Axiom Pinpointing in General Tableaux

Franz Baader   Rafael Peñaloza

LTCS-Report 07-01

Axiom Pinpointing in General Tableaux

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

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

February 14, 2007

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 concep-

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

tual 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 [12] as standard ontology language for the semantic web. As a consequence of this standardization, several ontology editors support OWL [14, 17, 13], 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 [11, 9, 23] employ tableau-based algorithms [5], which can be used to detect inconsistencies and to infer other implicit consequences, such as subsumption relationships between concepts or instance relationships between individuals and concepts.

For a developer or user of a DL-based ontology, it is often quite hard to understand why a certain consequence holds,[1] and even harder to decide how to change the ontology in case the consequence is unwanted. For example, in the current version of the medical ontology SNOMED [24], 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 [21] 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 [21] 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 $\mathcal{ALC}$ [22], 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 [19], Schlobach [20] 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

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

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

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 [10], which is very similar to the one described later in [21]. 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 [21] was extended by Parsia et al. [18] to more expressive DLs, and the one of Baader and Hollunder [4] was extended by Meyer et al. [16] 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 [21] 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 [16] follows the approach introduced in [4], but since GCIs require the introduction of so-called blocking conditions into the tableau-based algorithm to ensure termination, there are some new problems to be solved.

Thus, one can ask to which DLs and tableau-based algorithms the approaches described in [4, 21] 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

[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.

tree-like structures.

Axiom pinpointing has also been considered in other research area, 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 [15, 6], but also algorithms that extend a resolution-based SAT solver directly [7, 25]. 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 tableaux, and in Section 4 we show how to obtain pinpointing extension of such tableaux.

2 Basic definitions

Before we can define our general notion of a tableaux 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 $\mathcal{T}' \supseteq \mathcal{T}$.*

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 nonmonotonic 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:

$$\mathrm{ax}_1\colon\ \ \rightarrow q, \qquad \mathrm{ax}_2\colon\ \ \rightarrow s, \qquad \mathrm{ax}_3\colon\ s \rightarrow q, \qquad \mathrm{ax}_4\colon\ q \wedge s \rightarrow p \tag{1}$$

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 $\Gamma' := ((p, \{\mathrm{ax}_3, \mathrm{ax}_4\}), \{\mathrm{ax}_1, \mathrm{ax}_2\})$, 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 $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \{\{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_4\}, \{\mathrm{ax}_2, \mathrm{ax}_3, \mathrm{ax}_4\}\}$. In the variant of the example where only subsets of the facts $\{\mathrm{ax}_1, \mathrm{ax}_2\}$ can be taken, we have $\mathsf{MIN}_{\mathcal{P}'(\Gamma')} = \{\{\mathrm{ax}_2\}\}$.

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 [21, 15]. 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 $\{\mathrm{ax}_1, \mathrm{ax}_3\}$, $\{\mathrm{ax}_2\}$, $\{\mathrm{ax}_4\}$, and thus $\mathsf{MAX}_{\mathcal{P}(\Gamma)} = \{\{\mathrm{ax}_2, \mathrm{ax}_4\}, \{\mathrm{ax}_1, \mathrm{ax}_3, \mathrm{ax}_4\}, \{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_3\}\}$. Intuitively, to get a set of axioms that does not have the consequence, we must remove from $\mathcal{T}$ at least one axiom for every MinA, and thus the minimal hitting sets give us the minimal sets to be removed.

The reduction we have just sketched shows that it is enough to design an algorithm for computing all MinA, since the MaNA can then be obtained by a hitting set computation. It should be noted, however, that this reduction is not polynomial: there may be exponentially many hitting sets of a given collection of sets, and even deciding whether such a collection has a hitting set of cardinality $\leq n$ is an NP-complete problem [8]. 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*

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

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-know 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 $\mathsf{MIN}_{\mathcal{P}(\Gamma)} = \{\{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_4\}, \{\mathrm{ax}_2, \mathrm{ax}_3, \mathrm{ax}_4\}\}$ 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

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

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, assertion 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

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

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 $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ where $B_0, B_1, \ldots, B_m$ are finite sets of assertions and $\mathcal{S}$ is a finite set of axioms. Instead of working on a single set of assertions, the non-deterministic algorithm thus works on a finite set $\mathcal{M}$ of sets of assertions. The non-deterministic rule $(B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ 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 [5] 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 Σ-*pattern* is of the form $P(x_1, \ldots, x_n)$ where $P \in \Sigma$ is an n-ary predicate and $x_1, \ldots, x_n \in \mathcal{V}$. If the signature is clear from the context, we will often just say pattern (assertion). For a set of assertions A (patterns B), $\mathsf{cons}(A)$ ($\mathsf{var}(B)$) denotes the set of constants (variables) occurring in A (B).

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

Definition 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 B_0.*

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 $\mathcal{M}_0 \to_S \mathcal{M}_1 \to_S \mathcal{M}_2 \to_S \ldots$ starting with $\mathcal{M}_0 := \Gamma^S$.*

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

The simple decision procedure sketched in our Horn clause example is a correct tableau in the sense of this definition. More precisely, it is a tableau with unstructured assertions (i.e., the signature contains only nullary predicate symbols) and deterministic rules. It is easy to see that 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 $\mathcal{ALC}$ [22] 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.*

This proposition is a special case of Lemma 8 which will proven at the end of this paper. A proof for this particular case would be almost identical to the one given for that lemma.

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

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

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

Definition 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.*

Proof. If $\mathcal{V}(\mathcal{M})$ is full of clashes, then for every S-state $\mathfrak{S}_i \in \mathcal{M}$, $\mathcal{V}(\mathfrak{S}_i)$ contains a clash. Thus, there is a clash set A' in $\mathfrak{S}_i$ such that the labels $\mathsf{lab}(a)$ are satisfied by $\mathcal{V}$, for every $a \in A'$. This implies that the $\mathcal{V}$ satisfies the label $\psi_{A'}$ of this clash set. Hence, $\mathcal{V}$ satisfies the formula

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

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

Conversely, if there is a $\mathfrak{S} \in \mathcal{M}$ such that $\mathcal{V}(\mathfrak{S})$ does not contain a clash, it must be the case that for every clash set A' in $\mathfrak{S}$, there is an assertion $a \in A'$ such that $\mathcal{V}$ doesn't satisfy $\mathsf{lab}(a)$. This means that $\mathcal{V}$ does not satisfy the label of any of the clash sets $\psi_{A'}$. Hence, this valuation cannot satisfy the disjunction of such labels, nor the clash formula. ■

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

Lemma 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.*

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 applicable to $\mathcal{V}(\mathfrak{S})$ with a substitution ρ. By definition of applicability, it holds then that $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V}$, $B_0\rho \subseteq A_\mathcal{V}$, and for every $i, 1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ, $B_i\rho' \not\subseteq A_\mathcal{V}$.

Suppose now that R is not pinpointing applicable to $\mathfrak{S}$ with the same substitution ρ. Since $\mathcal{S} \subseteq \mathcal{T}_\mathcal{V} \subseteq \mathcal{T}$ and $B_0\rho \subseteq A_\mathcal{V} \subseteq A$, there must be an i and a substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ such that $\mathsf{ins}_\psi(B_i\rho', A) = \emptyset$, where $\psi := \bigwedge_{b \in B_0} \mathsf{lab}(b\rho) \wedge \bigwedge_{s \in \mathcal{S}} \mathsf{lab}(s)$. Then, for every $b \in B_i$ it holds that $b\rho' \in A$ and $\psi \models \mathsf{lab}(b\rho')$. But we know that $\mathcal{V}$ satisfies ψ; thus, it must also satisfy $\mathsf{lab}(b\rho')$, and thus $b\rho' \in A_{\mathcal{V}}$, contradicting the fact the $B_i\rho' \not\subseteq A_{\mathcal{V}}$. ■

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 $\mathcal{M}_0 \xrightarrow{*}_S \mathcal{M}$ and $\mathcal{M}_0 \xrightarrow{*}_{S^m} \mathcal{M}'$ 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 $\mathcal{M}_0 \xrightarrow{*}_{S^{pin}} \mathcal{M}$ implies $\mathcal{V}(\mathcal{M}_0) \xrightarrow{*}_{S^m} \mathcal{V}(\mathcal{M})$.*

Proof. Part 1 of this lemma is a corollary to Lemma 8, which will be proven later in this section.

For the second part of the lemma, let $\mathfrak{S} = (A, \mathcal{T}) \in \mathcal{M}$ and $\mathsf{R} : (B_0, \mathcal{S}) \rightarrow \{B_1, \ldots, B_m\}$ be such that R is pinpointing applicable to $\mathfrak{S}$ with substitution ρ. Then $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \Cup_\psi B_i\sigma, \mathcal{T}) \mid 1 \leq i \leq m\}$ where σ and ψ are as in the definition of pinpointing application.

Take one of the newly added S-states $\mathfrak{S}_i = (A \Cup_\psi B_i\sigma, \mathcal{T}) \in \mathcal{M}'$. By the definition of ψ-insertion, we know that every assertion $a \in A \setminus \mathsf{ins}_\psi(B_i\sigma, A)$ keeps its old label $\mathsf{lab}(a)$, each assertion in $\mathsf{ins}_\psi(B_i\sigma, A) \setminus A$ gets a label ψ and each assertion $b \in A \cap \mathsf{ins}_\psi(B_i\sigma, A)$ gets the new label $\psi \vee \mathsf{lab}(b)$. Thus, if $\mathcal{V}$ satisfies ψ, it holds that $(A \Cup_\psi B_i\sigma)_\mathcal{V} = A_\mathcal{V} \cup B_i\sigma$, because all the newly added and modified labels will be satisfied by $\mathcal{V}$. This implies that $\mathcal{V}(\mathcal{M}) \rightarrow_{S^m} \mathcal{V}(\mathcal{M}')$.

On the other hand, if $\mathcal{V}$ does not satisfy ψ, then $(A \Cup_\psi B_i\sigma)_\mathcal{V} = A_\mathcal{V}$, since none of the newly added labels will be satisfied by $\mathcal{V}$, and modifying a previously existing one by disjoining it with ψ does not change its satisfiability under $\mathcal{V}$. In this case, $\mathcal{V}(\mathcal{M}) = \mathcal{V}(\mathcal{M}')$. ■

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 $\mathcal{N}_0 \xrightarrow{*}_S \mathcal{N}$. Since S is correct for $\mathcal{P}$ and $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$, we know that $\mathcal{N}$ is full of clashes.

By 2. of Lemma 4, $\mathcal{M}_0 \xrightarrow{*}_{S^{pin}} \mathcal{M}_n$ implies $\mathcal{V}(\mathcal{M}_0) \xrightarrow{*}_{S^m} \mathcal{V}(\mathcal{M}_n)$. In addition, we know that $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$, and Lemma 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 $\mathcal{N}_0 = (\mathcal{I}, \mathcal{T}_\mathcal{V})^S \xrightarrow{*}_S \mathcal{N}$ where $\mathcal{N}$ is saturated. We have $(\mathcal{I}, \mathcal{T}_\mathcal{V}) \in \mathcal{P}$ if we can show that $\mathcal{N}$ is full of clashes.

As in the proof of the previous lemma, we have that $\mathcal{V}(\mathcal{M}_0) \xrightarrow{*}_{S^m} \mathcal{V}(\mathcal{M}_n)$, $\mathcal{V}(\mathcal{M}_0) = \mathcal{N}_0$, and $\mathcal{V}(\mathcal{M}_n)$ is saturated. Since $\mathcal{V}$ satisfies 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. ∎

To complete the proof of Theorem 1, it remains to show the first part of Lemma 4. Before proving this result, we will introduce the concept of *substate*. Intuitively, a S-state $\mathfrak{S}$ is a substate of a S-state $\mathfrak{S}'$ if every assertion and axiom in $\mathfrak{S}$ is also part of $\mathfrak{S}'$. In our case, we want to make this notion more general, allowing for different constants to appear in both S-states, as long as there is a "renaming" from the constants of $\mathfrak{S}$ into the ones of $\mathfrak{S}'$.

Definition 10 (substate) *A S-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 there exists 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, then $P(f(a_1), \ldots, f(a_k))$ is in A'.*

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

Lemma 7 *Let $\mathcal{N}$ and $\mathcal{N}_0$ be two sets of S-states, where $\mathcal{N}_0$ is saturated. If there exist $\mathfrak{S} \in \mathcal{N}$ and $\mathfrak{S}_0 \in \mathcal{N}_0$ such that $\mathfrak{S} \preceq \mathfrak{S}_0$, then for every $\mathcal{N} \to_{\mathsf{R}^m} \mathcal{N}'$ there is a $\mathfrak{S}' \in \mathcal{N}'$ such that $\mathfrak{S}' \preceq \mathfrak{S}_0$.*

Proof. If $\mathcal{N}'$ is obtained by the application of R to a S-state different from $\mathfrak{S}$, then $\mathfrak{S} \in \mathcal{N}'$. Suppose now that the rule $\mathsf{R} : (B_0, \mathcal{S}) \to \{B_1, \ldots, B_m\}$ is applied to $\mathfrak{S}$ with substitution ρ to obtain $\mathcal{N}'$, and let $\mathfrak{S} = (A, \mathcal{T})$ and $\mathfrak{S}_0 = (A_0, \mathcal{T}_0)$. Since $\mathfrak{S} \preceq \mathfrak{S}_0$, it holds that $\mathcal{T} \subseteq \mathcal{T}_0$ and there is a substitution ρ' on $\mathsf{var}(B_0)$ such that $B_0\rho' \subseteq A_0$. Thus, conditions (i) and (ii) from the definition of R being applicable to $\mathfrak{S}_0$ with ρ' (see Definition 5) are satisfied. As $\mathcal{N}$ is saturated, this definition cannot be satisfied; hence, condition (iii) must not hold. This means that there must exist a $1 \leq i \leq m$ and a substitution ς on $\mathsf{var}(B_0 \cup B_i)$ extending ρ' such that $B_0\varsigma \subseteq A_0$. On the other hand, a substitution σ is used to construct the new set $\mathcal{N}'$ after the application to the rule to $\mathfrak{S}$ with ρ in $\mathcal{N}$. Let $\mathfrak{S}' = (A \cup B_i\sigma, \mathcal{T})$. Extend the renaming function f to f' including the new constants $f' : \mathsf{cons}(A \cup B_i\sigma) \to \mathsf{cons}(A_0)$ by setting for every fresh variable x of R appearing in B_i, $f'(\sigma(x)) = \varsigma(x)$. This defines a complete renaming function for the constants in $A \cup B_i\sigma$, and by definition is such that $\sigma \circ f' = \varsigma$. Suppose that an assertion $P(a_1, \ldots, a_k)$ is in $A \cup B_i\sigma$. If it is in A, then, since $\mathfrak{S} \preceq \mathfrak{S}_0$, the assertion $P(f(a_1), \ldots, f(a_k))$ must be in A_0. In the other case, it must be in $B_i\sigma$, in other words, $P(a_1, \ldots, a_k) = P(\sigma(x_1), \ldots, \sigma(x_k))$, for some $x_1, \ldots, x_k \in \mathsf{var}(B_0 \cup B_i)$. As $\sigma \circ f' = \varsigma$, then the assertion $P(f'(a_1), \ldots, f'(a_k)) = P(\varsigma(a_1), \ldots, \varsigma(a_k))$ is in A_0. Thus, $\mathfrak{S}' \preceq \mathfrak{S}_0$. ■

The following lemma proves Proposition 1 and the first part of Lemma 4.

Lemma 8 *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 $\mathcal{M}_0 \overset{*}{\to}_{S^m} \mathcal{M}$ and $\mathcal{M}_0 \overset{*}{\to}_{S^m} \mathcal{M}'$, then $\mathcal{M}$ is full of clashes iff $\mathcal{M}'$ is full of clashes.*

Proof. Recall that the application of a rule to a set of S-states removes one of these S-states, and adds a finite amount of S-states that extend the removed one. Then, for every S-state $\mathfrak{S} \in \mathcal{M}'$, there is a S-state $\mathfrak{S}_0 \in \mathcal{M}_0$ such that $\mathfrak{S}_0 \preceq \mathfrak{S}$. Let $\mathcal{M}_0 \to_{S^m} \mathcal{M}_1 \to_{S^m} \ldots \to_{S^m} \mathcal{M}_n = \mathcal{M}$ be the chain of rule applications from $\mathcal{M}_0$ to $\mathcal{M}$. Then, by Lemma 7, for every $\mathfrak{S} \in \mathcal{M}'$, there is a S-state $\mathfrak{S}_1 \in \mathcal{M}_1$ such that $\mathfrak{S}_1 \preceq \mathfrak{S}$. This argument gives us an inductive step from which it follows that for every $\mathfrak{S} \in \mathcal{M}'$ and every $0 \leq i \leq n$ there is a S-state $\mathfrak{S}_i \in \mathcal{M}_i$ such that $\mathfrak{S}_i \preceq \mathfrak{S}$. If $\mathcal{M}$ is full of clashes, then every element in $\mathcal{M}$ contains a clash. In particular, for every $\mathfrak{S} \in \mathcal{M}'$, there is an element $\mathfrak{S}_n \in \mathcal{M}$ such that $\mathfrak{S}_n \preceq \mathfrak{S}$; thus, $\mathfrak{S}$ contains also a clash. Hence, if $\mathcal{M}$ is full of clashes, then $\mathcal{M}'$ is full of clashes. The same argument can be used to prove the converse direction. ■

This completes the proof of Theorem 1. The theorem considers a terminating chain of pinpointing rule applications. Unfortunately, termination of

$$\begin{aligned}
\mathsf{R}_1 &: (\{P_1(x)\}, \{\mathrm{ax}_1\}) \to \{\{P(x), Q_1(x)\}\} \\
\mathsf{R}_2 &: (\{P_2(x)\}, \{\mathrm{ax}_2\}) \to \{\{P(x), Q_2(x)\}\} \\
\mathsf{R}_3 &: (\{P(x)\}, \emptyset) \to \{\{r(x,y), P(y)\}, \{Q_1(x)\}, \{Q_2(x)\}\}
\end{aligned}$$

Figure 1: Rules of a terminating tableau

a tableau S does not imply termination of its pinpointing extension. This is the case since a rule may be pinpointing applicable in cases where it is not applicable in the normal sense (see the discussion above Definition 9). Intuitively, one could think that whenever this happens, the rule generates assertions with a label that does not imply the labels of the existing assertions that blocked the normal application of the rule. Termination of pinpointing rule application should follow from the fact that this can happen only finitely often since $\mathsf{lab}(\mathcal{T})$ is finite, and thus there are only finitely many monotone Boolean formulae over $\mathsf{lab}(\mathcal{T})$. We will now show with a counterexample that this intuition is not correct.

Consider a tableau S where the function $\cdot^S$ maps every input $\mathcal{I} \in \mathfrak{I}$ to the singleton set $\mathcal{I}^S = \{\{P_1(a), P_2(a)\}\}$ and $\mathcal{R}$ contains only the three rules shown in Figure 1. Suppose that the tableau algorithm is executed over an input of the form $\Gamma = (\mathcal{I}, \{\mathrm{ax}_1, \mathrm{ax}_2\})$. This execution begins with the singleton set of S-states $\Gamma^S = \{(\{P_1(a), P_2(a)\}, \{\mathrm{ax}_1, \mathrm{ax}_2\})\}$. At this point, any of the rules R_1 or R_2 is applicable, but not R_3 since there is no assertion of the form $P(d)$ with d a constant. For simplicity, on the following we will represent every S-state by showing only the set of assertions it contains; the set of axioms is always the same, namely $\{\mathrm{ax}_1, \mathrm{ax}_2\}$.

If rule R_1 is applied, then the rule R_2 is still applicable in the S-state obtained from such application, but R_3 is still not applicable because, although the assertion $P(a)$ can be found there, it also contains the assertion $Q_1(a)$, violating this way the applicability conditions. Further applying the rule R_2 to the resulting S-state leads to a saturated S-state. Analogously, if one starts by applying R_2, then the only further applicable rule is R_1, leading to a saturated S-state. Figure 2 shows the tree of possible rule applications for all sets of S-states reachable from Γ^S. In the figure, only a series of assertions are shown, since every set of S-states that is reached is a singleton. The left branch shows an application of rule R_1 followed by R_2 while the right branch shows the inverse ordering.

This shows that, regardless of the ordering chosen, the execution of the tableau will reach a saturated set of states after finitely many rule appli-

$$\begin{tikzcd}[column sep=small]
 & P_1(a), P_2(a) \arrow[dl, no head] \arrow[dr, no head] & \\
P(a), Q_1(a) \arrow[d, no head] & & P(a), Q_2(a) \arrow[d, no head] \\
P(a), Q_2(a) & & P(a), Q_1(a)
\end{tikzcd}$$

Figure 2: The complete choice tree for rule application of the tableau using as axioms ax_1 and ax_2. The tree shows only the assertions that are added to the parent S-state after an application of the rule.

cations. Notice that the rule R_3 is never applicable. The reason for this is that at the beginning, Condition *(ii)* of the definition of applicability is not fulfilled, and whenever a rule application adds the assertion required for satisfying it, it adds also more assertions that contradict Condition *(iii)* of the same definition.

We can now try to apply the modified algorithm for pinpointing over this input. For this, we will begin with the set of labeled S-states given by $\{\{P_1(a), P_2(a)\}\}$, where the label of both assertions is set to $\top$. An application of the rule R_1 on the only S-state substitutes it with another one having two additional assertions $P(a), Q_1(a)$, labeled with ax_1. A pinpointing application of rule R_2 to the resulting S-state adds the assertion: $Q_2(a)$ with label ax_2, and modifies $\mathsf{lab}(P(a))$ to the formula $\mathrm{ax}_1 \vee \mathrm{ax}_2$.

Summarizing, after applying both rules, we have one single S-state defined by the set of assertions $\mathcal{A} = \{P_1(a), P_2(a), P(a), Q_1(a), Q_2(a)\}$ where $\mathsf{lab}(P_1(a)) = \mathsf{lab}(P_2(a)) = \top, \mathsf{lab}(P(a)) = \mathrm{ax}_1 \vee \mathrm{ax}_2, \mathsf{lab}(Q_1(a)) = \mathrm{ax}_1$, and $\mathsf{lab}(Q_2(a)) = \mathrm{ax}_2$.

We then have an assertion $P(a)$ in this labeled S-state, and although the assertions $Q_1(a)$ and $Q_2(a)$ are also there, their labels are such that $\mathsf{lab}(P(a)) \not\models \mathsf{lab}(Q_1(a))$ and $\mathsf{lab}(P(a)) \not\models \mathsf{lab}(Q_2(a))$. Thus, rule R_3 is pinpointing applicable to this S-state. Applying it leads to the set of labeled S-states defined by:

$$\begin{array}{lll}\{ & \mathcal{A} \cup \{r(a,b), P(b)\} & \\ & \mathcal{A} \cup \{Q_1(a)\} & \\ & \mathcal{A} \cup \{Q_2(a)\} & \}.\end{array}$$

In fact, the last two sets define the same state, which is the same defined by $\mathcal{A}$. For the first of those S-states, the rule R_3 is again applicable, substituting it by three more S-states, one of which contains an extra assertion of the form $P(c)$ for which the same rule is again applicable. The same rule can be applied once and again, and hence the modified algorithm never reaches a pinpointing saturated set of S-states.

This example shows that the termination of a tableau S does not necessarily imply the termination of its pinpointing example. The reason in this case is that a rule that can never be applied with the "normal" applicability conditions produces a non-terminating cycle of addition of new constants to the S-states. Despite our efforts, we have not been able to fully characterize the tableaux for which this is not the case, nor any other conditions ensuring the transfer of termination from a tableau to its pinpointing extension.

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. From this formula, minimal axiom sets and maximal non-axiom sets can be derived.

The most important topic for future research is to solve the termination issue: fully characterize the conditions under which tableau termination is transfered to its pinpointing extension. In addition, our current framework has two restrictions that we will try to overcome in future work. First, our tableaux 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., [5]), cannot be modelled. A similar problem occurs for the tableau systems introduced in [3]. There, it was solved by modifying the applicability condition of rules 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 tableaux always terminates, without considering additional blocking conditions. As shown in [16], 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] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh (UK), 2005. Morgan Kaufmann, Los Altos.

[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, Jan Hladik, Carsten Lutz, and Frank Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57(2–4):247–279, 2003.

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

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

[6] James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In Manuel V. Hermenegildo and Daniel Cabeza, editors, *Proc. of the 7th Int. Symposium on Practical Aspects of Declarative Languages (PADL'05)*, volume 3350 of *Lecture Notes in Computer Science*, pages 174–186. Springer-Verlag, 2005.

[7] Gennady Davydov, Inna Davydova, and Hans 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.

[8] 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.

[9] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[10] Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of the German Workshop on Artificial Intelligence*, pages 38–47. Springer-Verlag, 1990.

[11] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[12] 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.

[13] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. Swoop: A Web ontology editing browser. *J. of Web Semantics*, 4(2), 2005.

[14] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In *Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan, 2004.

[15] Mark H. Liffiton and Karem A. Sakallah. On finding all minimally unsatisfiable subformulas. In Fahiem Bacchus and Toby Walsh, editors, *Proc. of the 8th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'05)*, volume 3569 of *Lecture Notes in Computer Science*, pages 173–186. Springer-Verlag, 2005.

[16] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*. AAAI Press/The MIT Press, 2006.

[17] Daniel Oberle, Raphael Volz, Boris Motik, and Steffen Staab. An extensible ontology software environment. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 311–333. Springer-Verlag, 2004.

[18] 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.

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

[20] 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.

[21] 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.

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

[23] Evren Sirin and Bijan Parsia. Pellet: An OWL DL reasoner. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 212–213, 2004.

[24] 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.

[25] Lintao Zhang and Sharad Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Proc. of the Conference on Design, Automation and Test in Europe (DATE'03)*, pages 10880–10885. IEEE Computer Society Press, 2003.