An Argumentation Machinery to Reason over Inconsistent Ontologies*

Martín O. Moguillansky[1], Renata Wassermann[2], and Marcelo A. Falappa[1]

[1] CONICET, AI R&D Lab (LIDIA), Universidad Nacional del Sur (UNS), Argentina
{mom,maf}@cs.uns.edu.ar
[2] Department of Computer Science (DCC), University of São Paulo (USP), Brazil
renata@ime.usp.br

Abstract. Widely accepted argumentation techniques are adapted to define a non-standard description logic (DL) reasoning machinery. A DL-based argumentation framework is introduced to reason about potentially inconsistent ontologies. Arguments in this framework can handle different DL families like $\mathcal{ALC}$, $\mathcal{EL}$, and DL-Lite. Afterwards, we propose an algorithm based on debugging techniques and classical tableau-based $\mathcal{ALC}$ satisfiability to build arguments, and discuss about the computational cost of reasoning through the proposed machinery.

1 Introduction

Much effort has been dedicated to the study of models for debugging and repairing ontologies, aiming at restoring consistency. One of the most influencing works on this matter is Schlobach and Cornet's [12]. However, in ontologies conceptualizing certain domains like medicine and law, it is well known the need to avoid loosing beliefs disregarding inconsistencies. In such areas, paraconsistent methods are usually needed in order to reason over inconsistency. Argumentation theory appears as an interesting alternative. For instance, in [10], ontologies (with some restrictions) are translated into the defeasible logic programming argumentation system (DELP) [9]. Nonetheless, it is more desirable to benefit from the continuous advances done in the area of ontology reasoning, and thus building DL-argumentation machineries on top of the corresponding specialized DL reasoner. This would also avoid any unnecessary additional complexity arising from the translation of ontologies to different logics. The main objective of this work is focused on the study of such DL-argumentation theories.

Some argumentation systems have been implemented with promisory results, as is the case of DELP among others. However, the study of complexity of reasoning through argumentation is still at its initial stages. Some results are available in [6]. In this article, we walk our first steps towards the complexity analysis of our proposed DL-argumentation machinery, which is defined upon DL semantic entailment. This allows to reason by relying on interpretation models, and therefore, tableaux techniques may be easily reused towards future implementations. Hence, since this methodology can be implemented on top of the corresponding DL reasoner, its complexity will be attached to that of the problem of reasoning in the underlying DL. We show this on ontologies

* Partially supported by UNS (PGI 24/ZN18) and CONICET (PIP 112-200801-02798).

based on an $\mathcal{ALC}$ fragment, known as unfolded $\mathcal{ALC}$, through an algorithm for building the set of defeaters of a given argument. Such algorithm is shown to correspond to the complexity class of the problem of reasoning in unfolded $\mathcal{ALC}$, which is in PSPACE. This algorithm would be useful for building dialectical trees (trees of arguments), as the elementary structure upon which the adopted argumentation semantics rely.

2 Description Logics: Brief Overview

An ontology $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ details knowledge in terms of intensional/extensional information described in $\mathcal{T}$ (TBox)/$\mathcal{A}$ (ABox). The symbols A and C identify atomic and general concepts, and P and E, atomic and general roles, respectively. Symbols $a, b, \ldots$ identify constant named individuals from the domain, and $x, y, \ldots$, free variables. The sets $\mathrm{N_V}$ and $\mathrm{N_C}$ are assumed to contain variables and constant names, respectively. A TBox usually contains axioms like $C_1 \sqsubseteq C_2$ identified as *general concept inclusions (GCIs)*; and an ABox, *membership assertions* like $C(a)$ and $E(a, b)$. However, different restrictions may appear depending on each specific description language. From now on we assume the reader is familiar with DLs. For more details on DLs refer to [2].

DL-semantics is given in terms of the standard set theoretic Tarskian semantics, through *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. If the interpretation $\mathcal{I}$ is model of the ontology Σ, we write $\mathcal{I} \models \Sigma$, implying $\mathcal{I} \models \phi$, for every assertion $\phi \in \Sigma$. An ontology is *satisfiable* (or *consistent*) if it admits at least one model. An ontology is *coherent* if it is satisfiable and for every concept C and every model $\mathcal{I}$, $C^{\mathcal{I}}$ is non-empty. An *incoherent* ontology considers a concept C which does not accept any individual in some model; however satisfiability could still hold in such ontology. Finally, an ontology Σ *logically implies* (or *entails*) an assertion ϕ, written $\Sigma \models \phi$, if for every model $\mathcal{I}$ of Σ, $\mathcal{I} \models \phi$.

To verify logical implications of DL formulae we rely on *reasoning services (RS)* like *subsumption* ($\Sigma \models C_1 \sqsubseteq C_2$), *instance checking* ($\Sigma \models C(a)$ or $\Sigma \models E(a, b)$), and *knowledge base satisfiability* (whether the ontology admits at least one model). In addition, *query answering* ($\Sigma \models q(\bar{x})$) has been most widely considered for querying DLs. A *conjunctive query (cq)* $q(\bar{x})$, with a tuple $\bar{x} \in (\mathrm{N_V})^n$ of arity $n \geq 0$, is a non empty set of atoms $C(z)$, $E(z_1, z_2)$, $z_1 = z_2$, or $z_1 \neq z_2$, where C and E are respectively a general concept and a general role of Σ, and only the names $\{z, z_1, z_2\} \cap \mathrm{N_V}$ are considered in $\bar{x}$. Function $\mathtt{var} : (\mathrm{N_V})^n \longrightarrow 2^{\mathrm{N_V}}$ identifies the variables in $\bar{x}$. When $\bar{x}$ is the empty tuple, no free variables are considered and the query is identified as *boolean*. Intuitively, $q(\bar{x})$ represents the conjunction of its elements. Let $\mathcal{I}$ be an interpretation, and $\mathfrak{m} : \mathtt{var}(\bar{x}) \cup \mathrm{N_C} \longrightarrow \Delta^{\mathcal{I}}$ a total function. If $z \in \mathrm{N_C}$ then $\mathfrak{m}(z) = z^{\mathcal{I}}$ otherwise $\mathfrak{m}(z) = a \in \Delta^{\mathcal{I}}$. We write $\mathcal{I} \models^{\mathfrak{m}} C(z)$ if $\mathfrak{m}(z) \in C^{\mathcal{I}}$, $\mathcal{I} \models^{\mathfrak{m}} E(z_1, z_2)$ if $(\mathfrak{m}(z_1), \mathfrak{m}(z_2)) \in E^{\mathcal{I}}$, $\mathcal{I} \models^{\mathfrak{m}} (z_1 = z_2)$ if $\mathfrak{m}(z_1) = \mathfrak{m}(z_2)$, and $\mathcal{I} \models^{\mathfrak{m}} (z_1 \neq z_2)$ if $\mathfrak{m}(z_1) \neq \mathfrak{m}(z_2)$. If $\mathcal{I} \models^{\mathfrak{m}} \phi$ for all $\phi \in q(\bar{x})$, we write $\mathcal{I} \models^{\mathfrak{m}} q(\bar{x})$ and call $\mathfrak{m}$ a *match* for $\mathcal{I}$ and $q(\bar{x})$. We say that $\mathcal{I}$ satisfies $q(\bar{x})$ and write $\mathcal{I} \models q(\bar{x})$ if there is a match $\mathfrak{m}$ for $\mathcal{I}$ and $q(\bar{x})$. If $\mathcal{I} \models q(\bar{x})$ for all models $\mathcal{I}$ of an ontology Σ, we write $\Sigma \models q(\bar{x})$ and say that Σ entails $q(\bar{x})$. Note that by answering cqs, instance checking ends up subsumed.

Reasoning in the standard $\mathcal{ALC}$ ($C \longrightarrow A | \bot | \top | \neg C | C \sqcap C | C \sqcup C | \forall P.C | \exists P.C$) is EXPTIME-complete, however, by restricting their use to unfoldable $\mathcal{ALC}$, satisfiability turns to PSPACE-complete. (A TBox is called unfoldable if the left-hand sides of ax-

ioms (defined concepts) are atomic, and if the right-hand sides (definitions) contain no direct or indirect reference to the defined concept.) The panorama gets worse by relying upon more expressive DLs. Different DLs have been proposed aiming at reducing the complexity of reasoning in detriment of their expressivity. That is the case of $\mathcal{EL}$ [1] ($C \longrightarrow A|\bot|\top|C \sqcap C|\exists P.C$), whose satisfiability checking was shown polinomial.

For $\mathcal{ALC}$ (and some extensions) and $\mathcal{EL}$, we will only consider the RSs of *subsumption* $\Sigma \models C_1 \sqsubseteq C_2$; and *query answering* $\Sigma \models q(\bar{x})$. In addition, for those languages whose names contain the letter $\mathcal{H}$, subsumption also corresponds to $\Sigma \models E_1 \sqsubseteq E_2$. This is the case of $\mathcal{ALCH}$, $\mathcal{ELH}$, and $\mathcal{ALCNH}^{-1,\neg}$ (see Ex. 5), among others.

A particular family of DLs which aims at lower complexity in detriment of expressivity is DL-Lite [5]. These logics allow to reason about large amounts of assertional data (ABox). Data complexity of query answering (wrt. the size of the ABox) is in LOGSPACE for most of the DL-Lite members, and polinomial regarding the whole ontology. Moreover, queries over DL-Lite ontologies may be rewritten as SQL queries so that standard database query engines can be used. The *DL-Lite*$_{\texttt{core}}$ grammar is given by $B \longrightarrow A|\exists R$, $C \longrightarrow B|\neg B$, $R \longrightarrow P|P^-$, and $E \longrightarrow R|\neg R$. In *DL-Lite*$_{\texttt{core}}$, the TBox is formed by axioms like $B \sqsubseteq C$, and the ABox by membership assertions like $A(a)$ and $P(a,b)$. Adding to *DL-Lite*$_{\texttt{core}}$ axioms like $R \sqsubseteq E$, or *functional restrictions* like $(\mathrm{funct}R)$, renders the languages *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$, respectively. (An interpretation $\mathcal{I}$ is a model of a $(\mathrm{funct}R)$ if the binary relation $R^{\mathcal{I}}$ is a function, *i.e.*, $(x, y_1) \in R^{\mathcal{I}}$ and $(x, y_2) \in R^{\mathcal{I}}$ implies $y_1 = y_2$. In this case, the *functionality checking* RS $\Sigma \models (\mathrm{funct}R)$ and $\Sigma \models \neg(\mathrm{funct}R)$, is considered.) Combining both extensions renders *DL-Lite*$_{(\mathcal{RF})}$, whose satisfiability turns to EXPTIME-hard.

3 DL Argumentation

An *argument* may be seen as a set of interrelated pieces of knowledge (in a language $\mathcal{L}$) providing support to a claim. A language $\mathcal{L}_{\texttt{cl}}$ for claims is assumed: for any $\phi \in \mathcal{L}_{\texttt{cl}}$, there is a set $\Phi \subseteq \mathcal{L}$ such that $\Phi \models \phi$. Caligraphic letters $\mathcal{A}, \mathcal{B}, \ldots$, denote arguments.

Definition 1 (Argument). *An **argument** $\mathcal{B}$ is a structure $\langle \Delta, \beta \rangle$, where $\Delta \subseteq \mathcal{L}$ is the **body**, $\beta \in \mathcal{L}_{\texttt{cl}}$ the **claim**, and it holds* (1) *$\Delta \models \beta$,* (2) *Δ is satisfiable (or $\Delta \not\models \bot$), and* (3) *$\nexists X \subset \Delta : X \models \beta$. We say that $\mathcal{B}$ **supports** β.*

Queries done to a knowledge base (KB) are supported through the claim of arguments. For ontology reasoning, we assume arguments' bodies to be included in an ontology $\Sigma \subseteq \mathcal{L}$, where $\mathcal{L}$ is the underlying DL in Σ. In this work, we write $\mathcal{L}$ to refer to general description languages (unless we reify $\mathcal{L}$ to a specific DL).

The *domain of arguments from an ontology* Σ is identified through the set $\mathbb{A}_\Sigma$. By means of $\mathfrak{bd} : \mathbb{A}_\Sigma \longrightarrow 2^{\mathcal{L}}$ and $\mathfrak{cl} : \mathbb{A}_\Sigma \longrightarrow \mathcal{L}_{\texttt{cl}}$, the body $\mathfrak{bd}(\mathcal{B})$ and claim $\mathfrak{cl}(\mathcal{B})$ of an argument $\mathcal{B} \in \mathbb{A}_\Sigma$, may be respectively identified. Observe that claims (within $\mathcal{L}_{\texttt{cl}}$) conform to $\mathcal{L}$ since they are not necessarily contained in Σ but entailed by Σ. The entailment $\models$ is obtained through usual DL interpretations, and for querying ontologies we refer to RSs as presented before. Hence, arguments should support RSs through their claims conforming the language $\mathcal{L}_{\texttt{cl}} \longrightarrow C_1 \sqsubseteq C_2|q(\langle\rangle)$, and we additionally consider $E_1 \sqsubseteq E_2$, when $\mathcal{L} =$ *DL-Lite*$_{\mathcal{R}}$ or the $\mathcal{L}$ string contains an $\mathcal{H}$; and

$(\mathrm{funct}R)|\neg(\mathrm{funct}R)$, when $\mathcal{L} =$ *DL-Lite*$_{(\mathcal{RF})}$ or *DL-Lite*$_{\mathcal{F}}$. Observe that from $\mathcal{L}_{\mathtt{cl}}$'s syntax, $q(\langle\rangle)$ refers to a cq with an empty tuple from $\mathrm{N_V}$. Hence, no argument supports a claim with free variables. That is, for any ontology $\Sigma \subseteq \mathcal{L}$ and any argument $\mathcal{B} \in \mathbb{A}_\Sigma$, if $\mathfrak{cl}(\mathcal{B}) = q(\bar{x})$ then $\bar{x} = \langle\rangle$. Given a query $\alpha \in \mathcal{L}_{\mathtt{cl}}$, an argument $\mathcal{B}$ is a *query supporter* if $\mathcal{B}$ supports α. We define the notion of query supporter to cope with any RS.

Definition 2 (Query Supporter). *Given an ontology $\Sigma \subseteq \mathcal{L}$ and a query $\alpha \in \mathcal{L}_{\mathtt{cl}} \cup \{q(\bar{x})\}$; an argument $\mathcal{B} \in \mathbb{A}_\Sigma$ is an α-**supporter** iff for every model $\mathcal{I}$ of $\mathfrak{cl}(\mathcal{B})$ there exists a match $\mathfrak{m}$ for $\mathcal{I}$ and α such that $\mathcal{I} \models^{\mathfrak{m}} \alpha$ holds, or equivalently $\mathfrak{cl}(\mathcal{B}) \models \alpha$.*

Primitive arguments like $\langle\{A(a)\}, A(a)\rangle$ appear when $A(a) \in \Sigma$; assuming $\{A(a), A \sqsubseteq B\} \subseteq \Sigma$, more complex arguments appear within $\mathbb{A}_\Sigma$, like $\langle\{A \sqsubseteq B\}, A \sqsubseteq B\rangle$ and $\langle\{A(a), A \sqsubseteq B\}, B(a)\rangle$. An argument $\mathcal{B} \in \mathbb{A}_\Sigma$ is *subargument* of $\mathcal{C} \in \mathbb{A}_\Sigma$ (and conversely, $\mathcal{C}$ is a *superargument* of $\mathcal{B}$) if it follows that $\mathfrak{bd}(\mathcal{B}) \subseteq \mathfrak{bd}(\mathcal{C})$ holds; moreover, $\mathcal{B}$ is a *proper subargument* of $\mathcal{C}$ if $\mathfrak{bd}(\mathcal{B}) \subset \mathfrak{bd}(\mathcal{C})$ holds. Observe that $\langle\{A \sqsubseteq B\}, A \sqsubseteq B\rangle$ is a subargument of $\langle\{A(a), A \sqsubseteq B\}, B(a)\rangle$. A functionality checking $(\mathrm{funct}R)$ is supported through $\langle\{(\mathrm{funct}R)\}, (\mathrm{funct}R)\rangle$. For other arguments considering functional assertions, assume for instance, $\neg P(a, c)$ is supported through both $\langle\{P(a, b), (\mathrm{funct}P)\}, \{\neg P(a, c)\}\rangle$ and $\langle\{P(b, c), (\mathrm{funct}P^-)\}, \{\neg P(a, c)\}\rangle$. CQ's like $q(\bar{x})$, where for instance $q(\bar{x}) = \{A(x), B(y)\}$ and $\bar{x} = \langle x, y\rangle$, might be supported through an argument $\mathcal{B} = \langle\{A(a), A \sqsubseteq B\}, q(\bar{y})\rangle$, where $q(\bar{y}) = \{A(a), B(a)\}$, $\bar{y} = \langle\rangle$, and a match $\mathfrak{m}(x) = \mathfrak{m}(y) = a$, appears. Observe that by changing the query to $q(\bar{x}) = \{A(x), B(y), x \neq y\}$, argument $\mathcal{B}$ is no longer a $q(\bar{x})$-supporter. From now on we refer to arguments like $\mathcal{B}$ as $\langle\{A(a), A \sqsubseteq B\}, \{A(a), B(a)\}\rangle$, avoiding the explicit writing of cq's like $q(\langle\rangle) = \{A(a), B(a)\}$ as claims.

Counterarguments are arguments whose claims pose justifications to disbelieve in other arguments. From the standpoint of an ontology Σ, counterarguments bring about sources of incoherence/inconsistency determined by their bodies. Counterarguments are usually formalized upon negation of formulae, nonetheless, this is problematic in DLs since negation of axioms can fall beyond the scope of the language. Hence, counterarguments are formalized upon DL satisfiability, avoiding the use of negation of axioms.

Definition 3 (Counterargument). *Given $\Sigma \subseteq \mathcal{L}$, arguments $\mathcal{B} \in \mathbb{A}_\Sigma$ and $\mathcal{C} \in \mathbb{A}_\Sigma$ are **in conflict** iff $\mathfrak{bd}(\mathcal{B}) \cup \mathfrak{cl}(\mathcal{C})$ is unsatisfiable. Argument $\mathcal{C}$ is a **counterargument** of $\mathcal{B}$.*

Argument-based preference relations usually determine *defeats* between arguments. However, for simplicity we avoid its usage, and identify *defeaters* strightforwardly from counterarguments: *$\mathcal{C}$ defeats $\mathcal{B}$* (noted $\mathcal{C} \hookrightarrow \mathcal{B}$) *iff* $\mathcal{C}$ counterargues $\mathcal{B}$.

Usually, arguments can be *rebutted* with an argument of opposite conclusion and they can be *undercut* with an argument whose conclusion opposes either explicitly or implicitly to the body of the defeated argument. Given two arguments $\mathcal{B}$ and $\mathcal{C}$ such that $\mathcal{C}$ counterargues $\mathcal{B}$, we usually say that $\mathcal{C}$ rebutts $\mathcal{B}$ if it follows that $\mathfrak{cl}(\mathcal{B}) = \neg\mathfrak{cl}(\mathcal{C})$. Nevertheless, for the case of DL-arguments this may not be so trivial. Let us analyze with more detail the negation of axioms in the context of DL arguments. Given an argument $\mathcal{B} = \langle\{A \sqsubseteq B, B \sqsubseteq C\}, A \sqsubseteq C\rangle$, a possible rebuttal would support an axiom like $\neg(A \sqsubseteq C)$. Negation of axioms like $A \sqsubseteq C$ was studied in [8] determining two kinds of negation: *consistency-negation* $\neg(A \sqsubseteq C) = \exists(A \sqcap \neg C)$ and *coherency-negation* $\sim(A \sqsubseteq C) = A \sqsubseteq \neg C$. For the former, an existence assertion like $\exists(A \sqcap \neg C)(x)$

would serve, however for languages without concept conjunction like *DL-Lite*$_{(\mathcal{RF})}$, it would fall out of the scope. For such cases, the existence assertion can be rewritten as a cq like $q(\langle x \rangle) = \{A(x), \neg C(x)\}$. For instance, $\mathcal{C} = \langle \{A(a), A \sqsubseteq \neg C\}, \{A(a), \neg C(a)\} \rangle$ would be a possible rebuttal supporting $q(\langle x \rangle)$ with $\mathfrak{m}(x) = a$. On the other hand, coherency-negation may be simply achieved by looking for an argument supporting $A \sqsubseteq \neg C$. In addition, for *DL-Lite*$_{\mathcal{F}}$, we extend negation of axioms to functional assertions, interpreting $\neg(\mathrm{funct} R)$ as a role R that does not conform to the definition of a function. An argument supporting such negation should consider extensional information (ABox). For instance, $\langle \{P(a,b), P'(a,c), P' \sqsubseteq P\}, \neg(\mathrm{funct} P) \rangle$.

Nonetheless, a major drawback appears when considering query answering: negation of cqs is in general undefined. As an alternative solution, we concentrate on finding sources of DL unsatisfiability through *undercuts*: $\mathcal{C}$ undercuts $\mathcal{B}$ *iff* $\mathcal{C}$ counterargues $\mathcal{B}$. Several undercuts may appear for a same argument. Furthermore, as is shown next, an undercut of an argument $\mathcal{B}$ may encompass other defeaters of $\mathcal{B}$.

Example 1. Consider the *DL-Lite*$_{\mathtt{core}}$ arguments $\mathcal{B} = \langle \{A(a), A \sqsubseteq B\}, \{B(a)\} \rangle$ and $\mathcal{C} = \langle \{A(b), A \sqsubseteq C, C \sqsubseteq \neg B\}, \{A(b), \neg B(b)\} \rangle$, and $\mathcal{C}$'s subargument $\mathcal{C}' = \langle \{A \sqsubseteq C, C \sqsubseteq \neg B\}, A \sqsubseteq \neg B \rangle$. Both $\mathcal{C}$ and $\mathcal{C}'$ counterargue $\mathcal{B}$.

Canonical undercuts [4] were defined (upon classic logic) as representatives of all defeaters of an argument. The claim of a canonical undercut $\mathcal{C}$ negates the conjunctive enumeration of all beliefs from the body of the counterargued argument $\mathcal{B}$, *i.e.*, $\mathfrak{cl}(\mathcal{C}) = \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$, where $\mathfrak{bd}(\mathcal{B}) = \{\alpha_1, \ldots, \alpha_n\}$. However, such kind of claims would fall out of the scope of a DL $\mathcal{L}$. Canonical undercuts constitute minimal sources of inconsistency (taken from the KB they are built) wrt. the body of the counterargued argument. Thus, by following such intuition, we propose *minimal undercuts*.

Definition 4 (Minimal Undercut). *Let $\mathcal{B}, \mathcal{C} \in \mathbb{A}_{\Sigma}$ be such that $\mathcal{C}$ counterargues $\mathcal{B}$. $\mathcal{C}$ is a **minimal undercut** of $\mathcal{B}$ iff there is no proper subargument of $\mathcal{C}$ counterarguing $\mathcal{B}$.*

To illustrate this notion, note from Ex. 1, that $\mathcal{C}'$ is a minimal undercut of $\mathcal{B}$. Through the use of minimal undercuts, we have restricted the consideration of counterarguments to those of minimal body. However, for some DLs several minimal undercuts with different claims may appear (see Ex. 2). In such cases, we will keep those of maximum entailment, identified as *maximally conservative undercuts* (*mcu*, for short).

Definition 5 (Maximally Conservative Undercut). *Let $\mathcal{B} \in \mathbb{A}_{\Sigma}$ and $\mathcal{C} \in \mathbb{A}_{\Sigma}$ be such that $\mathcal{C}$ minimally undercuts $\mathcal{B}$. Argument $\mathcal{C}$ is a **maximally conservative undercut (mcu)** of $\mathcal{B}$ iff for every subargument $\mathcal{C}'$ of $\mathcal{C}$ minimally undercutting $\mathcal{B}$, $\mathfrak{cl}(\mathcal{C}) \models \mathfrak{cl}(\mathcal{C}')$ holds.*

Example 2. Let $\mathcal{C} = \langle \Psi, A \sqcup B \sqsubseteq \neg C \rangle$, $\mathcal{C}' = \langle \Psi, A \sqsubseteq \neg C \rangle$, and $\mathcal{C}'' = \langle \Psi, B \sqsubseteq \neg C \rangle$, with $\Psi = \{A \sqcup B \sqsubseteq D, D \sqsubseteq \neg C\}$; be three $\mathcal{ALC}$ minimal undercuts of $\mathcal{B} = \langle \{(A \sqcap B)(a), A \sqcup B \sqsubseteq C\}, \{(A \sqcup B)(a), C(a)\} \rangle$. Note that only $\mathcal{C}$ is an mcu of $\mathcal{B}$.

Classic argumentation semantics like those from [7] (and others) can be applied to the DL-argumentation framework presented here. Nonetheless, ontology reasoning requires practical approaches. Hence, we should control the number of arguments needed to reason. Since querying ontologies can be performed via RSs (Sect. 2), the reasoning

methodology we assume is based on the acceptability of some query supporter obtained from $\mathbb{A}_\Sigma$. Thus, we rely on an argumentation semantics defined upon *dialectical trees*.

One of the essential elements of this semantics is the notion of *argumentation line*: a non-empty sequence $\lambda = [\mathcal{B}_1 \ldots, \mathcal{B}_n]$ of arguments from $\mathbb{A}_\Sigma$, where $\mathcal{B}_i \hookrightarrow \mathcal{B}_{i-1}$, for any $1 < i \leq n$. Argument $\mathcal{B}_1$ is identified as λ's root, and $\mathcal{B}_n$, as λ's leaf. An argumentation line could be seen as *two parties engaged in a discussion*: one standing by the root argument and the other arguing against it. Consequently, given a line λ, we identify the *set of pro (resp, con) arguments* containing all arguments placed on odd (resp, even) positions in λ, noted as λ^+ (resp, λ^-). We will abuse notation and write $\mathcal{B} \in \lambda$ to identify $\mathcal{B}$ from the argumentation line λ. An initial sequence of arguments in a line $\lambda = [\mathcal{B}_1, \ldots, \mathcal{B}_n]$ is identified through its *upper segment* $\lambda^{\uparrow}[\mathcal{B}_i] = [\mathcal{B}_1, \ldots, \mathcal{B}_i]$, with $1 \leq i \leq n$. Besides, the *proper upper segment* of λ wrt. $\mathcal{B}_i$ $(i \neq 1)$ is defined as $\lambda^{\uparrow}(\mathcal{B}_i) = [\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}]$. We refer to both proper and non-proper upper segments simply as "upper segments" and will be distinguishable only through the notation (round or square brackets respectively). *Acceptability conditions* are used to determine finite and non-fallacious lines, named *acceptable argumentation lines*, which are ensured to be non-circular and concordant. A line $[\mathcal{B}_1, \ldots, \mathcal{B}_n]$ is *non-circular iff* for any $1 \leq i \leq n$, the smallest subargument $\mathcal{C}$ of $\mathcal{B}_i$ such that $\mathcal{C}$ is defeated by $\mathcal{B}_{i+1}$, is not reintroduced in the rest of the line, that is $\mathcal{C}$ is not a subargument of any argument $\mathcal{B}_j$, with $i < j \leq n$. A line λ is *concordant iff* the sets $\bigcup_{\mathcal{B} \in \lambda^+} \mathfrak{bd}(\mathcal{B})$ and $\bigcup_{\mathcal{B} \in \lambda^-} \mathfrak{bd}(\mathcal{B})$ of bodies of pro and con arguments (respectively) are individually satisfiable. More on acceptability conditions can be found in [9]. Additionally, lines will be required to be *exhaustive*: if λ is acceptable, extending it with any defeater of its leaf determines a non-acceptable line.

On the other hand, the acceptability of a query supporter $\mathcal{R} \in \mathbb{A}_\Sigma$, namely the *warrant* status of $\mathcal{R}$, is determined by analizing the *dialectical tree* rooted in $\mathcal{R}$. Such tree is built from a maximal set of argumentation lines rooted in $\mathcal{R}$. Reducing the number of arguments' defeaters allows us to shrink the set of argumentation lines used to build dialectical trees. The usage of mcus results benefitting to such end, allowing to consider a single defeater from a set of minimal undercuts of a common argument. However, for DLs where logic equivalence of claims is possible, several mcus may appear.

Example 3. Given $\Sigma \subseteq \mathcal{ALC}$ and $\mathcal{B} = \langle \{A \sqsubseteq B', B' \sqsubseteq \neg B\}, A \sqsubseteq \neg B \rangle$ in $\mathbb{A}_\Sigma$. Assuming $\langle \{A(a), B(a)\}, \{(A \sqcap B)(a)\} \rangle$ and $\langle \{A(a), B(a)\}, \{A(a), B(a)\} \rangle$ can be built, both arguments are mcus of $\mathcal{B}$ with identical bodies and logically equivalent claims.

Building dialectical trees requires to identify *bundle sets*: maximal sets of argumentation lines rooted in a common argument. To such end, we define a domain $\mathbb{L}_\Sigma$ containing all the acceptable and exhaustive lines. $\mathbb{L}_\Sigma$ is ensured to be free of redundancies as seen in Ex. 3, by restricting the use of mcus with equivalent claims (non-redundancy).

Definition 6 (Argumentation Line Domain). *Given an ontology Σ, the **argumentation line domain** $\mathbb{L}_\Sigma$, is the maximal set of argumentation lines $\lambda = [\mathcal{B}_1, \ldots, \mathcal{B}_n]$, where $\mathcal{B}_i \in \mathbb{A}_\Sigma$ (for $1 \leq i \leq n$), such that λ is acceptable, exhaustive, **maximally conservative** ($\mathcal{B}_j$ is an mcu of $\mathcal{B}_{j-1}$ (for $1 < j \leq n$)), and **non-redundancy** is guaranteed:*

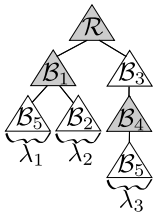
(non-redundancy) *for any $\{\lambda, \lambda'\} \subseteq \mathbb{L}_\Sigma$, $\mathcal{B} \in \lambda$ and $\mathcal{C} \in \lambda'$, if $\mathfrak{bd}(\mathcal{B}) = \mathfrak{bd}(\mathcal{C})$ and $\lambda^{\uparrow}(\mathcal{B}) = \lambda'^{\uparrow}(\mathcal{C})$ then $\lambda = \lambda'$.*

A *bundle set* for $\mathcal{R}$, noted as $\mathcal{S}(\mathcal{R})$, contains all the argumentation lines from $\mathbb{L}_\Sigma$ rooted in $\mathcal{R}$. From a bundle set $\mathcal{S}(\mathcal{R})$, the dialectical tree $\mathcal{T}(\mathcal{R})$ is constructed.

Definition 7 (Dialectical Tree). *Given an ontology $\Sigma \subseteq \mathcal{L}$, a **dialectical tree** $\mathcal{T}(\mathcal{R})$ rooted in $\mathcal{R} \in \mathbb{A}_\Sigma$ is determined by the bundle set $\mathcal{S}(\mathcal{R}) \subseteq \mathbb{L}_\Sigma$ such that an argument $\mathcal{C}$ in $\mathcal{T}(\mathcal{R})$ is: (1) a **node** iff $\mathcal{C} \in \lambda$, for any $\lambda \in \mathcal{S}(\mathcal{R})$; (2) a **child** of a node $\mathcal{B}$ in $\mathcal{T}(\mathcal{R})$ iff $\mathcal{C} \in \lambda$, $\mathcal{B} \in \lambda'$, for any $\{\lambda, \lambda'\} \subseteq \mathcal{S}(\mathcal{R})$, and $\lambda'^{\uparrow}[\mathcal{B}] = \lambda^{\uparrow}(\mathcal{C})$. The **leaves** in $\mathcal{T}(\mathcal{R})$ are the leaves of each line in $\mathcal{S}(\mathcal{R})$. The domain of trees from Σ is noted as $\mathbb{T}_\Sigma$.*

Example 4.

Given an ontology $\Sigma \subseteq \mathcal{L}$, and the bundle set $\mathcal{S}(\mathcal{R}) \subseteq \mathbb{L}_\Sigma$ determining the dialectical tree $\mathcal{T}(\mathcal{R}) \in \mathbb{T}_\Sigma$ (depicted on the right) such that $\mathcal{S}(\mathcal{R}) = \{\lambda_1, \lambda_2, \lambda_3\}$, where $\lambda_1 = [\mathcal{R}, \mathcal{B}_1, \mathcal{B}_5]$, $\lambda_2 = [\mathcal{R}, \mathcal{B}_1, \mathcal{B}_2]$, and $\lambda_3 = [\mathcal{R}, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5]$, are three acceptable, maximally conservative, and non-redundant lines. Observe that argument $\mathcal{B}_2$ is a child of $\mathcal{B}_1$ in $\mathcal{T}(\mathcal{R})$ given that $\lambda_1^{\uparrow}[\mathcal{B}_1] = \lambda_2^{\uparrow}(\mathcal{B}_2)$ (see Def. 7).



We write $\lambda \in \mathcal{T}(\mathcal{R})$ when λ is a line in $\mathcal{T}(\mathcal{R})$. Given $\Sigma \subseteq \mathcal{L}$, a query supporter $\mathcal{R} \in \mathbb{A}_\Sigma$ is finally accepted (or warranted) by analizing the dialectical tree $\mathcal{T}(\mathcal{R})$. To such end, a *marking function* $\mathfrak{mark} : \mathbb{A}_\Sigma \times \mathbb{L}_\Sigma \times \mathbb{T}_\Sigma \longrightarrow \mathbb{M}$ assigns to each argument in $\mathcal{T}(\mathcal{R})$ a mark from $\mathbb{M} = \{D, U\}$, where D/U means defeated/undefeated. The mark of an inner node in $\mathcal{T}(\mathcal{R})$ is obtained from its children (*i.e.*, its defeaters) by following a *marking criterion*. We adopt a skeptical marking criterion (as used in DELP [9]) defined as: (1) all leaves are marked U and (2) every inner node $\mathcal{B}$ is marked U *iff* every child of $\mathcal{B}$ is marked D, otherwise, $\mathcal{B}$ is marked D. The *warranting function* $\mathbf{warrant} : \mathbb{T}_\Sigma \longrightarrow \{true, false\}$ determines the root's acceptance verifying $\mathbf{warrant}(\mathcal{T}(\mathcal{R})) = true$ *iff* $\mathfrak{mark}(\mathcal{R}, \lambda, \mathcal{T}(\mathcal{R})) = U$. Hence, $\mathcal{R}$ is *warranted* from $\mathcal{T}(\mathcal{R})$ *iff* $\mathbf{warrant}(\mathcal{T}(\mathcal{R})) = true$. In such a case, $\mathcal{T}(\mathcal{R})$ is referred as *warranting tree*. These notions are illustrated with arguments painted in grey/white standing for D/U marks. For instance, in Ex. 4, $\mathcal{R}$ is defeated and thus $\mathcal{T}(\mathcal{R})$ is non-warranting.

Definition 8 (Argumentative Entailment). *Given $\Sigma \subseteq \mathcal{L}$ and a query $\alpha \in \mathcal{L}_{\mathtt{c1}} \cup \{q(\bar{x})\}$; $\Sigma \mid\!\!\approx \alpha$ iff there is a warranted α-supporter $\mathcal{R} \in \mathbb{A}_\Sigma$. If there is no warranted α-supporter from $\mathbb{A}_\Sigma$ then α is not entailed by Σ, noted as $\Sigma \not\mid\!\!\approx \alpha$.*

Theorem 1. *Given an ontology $\Sigma \subseteq \mathcal{L}$, if Σ is coherent and consistent then for any query $\alpha \in \mathcal{L}_{\mathtt{c1}} \cup \{q(\bar{x})\}$ it holds $\Sigma \models \alpha$ iff $\Sigma \mid\!\!\approx \alpha$.*

Example 5. To analyze whether a given presidential formula is reliable, we study how candidates voted for the last most relevant laws in the parlament, deciding in this manner whether a pair of candidates might be coalitionable. For that matter, we consider a role P, standing for presidential formula such that for any $(x, y) \in P^{\mathcal{I}}$, individual x is candidate for president and y for vice-president; role C (coalitionable candidates) such that for any $(x, y) \in C^{\mathcal{I}}$, individual x and y are two politicians that are assumed to agree according to their ideology on the most important national matters; concepts L and L', standing for the two most relevant laws being promulged during the last presidential period such that $a \in L^{\mathcal{I}}$ (resp., $a \in L'^{\mathcal{I}}$) identifies the politician that voted in favor of the L's (resp, L''s) promulgation; concept L_1, standing for one of the most controversial

articles from L's promulgation such that $a \in L_1^{\mathcal{I}}$ identifies the politician that voted in favor of L_1. The ontology $\Sigma \subseteq \mathcal{ALCNH}^{-1,\neg}$ will contain axioms $P \sqsubseteq C$ (every presidential formula is coalitionable); $C \sqsubseteq C^-$ (every coalitionable pair of politicians is commutative); $L_1 \sqsubseteq L$ (politicians in favor of article L_1 should have voted in favor of law L); $\forall P.\top \sqsubseteq= 1P$ and $\forall P^-.\top \sqsubseteq= 1P^-$ (a presidential formula should be unique and the candidates should have the expected position explicitly announced, and candidates presented in several presidential formulas are assumed to be less reliable); and $\forall C.L \sqsubseteq L$ and $\forall C.L' \sqsubseteq L'$ (politicians agreeing in L or L' are coalitionable). The Σ's ABox will include $P(a,b)$, $P(a,c)$, $P(d,e)$, $\neg C(d,e)$, $L(a)$, $\neg L(b)$, $L_1(b)$, $\neg L'(a)$, $L'(c)$, where individuals a, b, c, d, e, are currently active politicians.

Checking the reliability of the presidential formula $P(a,b)$ implies finding out whether $\Sigma \mid\!\!\approx P(a,b)$ holds. From $\mathbb{A}_\Sigma$, $\mathcal{R} = \langle\{P(a,b)\}, \{P(a,b)\}\rangle$ is a query supporter, and $\mathcal{B}_1 = \langle\{P \sqsubseteq C, C \sqsubseteq C^-, \forall C.L \sqsubseteq L, L(a), \neg L(b)\}, \{\neg P(a,b)\}\rangle$, $\mathcal{B}_2 = \langle\{L_1(b), L_1 \sqsubseteq L\}, \{L(b)\}\rangle$, $\mathcal{B}_3 = \langle\{P(a,c), \forall P.\top \sqsubseteq= 1P\}, \{\neg P(a,b)\}\rangle$, $\mathcal{B}_4 = \langle\{P \sqsubseteq C, \forall C.L' \sqsubseteq L', \neg L'(a), L'(c)\}, \{\neg P(a,c)\}\rangle$, and $\mathcal{B}_5 = \langle\{P(d,e), \neg C(d,e)\}, \{P(d,e), \neg C(d,e)\}\rangle$, determine the tree $\mathcal{T}(\mathcal{R})$ depicted in Ex. 4. Since $\mathcal{T}(\mathcal{R})$ is non-warranting and $\mathcal{R}$ is the only query supporter, we conclude $\Sigma \not\mid\!\!\approx P(a,b)$.

4 How Feasible is this Non-Standard DL-Reasoning Methodology?

To implement our argumentation-DL machinery two questions need to be addressed: how to construct (1) an argument supporting a query, and (2) the defeaters of a given argument. For (1) techniques upon the reasoning procedure on the DL at issue may be used. Some works on this matter are [11] (in $\mathcal{ALC}$) and [3] (in $\mathcal{EL}$). Here, we address (2) by relying on MUPS (minimal unsatisfiability-preserving sub-TBoxes) and MIPS (minimal incoherence-preserving sub-TBoxes) [12]. Such structures are defined by following an extension of the standard $\mathcal{ALC}$-tableau [2] applied to unfoldable $\mathcal{ALC}$. This algorithm is referred by the authors as *axiom pinpointing*. Next we introduce the intuitions to calculate MUPS, and afterwards extend them to propose an algorithm for recognizing defeaters of a given argument. We will rely upon unfolded $\mathcal{ALC}$ ontologies.

Unsatisfiability of a concept is detected with a labelled saturated tableau. A labelled tableau is a set of labelled branches. A labelled branch is a set of labelled formulas of the form $(a : C)^X$, where a is an individual name, C is a concept, and X is the label containing a set of axioms which leads to the inference of the formula $(a : C)$. A formula can occur with different labels on the same branch. A labelled tableau is saturated if all its branches are closed. A branch is closed if it contains a clash, *i.e.*, if there is at least one pair of formulas with contradictory atoms on the same individual. Hence, the information on which axioms are relevant for the closure (clash) of a branch is contained in the labels of contradictory formulas. For instance, a branch λ is closed if there is some pair $(a : A)^X \in \lambda$ and $(a : \neg A)^Y \in \lambda$, where A is an atomic concept. That is, axioms from X and Y lead to clashes, *i.e.*, $X \cup Y$ is unsatisfiable.

The closure of a branch is pursued by applying expansion rules which progressively unfold axioms in a lazy manner. An example of expansion rule over a branch λ is, If $(a : A)^X \in \lambda$ and $A \sqsubseteq C \in \Sigma$ then λ is replaced in the tableaux by $\lambda \cup \{(a : C)^{X \cup \{A \sqsubseteq C\}}\}$. Additional branches may be progressively included in the tableaux by

following a disjunctive rule which operates over formulas like $(a : C_1 \sqcap C_2)$. (For an account of the complete set of expansion rules, please refer to [12].) Once no more rules can be applied, a closed tableau is obtained through the set S of closed branches.

MUPS are constructed by building a labelled tableau for a branch initially containing only $(a : A)^{\emptyset}$, and by applying a *minimization function* φ which also starts in $(a : A)^{\emptyset}$. As expansion rules are applied to close the tableau, different rules also expand the minimization function. The idea is to obtain the smallest conjunction of axioms $\alpha_1 \wedge \ldots \wedge \alpha_n$, called *prime implicant*, implying φ. This is built from labels of contradictory formulae in each closed branch of the tableaux, hence $\alpha_i \in \Sigma$, for any $1 \leq i \leq n$. As φ is a minimization function every implicant of φ is also a minimization function. Finally, the prime implicant is also a minimization function. This means that A is unsatisfiable when $\alpha_1 \wedge \ldots \wedge \alpha_n$ is true, or equivalently A is unsatisfiable wrt. the set $M = \{\alpha_1, \ldots, \alpha_n\}$, which is minimal since it comes from a prime implicant (smallest conjunction of axioms implying φ). The MUPS for A wrt. Σ is $mups(A, \Sigma) = \{M \subseteq \Sigma | A$ is unsatisfiable in M but A is satisfiable in any $M' \subset M\}$.

To calculate the defeaters of a given argument $\mathcal{B}$, we first calculate $mups(A, \Sigma)$ where $A \sqsubseteq C \in \mathfrak{bd}(\mathcal{B})$. For any axiom $A' \sqsubseteq C' \in \mathfrak{bd}(\mathcal{B})$ which was not considered by the process to close the tableau (*i.e.*, not included in any label), a new MUPS $mups(A', \Sigma)$ should be obtained. (Only in the worst case a MUPS for every axiom within $\mathcal{B}$ should be constructed.) Once all the necessary MUPS are obtained, we join them into a set $mupsArg(\mathcal{B}, \Sigma) = \bigcup_{A \sqsubseteq C \in \mathfrak{bd}(\mathcal{B})} mups(A, \Sigma)$. Finally, for any $M \in mupsArg(\mathcal{B}, \Sigma)$, it holds $M \setminus \mathfrak{bd}(\mathcal{B})$ is the body of a minimal undercut of $\mathcal{B}$, conforming definitions 1, 3, and 4 (an mcu conforming Def. 5 can be easily obtained by accommodating the claim). In order to find inconsistencies beyond incoherencies, *i.e.*, to find also contrary membership assertions form the ABox, the original MUPS algorithm should consider additional expansion rules, thus turning MUPS from sub-terminologies to sub-ontologies. This assumption should not affect the following analysis.

Calculating MUPS relies on the construction of a minimization function from a tableau. Building it in a depth-first way allows to keep one single branch in memory at a time. Hence, the complexity class of the MUPS problem corresponds to that of the satisfiability checking in unfoldable $\mathcal{ALC}$, *i.e.*, PSPACE. Since the size of prime implicants may be exponential wrt. the number of axioms in the TBox, approximation methods could avoid the construction of fully saturated tableaux to reduce the size of the minimization functions. In addition, in order to render all the necessary defeaters of a given argument $\mathcal{B}$, the construction of several MUPS could be necessary. However, the unfolding process would find (in general) most of $\mathcal{B}$'s axioms.

Theorem 2. *Calculating all the defeaters of a given $\mathcal{ALC}$ argument is in PSPACE.*

5 Conclusions and Future Work

A general DL-argumentation machinery was proposed. This new ontology reasoner provides argumentation techniques to reason over inconsistent/ incoherent ontologies, and behaves as a classical ontology reasoner when considering consistent ontologies (Theorem 1). A similar $\mathcal{ALC}$-argumentation framework (without cqs support) was proposed in [13], where difficulties regarding negation of DL-axioms were addressed by

specifying specialized semantics enriched with $\neg$, $\wedge$, and $\vee$, to define defeaters. From our viewpoint, this would require to extend widely accepted $\mathcal{ALC}$-reasoning techniques with classic logic characteristics in order to build arguments in practice.

As far as classic DL-reasoning methodologies are reused to construct machineries for DL-argumentation, argumentation will provide a useful alternative to reason over inconsistent ontologies –the complexity of the argumentative reasoner will depend on that of the adopted DL-reasoner. Nonetheless, certain increase in the complexity should be presumed when working with huge dialectical trees, so establishing a relation to the "level of inconsistency" (number of contradictory axioms) of the queried ontology.

The complexity analysis of answering whether a query α is accepted by the DL-argumentation machinery requires to construct and mark the dialectical tree rooted in an α-supporter. In unfolded $\mathcal{ALC}$, this problem approaches to that of calculating MIPS from all MUPS. Moreover, since the number of defeaters for a complete dialectical tree, may grow exponentially in the number of axioms of the TBox, we believe that the construction of the tree in unfolded $\mathcal{ALC}$ would be at least in EXPTIME (as satisfiability in $\mathcal{ALC}$). A deep analysis on this matter is underway. Future work also pursues complete algorithms for $\mathcal{ALC}$ and most importantly, for efficient DL families as DL-Lite and $\mathcal{EL}$.

References

1. Baader, F.: Terminological Cycles in a Description Logic with Existential Restrictions. In: IJCAI. pp. 325–330 (2003)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): Description Logic Handbook: Theory, Implementation and Application. Cambridge University Press, Cambridge (2003)
3. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the Description Logic EL. In: Description Logics (2007)
4. Besnard, P., Hunter, A.: A Logic-based Theory of Deductive Arguments. Artif. Intell. 128(1-2), 203–235 (2001)
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite family. JAR 39(3), 385–429 (2007)
6. Cecchi, L., Fillottrani, P., Simari, G.: On the complexity of DeLP through game semantics. In: NMR. pp. 386–394 (2006)
7. Dung, P.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming and n-person Games. Artif. Intell. 77, 321–357 (1995)
8. Flouris, G., Huang, Z., Pan, J., Plexousakis, D., Wache, H.: Inconsistencies, Negations and Changes in Ontologies. In: AAAI. pp. 1295–1300 (2006)
9. García, A., Simari, G.: Defeasible Logic Programming: An Argumentative Approach. TPLP 4(1-2), 95–138 (2004)
10. Gómez, S., Chesñevar, C., Simari, G.: Reasoning with Inconsistent Ontologies through Argumentation. Applied Artificial Intelligence 24(1&2), 102–148 (2010)
11. Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding Maximally Satisfiable Terminologies for the Description Logic ALC. In: AAAI (2006)
12. Schlobach, S., Cornet, R.: Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In: IJCAI. pp. 355–362 (2003)
13. Zhang, X., Zhang, Z., Lin, Z.: An Argumentative Semantics for Paraconsistent Reasoning in Description Logic ALC. In: Description Logics (2009)