An argumentation workflow for reasoning in Ontology Based Data Access[1]

Bruno YUN [a], Madalina CROITORU [a,2]
[a] *INRIA Graphik/LIRMM University Montpellier, France*

Abstract. In this paper we demonstrate how to benefit from structured argumentation frameworks and their implementations to provide for reasoning capabilities of Ontology Based Data Access systems under inconsistency tolerant semantics. More precisely, given an inconsistent $Datalog^{\pm}$ knowledge base we instantiate it using the $ASPIC^{+}$ framework and show that the reasoning provided by $ASPIC^{+}$ is equivalent to the main inconsistent tolerant semantics in the literature. We provide a workflow that shows the practical interoperability of the logic based frameworks handling $Datalog^{\pm}$ and $ASPIC^{+}$.

Keywords. Applications and Structured Argumentation and Datalog+/-

Ontology Based Data Access and Inconsistency Handling

Ontology Based Data Access (OBDA) is a popular setting used by many Semantic Web applications that encodes the *access to data sources using an ontology (vocabulary)* [19, 20,10]. The use of the ontology will help obtain a unified view over heterogeneous data sources. Moreover, the ontology will enable the exploitation of implicit knowledge not explicitly stored in the data sources alone.

One of the main difficulties in OBDA consists in dealing with potentially inconsistent union of facts (data sources). Reasoning with inconsistency needs additional mechanisms because classical logic will infer everything out of *falsum*. It is classically assumed (and a hypothesis that we will also follow in this paper) that the inconsistency in OBDA occurs at the fact level and not due to the ontology [19,20,10]. The *facts are error prone* due to their unrestrained provenance while *ontologies are considered agreed upon* as shared conceptualisations.

We consider here two main methods of handling inconsistency. On one hand (and inspired from database research) we consider *repair based techniqu*es. A repair is a maximally consistent set of facts. Reasoning with inconsistency using repairs relies on reasoning with repairs and combining the results using various methods (called *inconsistency tolerant semantics*). [7,8,16] Despite them being the the mainstream techniques for OBDA reasoning, the main drawback of inconsistent tolerant semantics is the *lack of implementations available* today.

[1] Paper submitted to the application track
[2] Corresponding Author: E-mail: croitoru@lirmm.fr

A second method consists of using argumentation techniques. A Dung argumentation system [15] is a pair $\mathcal{AS} = (\mathcal{A}, \mathcal{C})$, where $\mathcal{A}$ is a set of arguments and $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{A}$ is a binary attack relation on them. We say that $a \in \mathcal{A}$ is *acceptable* w.r.t a set of arguments $\varepsilon \subseteq \mathcal{A}$ iff $\forall b \in \mathcal{A}$ such that $(b, a) \in \mathcal{C}, \exists c \in \varepsilon$ such that $(c, b) \in \mathcal{C}$. ε is *conflict-free* iff $\nexists a, b \in \varepsilon$ such that $(a, b) \in \mathcal{C}$. ε is *admissible* iff ε is conflict-free and all arguments of ε are acceptable w.r.t ε. ε is preferred iff it is maximal (for set inclusion) and admissible. ε is *stable* iff it is conflict-free and $\forall a \in \mathcal{A} \backslash \varepsilon, \exists b \in \varepsilon$ such that $(b, a) \in \mathcal{C}$. ε is *complete* iff it contains all arguments that are acceptable w.r.t ε. ε is *grounded* iff it is minimal (for set inclusion) and complete. Reasoning takes place on the various ε (also called extensions).

In this paper we demonstrate how to benefit from structured argumentation frameworks and their implementations to provide for reasoning capabilities of OBDA systems under inconsistency tolerant semantics. More precisely, given an inconsistent $Datalog^{\pm}$ knowledge base we instantiate it using the $ASPIC^{+}$ framework and show that the reasoning provided by $ASPIC^{+}$ is equivalent to the main inconsistent tolerant semantics in the literature.

The *significance of the paper* is proving the practical application of reasoning under inconsistency using argumentation for OBDA inconsistency tolerant semantics in $Datalog^{\pm}$ knowledge bases.

This practical proof is based on the following *new technical results*:

- The first instantiation of the $ASPIC^{+}$ framework using the $Datalog^{\pm}$ language.
- The proof of the soundness and completeness of this instantiation with respect to inconsistency tolerant semantics.

The paper organised as follows. We introduce the $Datalog^{\pm}$ language and the main inconsistency tolerant semantics: IAR, AR, ICR. We then show the $ASPIC^{+}$ instantiation and prove its properties with respect to the state of the art. Last we demonstrate the workflow that allows to use $ASPIC^{+}$ as a reasoning engine over inconsistent $Datalog^{\pm}$ knowledge bases.

The Logical Language: $Datalog^{\pm}$

In this section we explain the logical language $Datalog^{\pm}$ used throughout the paper. We define the notion of $Datalog^{\pm}$ knowledge base, inconsistent knowledge base and explain the three inconsistency tolerant semantics mostly used in the literature.

In OBDA there are two major approaches to represent an ontology: Description Logics (such as $\mathcal{EL}$ [3] and DL-Lite families [11]) and rule-based languages (such as $Datalog^{\pm}$ language [9], a generalization of Datalog that allows for existentially quantified variables in rules heads). Despite $Datalog^{\pm}$ undecidability when answering conjunctive queries, different decidable fragments are studied in the literature [6]. These fragments generalize the aforementioned Description Logics families and overcome their limitations by allowing any predicate arity as well as cyclic structures.

Here we use the general rule-based setting knowledge representation language $Datalog^{\pm}$, i.e, *the positive existential* conjunctive fragment of first-order logic FOL [12,5]. Its language $\mathcal{L}$ is composed of formulas built with the usual quantifier $(\exists, \forall)$ and *only* the connectors implication $(\rightarrow)$ and conjunction $(\wedge)$. We consider first-order vocab-

ularies with constants but no other function symbol. A vocabulary is a pair $\mathcal{V} = (\mathcal{P}, \mathcal{C})$, where $\mathcal{P}$ is a finite set of predicates and $\mathcal{C}$ is a possibly infinite set of constants. A term t over $\mathcal{V}$ is a constant or a variable, different constants represent different values (unique name assumption). We use uppercase letters for constants and lowercase letters for variables. An **atomic formula** (or atom) over $\mathcal{V}$ is of the form $p(t_1, ..., t_n)$ where $p \in \mathcal{P}$ is an n-ary predicate, and $t_1, ..., t_n$ are terms. A **ground atom** is an atom with no variables. A conjunction of atoms is called a **conjunct**. A conjunction of **ground atoms** is called a **ground conjunct**. By convention a ground atom is a ground conjunct. A variable in a formula is free if it is not in the scope of any quantifier. A formula is **closed** if it has no free variables (also known as **sentence**). Classically, a fact is a ground atom. The notion was extended in [5], so that a fact may contain existentially quantified variables and not only constants. Thus, a **fact** on $\mathcal{V}$ is the existential closure of a conjunction of atoms over $\mathcal{V}$. We denote by *terms*(F) (resp. *vars*(F)) the set of terms (resp. variables) that occur in F. We exclude duplicate atoms in facts, which allows to see a fact as a set of atoms. We denote by $\vec{x}$ a vector of variables. An *existential rule* (or simply a rule) is a closed formula of the form $R = \forall \vec{x} \forall \vec{y}(B \rightarrow \exists \vec{z} H)$, where B and H are conjuncts, with *vars*$(B) = \vec{x} \cup \vec{y}$, and *vars*$(H) = \vec{x} \cup \vec{z}$. The variables $\vec{z}$ are called the existential variables of the rule R. B and H are respectively called the *body* and the *head* of R. We denote them respectively $body(R)$ for B and $head(R)$ for H. We may sometimes omit quantifiers and write $R = B \rightarrow H$. A *negative constraint* (or simply a constraint) is a rule of the form $N = \forall \vec{x}(B \rightarrow \bot)$. Given a set of variables $\mathcal{X}$ and a set of terms $\mathcal{T}$, a **substitution** σ of $\mathcal{X}$ by $\mathcal{T}$ (notation $\sigma : \mathcal{X} \rightarrow \mathcal{T}$) is a mapping from $\mathcal{X}$ to $\mathcal{T}$. Given a fact F, $\sigma(F)$ denotes the fact obtained from F by replacing each occurrence of $x \in X \cap vars(F)$ by $\sigma(x)$. A **homomorphism** from a fact F to a fact F' is a substitution σ of $vars(F)$ by (a subset of) $terms(F')$ such that $\sigma(F) \subseteq F'$ [5].

A rule $R = B \rightarrow H$ is **applicable** to a fact F if there is a homomorphism σ from B to F. The *application of R to F w.r.t. σ* produces a fact $\alpha(F, R, \sigma) = F \cup \sigma(safe(H))$, where *safe*$(H)$ is obtained from H by replacing existential variables with fresh variables (not used variables). $\alpha(F, R, \sigma)$ is said to be an **immediate derivation** from F. Let F be a fact and $\mathcal{R}$ be a set of rules. A fact F' is called an **$\mathcal{R}$-derivation** of F if there is a finite sequence (called the **derivation sequence**) $(F_0 = F, ..., F_n = F')$ such that for all $0 \leq i < n$ there is a rule R which is applicable to F_i and F_{i+1} is an immediate derivation from F_i. Given a fact F and a set of rules $\mathcal{R}$, the **chase** (or saturation) procedure starts from F and performs rule applications in a breadth-first manner. The chase computes the **closure** of F, i.e. $CL_{\mathcal{R}}(F)$, which is the smallest set that contains F and that is closed under R-derivation, i.e. for every $\mathcal{R}$-derivation F' of F we have $F' \in CL_{\mathcal{R}}(F)$. Given a chase variant C [4], we call C-finite the class of set of rules $\mathcal{R}$, such that the C-chase halts on any fact F, consequently produces a finite $CL_{\mathcal{R}}(F)$. We limit our work in this paper to these kind of classes.

Let F and F' be two facts. $F \models F'$ if and only if there is a homomorphism from F' to F. Given two facts F and F' and a set of rules $\mathcal{R}$ we say $F, \mathcal{R} \models F'$ if and only if $CL_{\mathcal{R}}(F) \models F'$ where $\models$ is the classical first-order entailment [18].

Example 1 ($\mathcal{R}$-derivation) *For readability of this example only, we will use simple notation for predicates names. Let $F = \{q(A, B), r(D), p(x_1, C)\}$ and $\mathcal{R} = \{R_1, R_2\}$ such that $R_1 = q(x_1, y_1) \wedge r(z_1) \rightarrow d(x_1, z_1)$ and $R_2 = p(x_2, y_2) \wedge r(z_2) \rightarrow m(z_2, x_2)$. The following is a derivation sequence: $\langle F_0, F_1, F_2 \rangle$ where $F_0 = F$, $F_1 =$*

$\{q(A,B), r(D), d(A,D), p(x_1,C)\}$ *and* $F_2 = F_1 \cup \{m(D,x_1)\}$. *We get* F_1 *by applying* $\mathcal{R}_1$ *on* F *then we get* F_2 *by applying* R_2 *on* F_1. *We say* F_2 *is an* $\mathcal{R}$*-derivation of* F. *The closure of* F *is* $CL_{\mathcal{R}}(F) = F \cup \{d(A,D), m(D,x_1)\}$.

Knowledge base and inconsistency Let us denote by $\mathcal{L}$ the language described so far, A knowledge base $\mathcal{K}$ is a finite subset of $\mathcal{L}$. Precisely, $\mathcal{K}$ is a tuple $(\mathcal{F}, \mathcal{R}, \mathcal{N})$ of a finite set of facts $\mathcal{F}$, rules $\mathcal{R}$ and constrains $\mathcal{N}$. Saying that $\mathcal{K} \models F$ means $CL_{\mathcal{R}}(\mathcal{F}) \models F$. We say a set of facts $\mathcal{F}$ is $\mathcal{R}$-inconsistent with respect to a set of constraints $\mathcal{N}$ and rules $\mathcal{R}$ if and only if there exists $N \in \mathcal{N}$ such that $CL_{\mathcal{R}}(\mathcal{F}) \models body(N)$, otherwise $\mathcal{F}$ is $\mathcal{R}$-consistent. A knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ is said to be inconsistent with respect to $\mathcal{R}$ and $\mathcal{N}$ (inconsistent for short) if $\mathcal{F}$ is $\mathcal{R}$-inconsistent. We may use the notation $CL_{\mathcal{R}}(\mathcal{F}) \models \bot$ to mean the same thing.

In the area of inconsistent ontological knowledge base query answering, we usually check what can be inferred from an inconsistent ontology. We usually begin by calculating all maximal consistent subsets of $\mathcal{K}$ called *repairs*. Given a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$, we call by $Repairs(\mathcal{K})$ the set of all repairs defined as:

$$Repairs(\mathcal{K}) = \{\mathcal{F}' \subseteq \mathcal{F} | \mathcal{F}' \text{ is maximal for } \subseteq \text{ and } \mathcal{R}\text{-consistent}\}$$

Different inconsistency tolerant semantics are used for inconsistent ontology knowledge base query answering (Intersection of All Repairs: IAR, All Repairs: AR, Intersection of Closed Repairs: ICR). It is good to notice that these semantics can yield different results.

Definition 1 ***[cf [14]].*** *Let* $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ *be a knowledge base and* α *be a query.*

- α *is* AR-entailed *from* $\mathcal{K}$, *written* $\mathcal{K} \models_{AR} \alpha$ *iff for* $\forall r \in Repairs(\mathcal{K}), Cl_{\mathcal{R}}(r) \models \alpha$.
- α *is* ICR-entailed *from* $\mathcal{K}$, *written* $\mathcal{K} \models_{ICR} \alpha$ *iff* $\bigcap_{r \in Repairs(\mathcal{K})} Cl_{\mathcal{R}}(r) \models \alpha$
- α *is* IAR-entailed *from* $\mathcal{K}$, *written* $\mathcal{K} \models_{IAR} \alpha$ *iff* $CL_{\mathcal{R}}(\bigcap_{r \in Repairs(\mathcal{K})} r) \models \alpha$

Structured Argumentation for $Datalog^{\pm}$

In this section we address the problem of how to use structured argumentation for $Datalog^{\pm}$. We show how the $ASPIC^+$ framework can be instantiated to yield results equivalent to the state of the art in OBDA inconsistency tolerant semantics. This will help us establish the interoperability workflow where the $ASPIC^+$ engine is used to compute knowledge base queries following inconsistent tolerant semantics. The section is structured as follows. We define the first instantiation in the literature of $ASPIC^+$ using $Datalog^{\pm}$. We then prove that this instantiation yields equivalent results to the state of the art (namely to existing instantiations for $Datalog^{\pm}$ that do not follow the $ASPIC^+$ framework and, respectively, to inconsistency tolerant semantics).

$ASPIC^+$ *Instantiation For* $Datalog^{\pm}$

$ASPIC^+$ [17] is a framework for obtaining logical based argumentation system using any logical language $\mathcal{L}$. It is meant to generate an abstract argumentation framework and

was created because abstract argumentation does not specify the structure of arguments and the nature of attacks. $ASPIC^+$ is meant to provide guidance to those aspects without losing a large range of instantiating logics. Following [17], to use $ASPIC^+$, we need to choose a logical language $\mathcal{L}$ closed under negation ($\neg$), provide a set of rules $\mathcal{R} = \mathcal{R}_d \cup \mathcal{R}_s$ composed of defeasible rules and strict rules with $\mathcal{R}_d \cap \mathcal{R}_s = \emptyset$, specify a contrariness function $cf : \mathcal{L} \to 2^{\mathcal{L}}$ and a partial naming function $n : \mathcal{R}_d \to \mathcal{L}$ that associates a well-formed formulas of $\mathcal{L}$ to a defeasible rule. The function n will not be used in this instantiation. In $ASPIC^+$, an argumentation system is a triple $\mathcal{AS} = (\mathcal{L}, \mathcal{R}, n)$ and a knowledge base is $\mathcal{K} \subseteq \mathcal{L}$ consisting of two disjoint subsets $\mathcal{K}_n$ (the axioms) and $\mathcal{K}_p$ (the ordinary premises).

To instantiate $ASPIC^+$ for $Datalog^{\pm}$, we define $\mathcal{L}$ as $Datalog^{\pm}$, rules in definition 2 and the contrariness function in definition 3. Please note that definition 4 and 7 are new w.r.t state of art regarding $Datalog^{\pm}$ instantiations conform [17].

Definition 2 (Rules in $ASPIC^+$ - modified version of [17]) *Strict rules (resp. defeasible rules) are of the form $\forall\vec{x}\forall\vec{y}(B \to \exists\vec{z}H)$ (resp. $\forall\vec{x}\forall\vec{y}(B \Rightarrow \exists\vec{z}H)$) with B, the body and H, the head are atoms or conjunction of atoms with* vars$(B) = \vec{x} \cup \vec{y}$, *and* vars$(H) = \vec{x} \cup \vec{z}$..

Definition 3 ***[Contrariness function [17]].*** *cf is a function from $\mathcal{L}$ to $2^{\mathcal{L}}$ such that:*

- *φ is the contrary of ψ if $\varphi \in cf(\psi), \psi \notin cf(\varphi)$*
- *φ is the contradictory of ψ if $\varphi \in cf(\psi), \psi \in cf(\varphi)$*
- *Each $\varphi \in \mathcal{L}$ has at least one contradictory.*

We define our own contrariness function to instantiate $ASPIC^+$ for $Datalog^{\pm}$ ($\mathcal{L} = Datalog^{\pm}$). This contrariness function is necessary because it is used in the attack relation. It is worth noting that idea that we want to capture (as also defined in [1]) is that x is the contrary of y iff they cannot be both true but they can be both false. They are contradictory if the truth of one implies the falsity of the other and vise versa.

Definition 4 ($Datalog^{\pm}$ contrariness function) *Let $a \in \mathcal{L}$ and b be an atom or a conjunction of atoms. $b \in cf(a)$ iff $\exists\psi$ an atom such that $a \models \psi$ and $\{b, \psi\}$ is $\mathcal{R}$-inconsistent.*

Here we recall that an $ASPIC^+$ argument can be built from axioms and ordinary premises or from rules and other arguments. The arguments are built once $\mathcal{R}_d, \mathcal{R}_s, cf$ and $\mathcal{K}$ are known.

Definition 5 (Argument cf [17]) *Arguments in $ASPIC^+$ can be in two forms:*

- *$\emptyset \to c$ (resp. $\emptyset \Rightarrow c$) with $c \in \mathcal{K}_n$ (resp. $c \in \mathcal{K}_p$ or $\emptyset \Rightarrow c \in \mathcal{R}_d$) such that $Prem(A) = \{c\}, Conc(A) = c,$ Sub$(A) = \{A\}$ with $Prem$ returns premisses of A and $Conc$ returns its conclusion.*
  $DefRules(A) = \emptyset$.
- *$A_1, ..., A_m \to c$ (resp. $A_1, ..., A_m \Rightarrow c$), such that there exists a strict (resp. defeasible) rule $r = B \to H$ (resp. $r = B \Rightarrow H$) and a homomorphism σ from B to $X = Conc(A_1) \wedge Conc(A_2) \wedge \cdots \wedge Conc(A_m)$.*
  $Prem(A) = Prem(A_1) \cup ... \cup Prem(A_m)$,

$Conc(A) = c = \alpha(X, r, \sigma)$,
Sub(A) = *Sub*$(A_1) \cup ... \cup$ *Sub*$(A_m) \cup \{A\}$,
$TopRule(A) =$ *rule* $r = B \rightarrow H$ *(resp.* $r = B \Rightarrow H$*), such that there exists an homomorphism* σ *from* B *to* X.
$DefRules(A) = DefRules(A_1) \cup \cdots \cup DefRules(A_m)$ *(resp.* $DefRules(A) = DefRules(A_1) \cup \cdots \cup DefRules(A_m) \cup \{TopRule(A)\}$*).*

Attacks in $ASPIC^+$ are based on three notions (undercutting, undermining and rebutting). Each of those notions are useful as they capture different aspects of conflicts. In short, arguments can be attacked on a conclusion of a defeasible inference (rebutting attack), on a defeasible inference step itself (undercutting attack), or on an ordinary premise (undermining attack).

Definition 6 ***[cf [17]].*** *Let* a *and* b *be arguments, we say that* a *attacks* b *iff* a *undercuts, undermines or rebuts* b*, where:*

- a *undercuts argument* b *(on* b'*) iff* $Conc(a) \in cf(n(r))$ *for some* $b' \in Sub(b)$ *such that* b'*'s top rule* r *is defeasible.*
- a *rebuts argument* b *(on* b'*) iff* $Conc(a) \in cf(\psi)$ *for some* $b' \in Sub(b)$ *of the form* $b'_0, \ldots, b'_n \Rightarrow \psi$.
- a *undermines* b *(on* ψ*) iff* $Conc(a) \in cf(\psi)$ *for an ordinary premise* ψ *of* b.

We are now ready to define the mapping that allows the instantiation of $ASPIC^+$ with $Datalog^{\pm}$. The mapping will consider each fact as a defeasible rule because the inconsistency in the OBDA setting is assumed to come from the facts level. Therefore the only attack we consider in this instantiation is the undermine attack because we have simple defeasable rules. The rules of the ontology become strict rules.

Definition 7 (Mapping For $ASPIC^+$ **Instantiation of** $Datalog^{\pm}$ **)** *We denote by* $\mathcal{S}$ *the set of all possible inconsistent knowledge bases of the form* $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ *and* $\mathcal{G}$ *the set of all* $ASPIC^+$ *instantiation using* $Datalog^{\pm}$ *language. The mapping* $\tau : S \rightarrow \mathcal{G}$ *is defined as follows:*

1. *The mapping* τ *associates every* $\mathcal{R}$*-consistent subset* $F_i \subseteq \mathcal{F}$ *to its defeasible rule* $\emptyset \Rightarrow conjunct(F_i)$ *where* $conjunct(F_i)$ *denotes the conjunction of facts contained in* F_i.
2. *The mapping* τ *associates every rules* $r_i \in \mathcal{R}$ *to the same rule* $r_i \in \mathcal{R}_s$.

We will considerate that if $\emptyset \Rightarrow c$*, then* c *is an ordinary premise* $(c \in \mathcal{K}_p)$.

Properties of the $Datalog^{\pm}$ $ASPIC^+$ *Instantiation*

In order to give properties of the $ASPIC^+$ instantiation presented in this paper we remind few notions. ε is *admissible* iff ε is conflict-free and all arguments of ε are acceptable w.r.t ε. ε is preferred iff it is maximal (for set inclusion) and admissible. ε is *stable* iff it is conflict-free and $\forall a \in \mathcal{A} \backslash \varepsilon, \exists b \in \varepsilon$ such that $(b, a) \in \mathcal{C}$.

We denote by $AF^A_{\mathcal{K}}$ the $ASPIC^+$ argumentation framework constructed from $\mathcal{K}$ using the mapping of definition 7. We restate that attacks in $AF^A_{\mathcal{K}}$ are composed only of undermining because we only have simple defeasible rules of the form $\emptyset \Rightarrow c$. The following lemma shows that stable extensions are closed under sub-arguments in the $Datalog^{\pm}$ instantiation of $ASPIC^+$.

Lemma 1 *Let ε be an $ASPIC^+$ stable extension and $A \in \varepsilon$ an argument contained in ε. Then $Sub(A) \subseteq \varepsilon$.*

Proof By means of contradiction we suppose the contrary, i.e. there is an argument $A_1 \in Sub(A)$ that does not belong to ε. Since ε is stable, then there is an argument $B \in \varepsilon$ such that B undermines A_1 on $\phi \in Prem(A_1)$. Therefore, B undermines A since $Prem(A_1) \subseteq Prem(A)$, contradiction. Consequently, $Sub(A) \subseteq \varepsilon$.

Notation Let $c = \alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n$ be a conjunction of facts. $Elimination(c) = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is the set resulting from eliminating the conjunction of c. Let S be a set of facts. We denote by $\mathcal{P}(S)$ the superset of S which correspond to all subsets of S.

We can now define the set of arguments constructed on a consistent set of facts.

Definition 8 (new definition) *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and $AF_{\mathcal{K}}^{A}$ be the corresponding $ASPIC^+$ instantiation and $S \subseteq \mathcal{F}$ a $\mathcal{R}$-consistent subset of $\mathcal{F}$. We denote by $Arg^A(S)$ the set of arguments such that their premises are contained in S.*

$$Arg^A(S) = \{ASPIC^+ \textit{ argument } a | \bigcup_{c \in Prem(a)} Elimination(c) \subseteq \mathcal{P}(S)\}$$

The main result shows that the set of stable extension coincides with the set of preferred one and it is obtained from the arguments built on repairs. This result is important as it will allow to formally underpin the interoperability workflow presented in the next section.

Theorem 1 (Repair Equivalence for $ASPIC^+$ Instantiation) *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, $AF_{\mathcal{K}}^{A}$ be the corresponding $ASPIC^+$ instantiation and $\sigma \in \{preferred, stable\}$. Then:*

$$\{Arg^A(R) | R \in Repair(\mathcal{K})\} = Ext_{\sigma}(AF_{\mathcal{K}}^{A})$$

Proof The plan of the proof is as follows:

1. We prove that $\{Arg^A(R) | R \in Repair(\mathcal{K})\} \subseteq Ext_{stable}(AF_{\mathcal{K}}^{A})$.
2. We prove that $Ext_{preferred}(AF_{\mathcal{K}}^{A}) \subseteq \{Arg^A(R) | R \in Repair(\mathcal{K})\}$.
3. Since every stable extension is a preferred one, we can proceed as follows. From the first item, we have that $\{Arg^A(R) | R \in Repair(\mathcal{K})\} \subseteq Ext_{stable}(AF_{\mathcal{K}}^{A})$, thus the theorem holds for preferred semantics. From the second item we have that $Ext_{preferred}(AF_{\mathcal{K}}^{A}) \subseteq \{Arg^A(R) | R \in Repair(\mathcal{K})\}$, thus the theorem holds for stable semantics.

1. We show that $\{Arg^A(R) | R \in Repair(\mathcal{K})\} \subseteq Ext_{stable}(AF_{\mathcal{K}}^{A})$. Let $R \in Repair(\mathcal{K})$ and let $\varepsilon = Arg^A(R)$. We first prove that ε is a conflict-free extension. Aiming to a contradiction, let $A, B \in \varepsilon$ and A attacks B. From the definition of undermining, $Conc(A) \in cf(\phi)$ and $\phi \in Prem(B)$. From the definition of the contrariness function, $\exists \psi$ atom of ϕ such that $\{Conc(A), \psi\}$ is $\mathcal{R}$-inconsistent. Thus, $Prem(B) \cup \{Conc(A)\}$ is $\mathcal{R}$-inconsistent. Consequently R is $\mathcal{R}$-inconsistent, contradiction. Therefore, ε is conflict-free.

Let us now prove that ε is a stable extension. Let $B \in Arg^A(\mathcal{F}) \backslash Arg^A(R)$ and $\psi \in Prem(B)$ such that $\psi \notin R$ (such ψ exists otherwise $B \in Arg^A(R)$). Let $c = conjunct(R)$ be the conjunction of elements of R and let $A : \emptyset \Rightarrow c$ ($A \in Arg^A(R)$). We have $\psi \notin R$, so, due to the set inclusion maximality for the repairs, $\{c, \psi\}$ is $\mathcal{R}$-inconsistent. Therefore, $c \in cf(\psi)$ and A undermines B. Consequently, ε is a stable extension.

2. We show that $Ext_{preferred}(AF^A_{\mathcal{K}}) \subseteq \{Arg^A(R) | R \in Repair(\mathcal{K})\}$. Let $\varepsilon \in Ext_{preferred}(AF^A_{\mathcal{K}})$. Let $\varepsilon \in Ext_{preferred}(AF^A_{\mathcal{K}})$ and let us prove that there exists a repair R such that $\varepsilon = Arg^A(R)$. Let $S = \bigcup_{a \in \varepsilon} Prem(a)$. Let us prove that S is $\mathcal{R}$-consistent. Aiming to a contradiction, suppose that S is $\mathcal{R}$-inconsistent. Let $S' \subseteq S$ be such that S' is $\mathcal{R}$-inconsistent and every proper set of S' is $\mathcal{R}$-consistent. Let us denote $S' = \{\phi_1, \phi_2, \ldots, \phi_n\}$. Let $A \in \varepsilon$ be an argument such that $\phi_n \in Prem(A)$. From Lemma 1, there is an argument $A_1 \in Sub(A)$ such that $A_1 : \emptyset \Rightarrow \phi_n$ and $A_1 \in \varepsilon$. Let $c' = conjunct(S' \backslash \{\phi_n\})$ and let $A' : \emptyset \Rightarrow c'$. We have that $c' \in cf(\phi_n)$, thus A' attacks A_1. Since ε is conflict-free, then $A' \notin \varepsilon$. Since ε is an admissible set, there exists $B \in \varepsilon$ such that B attacks A'. Since B attacks A' then $Conc(B) \in cf(c')$ and there exists $i \in \{1, 2, \ldots, n-1\}$ such that $\{Conc(B), \phi_i\}$ is $\mathcal{R}$-inconsistent. Since $\phi_i \in S$, then there exists $C \in \varepsilon$ such that $\phi_i \in Prem(C)$. Thus, using Lemma 1, there exists $C' \in Sub(C)$ and $C' : \emptyset \Rightarrow \phi_i$. Therefore $Conc(B) \in cf(\phi_i)$ and B attacks C', contradiction. So it must be that S is $\mathcal{R}$-consistent.
   Let us now prove that S is maximal, i.e. there exists no $S' \subseteq \mathcal{F}$ such that $S \subset S'$ and S' is $\mathcal{R}$-consistent. We use the proof by contradiction. Thus, suppose that S is not a maximal $\mathcal{R}$-consistent subset of $\mathcal{F}$. Then, there exists $S' \in Repair(\mathcal{K})$, such that $S \subset S'$. We have that $\varepsilon \subseteq Arg^A(S)$. Denote $\varepsilon' = Arg^A(S')$. Since $S \subset S'$ then $Arg^A(S) \subset \varepsilon'$. Thus, $\varepsilon \subset \varepsilon'$. From the first part of the proof, $\varepsilon' \in Ext_{stable}(AF^A_{\mathcal{K}})$. Consequently, $\varepsilon' \in Ext_{preferred}(AF^A_{\mathcal{K}})$. We also know that $\varepsilon \in Ext_{preferred}(AF^A_{\mathcal{K}})$. Contradiction, since no preferred set can be a proper subset of another preferred set. Thus, we conclude that $S \in Repair(K)$. Let us show that $\varepsilon = Arg^A(S)$. It must be that $\varepsilon \subseteq Arg^A(S)$. Also, we know (from the first part) that $Arg^A(S)$ is a stable and a preferred extension, thus the case $\varepsilon \subset Arg^A(S)$ is not possible.
3. Now we know that $\{Arg^A(R) | R \in Repair(\mathcal{K})\} \subseteq Ext_{stable}(AF^A_{\mathcal{K}})$ and $Ext_{preferred}(AF^A_{\mathcal{K}}) \subseteq \{Arg^A(R) | R \in Repair(\mathcal{K})\}$. The theorem follows from those two facts, as explained at the beginning of the proof.

Let us highlight that the state of the art can also provide a structured argumentation framework of $Datalog^{\pm}$ [14,13]. The authors define the notion of argument and attack and prove that the argumentation framework thus obtained yields the same extensions as the repairs of the inconsistent knowledge base that has been used to build the arguments and the attacks.

Definition 9 (Argument cf [14,13]) *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, an argument a as described in [13,14] is a tuple $a = (\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_n)$ where:*

- *$\mathcal{F}_0 \subseteq \mathcal{F}$ is $\mathcal{R}$-consistent and $(\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_{n-1})$ is a derivation sequence.*
- *$\mathcal{F}_n$ is an atom, a conjunction of atoms, the existential closure of an atom or the existential closure of a conjunction of atoms such that $\mathcal{F}_{n-1} \models \mathcal{F}_n$*

We denote by $Supp(a) = \mathcal{F}_0$ the support of an argument, $Conc(a) = \mathcal{F}_n$ its conclusion and $\forall X \subseteq \mathcal{F}, Arg(X)$ is the set of all arguments a such that $Supp(a) \subseteq X$.

The attack relation between arguments is defined below and is not symmetric (following [2]).

Definition 10 (Attack cf [14,13]) *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and let a and b be two arguments. We say that an argument a attacks an argument b denoted by $(a, b) \in Att$ iff $\exists \varphi \in Supp(b)$ such that $\{Conc(a), \varphi\}$ is $\mathcal{R}$-inconsistent.*

This attack relation is not symmetric. Please note that it have been showed that symmetric attack relations violate some desirable properties [2].

Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, we denote by $AF_{\mathcal{K}}^{M}$ the instantiated logical argumentation framework $(\mathcal{A}, \mathcal{C})$ with $\mathcal{A} = Arg(\mathcal{F})$ and $\mathcal{C}$ defined in definition 10. Conform [14] the arguments constructed on the set of repairs coincide with the arguments in the stable and preferred extension: $\{Arg(R) | R \in Repair(\mathcal{K})\} = Ext_{\sigma}(AF_{\mathcal{K}}^{M})$.

We can thus conclude that the preferred/stable extensions in the two instantiated frameworks are the same and that for each stable/preferred extension of one framework, there is a corresponding stable/preferred extension in the other and vice-versa. This is formalised int he theorem below.

Theorem 2 (Instantiations Equivalence) *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, $AF_{\mathcal{K}}^{M}$ and $AF_{\mathcal{K}}^{A}$ be the two argumentation framework instantiations. Then if $\sigma \in \{preferred, stable\}$, $|Ext_{\sigma}(AF_{\mathcal{K}}^{M})| = |Ext_{\sigma}(AF_{\mathcal{K}}^{A})|$ and for each extension under semantics $\sigma, \varepsilon \in Ext_{\sigma}(AF_{\mathcal{K}}^{M})$, there is a corresponding extension $\varepsilon_2 \in Ext_{\sigma}(AF_{\mathcal{K}}^{A})$ and vice-versa (the corresponding extension can be found via repairs).*

We have thus proved that the $ASPIC^{+}$ instantiation presented in the paper yields the same semantic results as the repair based techniques. Therefore one can use the $ASPIC^{+}$ reasoning engine in order to provide reasoning techniques for OBDA in inconsistent $Datalog^{\pm}$ knowledge bases. This is explained in the next section.

Workflow for $ASPIC^{+}$ $Datalog^{\pm}$ Instantiation in OBDA

The *significance of this work* is that the proposed workflow it will enable $Datalog^{\pm}$ frameworks to handle inconsistencies in knowledge bases by means of the $ASPIC^{+}$ framework. Here we use two frameworks:

- **Graal**, a Java toolkit dedicated to querying knowledge bases within the framework of $Datalog^{\pm}$ and maintained by GraphIK team. Graal takes as input a Dlgp file and a query and answer the query using various means (saturation, query rewriting). This toolkit can be found at `https://graphik-team.github.io/graal/`.
- **ACL's ASPIC** project that takes as input a query and $ASPIC^{+}$ knowledge base, i.e. rules (strict and defeasible), ordinary premises, axioms and preferences. The output is the answer to the query. This inference engine can be found at `http://aspic.cossac.org/components.html`.

We use Graal's representation of a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ and construct the necessary input for the $ASPIC^+$ argumentation inference engine. The difficulty of this work resides in the definition of the contrariness function that ensures the semantic equivalence proved in the previous section.

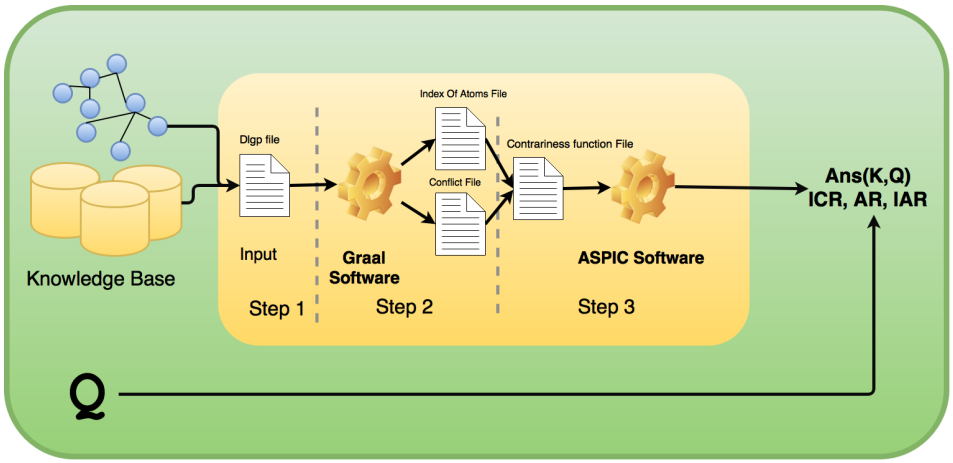

Figure 1. Interoperability Workflow of ACL and Graal.

In Figure 1 the interoperability workflow of the Graal software and the $ASPIC^+$ implementation are shown. Let us detail here how the workflow functions:

- **Step 1**. The input of the software is a $Datalog^{\pm}$ knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ obtained from the OBDA setting (that considers several data sources unified under the same ontology). The *dlgp* file that encodes this knowledge base (a textual format for the existential rule / Datalog framework) is parsed by the Graal framework. Each line in a *dlgp* file corresponds either to a fact, existential rule, negative constraint or conjunctive query. Please note that a complete grammar of the *dlgp* format is available here: `https://graphik-team.github.io/graal/papers/datalog+_v2.0_en.pdf`.
- **Step 2**. The intermediary files are meant to serve as input for the contrariness function computation: *Index Of Atoms* and *Conflict File*. The *Index Of Atoms* contains an index of all facts that can be obtained. The *Conflict File* contains the conflicts between the atoms and represent the constraints expressed by the negative constraints. The first line of this file represents the number of conflicts and each line describes the atoms participating in the conflict.

  Let us suppose that the knowledge base obtained from the parser is $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ where $\mathcal{F} = \{mammal(mouse), peripheral(mouse)\}$, $\mathcal{R} = \{\forall x(mammal(x) \rightarrow living(x))\}$ and $\mathcal{N} = \{\forall x\,(\,living(x) \wedge peripheral(x) \rightarrow \bot)\}$. The corresponding *Index Of Atoms File* contains:
  0 mammal(mouse).
  1 peripheral(mouse).
  2 living(mouse).
  In this example, the fact $mammal(mouse)$ is represented with the number 0.

Similarly, the corresponding *Conflict File* contains:
2
2 1
1 0
In this example, there is two conflicts and the first one involves the facts $living(mouse)$ and $peripheral(mouse)$.

- **Step 3**. The *Contrariness Function File* is build from the *Index Of Atoms File* and the *Conflict File* and explicitly details the sets $cf(a)$ where a is conjunction contained of facts contained in a $\mathcal{R}$-consistent subset of $\mathcal{F}$. Using the same knowledge base, the *Contrariness Function File* contains:
(0) [(1)]
(1) [(2), (0)]
The first line express the idea that a conjunction that contains the atom $peripheral(mouse)$ is contained in $cf(mammal(mouse))$ and the second line that a conjunction in $cf(peripheral(mouse))$ contains either $mammal(mouse)$ or either $living(mouse)$ (or both). This *Contrariness Function File* is then passed to the inference engine which is instantiated using the method stated previously.
- **Querying**. The output of this inference engine is the answer to the query w.r.t the inconsistent knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$, i.e. *true* or *false* if the query is a boolean conjunctive query or all the substitutions of answer variables by constants in $\mathcal{F}$. The soundness and completeness of the answer with respect to inconsistency tolerant semantics is ensured by the equivalence results presented in the previous section.

While this workflow has not yet been implemented, this is solely due to temporary technical details in obtaining the source files of the two implementations and will not put foundational challenges as explained above.

Conclusions

In this paper we demonstrated how to benefit from structured argumentation frameworks and their implementations to provide for reasoning capabilities of OBDA systems under inconsistency tolerant semantics. More precisely, given an inconsistent $Datalog^{\pm}$ knowledge base we instantiated it using the $ASPIC^{+}$ framework and showed that the reasoning provided by $ASPIC^{+}$ is equivalent to the main inconsistent tolerant semantics in the literature. A workflow of interoperability between $ASPIC^{+}$ ACL framework and Graal $Datalog^{\pm}$ framework was also demonstrated.

The future work avenues opened by this work are two fold. First, one could remove the hypothesis of inconsistency solely due to facts and explore other semantics obtained naturally using argumentation instantiation. Second, the investigation of how preferences are handled in various systems could also be of great practical benefit (since it will allow to explicitly consider the provenance of facts).

Acknowledgments

The authors acknowledge the support of ANR grants ASPIQ (ANR-12-BS02-0003), QUALINCA (ANR-12-0012) and DUR-DUR (ANR-13-ALID-0002). The work of the

second author has been carried out part of the research delegation at INRA MISTEA Montpellier and INRA IATE CEPIA Axe 5 Montpellier.

References

[1] L. Amgoud. Five weaknesses of ASPIC +. In *Advances in Computational Intelligence - 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2012, Catania, Italy, July 9-13, 2012, Proceedings, Part III*, pages 122–131, 2012.

[2] L. Amgoud and P. Besnard. Bridging the gap between abstract argumentation systems and logic. In *Scalable Uncertainty Management, Third International Conference, SUM 2009, Washington, DC, USA, September 28-30, 2009. Proceedings*, pages 12–27, 2009.

[3] F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope. In *Proc. of IJCAI 2005*, 2005.

[4] J.-F. Baget, F. Garreau, M.-L. Mugnier, and S. Rocher. Revisiting Chase Termination for Existential Rules and their Extension to Nonmonotonic Negation. *ArXiv e-prints*, May 2014.

[5] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(10):1620 – 1654, 2011.

[6] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI'11*, pages 712–717, 2011.

[7] M. Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[8] M. Bienvenu and R. Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013.

[9] A. Calì, G. Gottlob, and T. Lukasiewicz. Datalog$+/-$: a unified approach to ontologies and integrity constraints. In *Proc of ICDT'09*, pages 14–30. ACM, 2009.

[10] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[12] M. Chein and M. Mugnier. *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.

[13] M. Croitoru, R. Thomopoulos, and S. Vesic. Introducing preference-based argumentation to inconsistent ontological knowledge bases. In *PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings*, pages 594–602, 2015.

[14] M. Croitoru and S. Vesic. What can argumentation do for inconsistent ontology query answering? In *Scalable Uncertainty Management - 7th International Conference, SUM 2013, Washington, DC, USA, September 16-18, 2013. Proceedings*, pages 15–29, 2013.

[15] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[16] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, pages 103–117, 2010.

[17] S. Modgil and H. Prakken. The *ASPIC*$^+$ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[18] M. Mugnier. Ontological query answering with existential rules. In *Web Reasoning and Rule Systems - 5th International Conference, RR 2011, Galway, Ireland, August 29-30, 2011. Proceedings*, pages 2–23, 2011.

[19] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[20] L. Popa, S. Abiteboul, and P. G. Kolaitis, editors. *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*. ACM, 2002.