Adding Context to Tableaux for DLs

Weili Fu and Rafael Peñaloza

Theoretical Computer Science,
TU Dresden, Germany
weili.fu77@googlemail.com, penaloza@tcs.inf.tu-dresden.de

Abstract. We consider the problem of reasoning with ontologies where every axiom is associated to a context, and contexts are related through a total order. These contexts could represent, for example, a degree of trust associated to the axiom, or a level of granularity for the knowledge provided. We describe an extension of tableaux-based decision procedures into methods that compute the best-fitting context for the consequences of an ontology, and apply it to the tableaux algorithm for $\mathcal{ALC}$. We also describe an execution strategy that preserves most of the standard optimizations used in modern DL reasoners.

1 Introduction

Several non-standard reasoning tasks for DL ontologies can be described as repeated standard reasoning, over some of its sub-ontologies. This is the case, for example, in axiom-pinpointing [14,12,3], where the tasks is to identify the class of sub-ontologies entailing a consequence, or access control [2], where users are assigned views to specific subontologies, and one wants to decide which users can or cannot deduce some implicit consequence.

One can think of the sub-ontologies over which standard reasoning is performed as contextual views to the whole ontology. The task is then to partition the contexts into those that entail and those that do not entail the desired consequence. An obvious way to solve this task is to test each of the potentially exponentially many sub-ontologies for the consequence, and classify them according to the result. However, it is possible to exploit the structure of the set of contexts to obtain a more efficient method.

Notice that every set of sub-ontologies defines a partial ordering $\leq$ via the superset relationship: $\mathcal{O} \leq \mathcal{O}'$ iff $\mathcal{O}' \subseteq \mathcal{O}$. This partial order can always be extended to a lattice in which every context is join-prime [6]. It has been shown that there is an element ν in this lattice, called the *boundary*, such that the following holds for every context $\mathcal{O}$ represented by a label $\ell_{\mathcal{O}}$: $\mathcal{O}$ entails the consequence iff $\ell_{\mathcal{O}} \leq \nu$ [2]. If the lattice is distributive, then the boundary can be computed in polynomial time on the size of the input ontology [3].

So far, all the methods implemented for computing boundaries are based on a black-box approach, in which an unmodified reasoner is called repeatedly until the boundary has been found. In fact, attempts to produce a glass-box approach for e.g. axiom-pinpointing [4], where the reasoner is modified to directly compute the boundary, encounter the problem that most of the optimizations used in implemented DL reasoners do not work in the modified procedures, making the glass-box approaches much less efficient than black-box ones. Moreover, glass-box extensions are not even guaranteed to terminate.

In this paper we focus on a special case in which the contexts are linearly ordered. That is, for every pair of sub-ontologies $\mathcal{O}, \mathcal{O}'$, either $\mathcal{O} \subseteq \mathcal{O}'$ or $\mathcal{O}' \subseteq \mathcal{O}$. This simple setting still covers a wide variety of applications, such as possibilistic reasoning [13,11], trust management, or granularity reasoning. We develop a glass-box approach for extending tableau-based decision algorithms into procedures that compute the boundary for the consequence decided by the original tableaux. A prioritization ordering of the rule applications of these procedures ensures that the boundary-computation procedure behaves almost identically to the original tableau. This entails not only termination of the algorithm, but also that most of the optimization techniques used by modern DL reasoners are still applicable to the extensions. An additional benefit of our method is that it can deal with arbitrary blocking conditions, without losing its correctness.

The paper is divided as follows. We first introduce the basic reasoning problems we are interested in, followed by a general notion of tableaux. In Section 4 we describe the labeled extensions of tableaux, that output a boundary for the desired property, and provide the prioritization ordering that ensures that these extensions behave well. We wrap up by arguing that our approach can be seen as a special case of incremental reasoning, and hence should be easy to implement in modern DL reasoners.

2 Basic Definitions

We start by introducing the general notion of consequence properties that are decided by general tableaux. To avoid unnecessary confusion, we present slightly simplified versions of the notions of consequence properties and tableaux, which suffice for the goals of this paper. We consider also a general notion of axioms, which can e.g. be assertions, GCIs, or concept definitions.

Definition 1. *Let $\mathfrak{O}$ be a set of* axioms*, and let $\mathscr{P}_{fin}(\mathfrak{O})$ denote the set of all finite subsets of $\mathfrak{O}$. A* consequence property (c-property) *is a set $\mathcal{P} \subseteq \mathscr{P}_{fin}(\mathfrak{O})$ such that $\mathcal{O} \in \mathcal{P}$ implies $\mathcal{O}' \in \mathcal{P}$ for every $\mathcal{O} \subseteq \mathcal{O}'$.*

Intuitively, c-properties model consequence relations in logic that are monotonic; i.e. they describe which sets of axioms have or entail a desired consequence. For example, one can consider the set $\mathfrak{O}$ of all concept- and role-assertions in the DL $\mathcal{ALC}$ and $\mathcal{P}_{\mathsf{exa}}$ the set of all *in*consistent $\mathcal{ALC}$ ABoxes. If $\mathcal{O}_{\mathsf{exa}}$ is the ABox having the assertions

$$\mathrm{ax}_1 : \neg\exists r.B(a) \qquad \mathrm{ax}_2 : (\neg A \sqcap B)(b) \qquad \mathrm{ax}_3 : \forall r.A(a) \qquad \mathrm{ax}_4 : r(a,b) \quad (1)$$

then $\mathcal{O}_{\mathsf{exa}} \in \mathcal{P}_{\mathsf{exa}}$; that is, $\mathcal{O}_{\mathsf{exa}}$ is inconsistent. In general, we will call the sets $\mathcal{O} \in \mathscr{P}_{fin}(\mathfrak{O})$ *ontologies.*

For several applications, axioms cannot be considered to have all equal importance, but are provided with a label that represents the context to which they belong. We consider only labels that come from a finite totally ordered set $(L, \leq)$. For the rest of this paper, we will denote the elements of L as $\ell_0, \ell_1, \ldots, \ell_k$, with the implicit ordering $\ell_i \leq \ell_j$ iff $i \leq j$. In particular, ℓ_0 is the least- and ℓ_k is the greatest-element of L.

We will use the elements of the set L to define different *contexts* of an ontology. Depending on the specific application at hand, these contexts can have different meanings, such as access rights, level of expertise, trustworthiness, necessity degree, etc.

Every axiom t in an ontology $\mathcal{O}$ is assigned a label $\mathsf{lab}(t) \in L$, which expresses the contexts from which this axiom can be accessed. An ontology extended with such a labeling function lab will be called a *labelled ontology.* Each element $\ell \in L$ defines the context sub-ontology

$$\mathcal{O}_{\geq \ell} := \{t \in \mathcal{O} \mid \mathsf{lab}(t) \geq \ell\}.$$

Clearly, if $\ell \leq \ell'$, then $\mathcal{O}_{\geq \ell'} \subseteq \mathcal{O}_{\geq \ell}$. Conversely, every ontology $\mathcal{O}$ defines an element $\lambda_{\mathcal{O}} \in L$ (called the *label* of $\mathcal{O}$), given by $\lambda_{\mathcal{O}} := \min\{\mathsf{lab}(t) \mid t \in \mathcal{O}\}$. It follows from this definition that $\lambda_{\mathcal{O}} \leq \lambda_{\mathcal{O}'}$ whenever $\mathcal{O}' \subseteq \mathcal{O}$.

For a given labeled ontology, we want to compute the so-called *boundary* of the c-property $\mathcal{P}$. Intuitively, the boundary divides the contexts where $\mathcal{P}$ follows from those where it does not follow.

Definition 2. *Let $\mathcal{O}$ be a labeled ontology and $\mathcal{P}$ a c-property. An element $\nu \in L$ is called a $(\mathcal{O}, \mathcal{P})$-boundary if for every element $\ell \in L$ the following holds:*

$$\ell \leq \nu \quad \textit{iff} \quad \mathcal{O}_{\geq \ell} \in \mathcal{P}.$$

When it is clear from the context, we will usually omit the prefix $(\mathcal{O}, \mathcal{P})$, and call this ν a boundary.

Continuing with our example, let the label of every axiom appearing in (1) be $\mathsf{lab}(\mathrm{ax}_i) = \ell_i, 1 \leq i \leq 4$. The $(\mathcal{O}_{\mathsf{exa}}, \mathcal{P}_{\mathsf{exa}})$-boundary is then ℓ_2 since the sub-ontology $\mathcal{O}_{\geq \ell_2} = \{\neg A \sqcap B(b), \forall r.A(a), r(a,b)\}$ is inconsistent, while the ontology $\mathcal{O}_{\geq \ell_3} = \{\forall r.A(a), r(a,b)\}$ is consistent, i.e. does not belong to $\mathcal{P}_{\mathsf{exa}}$.

It was shown in [2] that if $\mathcal{O} \in \mathcal{P}$, then the $(\mathcal{O}, \mathcal{P})$-boundary always exists, is unique, and can be computed using axiom-pinpointing techniques. Given an ontology $\mathcal{O}$ that belongs to a c-property $\mathcal{P}$, a minimal (w.r.t. set inclusion) sub-ontology $\mathcal{O}' \subseteq \mathcal{O}$ that still belongs to $\mathcal{P}$ is called a *MinA* for $\mathcal{O}, \mathcal{P}$.[1] If $\mathcal{O}_1, \ldots, \mathcal{O}_m$ are all the MinAs for $\mathcal{O}, \mathcal{P}$, then

$$\nu = \max\{\lambda_{\mathcal{O}_i} \mid 1 \leq i \leq m\} \tag{2}$$

is the $(\mathcal{O}, \mathcal{P})$-boundary. However, using this method for computing the boundary may result in an unnecessary overhead. In our running example, we can see that $\{\mathrm{ax}_1, \mathrm{ax}_2, \mathrm{ax}_4\}$ and $\{\mathrm{ax}_2, \mathrm{ax}_3, \mathrm{ax}_4\}$ are the MinAs for $\mathcal{O}_{\mathsf{exa}}, \mathcal{P}_{\mathsf{exa}}$. The second

[1] These minimal sub-ontologies are also called *justifications* [9].

MinA has label ℓ_2. If we happen to compute this second MinA first, then there is no need to continue computing MinAs, as it is clear that any new MinA $\mathcal{O}'$ must contain the axiom ax_1, and hence have label $\ell_1 < \ell_2$. This new MinA will have no influence in the computation of the maximum from Equation (2). This problem is in fact more pronounced than what the example shows, since a single ontology can have exponentially many MinAs [5], and the boundary can be computed by a black-box algorithm that calls a decision procedure for $\mathcal{P}$ at most $k = |L|$ times; namely, once for each $\ell \in L$, to decide whether or not $\mathcal{O}_{\geq \ell} \in \mathcal{P}$.

3 General Tableaux

In this section we recall some of the notions of general tableaux [4]. We use $\mathcal{V}$ and $\mathcal{D}$ to denote two disjoint, countably infinite sets of *variables* and *constants*, respectively. A *signature* Σ is a set of predicate symbols, where each predicate $P \in \Sigma$ is equipped with an arity. A *Σ-assertion* is of the form $P(a_1, \ldots, a_n)$ where $P \in \Sigma$ is an n-ary predicate and $a_1, \ldots, a_n \in \mathcal{D}$. Similarly, a *$\Sigma$-pattern* is of the form $P(x_1, \ldots, x_n)$ where $P \in \Sigma$ and $x_1, \ldots, x_n \in \mathcal{V}$. Whenever the signature is clear from the context, we simply say pattern (assertion). Given a set of assertions A (resp. patterns B), $\mathsf{cons}(A)$ (resp. $\mathsf{var}(B)$) denotes the set of constants (resp. variables) occurring in A (resp. B).

A *substitution* is a mapping $\sigma : V \to \mathcal{D}$, where $V \in \mathscr{P}_{fin}(\mathcal{V})$. In this case, we say that σ 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 $x \in V$. If B is a set of patterns with $\mathsf{var}(B) \subseteq V$, then $B\sigma$ denotes the set of assertions obtained from B by replacing each variable by its image over the substitution σ.

Definition 3. *Let $\mathfrak{O}$ be a set of axioms. A* tableau *for $\mathfrak{O}$ is a tuple $S = (\Sigma, \mathcal{R}, \mathcal{C})$ where*

- *Σ is a signature,*
- *$\mathcal{R}$ is a set of* expansion rules *of the form $(B_0, \mathcal{N}) \to \{B_1, \ldots, B_m\}$, where $B_0, \ldots, B_m$ are finite sets of Σ-patterns and $\mathcal{N} \in \mathscr{P}_{fin}(\mathfrak{O})$, and* input rules *of the form $t \to B$, where $t \in \mathfrak{O}$ and B is a finite set of Σ-assertions, and*
- *$\mathcal{C}$ is a set of finite sets of Σ-patterns, called* clashes*.*

Given an expansion rule $R : (B_0, \mathcal{N}) \to \{B_1, \ldots, B_m\}$, 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 $\mathfrak{S} = (A, \mathcal{O})$ where A is a finite set of assertions and $\mathcal{O}$ is a finite set of axioms. The tableau algorithm works on sets of S-states. It starts with the set $\mathcal{M} = \{(\emptyset, \mathcal{O})\}$ and uses 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 that extend the first component of $\mathfrak{S}$. When no rules are applicable, then it tests whether the resulting set of S-states contains a clash; in that case, it accepts the ontology, and rejects it otherwise.

Definition 4. *An input rule $t \to B$ is* applicable *to an S-state $(A, \mathcal{O})$ if $t \in \mathcal{O}$ and $B \not\subseteq A$. An expansion rule $R : (B_0, \mathcal{N}) \to \{B_1, \ldots, B_m\}$ is* applicable *to an*

S-state $(A, \mathcal{O})$ with substitution ρ on $\mathsf{var}(B_0)$ if (i) $\mathcal{N} \subseteq \mathcal{O}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ we have $B_i\rho' \not\subseteq A$.

Given a set of S-states $\mathcal{M}$, application of an input rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ yields the set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B, \mathcal{O})\}$; application of an expansion rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ with ρ yields $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \cup B_i\sigma, \mathcal{O}) \mid 1 \leq i \leq m\}$, where σ is a substitution that extends ρ and maps the fresh variables of R to distinct new constants. If $\mathcal{M}'$ is obtained from $\mathcal{M}$ by a rule application, we write $\mathcal{M} \rightarrow_S \mathcal{M}'$. $\mathcal{M}$ is saturated *if there is no $\mathcal{M}'$ with $\mathcal{M} \rightarrow_S \mathcal{M}'$.*

The S-state $(A, \mathcal{O})$ contains a clash *if there is a $C \in \mathcal{C}$ and a substitution ρ on $\mathsf{var}(C)$ with $C\rho \subseteq A$. $\mathcal{M}$ is* full of clashes *if every $\mathfrak{S} \in \mathcal{M}$ contains a clash.*

A simple example of a tableau is the one for deciding inconsistency of $\mathcal{ALC}$ ABoxes. Its clashes are all the sets $\{D, \neg D\}$, where D is a concept name. It has different expansion rules dealing with the constructors of $\mathcal{ALC}$. For example, the rules for existential restrictions and disjunction are as follows:

$$R_\exists : (\{\exists r.C(x)\}, \emptyset) \rightarrow \{\{r(x, y), C(y)\}\},$$
$$R_\sqcup : (\{C \sqcup D(x)\}, \emptyset) \rightarrow \{\{C(x)\}, \{D(x)\}\}.$$

It also has input rules, for dealing with the ABox axioms used. For example, for concept assertions, we use

$$R_\mathsf{a} : C(a) \rightarrow \{C(\overline{a})\},$$

where $\overline{a}$ is the constant representing the individual name a.

A tableau is *correct* for a c-property $\mathcal{P}$ if every chain of rule applications starting with $\mathcal{M}_0 = \{(\emptyset, \mathcal{O})\}$ eventually reaches a saturated set of S-states $\mathcal{M}$ (i.e. it terminates) and it holds that $\mathcal{O} \in \mathcal{P}$ iff $\mathcal{M}$ is full of clashes.

In general, tableaux are not guaranteed to terminate, as their expansion rules may trigger the applicability of new expansion rules. This happens, for instance, in the tableaux algorithm for deciding inconsistency w.r.t. general TBoxes. To avoid this problem, a *blocking condition* is usually introduced. Intuitively, blocking disallows the application of a rule, whenever its application would not lead to the production of any new clashes; that is, when further expansions would not change the acceptance status of the input ontology. Several blocking conditions—like e.g. subset blocking [1], equality blocking [7], pairwise blocking [8], etc—have been studied in the literature. Rather than trying to define and deal with each of these conditions independently, we consider a general notion of blocking.

A *blocking condition* is simply a set of finite sets of Σ-assertions and Σ-patterns, using only the variables from $\mathsf{var}(B_0)$. Intuitively, these describe the situations in which the rule should not be applied. Every expansion rule is then extended with a blocking condition $\mathcal{B}$, and the rule applicability condition is restricted to satisfy additionally (iv) $A \notin \mathcal{B}\rho$. The notions of saturated sets of S-states and correctness w.r.t. a c-property are adapted accordingly. For example, the subset blocking condition disallowing the applicability of the existential rule in $\mathcal{ALC}$ is described by the set of all finite sets of Σ-assertion and Σ-patterns B

where the only variable used is x and such that there exist constants $a_1, \dots, a_n$ and roles $r_1, \dots, r_n$ such that $r_i(a_i, a_i + 1)$ for all $i, 1 \leq i < n$, $r_i(a_n, x)$, and $\{C \mid C(x) \in B\} \subseteq \{C \mid C(a_1) \in B\}$.

In the next section, we show how to transform tableaux that are correct for a c-property $\mathcal{P}$ into procedures that directly compute the $(\mathcal{O}, \mathcal{P})$-behaviour, for any given ontology $\mathcal{O}$. We focus first on tableaux without any blocking condition, which can be used for deciding consistency of $\mathcal{ALC}$ ABoxes. Later on, we show how to extend this construction to tableaux with blocking conditions, allowing us to reason w.r.t. general TBoxes also.

4 Boundary Extensions of General Tableaux

Given a tableau S that is correct for the c-property $\mathcal{P}$, we show how to construct an algorithm that computes the boundary for $\mathcal{P}$. For the moment, we focus on tableaux without blocking conditions, but later describe how to deal with blocking also.

For an input labeled ontology $\mathcal{O}$, the modified algorithm also works on sets of S-states, but every assertion a occurring in the first component of an S-state is also equipped with a label $\mathsf{lab}(a)$; we call this a *labeled S-state*. The application of rules must take the labels of the assertions and axioms into account.

If A is a set of labeled assertions and $\ell \in L$, then we say that an assertion a is *ℓ-insertable into A* if either (i) $a \notin A$ or (ii) $a \in A$ but $\ell > \mathsf{lab}(a)$. Given a set B of (unlabeled) assertions and a set A of labeled assertions, the set of *ℓ-insertable elements of B into A* is

$$\mathsf{ins}_\ell(B, A) := \{b \in B \mid b \text{ is } \ell\text{-insertable into } A\}.$$

The ℓ-insertion of these elements into A yields the set of labeled assertions $A \uplus_\ell B := A \cup \mathsf{ins}_\ell(B, A)$, where each assertion $a \in A \setminus \mathsf{ins}_\ell(B, A)$ keeps its old label $\mathsf{lab}(a)$, each assertion in $\mathsf{ins}_\ell(B, A) \setminus A$ is labeled with ℓ, and the label of each assertion $b \in A \cap \mathsf{ins}_\ell(B, A)$ is updated to ℓ.

Definition 5. *The input rule $t \to B$ is* labeled applicable *to a labeled S-state $\mathfrak{S} = (A, \mathcal{O})$ if $t \in \mathcal{O}$ and $\mathsf{ins}_{\mathsf{lab}(t)}(B, A) \neq \emptyset$. An expansion rule of the form $R : (B_0, \mathcal{N}) \to \{B_1, \dots, B_m\}$ is* labeled applicable *to a labeled S-state $(A, \mathcal{O})$ with substitution ρ on $\mathsf{var}(B_0)$ if (i) $\mathcal{N} \subseteq \mathcal{O}$, (ii) $B_0\rho \subseteq A$, and (iii) for every $1 \leq i \leq m$ and every substitution ρ' on $\mathsf{var}(B_0 \cup B_i)$ extending ρ we have $\mathsf{ins}_\ell(B_i\rho', A) \neq \emptyset$, where $\ell := \min\{\mathsf{lab}(\alpha) \mid \alpha = b\rho, b \in B_0 \text{ or } \alpha = s, s \in \mathcal{N}\}$. We call this ℓ the* degree of the rule application*.*

Given a set of labeled S-states $\mathcal{M}$, the labeled application of the input rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ yields the set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \uplus_{\mathsf{lab}(t)} B, \mathcal{O})\}$; labeled application of the expansion rule R to $\mathfrak{S} = (A, \mathcal{O}) \in \mathcal{M}$ with ρ yields the new set $\mathcal{M}' = (\mathcal{M} \setminus \{\mathfrak{S}\}) \cup \{(A \uplus_\ell B_i\sigma, \mathcal{O}) \mid 1 \leq i \leq m\}$, where ℓ is the degree of the rule application, defined above, and σ is a substitution that extends ρ and maps the fresh variables of R to distinct new constants. If $\mathcal{M}'$ is obtained from $\mathcal{M}$ by a labeled rule application, we write $\mathcal{M} \to_{S^{\mathsf{lab}}} \mathcal{M}'$. $\mathcal{M}$ is labeled saturated *if there is no $\mathcal{M}'$ with $\mathcal{M} \to_{S^{\mathsf{lab}}} \mathcal{M}'$.*

Consider a finite chain of labeled rule applications $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\mathsf{lab}}} \mathcal{M}$ such that $\mathcal{M}$ is labeled saturated. The label of an assertion appearing in $\mathcal{M}$ expresses the contexts that can derive this assertion. A clash in an S-state depends on the joint presence of possibly several assertions; thus, we need to find the contexts from which all of these assertions can be derived: this is given by the minimum of the labels of all these assertions. To decide the property $\mathcal{P}$, it suffices to have one clash per S-state $\mathfrak{S}$; hence, we are only interested in the *maximum* of the labels of all the clashes in a given state. As every S-state in $\mathcal{M}$ is required to have at least one clash, we again compute the minimum of the labels obtained from each S-state. We formalize this next.

Definition 6. *A set of assertions A' is called a* clash set *in a labeled S-state $\mathfrak{S} = (A, \mathcal{O})$ if there is a clash $C \in \mathcal{C}$ and a substitution ρ on $\mathsf{var}(C)$ with $A' = C\rho$. The* label *of this clash set is $\ell_{A'} = \min\{\mathsf{lab}(a) \mid a \in A'\}$.*

Given a set of labeled S-states $\mathcal{M} = \{\mathfrak{S}_1, \ldots, \mathfrak{S}_n\}$, the clash degree *induced by $\mathcal{M}$ is $\ell_{\mathcal{M}} := \min\{\max\{\ell_{A'} \mid A' \text{ is a clash set in } \mathfrak{S}_i\} \mid 1 \leq i \leq n\}$*

It can be shown, using similar techniques to the ones developed in [4] for pinpointing extensions of general tableaux that the clash degree is always the boundary for the c-property decided by the original tableau.

Theorem 7. *Let $\mathcal{P}$ be a c-property on $\mathfrak{O}$ and S a correct tableau for $\mathcal{P}$. For every ontology $\mathcal{O} \in \mathscr{P}_{fin}(\mathfrak{O})$ the following holds:*

> *for every finite chain of rule applications $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\mathsf{lab}}} \mathcal{M}$ with $\mathcal{M}$ labeled saturated, the clash degree $\ell_{\mathcal{M}}$ induced by $\mathcal{M}$ is a $(\mathcal{O}, \mathcal{P})$-boundary.*

Notice that this result does not depend on the order in which rules are applied. From a given set of S-states, several rules could be applicable, but the correctness of the labeled extension does not depend on which one is chosen first. This is an important feature of tableaux and their extensions, as it allows optimizations based on the choice of an ordering that leads to shorter chains of rule applications.

Unfortunately, this general approach for extending tableaux into boundary-computation methods also inherits some of the negative properties of pinpointing extensions of tableaux. In particular, (i) the labeled extension of a terminating tableau is not guaranteed to terminate (see [4] for an example), and (ii) even if it terminates, the labeled extension needs to run until saturation to guarantee that the clash label computed is indeed the boundary. The second point is used as an argument against glass-box approaches, as it disallows one of the most basic optimizations for tableaux, namely stopping the application of expansion rules on states where a clash has already been detected.

We now show that if we prioritize rule applications with a higher degree, then both of these problems disappear. Moreover, the same approach will allow us to deal with general blocking conditions in Section 4.2.

4.1 Prioritized Rule Applications

By definition, the boundary is the largest context from which the c-property can be derived. A simple idea to compute this boundary is then to try to first produce all the assertions that can be derived from a context—that is, from all the axioms having a label greater or equal to some ℓ_i—before applying rules that use axioms with a smaller label. We implement this idea by prioritizing labeled rule applications with a higher degree.

Definition 8. *The* ordered extension *of a tableau S is the labeled extension of S where, if several rules are applicable to a set of labeled S-states $\mathcal{M}$, then one rule application having the highest degree is applied.*

We write $\mathcal{M} \to_{S^{\mathsf{ord}}} \mathcal{M}'$ if $\mathcal{M}'$ is obtained from $\mathcal{M}$ by an ordered rule application. $\mathcal{M}$ is ordered saturated *if there is no $\mathcal{M}'$ with $\mathcal{M} \to_{S^{\mathsf{ord}}} \mathcal{M}'$.*

Several rule applications may have the same (highest) degree. In that case, it is irrelevant which one of these is chosen. This prioritization ordering has the consequence that the degree of successive rule applications is non-increasing.

Lemma 9. *Let $\mathcal{M}_0 \to_{S^{\mathsf{ord}}} \mathcal{M}_1 \to_{S^{\mathsf{ord}}} \mathcal{M}_2$. If $\mathcal{M}_2$ is obtained by applying a rule with degree ℓ to $\mathcal{M}_1$, and $\mathcal{M}_1$ is obtained by applying a rule with degree ℓ' to $\mathcal{M}_0$, then $\ell \leq \ell'$.*

Proof. $\mathcal{M}_1$ differs from $\mathcal{M}_0$ by having some S-states $\mathfrak{S}_1, \ldots, \mathfrak{S}_m$ that modify the assertional component of a S-state $\mathfrak{S} \in \mathcal{M}_0$ in the following way: some new assertions labeled with ℓ' are added, and some existing assertions get their label increased to ℓ'. In any case, all assertions appearing in some S-state of $\mathcal{M}_1$ with label greater than ℓ' are also in an S-state of $\mathcal{M}_0$. If $\ell > \ell'$, then all the assertions and axioms used to execute the rule $\mathcal{M}_1 \to_{S^{\mathsf{ord}}} \mathcal{M}_2$ have a label strictly greater than ℓ', and hence this rule was applicable also to an S-state in $\mathcal{M}_0$. But $\ell > \ell'$ implies that ℓ' would not be applied, since it violates the prioritization ordering. hence $\ell \leq \ell'$. □

A simple consequence of this lemma is that the labels of assertions appearing in the S-states are never modified by further ordered rule applications, as such modification would imply a rule application with a higher degree than the rule that originally created the assertion. This means that a rule is only ordered applicable if it adds new assertions to the S-state, and thus, for any set of S-states reachable from the initial set $\{(\emptyset, \mathcal{O})\}$ a rule is ordered applicable iff it is applicable in the original sense of Definition 4, as described by the following theorem.

Theorem 10. *Let S be a tableau over $\mathfrak{D}$ and $\mathcal{O} \in \mathscr{P}_{fin}(\mathfrak{D})$. For every set of labeled S-states $\mathcal{M}$, it holds that*

$$\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\mathsf{ord}}} \mathcal{M} \quad \textit{iff} \quad \{(\emptyset, \mathcal{O}_{\geq \ell})\} \xrightarrow{*}_{S} \mathcal{M},$$

where ℓ is the smallest label appearing in $\mathcal{M}$.

In particular, if S terminates in every input—as is the case for every tableau that is correct for some c-property—then its ordered extension must also terminate.

Corollary 11. *The ordered extension of any terminating tableau is terminating.*

More interesting, the execution of the ordered extension can be stopped as soon as the set of S-states $\mathcal{M}$ is full of clashes, as further rule applications will not modify the clash degree.

Corollary 12. *Let $\mathcal{M}, \mathcal{M}'$ be sets of S-states such that $\mathcal{M}$ is full of clashes and $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\mathsf{ord}}} \mathcal{M} \xrightarrow{*}_{S^{\mathsf{ord}}} \mathcal{M}'$. Then $\ell_{\mathcal{M}} = \ell_{\mathcal{M}'}$.*

4.2 Dealing with Blocking

Consider now a tableau with blocking conditions associated to its rules, that is correct for a property $\mathcal{P}$. In general, the same blocking condition cannot be used for restricting rule applications in its labeled extension. This would lead to a clash degree that may be strictly smaller than the actual boundary that this extension tries to compute, as can be seen by a simple adaptation from the examples provided in [10] for subset blocking and in [4] for equality blocking. The reason for this is that, since the blocking condition is independent of the labels used, it might be that the assertions that trigger the blocking of a rule application may depend on different contexts.

To solve this problem, a modification of the blocking condition that also takes into account the labels was proposed in [10,4]. However, it is not clear whether more complex blocking conditions would require a more elaborate labeled extension. Moreover, we want to produce a boundary-computing algorithm that works for any kind of blocking condition that fits our very general framework.

Fortunately, if we consider the prioritized rule-application ordering of ordered extensions of tableaux, we can use Theorem 10 and Corollary 12 to show that the same blocking conditions yield a correct computation of the boundary.

To define the labeled extensions of tableaux with blocking, we adapt the notion of labeled applicability of a rule to take into account the blocking condition. An expansion rule R with a blocking condition $\mathcal{B}$ is *labeled applicable* to a labeled S-state $(A, \mathcal{O})$ if it satisfies the three conditions from Definition 5 and additionally (iv) $A \notin \mathcal{B}$.

If a tableau with blocking is correct for a c-property $\mathcal{P}$, then it terminates on every input and when it reaches a saturated set of S-states, this is full of clashes iff the input ontology satisfied the property. We thus have the following result.

Theorem 13. *Let $\mathcal{P}$ be a c-property on $\mathfrak{O}$ and S a correct tableau (with blocking) for $\mathcal{P}$. For every ontology $\mathcal{O} \in \mathscr{P}_{fin}(\mathfrak{O})$ the following holds:*

- *the ordered extension of S terminates on $\mathcal{O}$; that is, there is no infinite chain of rule applications $\{(\emptyset, \mathcal{O})\} \to_{S^{\mathsf{ord}}} \mathcal{M}_1 \to_{S^{\mathsf{ord}}} \mathcal{M}_2 \to_{S^{\mathsf{ord}}} \ldots$;*
- *for every chain of rule applications $\{(\emptyset, \mathcal{O})\} \xrightarrow{*}_{S^{\mathsf{ord}}} \mathcal{M}$ with $\mathcal{M}$ labeled saturated, the clash degree $\ell_{\mathcal{M}}$ induced by $\mathcal{M}$ is a $(\mathcal{O}, \mathcal{P})$-boundary.*

5 Conclusions

We have presented a general approach for extending general tableaux algorithms that can decide a c-property $\mathcal{P}$, into methods that can compute the boundary for $\mathcal{P}$ w.r.t. to a set of linearly ordered contexts. Our approach can be used for extending the tableau-based algorithm for deciding consistency of $\mathcal{ALC}$ ABoxes w.r.t. general TBoxes, but is not limited to this logic. In fact, adding transitive and inverse roles, or considering more complex blocking conditions would not be a problem for our framework. However, our notion of general tableaux requires a monotonic extension of the consequences deduced, and cannot identify previously introduced individuals in a clean way. In future work we will study whether we can construct a tableau for reasoning in full $\mathcal{SROIQ}(\mathcal{D})$.

Our approach follows the lines of pinpointing extensions of tableaux, but has several advantages over these, mainly due to the simplicity of the linear ordering. By prioritizing the rule applications according to their degree—applying rules with higher degree first—we obtained an algorithm that does not differ much from the original tableau. The main intuition behind this prioritization of the rule applications is that it deduces all the consequences from a context higher in the ordering before testing contexts defined by a smaller label. Labeled rule applications using this ordering never modify the label of a previously existing assertion, and always add new assertions with non-increasing labels. Thus, chains of ordered rule applications correspond to rule applications of the original tableau. This in particular ensures that termination of a tableau transfers to its ordered extension, a property that is not shared by pinpointing extensions, or extensions based on lattices that are not linearly ordered.

The execution of ordered extensions of tableaux can be seen as incremental reasoning: given an ontology $\mathcal{O}$, the ordered extension first saturates the tableau using only the axioms having the greatest label ℓ_n. If this produces a set of S-states that is full of clashes, then we know that ℓ_n is the boundary for $(\mathcal{O}, \mathcal{P})$. Otherwise, it adds the axioms with label ℓ_{n-1}, and continues the tableau execution. Since several DL reasoners now implement incremental reasoning (e.g. Pellet,[2] FaCT++,[3] or CEL[4]), we believe that an implementation of our glass-box approach for reasoning in DLs with linearly ordered contexts will be a simple step.

References

1. F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.
2. F. Baader, M. Knechtel, and R. Peñaloza. Context-dependent views to axioms and consequences of semantic web ontologies. *Journal of Web Semantics*, 12–13:22–40, April 2012. Available at http://dx.doi.org/10.1016/j.websem.2011.11.006.

[2] http://clarkparsia.com/pellet/
[3] http://code.google.com/p/factplusplus/
[4] http://lat.inf.tu-dresden.de/systems/cel/

3. F. Baader and R. Peñaloza. Automata-based axiom pinpointing. *Journal of Automated Reasoning*, 45(2):91–129, August 2010. Special Issue: Selected Papers from IJCAR 2008.
4. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, February 2010. Special Issue: Tableaux and Analytic Proof Methods.
5. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In J. Hertzberg, M. Beetz, and R. Englert, editors, *Proceedings of the 30th German Annual Conference on Artificial Intelligence (KI'07)*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 52–67, Osnabrück, Germany, 2007. Springer-Verlag.
6. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2 edition, 2002.
7. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
8. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Journal of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.
9. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In K. Aberer, K.-S. Choi, N. F. Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *Proc. of the 6th Int. Semantic Web Conf. and 2nd Asian Semantic Web Conf. (ISWC'07,ASWC'07)*, volume 4825 of *LNCS*, pages 267–280, Busan, Korea, 2007. Springer-Verlag.
10. K. Lee, T. Meyer, and J. Z. Pan. Computing maximally satisfiable terminologies for the description logic ALC with GCIs. In B. Parsia, U. Sattler, and D. Toman, editors, *Proc. of the 2006 Description Logic Workshop (DL'06)*, Lake District, UK, 2006.
11. M.-J. Lesot, O. Couchariere, B. Bouchon-Meunier, and J.-L. Rogier. Inconsistency degree computation for possibilistic description logic: An extension of the tableau algorithm. In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2008)*, pages 1–6. IEEE Computer Society Press, 2008.
12. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In A. Ellis and T. Hagino, editors, *Proc. of the 14th Int. Conf. on World Wide Web (WWW'05)*, pages 633–640. ACM, 2005.
13. G. Qi and J. Z. Pan. A tableau algorithm for possibilistic description logic. In J. Domingue and C. Anutariya, editors, *Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008)*, volume 5367 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 2008.
14. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.