollunder [4] was extended by Meyer et al. [17] to the case of $\mathcal{ALC}$-terminologies with general concept inclusions (GCIs), which are no longer unfoldable. The choice of the DL $\mathcal{ALC}$ in [4] and [22] 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 [17] follows the approach introduced in [4], but since GCIs require the introduction of so-called

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

blocking conditions into the tableau-based algorithm to ensure termination, there are some new problems to be solved.

Thus, one can ask to which DLs and tableau-based algorithms the approaches described in [4, 22] 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 the ideas underlying the pinpointing algorithm described in [4]. To this purpose, we define a general notion of "tableaux algorithm," which captures many of the known tableau-based algorithms for DLs and Modal Logics,[3] but also other kinds of decision procedures, like the polynomial-time subsumption algorithm for the DL $\mathcal{EL}$ [1]. This notion is simpler than the tableau systems introduced in [3] in the context of translating tableaux into tree automata, and it is not restricted to tableau-based algorithms that generate tree-like structures.

Axiom pinpointing has also been considered in other research areas, though usually not under this name. For example, in the SAT community, people have considered the problem of computing maximally satisfiable and minimally unsatisfiable subsets of a set of propositional formulae. The approaches for computing these sets developed there include special purpose algorithms that call a SAT solver as a black box [16, 7], but also algorithms that extend a resolution-based SAT solver directly [8, 26]. To the best of our knowledge, extensions of tableau-based algorithms have not been considered in this context, and there are no general schemes for extending resolution-based solvers.

In the next section, we define the notions of minimal (maximal) sets having (not having) a given consequence in a general setting, and show some interesting connections between these two notions. In Section 3 we introduce our general notion of a tableau, and in Section 4 we show how to obtain pinpointing extension of such tableaux. Because of the space restriction, we cannot give complete proofs of our results. They can be found in [5].

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 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*. 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 (c-property) *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 $\mathcal{T}' \supseteq \mathcal{T}$.*

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

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 $\mathcal{T}$. 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 1, assume that $\mathfrak{I}$ is a countably infinite set of propositional variables, and 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 $p_1, \ldots, p_n, q \in \mathfrak{I}$. 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:

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

It is easy to see that $\Gamma \in \mathcal{P}$. Note that Definition 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. *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 the above 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 [22, 16]. 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 $\mathsf{MAX}_{\mathcal{P}(\Gamma)} = \{\{\text{ax}_2, \text{ax}_4\},\ \{\text{ax}_1, \text{ax}_3, \text{ax}_4\},\ \{\text{ax}_1, \text{ax}_2, \text{ax}_3\}\}$. 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 [9]. 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.[4] 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 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 [4].

Lemma 1. *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*

$$\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \{T_\mathcal{V} \mid \mathcal{V} \text{ is a minimal valuation satisfying } \phi\}$$
$$\mathsf{MAX}_{\mathcal{P}(\Gamma)} = \{T_\mathcal{V} \mid \mathcal{V} \text{ is a maximal valuation falsifying } \phi\}$$

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 from 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 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. Conversely, however, 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 pinpointing formula obtained from the set $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \{\{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_4\}, \{\mathrm{ax}_2, \mathrm{ax}_3, \mathrm{ax}_4\}\}$ is $(\mathrm{ax}_1 \wedge \mathrm{ax}_2 \wedge \mathrm{ax}_4) \vee (\mathrm{ax}_2 \wedge \mathrm{ax}_3 \wedge \mathrm{ax}_4)$.

[4] This corresponds to the clash formula introduced in [4]. 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.

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

$$\text{If } \{p_1, \ldots, p_n\} \subseteq A \text{ and } p_1 \wedge \ldots \wedge p_n \rightarrow q \in \mathcal{T} \text{ then } A := A \cup \{q\} \qquad (2)$$

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 $(B_0, \mathcal{S}) \rightarrow B_1$ 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 $A \cup B_1$.[5] 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 until now 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

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

thus works on a finite set $\mathcal{M}$ of sets of assertions. The non-deterministic rule $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ 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: $\to p \vee s$, 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 variables. 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 [6] 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:

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

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 *$\Sigma$-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 4 (Tableau). *Let $\mathfrak{I}$ be a set of inputs and $\mathfrak{T}$ a set of axioms. 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 a function that maps every $\mathcal{I} \in \mathfrak{I}$ to a finite set of finite sets 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 $(\mathcal{I}, \mathcal{T})^S := \{(A, \mathcal{T}) \mid A \in \mathcal{I}^S\}$.

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 5 (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 $\xrightarrow{*}_S$. 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 can now define under what conditions a tableau S is correct for a c-property.

Definition 6 (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 $\mathfrak{T}$. 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 $\mathcal{M}_0 \to_S \mathcal{M}_1 \to_S \mathcal{M}_2 \to_S \dots$ 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 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 also the polynomial-time subsumption algorithm for the DL $\mathcal{EL}$ and its extensions introduced in [1] can be viewed as a correct deterministic tableau with unstructured assertions. The standard tableau-based decision procedure for concept unsatisfiability in the DL $\mathcal{ALC}$ [23] is a correct tableau that uses structured assertions and has a non-deterministic rule.

In 2. of Definition 6, 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 1. *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.*

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 $\neg p$. The label of this

initial assertion 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 the initial set of S-states $\mathcal{M} = (\mathcal{I}, \mathcal{T})^S$, every assertion is labeled with $\top$.

The definition of rule application must 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 7 (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 $\mathfrak{S} \in \mathcal{M}$ 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 (1) 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 8 (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 2. *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.*

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

Lemma 3. *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.*

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

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

To prove this theorem, we 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 $\mathcal{M} \rightarrow_{S^{pin}} \mathcal{M}'$ does not imply $\mathcal{V}(\mathcal{M}) \rightarrow_S \mathcal{V}(\mathcal{M}')$. 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 5 is removed.

Definition 9 (modified rule application). *Given an S-state $\mathfrak{S} = (A, \mathcal{T})$, a rule $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$, and a substitution ρ on $\mathsf{var}(B_0)$, this rule is m-*applicable to $\mathfrak{S}$ with ρ *if (i) $\mathcal{S} \subseteq \mathcal{T}$ and (ii) $B_0\rho \subseteq A$. In this case, we write $\mathcal{M} \rightarrow_{S^m} \mathcal{M}'$ if $\mathfrak{S} \in \mathcal{M}$ and $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{T}) \mid i = 1, \ldots, m\}$, where σ is a substitution on the variables occurring in R that extends ρ and maps the fresh variables of R to distinct new constants.*

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. $\rightarrow_S$, as introduced in Definition 5.

Lemma 4. *Let $\Gamma = (\mathcal{I}, \mathcal{T})$ be an axiomatized input and $\mathcal{M}_0 = \Gamma^S$.*

1. *Assume that $\mathcal{M}_0 \stackrel{*}{\rightarrow}_S \mathcal{M}$ and $\mathcal{M}_0 \stackrel{*}{\rightarrow}_{S^m} \mathcal{M}'$ 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 $\mathcal{M} \rightarrow_{S^{pin}} \mathcal{M}'$ implies $\mathcal{V}(\mathcal{M}) \rightarrow_{S^m} \mathcal{V}(\mathcal{M}')$ or $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$. In particular, this shows that $\mathcal{M}_0 \stackrel{*}{\rightarrow}_{S^{pin}} \mathcal{M}$ implies $\mathcal{V}(\mathcal{M}_0) \stackrel{*}{\rightarrow}_{S^m} \mathcal{V}(\mathcal{M})$.*

We are now ready to prove Theorem 1. Let $\Gamma = (\mathcal{I}, \mathcal{T})$ be an axiomatized input, and assume that $\mathcal{M}_0 \rightarrow_{S^{pin}} \ldots \rightarrow_{S^{pin}} \mathcal{M}_n$ such that $\mathcal{M}_0 = \Gamma^S$ and $\mathcal{M}_n$ is pinpointing saturated. 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 5. *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 $\mathcal{N}_0 \stackrel{*}{\rightarrow}_S \mathcal{N}$. 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, $\mathcal{M}_0 \stackrel{*}{\rightarrow}_{S^{pin}} \mathcal{M}_n$ implies $\mathcal{V}(\mathcal{M}_0) \stackrel{*}{\rightarrow}_{S^m} \mathcal{V}(\mathcal{M}_n)$. In addition, we know that $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$, and Lemma 3 implies that $\mathcal{V}(\mathcal{M}_n)$ is saturated. Thus, 1. of Lemma 4, together with the fact that $\mathcal{N}$ is full of clashes, implies that $\mathcal{V}(\mathcal{M}_n)$ is full of clashes.

By Lemma 2, this implies that $\mathcal{V}$ satisfies $\psi = \psi_{\mathcal{M}_n}$. □

Lemma 6. *If $\mathcal{V}$ satisfies ψ then $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$.*

Proof. Consider again a chain of rule applications $\mathcal{N}_0 = (\mathcal{I}, \mathcal{T}_\mathcal{V})^S \stackrel{*}{\rightarrow}_S \mathcal{N}$ 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 $\mathcal{V}(\mathcal{M}_0) \stackrel{*}{\rightarrow}_{S^m} \mathcal{V}(\mathcal{M}_n)$, $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$, and $\mathcal{V}(\mathcal{M}_n)$ is saturated. Since $\mathcal{V}$ satisfies ψ, Lemma 2 implies that $\mathcal{V}(\mathcal{M}_n)$ is full of clashes.

By 1. of Lemma 4, this implies that $\mathcal{N}$ is full of clashes. □

This completes the proof of Theorem 1. 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 9).

Example 1. Consider the tableau S that has the following three rules

$$\mathsf{R}_1 : (\{P(x)\}, \{\mathrm{ax}_1\}) \to \{\{P'(x), Q_1(x)\}\},$$
$$\mathsf{R}_2 : (\{P(x)\}, \{\mathrm{ax}_2\}) \to \{\{P'(x), Q_2(x)\}\},$$
$$\mathsf{R}_3 : (\{P'(x)\}, \emptyset) \to \{\{r(x,y), P'(y)\}, \{Q_1(x)\}, \{Q_2(x)\}\},$$

and where the function $\cdot^S$ maps every input $\mathcal{I} \in \mathfrak{I}$ to the singleton set $\{\{P(a)\}\}$, and the set of axioms is $\mathfrak{T} = \{\mathrm{ax}_1, \mathrm{ax}_2\}$.

For any axiomatized input $\Gamma = (\mathcal{I}, \mathcal{T})$, we have $\Gamma^S = \{(\{P(a)\}, \mathcal{T})\}$, and thus R_3 is not applicable to Γ^S. Depending on which axioms are contained in $\mathcal{T}$, the rules R_1 and/or R_2 may be applicable. However, their application introduces $Q_1(a)$ or $Q_2(a)$ into the set of assertions, and thus the non-deterministic and potentially non-terminating rule R_3 is not applicable. Consequently, S terminates on every axiomatized input Γ.

It is possible, however, to construct an infinite chain of pinpointing rule applications starting with $\Gamma^S = \{(\{P(a)\}, \{\mathrm{ax}_1, \mathrm{ax}_2\})\}$ where $\mathsf{lab}(P(a)) = \top$. In fact, we can first apply the rule R_1. This adds the assertions $P'(a)$ and $Q_1(a)$, both with label ax_1. An application of the rule R_2 adds the assertion $Q_2(a)$ with label ax_2, and it modifies the label of the assertion $P'(a)$ to $\mathsf{lab}(P'(a)) = \mathrm{ax}_1 \vee \mathrm{ax}_2$. At this point, we have reached an S-state $\mathfrak{S}$ containing the assertions $P(a), P'(a), Q_1(a), Q_2(a)$ with labels $\mathsf{lab}(P(a)) = \top, \mathsf{lab}(P'(a)) = \mathrm{ax}_1 \vee \mathrm{ax}_2, \mathsf{lab}(Q_1(a)) = \mathrm{ax}_1$, and $\mathsf{lab}(Q_2(a)) = \mathrm{ax}_2$. The rule R_3 is pinpointing applicable to this S-state. Indeed, although both $Q_1(a)$ and $Q_2(a)$ are contained in the assertion set of $\mathfrak{S}$, their labels are not implied by $\mathsf{lab}(P'(a))$. The application of R_3 to $\mathfrak{S}$ replaces $\mathfrak{S}$ by three new S-states. One of these new S-states contains the assertion $P'(b)$ for a new constant b. Thus, R_3 is again applicable to this S-state, generating a new S-state with an assertion $P'(c)$ for a new constant c, etc. It is easy to see that this leads to an infinite chain of pinpointing rule applications.

The example shows that, to ensure termination of the pinpointing extension of a tableau, termination of this tableau on every axiomatized input is not sufficient. From the example one also gets the intuition that the reason why the tableau terminates, but its pinpointing extensions does not, is related to the applicability condition for *non*-deterministic rules. In fact, this condition ensures that the rule R_3, which causes non-termination, cannot be applied. The reason is that the assertion $P'(a)$ can only be generated together with $Q_1(a)$ or $Q_2(a)$. Once $Q_i(a)$ for $i \in \{1, 2\}$ is present, the definition of rule appication prevents R_3 from being applied. In the pinpointing case, this is no longer true since the labels must be taken into account, and thus the pure presence of $Q_i(a)$ is not sufficient to prevent the application of R_3.

Unfortunately, non-deterministic rules are not the only culprit that prevent transfer of termination. The following example introduces a terminating tableau with purely deterministic rules whose pinpointing extension is non-terminating.

Example 2. Consider the tableau S that has the following three rules

$$\mathsf{R}_1 : (\{P(x)\}, \{\mathrm{ax}_1\}) \to R,$$
$$\mathsf{R}_2 : (\{P(x)\}, \{\mathrm{ax}_2\}) \to R,$$
$$\mathsf{R}_3 : (\{Q_1(x), Q_2(y)\}, \emptyset) \to \{\{r(x,y,z), Q_1(y), Q_2(z)\}\},$$

with $R = \{Q_1(x), Q_1(y), Q_2(x), Q_2(y), r(x,x,x), r(x,y,x), r(y,x,x), r(y,y,x)\}$, and where the function $\cdot^S$ maps every input $\mathcal{I} \in \mathfrak{I}$ to the singleton set $\{\{P(a)\}\}$, and the set of axioms is $\mathfrak{T} = \{\mathrm{ax}_1, \mathrm{ax}_2\}$. 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

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

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 valuation that maps x to the constant a. Extending this valuation to map y to a as well violates Condition (iii) of the definition of rule application since the assertions $Q_1(a), Q_2(a)$ and $r(a,a,a)$ were already introduced by the first rule application. To satisfy Condition (ii) for rule R_3, we must choose a valuation ρ mapping x to a or b and y to a or b. In any case, the assertions $r(\rho(x), \rho(y), a), Q_1(\rho(x))$ and $Q_2(a)$ belong to $\mathcal{A}$, and thus extending ρ by mapping z to a violates Condition (iii). This shows that S indeed terminates on every axiomatized input.

It is possible to construct an infinite chain of pinpointing rule applications starting with $\Gamma^S = \{(\{P(a)\}, \{\mathrm{ax}_1, \mathrm{ax}_2\})\}$ 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 $P(a)$ are labeled with ax_1. Rule R_2 is pinpointing applicable to $\mathfrak{S}$ since, although there is an extension of the valuation 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 $Q_1(c), Q_2(c), r(a,c,a), r(c,a,a), r(c,c,a)$ with label ax_2, and modifies the label of $Q_1(a), Q_2(a), r(a,a,a)$ to $\mathrm{ax}_1 \vee \mathrm{ax}_2$. We can now apply R_3 to the resulting S-state $\mathfrak{S}'$ with the valuation 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 anymore. This rule application adds the assertions $r(b,c,d), Q_2(d)$ with label $\mathrm{ax}_1 \wedge \mathrm{ax}_2$. It is easy to see that the rule R_3 can now be repeatedly applied, producing an infinite chain of pinpointing rule applications.

5 Conclusion

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. The most important topic for future research is to address the termination issue: under what additional conditions does termination of a tableau transfer to its pinpointing extension?

In addition, our current framework has two restrictions that we will try to overcome in future work. First, our tableau rules always extend the current set of assertions. We do not allow for rules that can modify existing assertions. Thus, tableau-based algorithms that identify constants, like the rule treating at-most number restrictions in description logics (see, e.g., [6]), cannot be modelled. A similar problem occurs for the tableau systems introduced in [3]. There, it was solved by modifying the definition of rule application by allowing rules that introduce new individuals (in our notation: rules with fresh variables) to reuse existing individuals. However, this makes such rules intrinsically non-deterministic. In our setting, we believe that we can solve this problem more elegantly by introducing equality and inequality predicates.

Second, our approach currently assumes that a correct tableau always terminates, without considering additional blocking conditions. As shown in [17], extending a tableau with blocking to a pinpointing algorithm requires some additional effort. Solving this for the case of general tableaux will be a second important direction for future research.

References

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

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

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

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

[5] F. Baader and R. Penaloza. Axiom pinpointing in general tableaux. LTCS-Report 07-01, TU Dresden, Germany, 2007. See http://lat.inf.tu-dresden.de/research/reports.html.

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

[7] 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.

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

[9] 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.
[10] V. Haarslev and R. Möller. RACER system description. In *Proc. of IJCAR 2001*, 2001.
[11] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of German Workshop on AI*, pages 38–47. Springer-Verlag, 1990.
[12] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR'98*, pages 636–647, 1998.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of AAAI 2006*. AAAI Press/The MIT Press, 2006.
[18] D. Oberle, R. Volz, B. Motik, and S. Staab. An extensible ontology software environment. In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 311–333. Springer-Verlag, 2004.
[19] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW'05*, pages 633–640. ACM, 2005.
[20] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
[21] S. Schlobach. Diagnosing terminologies. In *Proc. of AAAI 2005*, pages 670–675. AAAI Press/The MIT Press, 2005.
[22] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI 2003*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.
[23] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
[24] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of DL 2004*, pages 212–213, 2004.
[25] 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.
[26] 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.