Logic-based argumentation with existential rules

Abdallah Arioua, Madalina Croitoru, Srdjan Vesic

Logic-Based Argumentation with Existential Rules

Abdallah Arioua[1], Madalina Croitoru[2], Srdjan Vesic[3]

[1] LIRIS CNRS, Univ. Claude Bernard Lyon 1, Lyon, France
[2]INRIA, LIRMM, Univ. Montpellier 2, France
[3] CRIL, CNRS & Univ. Artois, Lens, France

Abstract. In this paper we are interested in the use of argumentation for handling inconsistency in inconsistent knowledge bases expressed with existential rules. We propose an instantiation of an argumentation framework and demonstrate it is coherent, relatively grounded and non trivial, therefore satisfying the rationality postulates from the literature. We demonstrate how argumentation semantics relate to the state of the art of handling inconsistency in this setting, allowing us to propose the first dialectical proof in the literature for a given semantics.

Keywords: *Logic-based Argumentation, Existential Rules, Universal Acceptance, Dialectical Proof Theory.*

1 Introduction

We place ourselves in a concrete, practical setting called ONTOLOGY-BASED DATA ACCESS (OBDA) [41]. In this setting a consistent ontology (that we consider in this paper, for the sake of generality, to be expressed using existential rules [17]) is employed to "transparently access" a set of different, independently developed data sources with the benefit of unified querying and implicit information exposure. In some cases, the union of data sources can be inconsistent with the ontology. In order to still be able to reason and query data sources in presence of inconsistency, maximal consistent subsets of the union of the data sources, called *repairs* [36], are considered. Since the repairs are consistent, classical entailment can be used on the repairs in different manners. These different ways of considering entailment on repairs are called *inconsistency tolerant semantics* [36].

Another way of dealing with inconsistency in OBDA is argumentation. Logic-based argumentation considers constructing arguments from inconsistent knowledge bases, computing attacks between them and using so called argumentation semantics in order to select acceptable arguments and their conclusions. Several approaches for logic based argumentation exist in the literature: assumption-based argumentation frameworks (ABA) [16], DeLP [33], the deductive argumentation, where an argument is perceived as a tuple (H, C) of set of premises H and a conclusion C [12] or ASPIC/ASPIC+ [39].

In this paper, we are interested by the use of argumentation for handling inconsistency in OBDA. The benefit of using argumentation is that it allows to represent the data in a format that is easier to grasp by a user. It allows (by examining the support of an

argument) to track the provenance of different pieces of information used to conclude a given formula and to see (by examining the attacks between arguments) which pieces of information are not compatible together. More precisely, we ask the following research question: "Is it possible to use logic based argumentation in order to handle inconsistency in OBDA?" We show that the answer to the question is positive, as demonstrated by the following main four contributions of the paper:

- *We propose an instantiation* of an argumentation framework and demonstrate it is coherent, grounded and non trivial (Section 3).
- *We show the link* between the argumentation semantics and inconsistency tolerant semantics (Section 4).
- *We validate the proposed instantiation* by showing it abides by the rationality postulates proposed in the literature (Section 5).
- *We propose the first dialectical proof in the literature* for a given semantics and analyse its properties in terms of finiteness, soundness, completeness and dispute complexity (Section 6).

Our work is significant as it gives an alternative and promising method of handling inconsistency using argumentation in OBDA. On the one hand, it allows a user to better grasp a piece of information since it is represented in form of an argument. On the other hand, we prove that the output of our argumentation based system coincides with the output provided by existing inconsistency tolerant semantics from the literature. We also provide a novel dialectical proof for a given inconsistency tolerant semantics. Such result paves the way for improved explanation techniques of inconsistency tolerant semantics using argumentation.

2 Inconsistency Handling in OBDA with Existential Rules

There are two major approaches to representing ontologies in the OBDA setting: Description Logics (such as $\mathcal{EL}$ [8] and DL-Lite [18] families) and rule-based languages (such as *Datalog*$\pm$ [17] language, a generalization of Datalog [22] that allows for existentially quantified variables in rule heads). As a response to *Datalog*$\pm$ undecidability when answering conjunctive queries, different decidable fragments were proposed and studied in the literature [11]. These fragments generalize the aforementioned Description Logics families and overcome their limitations by allowing any predicate arity as well as cyclic structures. In the next section we detail the logical language used throughout the paper and namely existential rules.

2.1 Syntax and Semantics of Existential Rules

We consider *the positive existential* fragment of first-order logic $\texttt{FOL}(\exists, \wedge)$ [23, 10]. Its language $\mathcal{L}$ is composed of formulas built with the usual quantifiers $(\exists, \forall)$ and *only* two connectors: implication $(\rightarrow)$ and conjunction $(\wedge)$.

We consider usual first-order vocabularies with constants but no other function symbols. A term is thus a variable or a constant. A vocabulary is a pair of two disjoint sets

$\mathcal{V} = (\mathcal{P}, \mathcal{C})$, where $\mathcal{P}$ is a set of predicates and $\mathcal{C}$ is a set of constants. A term t over $\mathcal{V}$ is a constant or a variable; different constants represent different values (unique name assumption). As an example, we define the following vocabulary with the set of constants $\mathcal{C} = \{John, Tom\}$, $\mathcal{P} = \{student, teacher, teaches\}$. The unique name assumption dictates that John and Tom which have different names refer to different entities in the real world. We use uppercase letters for constants, if the constant is a word then the first letter is put in uppercase, and we use 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. For instance, $teaches(John, x_1)$ is a binary predicate where x_1 and $John$ are terms. A ground atom is an atom with no variables, for example $teaches(John, Tom)$ is a ground atom. A conjunction of atoms is called a *conjunct*. A conjunction of ground atoms is called a *ground conjunct*. For instance, $teaches(John, x_1) \wedge teacher(John)$ is a conjunct and $teacher(John) \wedge teaches(John, Tom) \wedge student(Tom)$ 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. A closed formula is called a *sentence*. For example, the formula $\exists x_1 teaches(John, x_1)$ is closed because x_1 is in the scope of the quantifier $\exists$, whereas the formula $(teacher(John) \wedge student(x_1) \wedge teaches(John, x_1))$ is not closed because x_1 appears free in the formula.

One way to represent knowledge about the world is to grasp its factual knowledge represented usually by *ground atoms*. In the Existential Rules framework this concept has been extended so that a fact on $\mathcal{V}$ is the existential closure of a conjunction of atoms over $\mathcal{V}$ [10]. For instance, $\exists x_1(student(x_1) \wedge enrolled(x_1, C1))$ is a fact. For a given fact F we exclude duplicate atoms in F, which allows to see a fact as a set of atoms. For instance, the fact $F = \exists x \exists y(r(x) \wedge p(A, y) \wedge r(x))$ can be seen as $\{p(A, y), r(x)\}$. By doing so we will be able to apply set-theoretic operations on facts like union, subset-inclusion, etc.

Let F be a fact, we denote by $terms(F)$ (resp. $vars(F)$) the set of terms (resp. variables) that occur in F. We recall the notions of *substitution* and *homomorphism* between facts to be used in order to query facts. Given a set of variables $\mathcal{X}$ and a set of terms $\mathcal{T}$, a **substitution** σ of $\mathcal{X}$ by $\mathcal{T}$ (notation $\sigma : \mathcal{X} \to \mathcal{T}$) is a function 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 $\sigma(F) \subseteq F'$.

Example 1 (Homomorphism). Consider the following vocabulary $\mathcal{V} = (\mathcal{C}, \mathcal{P})$:

– $\mathcal{C} = \{A, B\}$.
– $\mathcal{P} = \{q, r\}$.

Consider the following two facts over the vocabulary $\mathcal{V}$:

– $F = \{q(A, x)\}$ where $terms(F) = \{A, x\}$ and $vars(F) = \{x\}$.
– $F' = \{q(A, B), r(A)\}$ where $terms(F') = \{A, B\}$ and $vars(F') = \emptyset$.

Consider $vars(F)$ and $terms(F')$ as our set of variables and set of terms. We have two possible substitutions.

– $\sigma_1 = \{(x, A)\}$.
– $\sigma_2 = \{(x, B)\}$.

Where x is substituted by A in σ_1 and by B in σ_2. Let us see which of these substitutions is a homomorphism from F to F':

– When we apply σ_1 on F we get $\sigma_1(F) = \{q(A, A)\}$.
– When we apply σ_2 on F we get $\sigma_2(F) = \{q(A, B)\}$.

It is clear that the substitution σ_2 is a homomorphism from F to F' (unlike σ_1) because $\sigma_2(F) \subseteq F'$.

A conjunctive query (CQ) has the following form: $\mathcal{Q} = \texttt{ans}(x_1, \ldots, x_k) \leftarrow B$, where B (the "body" of $\mathcal{Q}$) is a fact, and $x_1, \ldots, x_k$ occur in B and `ans` is a special k-ary predicate, whose arguments are used to build an answer. Given a set of facts $\mathcal{F}$, an answer to $\mathcal{Q}$ in $\mathcal{F}$ is a tuple of constants $(A_1, \ldots, A_k)$ such that there is a homomorphism σ from B to $\mathcal{F}$, with $(\sigma(x_1), \ldots, \sigma(x_k)) = (A_1, \ldots, A_k)$. If $k = 0$, i.e. $\mathcal{Q} = \texttt{ans}() \leftarrow B$, $\mathcal{Q}$ is called a Boolean conjunctive query, the unique answer to $\mathcal{Q}$ is the empty tuple if there is a homomorphism from B to $\mathcal{F}$, otherwise there is no answer to $\mathcal{Q}$. Note that a query $\mathcal{Q}$ can be shortly referred to by its body B. For its simplicity, this notation will be used hereafter.

A complementary way to represent knowledge is to use rules that encode domain-specific knowledge. Rules are regarded as an ontological layer that reinforces the expressiveness of the knowledge base. Rules are logical formulae that allow us to infer new facts (conclusion) from existing facts (hypothesis). Existential rules [10, 17] introduce new variables in the conclusion having ability to represent unknown individuals (also known in database community as *value invention* [1]). This form of rules is also known as tuple-generating dependencies in database community [32]. We denote by $\boldsymbol{x}$ in a bold font a vector of variables. An *existential rule* (or simply a rule) is a closed formula of the form $R = \forall \boldsymbol{x} \forall \boldsymbol{y}(B \rightarrow \exists \boldsymbol{z} H)$, where B and H are conjuncts, with $vars(B) = \boldsymbol{x} \cup \boldsymbol{y}$, and $vars(H) = \boldsymbol{x} \cup \boldsymbol{z}$. The variables $\boldsymbol{z}$ 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.

Existential rules are more expressive than Description Logics as they can represent complex relations between individuals. For example, the existential rule $siblingOf(x, y) \rightarrow parentOf(z, x) \wedge parentOf(z, y)$ cannot be expressed in DL because of the "cycle on variables" [23].

To represent knowledge about the world we account for *negative constraints*, i.e. knowledge that dictates constraints about the world, also known as denial constraints in databases [17]. A negative constraint (or simply a constraint) is a rule of the form $N = \forall \boldsymbol{x}(B \rightarrow \bot)$. Negative constraints in the existential rules framework fully capture *concept disjointness* of DLs. For example, the following is a negative constraint $retiredFrom(x, y) \wedge worksIn(x, y) \rightarrow \bot$. From now on we omit quantifiers in front of formulae as there is no ambiguity.

A knowledge base is a tuple $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ of finite sets of facts, rules and negative constraints respectively. Reasoning with a knowledge base is done via a mechanism called saturation. In order to define saturation we need to define rule applicability.

A rule $R = B \rightarrow H$ is **applicable** [10] 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 $\texttt{safe}(H)$ 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**) $\langle F_0 = F, ..., F_n = F' \rangle$ such that for all $0 \leq i < n$ there is a rule $R \subseteq \mathcal{R}$ which is applicable to F_i and F_{i+1} is an immediate derivation from F_i.

Example 2. For instance, consider $R = q(x, y) \rightarrow p(z, y)$ and $F = \{q(A, B), r(A), s(v)\}$, R is applicable to F because there is a homomorphism from set $\{q(x, y)\}$ to set $\{q(A, B), r(A), s(v)\}$ that substitutes x by A and y by B. The immediate derivation from F is the fact:

$F' = \{q(A, B), r(A), s(v)\} \cup \{p(w, B)\}$ where w is a fresh variable not used before.

The reason to introduce fresh variables is to avoid relating variables from the initial set of facts with new introduced facts. For instance, in the above example if we were to use v instead of w then we would have $\{p(v, B)\}$ which dictates that: besides the fact that v has the property s indicated initially in F, it is also related to B by the predicate p, which is not faithful to the meaning of the rule.

We denote by $C\ell^*_{\mathcal{R}}(\mathcal{F})$ the saturation of $\mathcal{F}$ with respect to $\mathcal{R}$. Applying a rule to a set of facts is called a chase, and different chase mechanisms use different simplifications that prevent infinite redundancies. In this paper, we use the skolem chase [37]. Moreover, we restrict ourselves to recognisable FES classes of existential rules where the chase is guaranteed to stop [10]. Thus, $C\ell^*_{\mathcal{R}}(\mathcal{F})$ is a finite set.

2.2 Inconsistency-Tolerant Semantics

Existential rules framework is widely used in Semantic Web and in the so-called ONTOLOGY-BASED DATA ACCESS, where rules and constraints act as an ontology used to "access" different data sources. These sources are prone to inconsistencies. In this setting, we suppose that the set of rules is compatible with the set of negative constraints, i.e. the union of those two sets is satisfiable [36].

This assumption is made because in OBDA we assume that the ontology is believed to be reliable as it is the result of a robust construction by domain experts. However, as data can be large and heterogeneous due to merging and fusion, in the OBDA setting the data is assumed to be the source of inconsistency. This means that by applying the rules on the set of facts, we might violate a constraint.

In what follows we recall the formal definition of *inconsistency* in the existential rules framework; then we introduce the subset-repairing technique which is inspired by the work from the database community [25] and Description Logics [36, 14].

Definition 1 (Inconsistency). *A set of facts $\mathcal{F}$ is inconsistent with respect to a set of rules $\mathcal{R}$ and negative constraints $\mathcal{N}$ (or inconsistent for short) if and only if there exists a constraint $N \in \mathcal{N}$ such that $C\ell^*_{\mathcal{R}}(\mathcal{F}) \models body(N)$.*

This means that the set of facts *violates* the negative constraint N or triggers it. Correspondingly, a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ is inconsistent (with respect to $\mathcal{R}$ and $\mathcal{N}$) if and only if there exists a set of facts $\mathcal{F}' \subseteq \mathcal{F}$ such that $\mathcal{F}'$ is inconsistent. An alternative writing is $C\ell^*_{\mathcal{R}}(\mathcal{F}) \models \bot$.

Example 3. Let us consider the following knowledge base $\mathcal{K}$ with:
$\mathcal{F} = \{cat(Tom), bark(Tom)\}$, $\mathcal{R} = \{R_1 : cat(x_1) \rightarrow miaw(x_1)\}$, $\mathcal{N} = \{N_1 : bark(x_2) \wedge miaw(x_2) \rightarrow \bot\}$.
The saturation yields: $C\ell^*_{\mathcal{R}}(\mathcal{F}) = \{cat(Tom),\ bark(Tom), miaw(Tom)\}$. Observe that this knowledge base violates the negative constraint N_1.

One way to cope with inconsistency is to construct *maximal consistent subsets* of the knowledge base [44]. This corresponds to "Data Repairs" [5]. A data repair of a knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ is a set of facts $\mathcal{F}'$ such that $\mathcal{F}'$ is consistent and there exists no consistent subset of $\mathcal{F}$ that strictly contains $\mathcal{F}'$ [36].

Definition 2 (Repair). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base. A data repair (repair for short) of $\mathcal{K}$ is a set of facts $\mathcal{F}' \subseteq \mathcal{F}$ such that:*

– $C\ell^*_{\mathcal{R}}(\mathcal{F}') \not\models \bot$ (consistency).
– $\forall \mathcal{S} \subset \mathcal{F} \setminus \mathcal{F}'$, if $S \neq \emptyset$ then $C\ell^*_{\mathcal{R}}(\mathcal{F}' \cup \mathcal{S}) \models \bot$.

Since repairs are computed exclusively on the set of facts and given that the factual part of the knowledge base is the only source of inconsistency we, from now on, abuse slightly the notation and refer to $\mathcal{K}'$ by its set of facts $\mathcal{F}'$. The set of all repairs of $\mathcal{K}$ is denoted by $\mathcal{R}epair(\mathcal{K})$.

Example 4 (Example 3 cont'd). The two possible repairs are: $\mathcal{P}_1 = \{cat(Tom)\}$ and $\mathcal{P}_2 = \{bark(Tom)\}$. If we consider $\mathcal{F} \cup \{animal(Tom)\}$, then the repairs become:

$$\mathcal{P}_1 = \{cat(Tom), animal(Tom)\} \text{ and } \mathcal{P}_2 = \{bark(Tom), animal(Tom)\}$$

Once all repairs are computed, there are different ways to compute queries that follow from an inconsistent knowledge base. The most prominent way is to allow the entailment of a query if it is entailed from *all repairs*. This is called the CQA semantics (Consistent Query Answering semantics) or $\mathcal{AR}$*-semantics* (All Repairs semantics) . Please note that in our context a query under CQA has either a yes or a no answer. When we say a query is accepted that means it has a yes answer (entailed), otherwise it has a no answer (not entailed).

Definition 3 (CQA semantics). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and let $\mathcal{Q}$ be a query. Then $\mathcal{Q}$ is accepted under CQA in $\mathcal{K}$, written $\mathcal{K} \models_{\mathcal{CQA}} \mathcal{Q}$ iff for every repair $\mathcal{P} \in \mathcal{R}epair(\mathcal{K})$, it holds that $C\ell^*_{\mathcal{R}}(\mathcal{P}) \models \mathcal{Q}$.*

Example 5 (Example 4 cont'd). We have that:
$C\ell^*_{\mathcal{R}}(\mathcal{P}_1) = \{cat(Tom), animal(Tom), miaw(Tom)\}$
$C\ell^*_{\mathcal{R}}(\mathcal{P}_2) = \{bark(Tom), animal(Tom)\}$.
It is the case that $\mathcal{K} \models_{\mathcal{CQA}} animal(Tom)$ but it is not the case that $\mathcal{K} \models_{\mathcal{CQA}} miaw(Tom)$ because $miaw(Tom)$ is not entailed from $\mathcal{P}_2$.

Another possibility is to check whether the query is entailed from the *intersection of closed repairs* ($\mathcal{ICR}$*-semantics*).

Definition 4 **($\mathcal{ICR}$-*semantics*).** *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and let $\mathcal{Q}$ be a query. Then $\mathcal{S}$ is $\mathcal{ICR}$-**entailed** from $\mathcal{K}$, written $\mathcal{K} \models_{\mathcal{ICR}} \mathcal{Q}$ if and only if:*

$$\bigcap_{\mathcal{P} \in \mathcal{R}epair(\mathcal{K})} C\ell^*_{\mathcal{R}}(\mathcal{P}) \models \mathcal{Q}$$

Example 6 (Example 5 cont'd). It is *not* the case that $\mathcal{K} \models_{\mathcal{ICR}} cat(Tom)$.

Finally, another possibility is to consider the *intersection of all repairs* and then close this intersection under the rules ($\mathcal{IAR}$*-semantics*).

Definition 5 **($\mathcal{IAR}$-*semantics*).**
*Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and let $\mathcal{Q}$ be a query. Then $\mathcal{Q}$ is $\mathcal{IAR}$-**entailed** from $\mathcal{K}$, written $\mathcal{K} \models_{\mathcal{IAR}} \mathcal{Q}$ iff:*

$$C\ell^*_{\mathcal{R}}\left(\bigcap_{\mathcal{P} \in \mathcal{R}epair(\mathcal{K})} \mathcal{P}\right) \models \mathcal{Q}$$

Let us show that the three semantics can yield different results.

Example 7. ($\mathcal{ICR}$ and $\mathcal{IAR}$ different from $\mathcal{CQA}$) Consider $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$, with:

– $\mathcal{F} = \{haveCat(John, Tom), haveMouse(John, Jerry)\}$.
– $\mathcal{R} = \{haveCat(x_1, x_2) \rightarrow haveAnimal(x_1, x_2),$
$haveMouse(x_3, x_4) \rightarrow haveAnimal(x_3, x_4)\}$.
– $\mathcal{N} = \{haveCat(x_1, x_2) \wedge haveMouse(x_1, x_3) \rightarrow \bot\}$.

There are two repairs:

– $\mathcal{P}_1 = \{haveCat(John, Tom)\}$.
– $C\ell^*_{\mathcal{R}}(\mathcal{P}_1) = \{haveCat(John, Tom), haveAnimal(John, Tom)\}$.
– $\mathcal{P}_2 = \{haveMouse(John, Jerry)\}$.
– $C\ell^*_{\mathcal{R}}(\mathcal{P}_2) = \{haveMouse(John, Jerry), haveAnimal(John, Jerry)\}$.

Consider a query $\mathcal{Q} = haveAnimal(John, x)$ asking whether John has an animal, recall that this is boolean query that has a *yes* or *no* answer. It holds that $\mathcal{K} \models_{\mathcal{CQA}} \mathcal{Q}$ since $C\ell^*_{\mathcal{R}}(\mathcal{P}_1) \models \mathcal{Q}$ and $C\ell^*_{\mathcal{R}}(\mathcal{P}_2) \models \mathcal{Q}$, but neither $\mathcal{K} \models_{\mathcal{ICR}} \mathcal{Q}$ (since $C\ell^*_{\mathcal{R}}(\mathcal{P}_1) \cap C\ell^*_{\mathcal{R}}(\mathcal{P}_2) = \emptyset$) nor $\mathcal{K} \models_{\mathcal{IAR}} \mathcal{Q}$ (since $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$).

Let us now show that $\mathcal{CQA}$ and $\mathcal{ICR}$ are different from $\mathcal{IAR}$.

Example 8. ($\mathcal{CQA}$ and $\mathcal{ICR}$ different from $\mathcal{IAR}$) Consider the following knowledge base $\mathcal{K}$:

– $\mathcal{F} = \{cat(Tom), dog(Tom)\}$.
– $\mathcal{R} = \{cat(x_1) \rightarrow animal(x_1), dog(x_2) \rightarrow animal(x_2)\}$,

– $\mathcal{N} = \{cat(x_1) \wedge dog(x_1) \rightarrow \bot)\}$.

We have $\mathcal{R}epair(\mathcal{K}) = \{\mathcal{P}_1, \mathcal{P}_2\}$ with $\mathcal{P}_1 = \{cat(Tom)\}$ and $\mathcal{P}_2 = \{dog(Tom)\}$. $C\ell^*_{\mathcal{R}}(\mathcal{P}_1) = \{cat(Tom), animal(Tom)\}$, $C\ell^*_{\mathcal{R}}(\mathcal{P}_2) = \{dog(Tom),$ $animal(Tom)\}$. Observe that, it is *not* the case that $\mathcal{K} \models_{\mathcal{IAR}} animal(x)$ since we have $\mathcal{P}_1 \bigcap \mathcal{P}_2 = \emptyset$. However, $\mathcal{K} \models_{\mathcal{CQA}} animal(x)$. This is because $C\ell^*_{\mathcal{R}}(\mathcal{P}_1) \models animal(x)$ and $C\ell^*_{\mathcal{R}}(\mathcal{P}_2) \models animal(x)$. Also, we have $\mathcal{K} \models_{\mathcal{ICR}} animal(x)$ since $C\ell^*_{\mathcal{R}}(\mathcal{P}_1) \cap C\ell^*_{\mathcal{R}}(\mathcal{P}_2) = \{animal(Tom)\}$.

Inconsistency handling can rely on another concept called *minimal conflicts*. Given a knowledge base $\mathcal{K}$, a set of facts C is called a *minimal conflict* of $\mathcal{K}$ if and only if C is inconsistent and every subset of C is consistent.

Definition 6 (Minimal conflicts). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be an inconsistent knowledge base. A set of facts C is called a minimal conflict of $\mathcal{K}$ if and only if:*

– *$C\ell^*_{\mathcal{R}}(C) \models \bot$ (inconsistency).*
– *$\forall X \subset C$, if $X \neq \emptyset$ then $C\ell^*_{\mathcal{R}}(C \setminus X)$ is consistent (minimality).*

We denote by $\texttt{conflict}(\mathcal{K})$ *the set of all minimal conflicts of $\mathcal{K}$.*

Example 9 (Conflicts). Consider the following knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$:

– $\mathcal{F} = \{p(A, A), p(B, C), q(C, B), r(C), w(D)\}$.
– $\mathcal{R} = \{q(x, y) \rightarrow s(x, y)\}$.
– $\mathcal{N} = \{p(x, x) \rightarrow \bot, p(x, y) \wedge q(y, x) \wedge r(y) \rightarrow \bot\}$.

We have the following $\texttt{conflict}(\mathcal{K})$:

– $C_1 = \{p(A, A)\}$ and $C_2 = \{p(B, C), q(C, B), r(C)\}$.

Note that $\{p(B, C), s(C, B), r(C)\}$ is not a conflict because it is consistent. It is clear from this knowledge base that every fact in $\mathcal{F}$ except $w(D)$ is involved in some inconsistencies. This means that $w(D)$ will be in all repairs. On the contrary, $p(B, C)$ will not be in any repair that contains $q(C, B)$ and $r(C)$. Consequently, $p(B, C)$ will not be in all repairs and will not be accepted under CQA semantics.

The repairs are:

– $\mathcal{P}_1 = \{p(B, C), r(C), w(D)\}$
– $\mathcal{P}_2 = \{p(B, C), q(C, B), w(D)\}$
– $\mathcal{P}_3 = \{q(C, B), r(C), w(D)\}$

We saw how the CQA semantics handles inconsistency in the existential rules framework. In the next section we introduce logic-based argumentation with existential rules, which is another method to handle inconsistency.

3 Instantiating Dung's Abstract Framework with Existential Rules

In this section we show how to instantiate Dung's [28] abstract model with the logic presented in the previous section. Subsection 3.1 defines the instantiation and Subsection 3.2 studies its properties.

3.1 Arguments and attacks

An argument is composed of premises and a conclusion [12]. The set of premises is seen as a *justification*, a *support*, a *reason* or a *proof* for the conclusion.

Definition 7 (Argument). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be an inconsistent knowledge base. An argument is a couple $a = (H, C)$ such that:*

1. $H \subseteq \mathcal{F}$ and $C\ell^*_{\mathcal{R}}(H) \not\models \bot$ (consistency).
2. $C = \alpha_0 \wedge \ldots \wedge \alpha_n$ is an atom or a conjunct such that $\{\alpha_0, \ldots, \alpha_n\} \subseteq C\ell^*_{\mathcal{R}}(H)$ (entailment).

Given an argument a, we denote its support by $\texttt{Supp}(a) = H$ and its conclusion by $\texttt{Conc}(a) = C$.

An argument is defined as a couple in which the support is a set of facts responsible from the entailment of the conclusion C from the knowledge base $\mathcal{K}$. The first clause ensures that the support is consistent, which is an important property [12]. The second clause ensures the preservation of entailment from the support H to the conclusion C. Note that here arguments are constructed only from the factual part $\mathcal{F}$ of the knowledge base, and there are no rules or negative constraints in the support or the conclusion. The reason to exclude such formulas is due to an assumption inherent to OBDA which dictates that the rules and the negative constraints are satisfiable and reliable, therefore they should not be subjects of any attack.

Since all inconsistency comes from the facts, the only type of attack needed is the so-called *direct undercut*, also known as *assumption attack*.

Definition 8 (Attack). *An argument a attacks b if and only if $\exists h \in \texttt{Supp}(b)$ such that*

$$C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(a), h\}) \models \bot.$$

Note that the attack relation is not symmetric, as illustrated by the following example.

Example 10 (Non-symmetry). Let $\mathcal{F} = \{p(M), q(M), r(M)\}$, $\mathcal{R} = \emptyset$, $\mathcal{N} = \{p(x) \wedge q(x) \wedge r(x) \rightarrow \bot)\}$. Let $a = (\{p(M), q(M)\}, p(M) \wedge q(M))$, $b = (\{r(M)\}, r(M))$. The argument a attacks b but b does not attack a because there exists no $h \in \texttt{Supp}(a)$ which is inconsistent with $\texttt{Conc}(b) = r(M)$. However, the set $\{p(M), q(M)\}$ is indeed inconsistent with $r(M)$, but according to the definition of the attack we consider a single atom in the support of a.

This attack relation is irreflexive. This is guaranteed by the fact that for any argument a, $\texttt{Supp}(a)$ is consistent and it entails $\texttt{Conc}(a)$, therefore $\texttt{Supp}(a)$ and $\texttt{Conc}(a)$ are consistent together. Consequently, an argument cannot attack itself.

Note that the attack relation can be empty. This happens if the knowledge base is consistent or all the minimal conflicts in the knowledge base are unary.

Example 11. Consider $\mathcal{F} = \{p(M), r(M)\}$, $\mathcal{R} = \emptyset$, $\mathcal{N} = \{p(X) \rightarrow \bot)\}$. We have only one argument in this case $a = (\{r(M)\}, r(M))$. It is clear that $p(M)$ is self-contradictory because it triggers the negative constraint. It is not hard to see that the attack relation is empty.

Notation 1 *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, $\mathcal{F}' \subseteq \mathcal{F}$ be a set of facts and S be a set of arguments. We adapt the following notations:*

- $\texttt{Args}(\mathcal{F}') = \{a \mid a$ *is an argument such that* $\texttt{Supp}(a) \subseteq \mathcal{F}'\}$. *This refers to the set of all possible arguments that can be constructed from a given set of facts $\mathcal{F}'$.*
- $\texttt{Base}(\mathcal{S}) = \bigcup_{a_i \in \mathcal{S}} \texttt{Supp}(a_i)$. *A base of a set of arguments is the set of facts that contains the supports of all arguments from $\mathcal{S}$.*
- *The set of all arguments that can be constructed over $\mathcal{K}$ is denoted as* $\texttt{Args}(\mathcal{F})$.

An argumentation framework is defined as follows.

Definition 9 (Argumentation framework). *An argumentation framework is a pair $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ where $\mathcal{A}$ is the set of arguments and $\mathcal{X} \subseteq \mathcal{A} \times \mathcal{A}$ is the corresponding attack relation.*

In order to calculate the sets of arguments that can be accepted together, called *extensions*, different acceptability semantics were introduced in the literature.

Definition 10 (Extensions). *Let $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ be an argumentation framework, $\mathcal{E} \subseteq \mathcal{A}$ and $a \in \mathcal{A}$.*

- *We say that $\mathcal{E}$ is* conflict free *iff there exists no arguments $a, b \in \mathcal{E}$ such that $(a, b) \in \mathcal{X}$.*
- *$\mathcal{E}$* defends *a iff for every argument $b \in \mathcal{A}$, if we have $(b, a) \in \mathcal{X}$ then there exists $c \in \mathcal{E}$ such that $(c, b) \in \mathcal{X}$.*
- *$\mathcal{E}$ is* admissible *iff it is conflict free and defends all its arguments.*
- *$\mathcal{E}$ is a* complete extension *iff $\mathcal{E}$ is an admissible set which contains all the arguments it defends.*
- *$\mathcal{E}$ is a* preferred extension *iff it is maximal (with respect to set inclusion) admissible set.*
- *$\mathcal{E}$ is a* stable extension *iff it is conflict-free and for all $a \in \mathcal{A} \setminus \mathcal{E}$, there exists an argument $b \in \mathcal{E}$ such that $(b, a) \in \mathcal{X}$.*
- *$\mathcal{E}$ is a* grounded extension *iff $\mathcal{E}$ is a minimal (for set inclusion) complete extension.*
- *An argument is :*
  - *skeptically accepted with respect to a semantics x if it is in all extensions under x, and there exists at least one extension,*
  - *credulously accepted with respect to a semantics x if it is in at least one extension under x and*
  - *rejected if it is not in any extension under x.*

To facilitate the readability of the paper we adapt the following notations.

Notation 2 *For an argumentation framework $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ we denote by $\texttt{Ext}_x(\mathcal{H})$ the set of its extensions with respect to the semantics x. We use the abbreviations p, s, and g for respectively preferred, stable and grounded semantics. We denote by $\texttt{Sc}_x(\mathcal{H})$ the set of skeptically accepted arguments and by $\texttt{Cr}_x(\mathcal{H})$ the set of credulously accepted arguments of $\mathcal{H}$ under the semantics x. In addition, we also use the following notations:*

- $\texttt{range}^+(a) = \{b \mid (a, b) \in \mathcal{X}\}$ *(the set of arguments attacked by a)*

– $\texttt{range}^-(a) = \{b \mid (b, a) \in \mathcal{X}\}$ *(the set of arguments that attack a)*
– $\texttt{range}^+(S) = \bigcup_{a \in S} \texttt{range}^+(a)$
– $\texttt{range}^-(S) = \bigcup_{a \in S} \texttt{range}^-(a)$

To every knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ corresponds an argumentation framework $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ where $\mathcal{A} = \texttt{Args}(\mathcal{F}')$ and $\mathcal{X} \subseteq \mathcal{A} \times \mathcal{A}$ is the attack relation between the arguments of $\mathcal{A}$, as specified in Definition 8. We call $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework of $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$.

Let us use an example to show how to construct an argumentation framework from a given knowledge base.

Example 12 (Pick two!). Consider the "Fast, Good or Cheap. Pick two!" project management principle. It states the fact that the three properties Fast, Good and Cheap of a project are interrelated, and it is not possible to optimize all the three, then one should always pick two of the three. It can be represented as a knowledge base as follows:

$\mathcal{F} = \{project(P), isfast(P), isgood(P), ischeap(P)\}$
$\mathcal{R} = \{isfast(x) \wedge isgood(x) \rightarrow isexpensive(x)\}$
$\mathcal{N} = \{ischeap(x) \wedge isexpensive(x) \rightarrow \bot\}$

All the arguments that can be generated from $\mathcal{K}$ are presented in Appendix, Table 4. The attacks are defined with respect to the following set of conflicts:

$\texttt{conflict}(\mathcal{K}) = \{C_1, C_2\}$ such that $C_1 = \{ischeap(P), isfast(P), isgood(P)\}$ and $C_2 = \{ischeap(P), isexpensive(P)\}$.

There are three extensions under stable / preferred semantics:

– $\mathcal{E}_1 = \{a_2, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{22}, a_{23}, a_{27}, a_{28}, a_{29}, a_{32}, a_{33}, a_{38}, a_{39}, a_{40}, a_{41}, a_{43}, a_{44}, a_{45}, a_{46}, a_{47}, a_{48}, a_{49}, a_{50}, a_{51}, a_{52}, a_{53}, a_{54}\}$.
– $\mathcal{E}_2 = \{a_1, a_4, a_5, a_6, a_{19}, a_{20}, a_{21}, a_{22}, a_{25}, a_{26}, a_{32}, a_{33}, a_{34}, a_{35}, a_{36}, a_{37}, a_{40}, a_{41}, a_{42}, a_{44}\}$.
– $\mathcal{E}_3 = \{a_3, a_7, a_8, a_9, a_{16}, a_{17}, a_{18}, a_{23}, a_{24}, a_{25}, a_{28}, a_{29}, a_{30}, a_{31}, a_{34}, a_{35}, a_{40}, a_{42}, a_{43}\}$.

For this argumentation framework $\texttt{Sc}_s(\mathcal{H}) = \texttt{Sc}_p(\mathcal{H}) = \{a_{40}\}$. Note that the grounded extension is equal to $\{a_{40}\}$. We have $\texttt{Cr}_s(\mathcal{H}) = \texttt{Cr}_p(\mathcal{H}) = \mathcal{A}$. There are no rejected arguments.

3.2 Characterizing the outputs

Logic-based argumentation frameworks allow to exploit the structure of arguments to reason in terms of acceptable conclusions. We introduce two definitions allowing us to reason over such an argumentation framework. The output of an argumentation framework is usually defined [19, Definition 12] as the set of conclusions that appear in all the extensions (under a given semantics).

Definition 11 (Output of an argumentation framework). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework. The output of $\mathcal{H}$ under semantics x is defined as:*

$$\texttt{Output}_x(\mathcal{H}) = \bigcap_{\mathcal{E} \in \texttt{Ext}_x(\mathcal{H})} \texttt{Concs}(\mathcal{E}).$$

In the degenerate case when $\texttt{Ext}_x(\mathcal{H}) = \emptyset$, we define $\texttt{Output}(\mathcal{H}) = \emptyset$ by convention.

Note that the previous definition asks for existence of a conclusion in every extension. This kind of acceptance is usually referred to as *skeptical* acceptance. We say that a query $\mathcal{Q}$ is skeptically accepted if it is a logical consequence of the output of $\mathcal{H}$:

Definition 12 (Skeptical acceptance of a query). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework. A query $\mathcal{Q}$ is skeptically accepted under semantics x if and only if $\texttt{Output}_x(\mathcal{H}) \models \mathcal{Q}$.*

It is possible to make an alternative definition, which uses the notion of universal acceptance instead of skeptical one. According to universal criterion, a query $\mathcal{Q}$ is accepted if it is a logical consequence of conclusions of every extension:

Definition 13 (Universal acceptance of a query). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework. A query $\mathcal{Q}$ is universally accepted under semantics x if and only if for every extension $\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{Q})$, it holds that $\texttt{Concs}(\mathcal{E}_i) \models \mathcal{Q}$.*

In general, universal and skeptical acceptance of a query do not coincide. Take for instance the KB from Example 7. We can check that $\mathcal{Q} = haveAnimal(John, x)$ is universally accepted but not skeptically accepted.

Example 13 (Cont'd Example 7). Let us compute the corresponding argumentation framework $\mathcal{H}$, and compare the sets of universally and skeptically accepted queries under preferred semantics. We have the following $\texttt{Args}(\mathcal{F})$:

$a_1 = (\{haveCat(John, Tom)\}, haveCat(John, Tom))$
$a_2 = (\{haveMouse(John, Jerry)\}, haveMouse(John, Jerry))$
$a_3 = (\{haveMouse(John, Jerry)\}, haveAnimal(John, Jerry))$
$a_4 = (\{haveCat(John, Tom)\}, haveAnimal(John, Tom) \wedge haveCat(John, Tom))$
$a_5 = (\{haveCat(John, Tom)\}, haveAnimal(John, Tom))$
$a_6 = (\{haveMouse(John, Jerry)\}, haveAnimal(John, Jerry)$
$\wedge haveMouse(John, Jerry))$

The attack relation is drawn in Figure 1. Consequently, we get the following $\texttt{Ext}(\mathcal{H})$ under the stable and preferred semantics:

$$\mathcal{E}_1 = \{a_1, a_4, a_5\}\ \mathcal{E}_2 = \{a_2, a_3, a_6\}$$

The set of conclusions of the extensions are:

- $\texttt{Concs}(\mathcal{E}_1) = \{haveCat(John, Tom), haveAnimal(John, Tom) \wedge haveCat(John, Tom), haveAnimal(John, Tom)\}$.

– $\texttt{Concs}(\mathcal{E}_2) = \{haveMouse(John, Jerry), haveAnimal(John, Jerry), haveAnimal(Jo\text{-}hn, Jerry) \wedge haveMouse(John, Jerry)\}$.

The output $\texttt{Output}(\mathcal{H})$ under the stable and preferred semantics of the argumentation framework is empty because the intersection of $\texttt{Concs}(\mathcal{E}_1)$ and $\texttt{Concs}(\mathcal{E}_2)$ is empty. Consider now the boolean query $\mathcal{Q} = haveAnimal(John, x)$, it is clear that the query is universally accepted because $\texttt{Concs}(\mathcal{E}_1) \models \mathcal{Q}$ and $\texttt{Concs}(\mathcal{E}_2) \models \mathcal{Q}$. However, $\mathcal{Q}$ is not skeptically accepted because $\texttt{Output}(\mathcal{H}) \not\models \mathcal{Q}$.

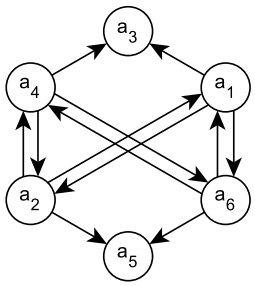


Fig. 1: Argument graph of Example 7.

Note that for single-extension semantics (e.g. grounded), the notions of skeptical and universal acceptance coincide. So we simply use word "accepted" in that case.

Definition 14 (Acceptance of a query). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework, x a single-extension semantics and let $\mathcal{E}$ be the unique extension of $\mathcal{H} = (\mathcal{A}, \mathcal{X})$. A query α is accepted under semantics x if and only if $\texttt{Concs}(\mathcal{E}) \models \mathcal{Q}$.*

In order to formally capture the subtle difference between universal and skeptical acceptance let us define the notion of a supporting argument. An argument a supports a query if and only if the query is entailed by the conclusion of the argument a.

Definition 15 (Support). *Let $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ be an argumentation framework over an inconsistent knowledge base $\mathcal{K}$ and let $\mathcal{Q}$ be a query. An argument $a \in \mathcal{A}$ supports the query $\mathcal{Q}$ if and only if $\texttt{Conc}(a) \models \mathcal{Q}$. We call a **a supporter** or **a supporting argument** of $\mathcal{Q}$. The set of all supporters of a given query $\mathcal{Q}$ is denoted by $\texttt{S}(\mathcal{Q})$.*

The universal and non-universal acceptance can be further characterized in a precise way. The goal of introducing such characterization is to be able to understand why a query is universally accepted or not. In what follows, when we do not specify the semantics, we suppose stable or preferred semantics.

Definition 16 (Reduct of extension). *Given an extension $\mathcal{E} \subseteq \mathcal{A}$ and a query $\mathcal{Q}$, set $\mathtt{S}(\mathcal{Q}) \bigcap \mathcal{E}$ is called the reduct $\mathcal{E}^{\mathcal{Q}} \subseteq \mathcal{E}$ of the extension $\mathcal{E}$ w.r.t the query $\mathcal{Q}$ if and only if it is non-empty. The previous definition says that the reduct of the set of all extensions $\mathtt{Ext}(\mathcal{H})$ with respect to $\mathcal{Q}$ is defined as $\mathtt{Ext}(\mathcal{H})^{\mathcal{Q}} = \{\mathcal{E}^{\mathcal{Q}} | \mathcal{E} \in \mathtt{Ext}(\mathcal{H})\}$.*

The reduct $\mathcal{E}^{\mathcal{Q}}$ of the extension $\mathcal{E}$ with respect to the query $\mathcal{Q}$ is defined as the set of all supporters of $\mathcal{Q}$ which belong to $\mathcal{E}$.

Definition 17 (Complete reduct). *The set of all reducts $\mathtt{Ext}(\mathcal{H})^{\mathcal{Q}}$ with respect to a query $\mathcal{Q}$ is complete if and only if there exists no $\mathcal{E} \in \mathtt{Ext}(\mathcal{H})$ such that $\mathcal{E}^{\mathcal{Q}} \notin \mathtt{Ext}(\mathcal{H})^{\mathcal{Q}}$.*

This means that the set of all reducts is complete if it covers all the extensions. However a further characterization can be developed to precisely determine when and how a reduct can be complete or not. It turns out that the latter has a tight relation with the hitting set problem.

Example 14 (Reducts). Let us see how we compute reducts. Consider the following knowledge base:

– $\mathcal{F} = \{jaguar(T), leopard(T)\}$.
– $\mathcal{R} = \{jaguar(x) \rightarrow animal(x), leopard(x) \rightarrow animal(x)\}$.
– $\mathcal{N} = \{jaguar(x) \wedge leopard(x) \rightarrow \bot\}$.

We have the following $\mathtt{Args}(\mathcal{F})$:

$a_1 = (\{jaguar(T)\}, jaguar(T))$ $\quad$ $a_2 = (\{leopard(T)\}, leopard(T))$
$a_3 = (\{leopard(T)\}, animal(T))$ $\quad$ $a_4 = (\{jaguar(T)\}, animal(T) \wedge jaguar(T))$
$a_5 = (\{jaguar(T)\}, animal(T))$ $\quad$ $a_6 = (\{leopard(T)\}, animal(T) \wedge leopard(T))$

The attack relation is drawn in Figure 2. Consequently, we get the following $\mathtt{Ext}_x(\mathcal{H})$ such that $x \in \{s, p\}$:

$$\mathcal{E}_1 = \{a_1, a_4, a_5\} \; \mathcal{E}_2 = \{a_2, a_3, a_6\}$$

– Consider $\mathcal{Q}_1 = \exists x(animal(x))$; we get $\mathcal{E}_1^{\mathcal{Q}_1} = \{a_5, a_4\}$ and $\mathcal{E}_2^{\mathcal{Q}_1} = \{a_3, a_6\}$.
– Consider now $\mathcal{Q}_2 = \exists x(animal(x) \wedge leopard(x))$; we get $\mathcal{E}_2^{\mathcal{Q}_2} = \{a_6\}$ and $\mathcal{E}_1^{\mathcal{Q}_2}$ does not exist.

$\mathtt{Ext}(\mathcal{H})^{\mathcal{Q}_1} = \{\mathcal{E}_1^{\mathcal{Q}_1}, \mathcal{E}_2^{\mathcal{Q}_1}\}$ is complete and $\mathtt{Ext}(\mathcal{H})^{\mathcal{Q}_2} = \{\mathcal{E}_2^{\mathcal{Q}_2}\}$ is not because the latter does not cover the extension $\mathcal{E}_1$ as it has no reduct.

Still the characterization of universal acceptance is not precise enough to give a complete account. Consider the set of all reducts w.r.t $\mathcal{Q} = \exists x(animal(x))$ which is $\{\{a_5, a_4\}, \{a_3, a_6\}\}$ (Example 14). This set holds the "reasons" to believe that $\mathcal{Q}$ is universally accepted. However, it would have been sufficient to just have $\{\{a_5\}, \{a_3\}\}$ or other combinations that keep in a *minimal way* those supporters which *preserve* the entailment of the query $\mathcal{Q}$ in each extension.

In combinatorics and diagnosis theory this problem is known as the hitting set problem [43]. It is also referred to as the transversal problem in hypergraph theory [31].

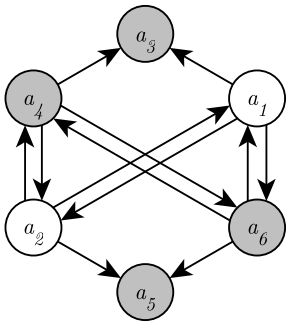


Fig. 2: Argument graph, the gray-colored arguments are the supporting arguments of the query $\mathcal{Q}_1$.

Definition 18 (Hitting set). *Given a collection $\mathcal{C} = \{S_1, ..., S_m\}$ of finite nonempty subsets of a set $\mathcal{B}$ (the background set), a* hitting set *of $\mathcal{C}$ is a set $A \subseteq \mathcal{B}$ such that $S_j \cap A \neq \emptyset$ for all $S_j \in \mathcal{C}$. A hitting set of $\mathcal{C}$ is minimal (w.r.t $\subseteq$) if and only if no proper subset of it is a hitting set of $\mathcal{C}$. A minimum hitting set is a minimal hitting set w.r.t set-cardinality.*

Finding one / all minimal / minimum hitting set is an interesting problem. It has a relation with different problems in different areas. In what follows we give a precise characterization of universal acceptance by means of the hitting set problem.

Definition 19 (Proponent set). *Let $\mathcal{Q}$ be a query and $\texttt{Ext}(\mathcal{H})^{\mathcal{Q}}$ be its complete reduct. The minimal hitting set S (w.r.t $\subseteq$) of $\texttt{Ext}(\mathcal{H})^{\mathcal{Q}}$ is called a proponent set. The minimum proponent set is defined as the smallest proponent set w.r.t to cardinality among all proponent sets.*

Example 15 (Example 14 cont'd). $\mathcal{Q} = \exists x(animal(x))$ has 4 proponent sets $S_1 = \{a_5, a_3\}$, $S_2 = \{a_5, a_6\}$, $S_3 = \{a_4, a_3\}$ and $S_4 = \{a_4, a_6\}$.

Following the definition, a proponent set is the smallest set of arguments that allows us to infer the query from all extensions. We obtain the following characterization, which is similar to the concept of a complete base [45].

Proposition 1. *$\mathcal{Q}$ is universally accepted $\Leftrightarrow$ $\mathcal{Q}$ has a proponent set.*

Proof. If Q is universally entailed then by definition $\forall \mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$, it holds that $\texttt{Concs}(\mathcal{E}_i) \models \mathcal{Q}$. This means that $\forall \mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$ there exists a set of facts $F_i \subseteq \texttt{Concs}(\mathcal{E}_i)$ such that $F_i \models \mathcal{Q}$, hence $\forall \mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$ we can construct a set of arguments A_i such that $\forall a_i \in A_i, F_i \subseteq \texttt{Conc}(a_i)$ and $F_i \models \mathcal{Q}$ (set of supporters from each $\mathcal{E}_i$). One can see that the set of all supporters $S = \{A_1, \ldots, \mathcal{A}_n\}$ over all extensions $\mathcal{E}_i$ where $i \in \{1, \ldots, n\}$ is actually a complete reduct. From here, it is straightforward to conclude that there does exist a proponent set as by definition for any collection of non-empty subsets there exists a minimal hitting set.

The left to right implication is trivial. The existence of a proponent set is entailed by the fact the query $\mathcal{Q}$ has a complete reduct (by definition). Therefore, for each extension $\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$ there exists at least one argument $a_i \in \mathcal{E}_i$ such that $\texttt{Conc}(a_i) \models \mathcal{Q}$. Consequently, for each extension $\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$ we have $\texttt{Conc}(a_i) \in \texttt{Concs}(\mathcal{E}_i)$, then this necessarily yields $\forall \mathcal{E}_i \in \texttt{Ext}_x(\mathcal{Q}), \texttt{Concs}(\mathcal{E}_i) \models \mathcal{Q}$.

A proponent set holds the smallest set of arguments which are distributed over all extensions and support the query $\mathcal{Q}$. So, if one extension does not contain any supporters then the query is not universally accepted.[1] The reason for the absence of such supporter is what we call the presence of a *block*. We follow the notion of a **block** from [38] and instantiate it in our setting. A block B is a set of arguments which are (1) all credulously accepted, (2) attack all the supporters of $\mathcal{Q}$, and (3) can all together be extended to form an extension.

Definition 20 (Block). *Let $\mathcal{Q}$ be a query and let $\mathcal{C} = \{\texttt{range}^-(a) | a \in \texttt{S}(\mathcal{Q})\}$. A set of arguments $B \subseteq \mathcal{A}$ is a block of $\mathcal{Q}$ under stable / preferred semantics if and only if:*

1. B is a hitting set of $\mathcal{C}$; and,
2. There exists an admissible set $B' \subseteq \mathcal{A}$ such that $B \subseteq B'$.

Note that a query may have more than one block. The minimum block is defined as the smallest block w.r.t to cardinality among all blocks.

Interestingly, a block is related to the hitting set problem. Informally, we take all the supporters of the query $\mathcal{Q}$ and for each supporter we get its attackers (i.e. $\texttt{range}^-(a)$) then we look for those hitting sets (over all sets of attackers) that can be extended to an extension, in other words those which belong to the same admissible set. Note that while a block is necessarily a hitting set it is not necessarily a minimal one.

The last requirement is very important as the following example shows.

Example 16 (Example 14 cont'd). Consider again the query $\mathcal{Q}_1 = \exists x(animal(x))$ and let us compute its block(s):

– $\texttt{S}(\mathcal{Q}_1) = \{a_3, a_4, a_5, a_6\}$.
– $\texttt{range}^-(a_3) = \texttt{range}^-(a_6) = \{a_1, a_4\}$.
– $\texttt{range}^-(a_4) = \texttt{range}^-(a_5) = \{a_2, a_6\}$.

We get the following hitting sets: $\{a_1, a_6\}$,$\{a_1, a_2\}$,$\{a_4, a_6\}$,$\{a_4, a_2\}$. Observe that none of them is considered as a block, because none of them can be extended to form an extension. In fact, they violate the conflict-freeness condition. While $\mathcal{Q}_1$ has no blocks, the query $\mathcal{Q}_2 = \exists x(animal(x) \wedge leopard(x))$ has two blocks $\{a_1\}$ and $\{a_4\}$.

Since the concept of a block characterizes exactly non-universal acceptance, we obtain the following result.

Proposition 2. *Suppose stable or preferred semantics. Then a query $\mathcal{Q}$ has a block if and only if it has no proponent set.*

[1] This is to be distinguished from universal non-acceptance, as the latter means that from all extensions the negation of the query is entailed.

Proof. Suppose that $\mathcal{Q}$ has a block B. Then, by definition, B attacks all the supporters of $\mathcal{Q}$. In addition, B is a subset of an admissible set, which means that there exists at least one extension $\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$ such that $B \subseteq \mathcal{E}_i$. By conflict-freeness of extensions, $\mathcal{E}_i$ does not contain any supporter of query $\mathcal{Q}$. Consequently, the reduct $\texttt{Ext}(\mathcal{H})^{\mathcal{Q}}$ is not complete. Hence, there is no proponent set for the query $\mathcal{Q}$.

Suppose now that $\mathcal{Q}$ has a proponent set. Then, it has a complete reduct. Consequently, for each extension there exists an argument $a_i \in \mathcal{E}_i$ that supports the query $\mathcal{Q}$. Therefore, there exists no admissible set A that contains a set of arguments that attacks all the supporters. This proves that there is no block for the query $\mathcal{Q}$.

Proponent sets and blocks can be seen as causes describing why a query is universally accepted or not. They describe precisely the reasons behind the acceptance or non-acceptance of the query. Moreover, if a query is (not) universally accepted then there is always an explanation (block or proponent set) to explain its state, which is an interesting feature.

In the next section we show the relation between the output of argumentation frameworks of this instantiation and inconsistency-tolerant semantics studied in Section 2.2.

4 Equivalences with Inconsistency-Tolerant Semantics

Given an inconsistent knowledge base $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$, a repair is a maximal (w.r.t $\subseteq$) consistent set of facts $F \subseteq \mathcal{F}$. We show that there is a correspondence between repairs of a knowledge base and the extensions of the corresponding argumentation framework.

Theorem 1. *Let $\mathcal{K}$ be a knowledge base, $\mathcal{H}$ its corresponding argumentation framework and $x \in \{s, p\}$. Then:*

$$\texttt{Ext}_x(\mathcal{H}) = \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$$

Proof. The plan of the proof is as follows:

1. We prove that $\{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\} \subseteq \texttt{Ext}_s(\mathcal{H})$.
2. We prove that $\texttt{Ext}_p(\mathcal{H}) \subseteq \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$.
3. Since every stable extension is a preferred one [29], we can proceed as follows. From the first item, we have that $\{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\} \subseteq \texttt{Ext}_p(\mathcal{H})$, thus the theorem holds for preferred semantics. From the second item we have that $\texttt{Ext}_s(\mathcal{H}) \subseteq \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$, thus the theorem holds for stable semantics.

1. We first show $\{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\} \subseteq \texttt{Ext}_s(\mathcal{H})$. Let $\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})$ and let $\mathcal{E} = \texttt{Args}(\mathcal{P}')$. Let us prove that $\mathcal{E}$ is a stable extension of $(\texttt{Args}(\mathcal{F}), \mathcal{X})$.
   We first prove that $\mathcal{E}$ is conflict-free. By means of contradiction we suppose the contrary, i.e. let $a, b \in \mathcal{E}$ such that $(a, b) \in \mathcal{X}$. From the definition of attack, there exists $\varphi \in \texttt{Supp}(b)$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(a), \varphi\})$ is inconsistent. Thus $C\ell^*_{\mathcal{R}}(\texttt{Supp}(a) \cup \{\varphi\})$ is inconsistent; consequently $\mathcal{P}'$ is inconsistent, contradiction. Therefore $\mathcal{E}$ must be conflict-free.

Let us prove that $\mathcal{E}$ attacks all arguments outside the set. Let $b \in \texttt{Args}(\mathcal{F}) \setminus \texttt{Args}(\mathcal{P}')$ and let $\varphi \in \texttt{Supp}(b)$, such that $\varphi \notin \mathcal{P}'$. For a set $X = \{x_1, \ldots, x_n\}$, let us denote by $\bigwedge X$ the formula $x_1 \wedge \ldots \wedge x_n$. Let $a = (\mathcal{P}', \bigwedge \mathcal{P}')$. We have $\varphi \notin \mathcal{P}'$, so, due to the set inclusion maximality of the repairs, $C\ell^*_{\mathcal{R}}(\{\bigwedge \mathcal{P}', \varphi\})$ is inconsistent. Therefore, $(a, b) \in \mathcal{X}$. Consequently, $\mathcal{E}$ is a stable extension.

2. We now need to prove that $\texttt{Ext}_p(\mathcal{H}) \subseteq \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$. Let $\mathcal{E} \in \texttt{Ext}_p(\mathcal{H})$ and let us prove that there exists a repair $\mathcal{P}'$ such that $\mathcal{E} = \texttt{Args}(\mathcal{P}')$. Let $S = \texttt{Base}(\mathcal{E})$. Let us prove that S is consistent. Aiming to a contradiction, suppose that S is inconsistent. Let $S' \subseteq S$ be such that (1) S' is inconsistent and (2) every proper set of S' is consistent. Let us denote $S' = \{\varphi_1, \varphi_2, ..., \varphi_n\}$. Let $a \in \mathcal{E}$ be an argument such that $\varphi_n \in \texttt{Supp}(a)$. Let $a' = (S' \setminus \{\varphi_n\}, \varphi_1 \wedge \ldots \wedge \varphi_{n-1})$. We have that $(a', a) \in \mathcal{X}$. Since $\mathcal{E}$ is conflict free, then $a' \notin \mathcal{E}$. Since $\mathcal{E}$ is an admissible set, there exists $b \in \mathcal{E}$ such that $(b, a') \in \mathcal{X}$. Since b attacks a' then there exists $i \in \{1, 2, ..., n-1\}$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b), \varphi_i\})$ is inconsistent. Since $\varphi_i \in \texttt{Base}(\mathcal{E})$, then there exists $c \in \mathcal{E}$ such that $\varphi_i \in \texttt{Supp}(c)$. Thus $(b, c) \in \mathcal{X}$, contradiction. So it must be that S is consistent.
   Let us now prove that there exists no $S' \subseteq \mathcal{F}$ such that $S \subsetneq S'$ and S' is consistent. We use the proof by contradiction. Thus, suppose that S is not a maximal consistent subset of $\mathcal{F}$. Then, there exists $S' \in \mathcal{R}epair(\mathcal{K})$, such that $S \subsetneq S'$. We have that $\mathcal{E} \subseteq \texttt{Args}(S)$, since $S = \texttt{Base}(\mathcal{E})$. Denote $\mathcal{E}' = \texttt{Args}(S')$. Since $S \subsetneq S'$ then $\texttt{Args}(S) \subsetneq \mathcal{E}'$. Thus, $\mathcal{E} \subsetneq \mathcal{E}'$. From the first part of the proof, $\mathcal{E}' \in \texttt{Ext}_s(\mathcal{H})$. Consequently, $\mathcal{E}' \in \texttt{Ext}_p(\mathcal{H})$. We also know that $\mathcal{E} \in \texttt{Ext}_p(\mathcal{H})$. Contradiction, since no preferred set can be a proper subset of another preferred set. Thus, we conclude that $\texttt{Base}(\mathcal{E}) \in \mathcal{R}epair(\mathcal{K})$.
   Let us show that $\mathcal{E} = \texttt{Args}(\texttt{Base}(\mathcal{E}))$. It must be that $\mathcal{E} \subseteq \texttt{Args}(S)$. Also, we know (from the first part) that $\texttt{Args}(S)$ is a stable and a preferred extension, thus the case $\mathcal{E} \subsetneq \texttt{Args}(S)$ is not possible.
3. Now we know that $\{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\} \subseteq \texttt{Ext}_s(\mathcal{H})$ and $\texttt{Ext}_p(\mathcal{H}) \subseteq \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$. The theorem follows from those two facts, as explained at the beginning of the proof.

To prove Theorem 2, we first prove the following lemma which says that if there are no rejected arguments under preferred semantics, then the grounded extension is equal to the intersection of all preferred extensions. Note that this result holds for every argumentation framework (not only for the one studied in this paper, where arguments are constructed from an ontological knowledge base). Thus, we only suppose that we are given a set and a binary relation on it (called attack relation).

Lemma 1. *Let $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ be an argumentation framework and* GE *its grounded extension.*

$$\textit{If } \mathcal{A} \subseteq \bigcup_{\mathcal{E}_i \in \texttt{Ext}_p(\mathcal{H})} \mathcal{E}_i \quad \textit{then} \quad \texttt{GE} = \bigcap_{\mathcal{E}_i \in \texttt{Ext}_p(\mathcal{H})} \mathcal{E}_i.$$

Proof. Let

$$\texttt{Iope} = \bigcap_{\mathcal{E}_i \in \texttt{Ext}_p(\mathcal{H})} \mathcal{E}_i$$

denote the intersection of all preferred extensions. It is known [28] that $\texttt{GE} \subseteq \texttt{Iope}$. Let us prove that in the case when there are no rejected arguments, it also holds that $\texttt{Iope} \subseteq \texttt{GE}$. Let $a \in \texttt{Iope}$. Let us show that no argument b attacks a. This holds since every argument b is in at least one preferred extension, say $\mathcal{E}_i$, and a is also in $\mathcal{E}_i$ (since a is in all preferred extensions) thus b does not attack a since both a and b are in $\mathcal{E}_i$ and $\mathcal{E}_i$ is a conflict-free set (since it is a preferred extension). All this means that arguments in $\texttt{Iope}$ are not attacked. Consequently, they must all belong to the grounded extension. In other words, $\texttt{Iope} \subseteq \texttt{GE}$.

We can now, using the previous result, prove the link between the intersection of repairs and the grounded extension.

Theorem 2. *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base and $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework. Denote the grounded extension of $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ by* $\texttt{GE}$*. Then:*

$$\texttt{GE} = \texttt{Args}\left(\bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} \mathcal{P}'\right)$$

Proof. Denote the intersection of all repairs by

$$\texttt{Ioar} = \bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} \mathcal{P}'$$

and the intersection of all preferred extensions by

$$\texttt{Iope} = \bigcap_{\mathcal{E}_i \in \texttt{Ext}_p(\mathcal{H})} \mathcal{E}_i.$$

From Theorem 1, we know that $\texttt{Ext}_p(\mathcal{H}) = \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$. Consequently,

$$\texttt{Iope} = \bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} \texttt{Args}(\mathcal{P}') \tag{1}$$

Since every argument has a consistent support, then its support is in at least one repair. From Theorem 1, that argument is in at least one preferred extension, (i.e. it is not rejected). From Lemma 1,

$$\texttt{Iope} = \texttt{GE} \tag{2}$$

From (1) and (2), we obtain that

$$\texttt{GE} = \bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} \texttt{Args}(\mathcal{P}') \tag{3}$$

Note that for every collection $\{S_1, \ldots, S_n\}$ of sets of formulae, we have $\texttt{Args}(S_1) \cap \ldots \cap \texttt{Args}(S_n) = \texttt{Args}(S_1 \cap \ldots \cap S_n)$. By applying this rule on the set of all repairs, we obtain:

$$\bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} \texttt{Args}(\mathcal{P}') = \texttt{Args}(\texttt{Ioar}) \tag{4}$$

From (3) and (4), we obtain $\texttt{GE} = \texttt{Args}(\texttt{Ioar})$ which ends the proof.

The previous two theorems allow us to show one of the main resutls of the paper: the links between semantics from argumentation theory (stable, preferred, grounded) and semantics from inconsistent ontology KB query answering (ICR, CQA, IAR). More precisely, we show that:

– skeptical acceptance under stable and preferred semantics corresponds to ICR semantics;
– universal acceptance under stable and preferred semantics corresponds to CQA semantics;
– acceptance under grounded semantics corresponds to IAR semantics.

Theorem 3. *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, let $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ be the corresponding argumentation framework and let $\mathcal{Q}$ be a query. Let $x \in \{s, p\}$ be stable or preferred semantics. Then:*

– *$\mathcal{K} \models_{\mathcal{ICR}} \mathcal{Q}$ iff $\mathcal{Q}$ is skeptically accepted under semantics x.*
– *$\mathcal{K} \models_{\mathcal{CQA}} \mathcal{Q}$ iff $\mathcal{Q}$ is universally accepted under semantics x.*

Proof. Theorem 1 implies $\texttt{Ext}_x(\mathcal{H}) = \{\texttt{Args}(\mathcal{P}') \mid \mathcal{P}' \in \mathcal{R}epair(\mathcal{K})\}$. In fact, the restriction of function Args on $\mathcal{R}epair(\mathcal{K})$ is a bijection between $\mathcal{R}epair(\mathcal{K})$ and $\texttt{Ext}_x(\mathcal{H})$. Note also that for every query $\mathcal{Q}$, for every repair $\mathcal{P}'$, we have that $C\ell^*_{\mathcal{R}}(\mathcal{P}') \models \mathcal{Q}$ if and only if $\texttt{Concs}(\texttt{Args}(\mathcal{P}')) \models \mathcal{Q}$. By using those two facts, the result of the theorem can be obtained as follows:

– For every query $\mathcal{Q}$, we have: $\mathcal{K} \models_{\mathcal{ICR}} \mathcal{Q}$ if and only if

$$\bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} C\ell^*_{\mathcal{R}}(\mathcal{P}') \models \mathcal{Q}$$

if and only if

$$\bigcap_{\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})} \texttt{Concs}(\mathcal{E}_i) \models \mathcal{Q}$$

if and only if

$$\texttt{Output}_x(\mathcal{H}) \models \mathcal{Q}$$

if and only if

$\mathcal{Q}$ is skeptically accepted.

– For every query $\mathcal{Q}$, we have: $\mathcal{K} \models_{\mathcal{CQA}} \mathcal{Q}$ if and only if

$$\text{for every } \mathcal{P}' \in \mathcal{R}epair(\mathcal{K}),\ C\ell^*_{\mathcal{R}}(\mathcal{P}') \models \mathcal{Q}$$

if and only if

$$\text{for every } \mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H}),\ \texttt{Concs}(\mathcal{E}_i) \models \mathcal{Q}$$

if and only if

$\mathcal{Q}$ is universally accepted.

This ends the proof.

The next theorem shows the link between grounded argumentation semantics and IAR semantics.

Theorem 4. *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, let $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ be the corresponding argumentation framework and let $\mathcal{Q}$ be a query. Then:*

$$\mathcal{K} \models_{\mathcal{IAR}} \mathcal{Q} \textit{ iff } \mathcal{Q} \textit{ is accepted under grounded semantics}.$$

Proof. Let us denote the grounded extension of $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ by $\mathtt{GE}$ and the intersection of all repairs by

$$\mathtt{Ioar} = \bigcap_{\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})} \mathcal{P}'.$$

From Definition 14, we have:

$$\mathcal{Q} \text{ is accepted under grounded semantics iff } \mathtt{Concs}(\mathtt{GE}) \models \mathcal{Q}. \tag{5}$$

From Theorem 2, we have:

$$\mathtt{GE} = \mathtt{Args}(\mathtt{Ioar}). \tag{6}$$

Note also that for every set of facts $\{F_1, \ldots, F_n\}$ and for every query $\mathcal{Q}$, we have that $C\ell^*_{\mathcal{R}}(\{F_1, \ldots, F_n\}) \models \mathcal{Q}$ if and only if $\mathtt{Concs}(\mathtt{Args}(\{F_1, \ldots, F_n\})) \models \mathcal{Q}$. Thus,

$$C\ell^*_{\mathcal{R}}(\mathtt{Ioar}) \models \mathcal{Q} \text{ if and only if } \mathtt{Concs}(\mathtt{Args}(\mathtt{Ioar})) \models \mathcal{Q}. \tag{7}$$

From (6) and (7) we have that:

$$C\ell^*_{\mathcal{R}}(\mathtt{Ioar}) \models \mathcal{Q} \text{ if and only if } \mathtt{Concs}(\mathtt{GE}) \models \mathcal{Q}. \tag{8}$$

From Definition 5, we have that:

$$C\ell^*_{\mathcal{R}}(\mathtt{Ioar}) \models \mathcal{Q} \text{ if and only if } \mathcal{K} \models_{\mathcal{IAR}} \mathcal{Q}. \tag{9}$$

The theorem now follows from (5), (8) and (9).

5 Properties of the Instantiation

This section studies the properties and postulates from the literature [2, 19] that are satisfied by our logic-based instantiation.

5.1 Basic properties

In this subsection we provide some important properties for the class of argumentation we defined in this section. We investigate finiteness, non-triviality, coherence, relative groundedness and well-foundedness.

Class	Name	Description
FINITE	Finite	$\mathcal{H}$ has a finite set of arguments.
NTRIVIAL	Non-trivial	$\forall \mathcal{H}, \texttt{Ext}_p(\mathcal{H}) \neq \{\{\}\}$.
COHERENT	Coherent	$\forall \mathcal{H}, \texttt{Ext}_p(\mathcal{H}) = \texttt{Ext}_s(\mathcal{H})$.
RGROUNDED	Relatively grounded	$\texttt{GE} = \bigcap \texttt{Ext}_p(\mathcal{H})$.
WFOUNDED	Well-founded	$\forall \mathcal{H}, \texttt{Ext}_c(\mathcal{H}) = \texttt{Ext}_g(\mathcal{H}) = \texttt{Ext}_p(\mathcal{H}) = \texttt{Ext}_s(\mathcal{H})$ and $\lvert\texttt{Ext}(\mathcal{H})\rvert = 1$ such that c refers to the complete semantics.

Table 1: Classes of argumentation frameworks studied in the literature. Note that these classes are with respect to those argumentation frameworks that contain at least one argument. Recall that p and s refers to preferred and stable respectively, and that GE denoted the grounded extension.

It is obvious that in the case of a consistent knowledge base, the corresponding argumentation framework has exactly one extension, which contains all arguments, independently of the semantics used.

As we handle inconsistency in existential rules, in this paper we are interested in the class of argumentation frameworks that are built over **inconsistent** knowledge bases having at least one minimal conflict of size two or more. This class is denoted as ARG$_\exists$.

It turns out that the argumentation frameworks of ARG$_\exists$ enjoy the finiteness property.

Definition 21 (Finite argument). *An argument a is finite if and only if $\texttt{Conc}(a)$ is finite.*

This means that the conclusion of the argument does not contain infinite conjunctions. Note that, since $\texttt{Supp}(a)$ is constructed from a finite set of facts $\mathcal{F}$, it is finite.

Proposition 3 (Finiteness). $\forall \mathcal{H} \in$ ARG$_\exists$ *the following holds: (i)* $\forall a \in \mathcal{A}$, *$a$ is finite; and (ii)* $\mathcal{H} \in$ FINITE.

Proof. (i) Let $\texttt{Conc}(a) = \alpha_0 \wedge \alpha_2 \wedge \ldots$ be infinite. By definition $\forall \alpha_i \in \{\alpha_0, \alpha_2, \ldots\}, \alpha_i \in C\ell^*_{\mathcal{R}}(\texttt{Supp}(a))$. This means $C\ell^*_{\mathcal{R}}(\texttt{Supp}(a))$ is infinite. This is in contradiction with the assumption we have made in the logical language section about the finiteness of the saturation. (ii) If $\mathcal{A}$ was not finite then we would have arguments with infinite conclusions, which is not the case.

Let us now prove that maximal consistent sets give rise to admissible sets of arguments. For a set $X = \{x_1, \ldots, x_n\}$, notation $\bigwedge X$ stands for $x_1 \wedge \ldots \wedge x_n$.

Proposition 4 (Sentinel).
Let $\mathcal{H} \in$ ARG$_\exists$ and $a = (A, \bigwedge A) \in \mathcal{A}$, if A is a repair then $\{a\}$ is an admissible set.

Proof. Assume that there exists an argument b that attacks a. That means that there exists $h \in \texttt{Supp}(a)$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b), h\})$ is inconsistent, consequently $C\ell^*_{\mathcal{R}}(\{\texttt{Supp}(b) \cup \{h\}\})$ is inconsistent.

By means of contradiction, assume that a does not attack b. Then there exists no $h' \in \texttt{Supp}(b)$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(a), h'\})$ is inconsistent. By maximality we conclude that $\texttt{Supp}(b) \subseteq A$. But according to the conclusion above, $C\ell^*_{\mathcal{R}}(\texttt{Supp}(b) \cup \{h\})$ is inconsistent which is a contradiction with the fact that A is consistent.

As a result of this proposition we get the following result.

Proposition 5. *Let $a \in \mathcal{A}$ be an argument of the form $a = (A, \bigwedge A)$, where A is a repair. Then, for every argument b such that $\texttt{Supp}(b) \subseteq \texttt{Supp}(a)$ we have that a defends b. Consequently, $\{a, b\}$ is an admissible set.*

Proof. If there exists an argument c such that c attacks b then there exists $h \in \texttt{Supp}(b)$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(c), h\})$ is inconsistent. Since $h \in \texttt{Supp}(a)$ then c also attacks b. From Proposition 4 the argument a defends itself from all attacks, hence a attacks c.

We saw that argumentation frameworks over **consistent** knowledge bases always had one non-empty extension. In what follows, we prove that every $\mathcal{H} \in \text{ARG}_\exists$ is non-trivial.

Proposition 6 (Non-triviality). *Under preferred / stable semantics we have that*

$$\text{ARG}_\exists \subseteq \text{NTRIVIAL}$$

Proof. By means of contradiction, suppose that there exists $\mathcal{H} \in \text{ARG}_\exists$ such that $\texttt{Ext}_p(\mathcal{H}) = \{\emptyset\}$. This means that the only admissible set is the empty set. This is in contradiction with Proposition 4 which states that the argument $a = (A, \bigwedge A)$, such that A is a repair, is an admissible set. Thus, there exists a non-empty admissible set. Consequently, there exists a non-empty preferred extension. Since preferred and stable semantics coincide, the result holds for stable semantics.

Note the curious fact that no framework in $\text{ARG}_\exists$ has rejected arguments.

Proposition 7 (Rejected arguments). *For all $\mathcal{H} \in \text{ARG}_\exists$, for all $x \in \{p, s\}$:*

$$\mathcal{A}_\mathcal{H} = \bigcup_{\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})} \mathcal{E}_i$$

Proof. According to Proposition 5, for every argument b there exists an argument $a = (A, \bigwedge A)$ such that $\texttt{Supp}(b) \subseteq \texttt{Supp}(a)$ and A is a repair and $\{a, b\}$ is admissible. Therefore all arguments are defended, hence there are no rejected arguments.

This means that all the arguments are credulously accepted under the preferred/stable semantics. An argumentation framework is said to be coherent if its preferred and stable extensions coincide. The class of coherent argumentation frameworks is denoted by COHERENT. Many algorithms and proof procedures have been developed for this class [38]. In what follows we prove that $\text{ARG}_\exists$ is within the class of coherent argumentation frameworks.

Proposition 8 (Coherence). $\text{ARG}_\exists \subseteq \text{COHERENT}$.

Proof. The proof follows directly from Theorem 1, page 17.

From Lemma 1 and Proposition 7 we conclude that the grounded extension is equal to the intersection of all preferred extensions:

Proposition 9 (Relative groundedness). $\text{ARG}_\exists \subseteq \text{RGROUNDED}$.

Let us now show that no framework in $\text{ARG}_\exists$ is well founded. We start with a lemma.

Lemma 2. *Let $\mathcal{K}$ be the knowledge base of an argumentation framework $\mathcal{H} \in \text{ARG}_\exists$. Let C be a minimal conflict of $\mathcal{K}$, let $C = S \cup S'$, such that $S \neq \emptyset$, $S' \neq \emptyset$ and $|S| = |S'| = |C| - 1$. Let $a = (S, \bigwedge S)$ and $b = (S', \bigwedge S')$. Then, $(a, b) \in \mathcal{X}$ and $(b, a) \in \mathcal{X}$.*

Proof. We show that the two arguments are well defined and attack each other.

– It is clear that $\texttt{Supp}(a)$ and $\texttt{Supp}(b)$ are consistent because by definition any subset of a minimal conflict is consistent.
– Let us prove that a attacks b. There exists $h \in S'$ such that $C = S \cup \{h\}$. Consequently, $C\ell^*_{\mathcal{R}}(S \cup \{h\})$ is inconsistent, in other words $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(a) \cup h\})$ is inconsistent, which means that a attacks b.
– For the same reasons, b attacks a.

Proposition 10 (Cyclicity). *$\forall \mathcal{H} \in \text{ARG}_\exists$, $\mathcal{H}$ has at least one cycle.*

Proof. The proof follows from Lemma 2.

Dung [29] introduces the notion of well-foundedness. He shows [29] that if an argumentation framework does not have an infinite sequence $a_0, a_1, \ldots$ such that for each i, a_{i+1} attacks a_i then it has a unique complete / preferred / stable / grounded extension. The following result is immediate.

Proposition 11 (Well-foundedness). $\text{ARG}_\exists \cap \text{WFOUNDED} = \emptyset$.

5.2 Postulate satisfaction

In this chapter, we prove that our instantiation satisfies the rationality postulates [2], namely Closure under $C\ell$, Consistency, Closure under sub-arguments, Strong consistency, Exhaustiveness and Free precedence. We first prove that the set of conclusions of every extension is closed under $C\ell$.

Proposition 12 (Closure under $C\ell$). *Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ the corresponding argumentation framework and $x \in \{s, p, g\}$. Then:*

– *for every $\mathcal{E}_i \in \texttt{Ext}_x(\mathcal{H})$, $\texttt{Concs}(\mathcal{E}_i) = C\ell^*_{\mathcal{R}}(\texttt{Concs}(\mathcal{E}_i))$.*

Proof. From the definition of $C\ell$, we see that $\texttt{Concs}(\mathcal{E}_i) \subseteq C\ell^*_{\mathcal{R}}(\texttt{Concs}(\mathcal{E}_i))$. Let us prove that $C\ell^*_{\mathcal{R}}(\texttt{Concs}(\mathcal{E}_i)) \subseteq \texttt{Concs}(\mathcal{E}_i)$. Suppose that $\alpha \in C\ell^*_{\mathcal{R}}(\texttt{Concs}(\mathcal{E}_i))$. Since $\mathcal{E}_i$ is a preferred, a stable or the grounded extension, Theorems 1 and 2 imply that there exists a set of formulae S such that $\mathcal{E}_i = \texttt{Args}(S)$. Consequently, $\mathcal{E}_i = \texttt{Args}(\texttt{Base}(\mathcal{E}_i))$. Since the supports of arguments of $\mathcal{E}_i$ contain only formulas from S, we have that $\alpha \in C\ell^*_{\mathcal{R}}(S)$. Thus, there exists an argument $a \in \mathcal{E}_i$ s.t. $\texttt{Conc}(a) = \alpha$.

We now prove that the set of conclusions of every extension is consistent.

Proposition 13 (Consistency). *For every $\mathcal{H} \in \text{ARG}_\exists$, for every $\mathcal{E} \in \texttt{Ext}_x(\mathcal{H})$ such that $x \in \{s, p, g\}$, $\texttt{Concs}(\mathcal{E})$ is consistent.*

Proof. Let $\mathcal{E}_i$ be a stable or a preferred extension of $\mathcal{H} = (\mathcal{A}, \mathcal{X})$. From Theorem 1, there exists a repair $\mathcal{P}' \in \mathcal{R}epair(\mathcal{K})$ such that $\mathcal{E}_i = \texttt{Args}(\mathcal{P}')$. Note that $\texttt{Concs}(\mathcal{E}_i) = C\ell^*_{\mathcal{R}}(\mathcal{P}')$. And since $\mathcal{P}'$ is a repair then $C\ell^*_{\mathcal{R}}(\mathcal{P}')$ is consistent. Therefore, $\texttt{Concs}(\mathcal{E}_i)$ is consistent.

Let us now consider the case of grounded semantics. Denote GE the grounded extension of $\mathcal{H} = (\mathcal{A}, \mathcal{X})$. We have just seen that for every $\mathcal{E}_i \in \texttt{Ext}_p(\mathcal{H})$, it holds that $\texttt{Concs}(\mathcal{E}_i)$ is a consistent set. Since the grounded extension is a subset of the intersection of all the preferred extensions, and since there is at least one preferred extension, say $\mathcal{E}_1$, then $\texttt{GE} \subseteq \mathcal{E}_1$. Since $\texttt{Concs}(\mathcal{E}_1)$ is consistent then $\texttt{Concs}(\texttt{GE})$ is also consistent.

An argument $a' = (H', C')$ is a sub-argument of another argument $a = (H, C)$ iff $H' \subseteq H$. We now show that every extension contains all sub-arguments of each argument it contains.

Proposition 14 (Closure under sub-arguments). *For every $\mathcal{H} \in \text{ARG}_\exists$, for every $\mathcal{E} \in \texttt{Ext}_x(\mathcal{H})$ such that $x \in \{s, p, g\}$, if $a \in \mathcal{E}$ then for every sub-argument a' of a we have that $a' \in \mathcal{E}$.*

Proof.

– *Preferred/stable*: Let $a \in \mathcal{E}$ and let a' be a sub-argument of a such that $a' \notin \mathcal{E}$. Aiming to a contradiction, suppose $a' \notin \mathcal{E}$. Since $\mathcal{E}$ is a stable extension, there exists $b \in \mathcal{E}$ such that b attacks a'. This means that $\exists h \in \texttt{Supp}(a')$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b), h\})$ is inconsistent. Since a' is a sub-argument of a then by definition $\texttt{Supp}(a') \subseteq \texttt{Supp}(a)$, consequently $h \in \texttt{Supp}(a)$. Therefore, $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b), h\})$ which yields b attacks a. This is clearly in contradiction with the fact that $\mathcal{E}$ is a stable extension that has to be conflict free.
– *Grounded*: Denote the grounded extension by GE. From Proposition 7, we know that there are no rejected arguments. From Lemma 1, the grounded extension is equal to the intersection of preferred extensions. This means that if $a \in \texttt{GE}$, then a is not attacked (since a and its attacker would be in the same preferred extension, which is impossible). Let $a \in \texttt{GE}$ and let a' be one of its sub-arguments. Since $\texttt{Supp}(a') \subseteq \texttt{Supp}(a)$ then a' is unattacked too. Observe that an unattacked argument must belong to the groudned extension. Hence, $a' \in \texttt{GE}$.

We now show that the base of every extension is consistent.

Corollary 1 (Strong consistency). *For every $\mathcal{H} \in \text{ARG}_{\exists}$, for every $\mathcal{E} \in \texttt{Ext}_x(\mathcal{H})$ such that $x \in \{s, p, g\}$, $\texttt{Base}(\mathcal{E})$ is consistent.*

Proof.
– *Preferred/stable*: Let $S = \texttt{Base}(\mathcal{E})$. From Theorem 1, $\mathcal{E} = \texttt{Args}(S)$ and S is a repair. Thus, S is consistent.
– *Grounded*: Let $S = \texttt{Base}(\mathcal{E})$. From Theorem 2, $\mathcal{E} = \texttt{Args}(S)$ and there exist a repair $\mathcal{P}$ such that $S \subseteq \mathcal{P}$. Thus, S is consistent.

Roughly speaking, exhaustiveness says that if the support and the conclusion of argument a belong to the conclusions of extension $\mathcal{E}$, then a belongs to $\mathcal{E}$.

Proposition 15 (Exhaustiveness). *For every $\mathcal{H} \in \text{ARG}_{\exists}$, for every $\mathcal{E} \in \texttt{Ext}_x(\mathcal{H})$ such that $x \in \{s, p, g\}$, for every $a = (H, C) \in \mathcal{A}$, if $H \cup \{C\} \subseteq \texttt{Concs}(\mathcal{E})$, then $a \in \mathcal{E}$.*

Proof.
– *Preferred and stable:* Aiming to a contradiction, assume that there exists $a = (H, C) \in \mathcal{A}$ such that $H \cup \{C\} \subseteq \texttt{Concs}(\mathcal{E})$ and $a \notin \mathcal{E}$. This means that there exists an argument $b \in \mathcal{E}$ such that b attacks a because $\mathcal{E}$ is a stable extension. Consequently, there exists $h \in \texttt{Supp}(a)$ such that $C\ell^*_{\mathcal{R}}(\{h, \texttt{Conc}(b)\})$ is inconsistent. However, we know that $\texttt{Supp}(a) \subseteq \texttt{Concs}(\mathcal{E})$ and $\texttt{Conc}(b) \in \texttt{Concs}(\mathcal{E})$. This indicates that $\texttt{Concs}(\mathcal{E})$ is inconsistent which is in contradiction with the Proposition 13.
– *Grounded:* Aiming to a contradiction, assume that there exists $a = (H, C) \in \mathcal{A}$, such that $H \cup \{C\} \subseteq \texttt{Concs}(\mathcal{E})$ and $a \notin \mathcal{E}$. Recall that the grounded extension is the intersection of preferred extensions. This means that there exists a preferred extension $\mathcal{E}'$ such that $a \notin \mathcal{E}'$. Therefore, there exists $b \in \mathcal{E}'$ such that b attacks a (as $\mathcal{E}'$ is also a stable extension). Consequently, there exists $h \in H$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b), h)\})$ is inconsistent. Therefore, $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b)\} \cup H)$ is inconsistent. Since the grounded extension $\mathcal{E}$ is a subset of the preferred extension $\mathcal{E}'$ and $H \subseteq \texttt{Concs}(\mathcal{E})$, we get $H \cup \{\texttt{Conc}(b)\} \subseteq \mathcal{E}'$. This yields the inconsistency of $\mathcal{E}'$, which is in contradiction with Proposition 13.

We now show that if an argument is constructed from information that does not appear in any minimal conflict, then it belongs to all extensions.

Proposition 16 (Free precedence). *For every $\mathcal{H} \in \text{ARG}_{\exists}$, for every $\mathcal{E} \in \texttt{Ext}_x(\mathcal{H})$ such that $x \in \{s, p, g\}$:*

$$\texttt{Args}(\texttt{Free}(\mathcal{K})) \subseteq \mathcal{E},$$

where

– *$\mathcal{K}$ is the knowledge base $\mathcal{H}$ was built from and*
– $\texttt{Free}(\mathcal{K}) = \mathcal{F} \setminus \bigcup_{C \text{ is a minimal conflict}} C$

Proof. Observe that $\texttt{Free}(\mathcal{K})$ is consistent with every other consistent set of facts $\mathcal{F}' \subseteq \mathcal{F}$. Let us now suppose that $a \in \mathcal{A}$ is an argument such that $\texttt{Supp}(a) \subseteq \texttt{Free}(\mathcal{K})$. We claim that a is unattacked. By aiming to a contradiction, assume that there exists an argument $b \in \mathcal{A}$ such that b attacks a. By definition, there exists $h \in \texttt{Supp}(a)$ such that $C\ell^*_{\mathcal{R}}(\{\texttt{Conc}(b), h\})$ is inconsistent. Consequently, $C\ell^*_{\mathcal{R}}(\texttt{Free}(\mathcal{K}) \cup \texttt{Supp}(b))$ is inconsistent. Recall that $\texttt{Supp}(b)$ is consistent. Therefore, $\texttt{Free}(\mathcal{K})$ is inconsistent with the consistent set of facts $\texttt{Supp}(b)$, which is a contradiction. So it must be that a is unattacked. Therefore, it belongs to every extension.

6 Dialectical Proof Theory for Universal Acceptance

In this section we are interested by the problem of query acceptance in logic-based argumentation frameworks, namely universal acceptance under preferred/stable semantics. We show how this problem can be solved through a dialogue game between an opponent and a proponent over a given query. This dialogue results in what is called *a dialectical proof*. Note that in this section we limit ourselves to *ground* boolean conjunctive queries (BCQ without variables). In what follows we describe an argument game that we will explain on a detailed example in Subsection 6.1. Next, in Subsection 6.2 we prove its completeness and soundness and we study certain aspects concerning its complexity.

Given a query $\mathcal{Q}$ and an argumentation framework $\mathcal{H}$, the preferred universal dialectical proof theory is a two-person argument game between a proponent (PRO) and an opponent (OPP). PRO takes the position of supporting the query $\mathcal{Q}$ while OPP takes the opposite position. The proponent and the opponent are engaged in an argumentation dialogue with *precisely defined* types of moves. The goal is to determine whether the query is universally accepted or not at the end of the dialogue.

Let us formally define what a dialogue is.

Definition 22 (Dialogue). *Let $\mathcal{H} = (\mathcal{A}, \mathcal{X})$ be an argumentation framework. A dialogue based on $\mathcal{H}$ is a finite non-empty sequence $d_n = (m_1, \ldots, m_n)$ of moves where each m_i is either:*

– *a support move:* $\text{SUPPORT}(a)$ *such that* $a \in \mathcal{A}$.
– *a counter move:* $\text{COUNTER}(A)$ *such that* $A \subseteq \mathcal{A}$.
– *a retrace move:* $\text{RETRACE}(A, i)$ *such that* $A \subseteq \mathcal{A}$ *and* $i < n$.

We denote by $\texttt{Player}(m_i)$ *the player of the move.* $\texttt{Player}(m_i) = \texttt{OPP}$ *iff* i *is even,* $\texttt{Player}(m_i) = \texttt{PRO}$ *iff* i *is odd. We denote by* $d \cdot d'$ *and* $d \cdot m$ *the concatenation of the dialogues* d *and* d' *and the dialogue* d *with the move* m *respectively.*

Given a move m, we denote by $\texttt{Arg}(m)$ and $\texttt{Sp}(m)$ the content and the speech act of the move as follows:

– $m = \text{SUPPORT}(a)$, $\texttt{Arg}(m) = a$ and $\texttt{Sp}(m) = \text{SUPPORT}$.
– $m = \text{COUNTER}(A)$, $\texttt{Arg}(m) = A$ and $\texttt{Sp}(m) = \text{COUNTER}$.
– $m = \text{RETRACE}(A, i)$, $\texttt{Arg}(m) = A$, $\texttt{Sp}(m) = \text{RETRACE}$.

The retrace move has a special parameter i called the index and is denoted as $\texttt{Idx}(m)$.

A dialogue is a sequence of moves with different types respecting a turn taking mechanism. The turn taking mechanism is simple and deterministic where odd indexed moves are advanced by PRO while even index moves are advanced by OPP. The moves of the dialogue are defined in terms of speech acts and content which can express, support, counter attack or retrace. The move $\text{SUPPORT}(a)$ advances an argument a which supports an arbitrary query. The move $\text{COUNTER}(A)$ counterattacks the position of PRO by advancing a set of arguments. The move $\text{RETRACE}(A, i)$ is used to retrace to earlier stage in the dialogue and continue from there. The first move can only be played by PRO, whereas COUNTER and RETRACE are only employed by OPP.

In this dialectical theory, any dialogue starts by PRO advancing a support move to support the query in question. Then, OPP presents an argument (or a set) that attacks the previously advanced argument. Next, PRO tries to avoid this attack and reinstate the query using another argument which is not attacked by the already advanced attackers. OPP in turn, tries to extend the previous set of attackers so that it attacks all the supporters advanced so far. When OPP fails to extend the set, he retraces back and chooses another set of arguments and continues the dialogue from thereafter. By doing so OPP is somehow trying to construct a set of arguments that attack all the supporters of the query $\mathcal{Q}$. In other words, he is trying to build a *block* for the query $\mathcal{Q}$.

Many questions can rise, for instance, what happens when OPP retraces? Would PRO play the supporters which were attacked before retracing or not? What is the nature of the advanced sets of arguments in COUNTER?

In order to answer these and other questions, we need to introduce a control structure that keeps track of the state of the dialogue. This structure will be used in defining the nature and legality of moves.

Definition 23 (Dialectical state). *Let d_k be a dialogue at stage k. The dialectical state of d_k is a tuple $\delta_k = (\pi_k, h_k, \theta_k, \beta_k, \Delta_k)$[2]. d_0 is the empty dialogue and δ_0 is its initial dialectical state.*

This state defines at any stage k of the dialogue d_k the set of arguments π_k available to PRO that can be used to support the query $\mathcal{Q}$. In the dialectical state, we find also the set h_k which shows the arguments that have been so far played by PRO. In addition, it presents the set θ_k of arguments that can be used to attack the arguments previously advanced by PRO. β_k presents the currently constructed block. When OPP fails to extend the current block to another that attacks all the previously played supporters, he uses the RETRACE move. By doing so we need to keep track of the sets of arguments that cannot be extended to a block. These are stored in Δ_k.

Given a query $\mathcal{Q}$, the initial state of the dialectical state is described as follows:

– $\pi_0 = \mathtt{S}(\mathcal{Q})$.
– $h_0 = \emptyset$, $\theta_0 = \emptyset$, $\beta_0 = \emptyset$, $\Delta_0 = \emptyset$.

Since the dialogue d_0 has not yet been started, the set of available arguments π_0 for PRO ranges over all the possible supporters of the query $\mathcal{Q}$. The played arguments h_0, the available arguments θ_0, current block β_0 and Δ_0 are empty since the first move has not yet been uttered.

The advancement of moves within the dialogue is usually controlled by a legal move function [38] which can be expressed in terms of rules, called dialogue rules. Every move depends on certain *preconditions* about the actual dialectical state and the previous move advanced by the other party. Every move also determines the next move (*postcondition*).

Moves affect the dialectical states of the dialogue. In fact, we can see the moves as transitions between possible dialectical states where a dialectical state δ_k for a dialogue

[2] To be able to understand the terms think of π as the first letter of **p**roponent, h as **h**istory, θ as **o**pponent and β as **b**lock.

d_k and a move m_{k+1} define a new dialectical state δ_{k+1}. This is called the *effect* of the move.

In what follows we recall for each move its informal description and we present the preconditions that should be satisfied so that the move is considered legal to be played. We then present its effect on the dialectical state and its postconditions.

Let d_k be a dialogue and δ_k the current dialectical state of d_k. Let m_{k+1} be a move and δ_{k+1} be the dialectical state of the dialogue $d_{k+1} = d_k \cdot m_{k+1}$ after playing the move m_{k+1}. Note that for a given move we index preconditions (resp. effects) by the first letter of the speech act of the move followed by P (resp. E) and subscripted by a number.

Move:
$m_{k+1} = \text{SUPPORT}(a)$.
Description:
this move advances an argument that supports the query in question.
Preconditions:
(SP_1) $k+1$ is odd.
(SP_2) $a \in \pi_k$.
Postconditions:
the next move can be either COUNTER or RETRACE.
Effects:
(SE_1) $\pi_{k+1} = \pi_k \setminus \{a\}$.
(SE_2) $h_{k+1} = h_k \cup \{a\}$.
(SE_3) $\theta_{k+1} = \texttt{range}^-(h_{k+1})$.
(SE_4) $\beta_{k+1} = \beta_k$.
(SE_5) $\Delta_{k+1} = \Delta_k$.

This move is advanced by PRO, therefore $k+1$ should be odd (SP_1). It advances an argument a that supports the query $\mathcal{Q}$ which is not attacked by the current block β_k presented so far (SP_2). To respond to this move, in the next turn OPP should either use COUNTER or RETRACE.

As one may notice, the support move m_{k+1} changes the set of available arguments π_{k+1} of PRO. In fact a supporting argument ceases to be available once it is played (SE_1). In contrast it is added to the history h_{k+1}. The support move alters the set of available arguments of OPP by adding all arguments that can be played in the future by OPP (SE_3).

As indicated in the postconditions of the support move, a counter move is allowed to be played next.

Move:
$m_{k+1} = \text{COUNTER}(A)$.
Description:
this move advances a set of arguments that attacks all the arguments presented so far.
Preconditions:
(CP_1) $k+1$ is even.

(CP$_2$) $A = \beta_k \cup S$ such that $S \subseteq \theta_k$.
(CP$_3$) A attacks h_k and belongs to (or is) an admissible set.
(CP$_4$) there is no $A' \in \Delta_k$ such that $A' \subseteq A$.
Postconditions:
the next move should be SUPPORT.
Effect:
(CE$_1$) $\pi_{k+1} = \pi_k \setminus \texttt{range}^+(A)$.
(CE$_2$) $h_{k+1} = h_k$.
(CE$_3$) $\theta_{k+1} = \texttt{range}^-(\pi_{k+1})$.
(CE$_4$) $\beta_{k+1} = A$.
(CE$_5$) $\Delta_{k+1} = \Delta_k$.

This move is advanced by OPP therefore $k+1$ should be even (CP$_1$). It tries to extend the current block β_k to another set of arguments that attacks all the supporters presented so far (CP$_2$ and CP$_3$). OPP does so by incorporating arguments from θ_k. The new current block ($\beta_{k+1} = A$) or one of its subsets should have not been already proven to be not a block (CP$_4$). After advancing m_{k+1}, π_{k+1} contains all the arguments from π_{k+1} except those which are attacked by A (CE$_1$), thus they are spared from further use. Note that the spared arguments may be readded afterwards, this is particularly the case when we use retrace as we shall mention later.

The set of available moves θ_{k+1} of OPP contains all the arguments that attack the supporting arguments that can be played by PRO (CE$_3$). Since A attacks all the supporting arguments so far provided, it is considered the current block (CE$_4$). The sets Δ_{k+1} and h_{k+1} are left unchanged (CE$_2$ and CE$_5$).

After a support move, OPP can also play a retrace move. This is particularly needed when he is unable to play a counter move. The formal details about the retrace move are presented hereafter:

Move:
$m_{k+1} = \text{RETRACE}(A, i)$.
Description:
this move retraces to the recent stage i from which it can extend the block of stage $i-1$.
Preconditions:
(RP$_1$) $k+1$ is even, $i < k+1$ and i is odd.
(RP$_2$) there is no $S \subseteq \theta_k$ such that $\beta_k \cup S$ is or belongs to an admissible set and attacks h_k.
(RP$_3$) $A = \beta_i \cup S$ such that $S \subseteq \theta_i$.
(RP$_4$) A attacks h_i and belongs to (or is) an admissible set.
(RP$_5$) there is no $A' \in \Delta_k$ such that $A' \subseteq A$.
Postconditions:
the next move should be SUPPORT.
Effect:
(RE$_1$) $\pi_{k+1} = \pi_i \setminus \texttt{range}^+(A)$.
(RE$_2$) $h_{k+1} = h_i$.
(RE$_3$) $\theta_{k+1} = \texttt{range}^-(h_{k+1})$.

(RE$_4$) $\beta_{k+1} = A$.
(RE$_5$) $\Delta_{k+1} = \Delta_k \cup \beta_k$.

When OPP cannot extend the current block β_k with arguments from θ_k (RP$_2$), he should retrace back and choose other arguments. The index i (which should be odd) determines the point of a support move from which OPP can mount another line of attack. By starting a new line of attack, OPP should opt for a new block that attacks all the supporters from Stage i up to the Stage 1 (RP$_3$) by extending β_i from θ_i. The new block $\beta_{k+1} = A$ or one of its subsets should have not been already proven to be not a block (CP$_5$).

When the retrace move is advanced, π_{k+1} is reset to its ancient state i in addition to excluding all the arguments that can be attacked afterwards (RE$_1$). The current block β_{k+1} is set to A (RE$_4$), while Δ_{k+1} is set to $\Delta_k \cup \beta_k$ (RE$_5$), i.e. the block of stage k which OPP could not extend.

If one of the preconditions is not satisfied, OPP goes further and look for other stages where he can mount a new attack. OPP follows the following procedure:

Procedure 1 *Let d_n be a dialogue and m_n be the last played move such that* $\mathtt{Sp}(m_n) =$ SUPPORT. *If* OPP *cannot play a counter move m_{n+1} then it plays the retrace move m_{n+1} as follows:* [3]

1. do $y = y - 1$ until $m_y =$ RETRACE(A, x) *or $m_y =$* SUPPORT(a) *or $y = 0$.*
2. if $m_y =$ RETRACE(A, x) *then:*
  (a) If there does not exist a move $m_{n+1} =$ RETRACE(A', x) *that respects the preconditions then set $y = x$ and goto 1 else play m_{n+1} and exit.*
3. if $m_y =$ SUPPORT(a) *then:*
  (b) If there does not exist a move $m_{n+1} =$ RETRACE(A', y) *that respects the preconditions then goto 1 else play m_{n+1} and exit.*

OPP starts by looking for the most recent retrace or support move (line 1). If a retrace move is found (line 2) then it tries to play a retrace to stage x that respects the preconditions (line a) by looking exhaustively for all possible sets A' that make the move respect the preconditions. If he succeeds to play such move, the procedure exits. Otherwise it continues the search by setting y to x. If a support move m_y is found (line 3) then it plays a retrace with index to y. Otherwise, it continues the search for other moves from which OPP can play.

To better illustrate the point let us apply the procedure on an example.

Example 17. Consider the dialogue example in the table of Figure 3. Let us see how OPP has played the retrace move at stage (8). Note that this is just an illustrative example and does not correspond to a real dialogue.

At stage (7), PRO played the argument c. OPP tried to play a counter move but he failed to do so. Now, OPP will follow Procedure 1 to play a retrace move. At this moment $n = 7$, $y = n$. OPP gets into the loop at line (1). When $y = 6$, OPP encounters the retrace move $m_6 =$ RETRACE$(\{p, r\}, 3)$, he tries to play a retrace move

[3] Note that y is initialized to n and $x < y$, and a, A are arbitrary (set of) arguments respectively.

i	Move
1	SUPPORT(a)
2	COUNTER$(\{p\})$
3	SUPPORT(b)
4	COUNTER$(\{p, q\})$
5	SUPPORT(c)
6	RETRACE$(\{p, r\}, 3)$
7	SUPPORT(c)
8	RETRACE$(\{s\}, 1)$
9	SUPPORT(d)
10	COUNTER$(\{s, t\})$
11	SUPPORT(e)
12	RETRACE$(\{s, v\}, 9)$

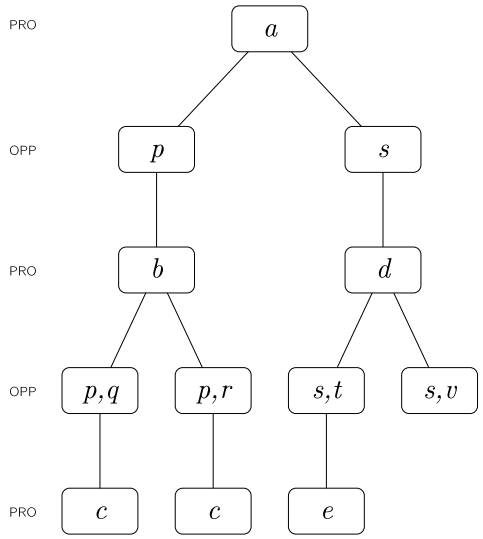


Fig. 3: The left table presents the dialogue, the right figure shows its associated dialogue tree.

$m_8 = \text{RETRACE}(A', 3)$ but it seems that he couldn't play such move because there is no A' that makes the retrace move respect the preconditions. OPP continues this time from 2 (y is set to 3 in line (b) and it gets decreased at line (1)) where he skips the counter move and stops at stage (1) where a support move $m_1 = \text{SUPPORT}(a)$ is found. At this point, OPP plays the retrace move $m_8 = \text{RETRACE}(\{s\}, 1)$ which seems to respect the preconditions. Afterwards, the dialogue continues normally by PRO until it stops at stage (12) with OPP playing the last move.

In fact, the sequence of moves represents a dialogical representation of a tree where retrace moves represent branching points. This tree is called the associated dialogue tree and it is defined as follows.

Definition 24 (Dialogue tree). *Given a dialogue $d_n = (m_1, \ldots, m_n)$, its dialogue tree is a labeled tree $\mathcal{T}(d_n) = (\mathcal{V}, \mathcal{D})$ such that $\mathcal{V}$ is a set of nodes and $\mathcal{D}$ is a binary relation over $\mathcal{V}$ defined as follows:*

– $\mathcal{V} = \{\texttt{Arg}(m_i) \mid m_i \in d_n\}$.
– $\mathcal{D} = \{(\texttt{Arg}(m_{i-1}), \texttt{Arg}(m_i)) \mid i \leqslant n \text{ and } m_i \neq \text{RETRACE}(A, j)\} \cup$
$\{(\texttt{Arg}(m_j), \texttt{Arg}(m_n)) \mid i \leqslant n \text{ and } m_i = \text{RETRACE}(A, j)\}$.

$\texttt{Arg}(m_1)$ *is the root node of the tree. Note that $|\mathcal{T}(d_n)| = |\mathcal{V}|$ refers to the size of the tree which is equal to the number of its nodes.*

It is a tree where nodes are arguments or set of arguments played by both parties. Odd-level nodes are played by PRO and even-level nodes are played by OPP.

Fact 1 *Let d_n be a dialogue, $\mathcal{T}(d_n)$ its associated dialogue tree. The following holds:*

1. $\mathcal{T}(d_n)$ is unique.
2. $|d_n| = |\mathcal{T}(d_n)|$.

The dialogue terminates when no one can further the dialogue with moves.

Definition 25 (Termination and wining). *Let d_k be a dialogue and δ_k the current dialectical state of d_k. The dialogue d_k is a terminated dialogue if and only if the player to play has run out of moves. The winner of the dialogue is* $\texttt{Player}(m_k)$ *(the player of the last move).*

It is easy to determine the winner of a dialogue from its tree.

Fact 2 *Le d_n be a terminated dialogue, $\mathcal{T}(d_n)$ its associated dialogue tree. The following statements are equivalent:*

– *The length of the right-most path is odd.*
– PRO *is the winner of d_n.*

It is clear that the last move corresponds to the leaf node in the right-most path of the tree. If the length of the path is odd then PRO is the last one who played, consequently PRO is the winner.

After having defined the theory, let us define the concept of a dialectical proof.

Definition 26 (Dialectical proof). *Let $\mathcal{Q}$ be a querry and d_n a terminated dialogue d_n. We call d_n a dialectical proof for the universal acceptance of $\mathcal{Q}$ if and only if* PRO *is the winner, otherwise it is called a dialectical proof for universal non-acceptance of $\mathcal{Q}$.*

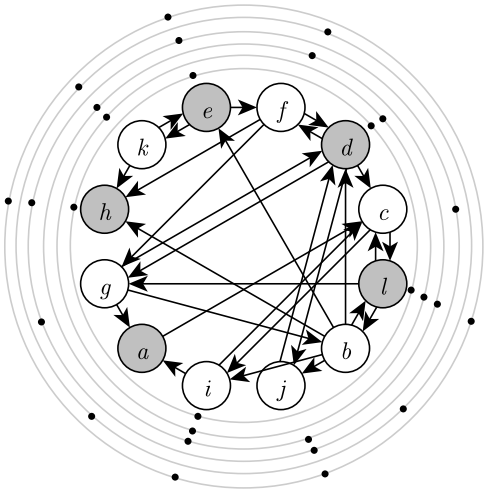
In the next subsection we give a detailed example of universal acceptance and non-universal acceptance on a real argumentation framework.

6.1 Dialogue example

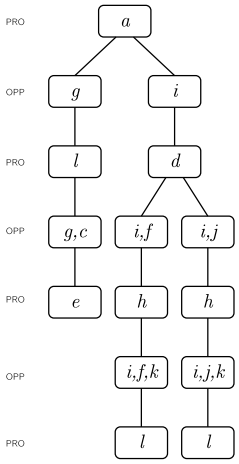
Consider the argumentation framework $\mathcal{H}$ of Figure 4(a). This argumentation framework is coherent. Suppose that the gray-colored arguments supports a query $\mathcal{Q}$ (i.e. $\texttt{S}(\mathcal{Q}) = \{a, d, e, l, h\}$). In what follows, we show how the query $\mathcal{Q}$ is universally accepted by providing a dialectical proof.

The dialectical proof is presented in Table 2 and its associated dialogue tree is shown in Figure 4(b).

At stage (0) the dialectical state is initialized as defined previously. The dialogue starts at stage (1) by PRO playing the supporter a from the available supporters in π_0. When PRO plays a, the argument a is moved from the available supporters π_1 to the history of advanced arguments by PRO h_1. The set of available attackers θ_1 becomes the set of all attackers of $\pi_1 \cup h_1$. This means when the turn of OPP comes at stage (2)



(a) The argumentation framework.



(b) The dialogue tree.

Fig. 4: The circles are the extensions presented in an increasing order with $\mathcal{E}_1$ being the inner circle.

he shall choose from this set. At stage (2) OPP advances a counter move with argument g that attacks all the advanced supporters (i.e. $h_1 = \{a\}$). After advancing such move, the argument d is removed from the set of available arguments π_2 since g attacks d, thus PRO will not be able to play d. Observe that j is removed also from θ_2 because it does not attack any argument in $\pi_2 \cup h_2$. Since $\{g\}$ attacks all the supporters advanced so far, it becomes the current block, i.e. $\beta_2 = \{g\}$. At stage (3), PRO responds by a support move with the argument l that is not attacked by the current block. At stage (4), OPP extends the current block $\beta_3 = \{g\}$ by the argument c which attacks l. Note that $\{g, c\}$ is a subset of the admissible set $\{g, c, e\}$. Now, $\beta_4 = \{g, c\}$ attacks all the presented supporters. At stage (5), PRO presents another unattacked supporter (i.e. e). Note that the choice of the supporters is arbitrary.

At stage (6), OPP could not extend the current block β_5 into another that attacks e too. Therefore OPP plays a retrace move $\mathtt{R}(\{i\}, 1)$ that can be read as "retrace to stage (1) and play a counter move with $\{i\}$". By doing so, OPP creates another line of dialogue and rolls back all the changes that have been made on the dialectical state up to the stage (1). That is why at stage (6) the sets π_6, h_6 and θ_6 are set to π_1, h_1 and θ_1 respectively. The current block is changed to $\{i\}$ and the ancient block β_5 is moved to $\Delta_6 = \{\beta_5\}$. The former means that this set or any of its supper sets will never form a block. This is important to avoid unnecessary moves. The same happens at stage (12) where OPP retraces to stage (7) because he cannot retrace to the stage (9). The current block β_{12} is set to $\{i, j\}$ which extends β_7.

The dialogue continues until stage (15) where PRO plays a support move with argument l against which OPP could neither attack nor retrace to previous stages. At this stage the dialogue ends and PRO is declared as the winner.

i	Move	π_i	h_i	θ_i	β_i	Δ_i
0	-	$\{a, d, e, l, h\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	S(a)	$\{d, e, l, h\}$	$\{a\}$	$\{g, i\}$	$\emptyset$	$\emptyset$
2	C($\{g\}$)	$\{e, l, h\}$	$\{a\}$	$\{g, i\}$	$\{g\}$	$\emptyset$
3	S(l)	$\{e, h\}$	$\{a, l\}$	$\{g, i, c, b\}$	$\{g\}$	$\emptyset$
4	C($\{g, c\}$)	$\{e, h\}$	$\{a, l\}$	$\{g, i, c, b\}$	$\{g, c\}$	$\emptyset$
5	S(e)	$\{h\}$	$\{a, l, e\}$	$\{g, i, c, b, k\}$	$\{g, c\}$	$\emptyset$
6	R($\{i\}, 1$)	$\{d, e, l, h\}$	$\{a\}$	$\{g, i\}$	$\{i\}$	$\{\beta_5\}$
7	S(d)	$\{e, l, h\}$	$\{a, d\}$	$\{g, i, f, j, b\}$	$\{i\}$	$\{\beta_5\}$
8	C($\{i, f\}$)	$\{l, h\}$	$\{a, d\}$	$\{g, i, f, j, b\}$	$\{i, f\}$	$\{\beta_5\}$
9	S(h)	$\{l\}$	$\{a, d, h\}$	$\{g, i, f, j, b, k\}$	$\{i, f\}$	$\{\beta_5\}$
10	C($\{i, f, k\}$)	$\{l\}$	$\{a, d, h\}$	$\{g, i, f, j, b, k\}$	$\{i, f, k\}$	$\{\beta_5\}$
11	S(l)	$\emptyset$	$\{a, d, h, l\}$	$\{g, i, f, j, b, k, c\}$	$\{i, f, k\}$	$\{\beta_5\}$
12	R($\{i, j\}, 7$)	$\{e, l, h\}$	$\{a, d\}$	$\{g, i, f, j, b\}$	$\{i, j\}$	$\{\beta_5, \beta_{11}\}$
13	S(h)	$\{e, l\}$	$\{a, d, h\}$	$\{g, i, f, j, b, k\}$	$\{i, j\}$	Δ_{12}
14	C($\{i, j, k\}$)	$\{l\}$	$\{a, d, h\}$	$\{g, i, f, j, b, k\}$	$\{i, j, k\}$	Δ_{12}
15	S(l)	$\emptyset$	$\{a, d, h, l\}$	$\{g, i, f, j, b, k, c\}$	$\{i, j, k\}$	$\Delta_{12} \cup \{\beta_{14}\}$

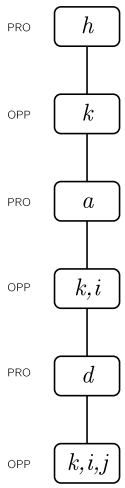
Table 2: A dialectical proof for the query $\mathcal{Q}$. For space reasons S(), C() and R() denote SUPPORT(), COUNTER() RETRACE() respectively.

The associated tree in Figure 4(b) shows clearly the relation between the advanced arguments played by both parties. The tree in Figure 5(b) is another dialogue tree for another dialogue where PRO is the winner. This can be easily observed since all leaf nodes are within an odd layer.

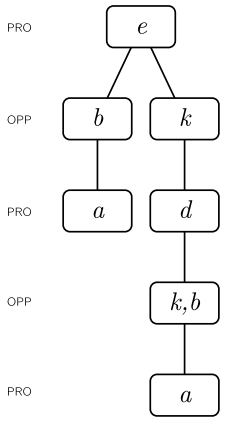
Let us now use the same argumentation framework, but consider another query $\mathcal{Q}'$, which is not universally accepted. The supporters are $\mathtt{S}(\mathcal{Q}') = \{a, d, e, h\}$. The dialogue is presented in Table 3 and its associated dialogue tree is shown in Figure 5(a). In this example, OPP has been able to construct the block $\beta_6 = \{k, i, j\}$ in the last move which attacks all the supporters. This made PRO unable to continue the dialogue. Note that we do not allow retracing for PRO because one block is sufficient to prove the unacceptability of the query.

i	Move	π_i	h_i	θ_i	β_i
0	-	$\{a, d, e, h\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	S(h)	$\{a, d, e\}$	$\{h\}$	$\{k, f, b\}$	$\emptyset$
2	C($\{k\}$)	$\{a, d\}$	$\{h\}$	$\{k, f, b\}$	$\{k\}$
3	S(a)	$\{d\}$	$\{a, h\}$	$\{k, f, b, g, i\}$	$\{k\}$
4	C($\{k, i\}$)	$\{d\}$	$\{a, h\}$	$\{k, f, b, g, i\}$	$\{k, i\}$
5	S(d)	$\emptyset$	$\{d, a, h\}$	$\{k, f, b, g, i, j\}$	$\{k, i\}$
6	C($\{k, i, j\}$)	$\emptyset$	$\{d, a, h\}$	$\{k, f, b, g, i, j\}$	$\{k, i, j\}$

Table 3: A dialectical proof for the non-universal acceptance of $\mathcal{Q}'$. Note that we omit Δ_i as it is always empty in this example.



(a) Non-universal acceptance.



(b) Universal acceptance

Fig. 5: The associated dialogue tree.

In the next section we show that OPP the winner if the query is not universally accepted and that PRO is the winner if the query is universally accepted. We also show some other properties.

6.2 Properties

In this section we show that our dialectical proof theory satisfies finiteness, soundness and completeness. Next we shift to studying the dispute complexity of dialectical proofs, defined by Dunne and Bench-Capon [30].

Termination is an important property for any dialogue [4, 35]. In what follows we show how our dialectical theory produces always finite dialogues. To establish such a property we need to show that for any dialogue d, its associated dialogue tree is finite. Such result can be established by showing that the height of the tree is finite and that for each node the number of its children is finite. Given a tree, we denote by $\texttt{Height}(.)$ the number of edges from the root to the farthest leaf. For a node v, we denote by $\mathcal{C}(v)$ the set of all child nodes of v.

Lemma 3. *Let $\mathcal{H}$ be an argumentation framework and $\mathcal{D}$ be the set of all possible dialogues over $\mathcal{H}$. Given $\mathcal{T}(d) = (\mathcal{V}, \mathcal{D})$ of any $d \in \mathcal{D}$ the following holds:*

1. $\texttt{Height}(\mathcal{T}(d))$ *is finite*
2. $\forall\, v \in \mathcal{V}, \exists l \in \mathbb{N}$ *such that* $|\mathcal{C}(v)| \leqslant l$.

Proof. Let us suppose that $\texttt{Height}(\mathcal{T}(d))$ is infinite, and let P be the longest path in $\mathcal{T}(d)$ starting from the root node. This means either there are infinitely many supporting

arguments used in P, or there are some infinite repetitions in supporting arguments used in P. The first one is impossible since we are dealing with finite argumentation framework (the set of all arguments is finite). The second is impossible since once an argument is played it cannot be advanced afterwards in the same path (see SE_1 of SUPPORT move).

Let us suppose that $|\mathcal{C}(v)|$ is infinite. This means that v is a supporting argument and it has infinitely many attackers, which is impossible since the argumentation framework is finite.

Amgoud et al. [4] introduced another constraint, called finiteness of the moves' contents. This constraint insures that the arguments advanced within the dialogue are finite. In our context we distinguish tow cases, (i) the argument in a support moves should be finite, and (ii) the set of arguments advanced in a counter move should be finite too. Fortunately, the two cases are verified in our argumentation framework because as shown in Proposition 3 the set of arguments $\mathcal{A}$ for any argumentation framework over a possibly inconsistent knowledge base is finite and the set of attacker for a given argument is finite. We get the following result on finiteness.

Theorem 5 (Finiteness). *Let $\mathcal{H}$ be an argumentation framework and $\mathcal{D}$ be the set of all possible dialogues over $\mathcal{H}$. Then, for every $d \in \mathcal{D}$, $\exists k \in \mathbb{N}$ such that $|d| \leqslant k$.*

Proof. Let us suppose that d is infinite. This means, either (i) $\texttt{Height}(\mathcal{T}(d))$ is infinite; or (ii) there is a node in $\mathcal{T}(d_n)$ with infinitely many child nodes. From Lemma 3, the two cases are impossible.

Finiteness is not sufficient alone. After all, if a dialectical proof theory gives finite dialogues but incorrect results then such proof theory is useless. Soundness is the property that insures that the proof theory gives only correct results. In other words, if one has a dialectical proof for universal acceptance (resp. non-universal acceptance) of a query then the query is universally accepted (resp. not universally accepted).

Let us show that the dialectical proof theory is *consistent* in the sense that there are no two dialogues about a query $\mathcal{Q}$ such that PRO wins one and loses the other.

Proposition 17. *Let $\Omega(\mathcal{H}, \mathcal{Q})$ be the set of all terminated dialogues about $\mathcal{Q}$ in $\mathcal{H}$ and let $d \in \Omega(\mathcal{H}, \mathcal{Q})$. If d is won by* PRO *(resp.* OPP*) then every $d' \in \Omega(\mathcal{H}, \mathcal{Q})$ is won by* PRO *(resp.* OPP*).*

Proof. Suppose that d is won by OPP and there exists another dialogue d' that is won by PRO. This means that OPP has failed to construct a block in d''. This means that either (i) there is no block, or (ii) the Procedure 1 is not exhaustive. The former is in contradiction with the fact that OPP has won d therefore a block does exist. The latter is in contradiction with the evident fact that the procedure indeed tries all possible moves.

This property is very important since we do not want to have a dialectical proof theory that is contradictory. It turns out that this property is important for soundness. In what follows soundness is characterized by the existence of one winning dialogue (by PRO or OPP).

Theorem 6 (Soundness). *Given a dialogue d_n about the query $\mathcal{Q}$, if d_n is won by* PRO *then $\mathcal{Q}$ is universally accepted.*

Proof. Let us proceed by contradiction. Suppose that d_n is won by PRO but $\mathcal{Q}$ is not universally accepted. On the one hand, recall that if $\mathcal{Q}$ is not universally accepted then there exists a block B against all $\mathcal{Q}$'s supporters. On the other hand, if PRO has won d_n then OPP could not find any block that attacks all supporters advanced in d_n. This means that either (i) OPP search was not exhaustive or (ii) there is no such block. As one can see, (ii) is in contradiction with the assumption and (i) is in contradiction with the fact that the Procedure 1 is exhaustive.

If the dialectical proof theory is sound but does not provide dialectical proofs for all universally (resp. not universally) accepted queries then it is incomplete. In what follows we provide a completeness proof.

Theorem 7 (Completeness). *If a query $\mathcal{Q}$ is universally accepted then* PRO *wins every dialogue about $\mathcal{Q}$.*

Proof. By contradiction, if $\mathcal{Q}$ is universally accepted and PRO loses then OPP has constructed a block β_n for $\mathcal{Q}$. This means that $\mathcal{Q}$ is not universally accepted, which is a contradiction.

Given a query $\mathcal{Q}$, what is the shortest dialogue allowing to establish universal acceptance (universal non-acceptance) of $\mathcal{Q}$? Dunne and Bench-Capon [30] introduced the so-called dispute complexity for a given argument in a given argumentation framework. We adapt this definition to our context as follows.

Definition 27 (Dispute complexity). *Let $\mathcal{H}$ be an argumentation framework and $\mathcal{Q}$ be a query. The dispute complexity $\delta(\mathcal{H}, \mathcal{Q})$ is defined as follows:*

$$\delta(\mathcal{H}, \mathcal{Q}) = min(|d| : d \text{ is a terminated dialogue about } \mathcal{Q} \text{ in } \mathcal{H})$$

It is read as the dispute complexity of the query $\mathcal{Q}$ in the argumentation framework $\mathcal{H}$.

The dispute complexity is the minimal number of moves that must be used to prove that $\mathcal{Q}$ is universally accepted or not universally accepted. Dunne and Bench-Capon [30] provided an exact characterization of such complexity for credulous acceptance by considering as an input the argumentation framework and all admissible sets. Our goal in what follows is to propose some bounds for such complexity in universal (or non-universal) acceptance. These bounds will be studied with respect to the size of blocks and proponent sets defined in page 16.

Recall that a block of a given query is necessarily a hitting set. A minimum block is the smallest block w.r.t cardinality. Similarly, a minimum proponent set is the smallest proponent set w.r.t cardinality.

Notation 3 *Let $\mathcal{Q}$ be a query, $\mathcal{H}$ an argumentation framework such that $\mathcal{Q}$ is not universally accepted in $\mathcal{H}$ and $\mathcal{C} = \{\texttt{range}^{-}(x) \mid x \in \texttt{S}(\mathcal{Q})\}$:*

– $HS(\mathcal{H}, \mathcal{Q})$ denotes the set of all hitting sets of $\mathcal{C}$.

– *MHS*$(\mathcal{H}, \mathcal{Q})$ *denotes the set of all minimal hitting sets of* $\mathcal{C}$.
– *BS*$(\mathcal{H}, \mathcal{Q})$ *denotes the set of all blocks of* $\mathcal{Q}$.
– *MinBS*$(\mathcal{H}, \mathcal{Q})$ *denotes the set of all minimum blocks of* $\mathcal{Q}$.
– *The block number of* $\mathcal{Q}$ *in* $\mathcal{H}$ *is the size of the minimum block:*
$\tau(\mathcal{H}, \mathcal{Q}) = min(|B| : B \in \textit{MinBS}(\mathcal{H}, \mathcal{Q}))$.
– *The hitting set number is the size of the minimum hitting set of* $\mathcal{C}$*:*
$\alpha(\mathcal{H}, \mathcal{Q}) = min(|S| : S \in \textit{MHS}(\mathcal{H}, \mathcal{Q}))$.

The block number corresponds to the minimum block which is the smallest block *(w.r.t set-cardinality)* among all blocks. Note that it is not necessary that every minimum hitting set of $\mathcal{C}$ is a minimum block, because a minimum block imposes that its members have to belong to the same admissible set (see Example 18 below). Therefore it is possible to have a block which is minimum but does not correspond to any minimum hitting set. In contrast, a minimum block has to be a hitting set. We get the following straightforward relations:

Fact 3 *The following statement holds:*

– *BS*$(\mathcal{H}, \mathcal{Q}) \subseteq$ *HS*$(\mathcal{H}, \mathcal{Q})$.

From this fact we can easily deduce that the block number can be equal or greater than the hitting set number of a query in an argumentation framework.

Fact 4 $\tau(\mathcal{H}, \mathcal{Q}) \geqslant \alpha(\mathcal{H}, \mathcal{Q})$.

Note that, given a query $\mathcal{Q}$, the complexity of the dispute in an argumentation framework is equal to the double of the block number. Indeed, if the size of the minimum block B equals n then OPP extends his current block by advancing one attacker at each stage. Therefore, for each SUPPORT move we have a COUNTER move that extends the current block by one argument. When the current block reaches the size n, that means OPP has played all the arguments of the minimum block; PRO will have no supporting argument to advance, thus the dialogue will terminate after $2 \times n$ moves.

Fact 5 *For any terminated dialogue* d *about* $\mathcal{Q}$ *in an argumentation framework* $\mathcal{H}$ *where* $\mathcal{Q}$ *is not universally accepted:*

$$\delta(\mathcal{H}, \mathcal{Q}) = 2 \times \tau(\mathcal{H}, \mathcal{Q}).$$

Example 18. Consider the example in Table 3, page 35.

– $\mathtt{S}(\mathcal{Q}) = \{a, d, e, h\}$.
– the set of all sets of attackers $\mathcal{C}$ is as follows:
1. $\mathtt{range}^-(a) = \{i, j\}$.
2. $\mathtt{range}^-(d) = \{b, j, g, f\}$.
3. $\mathtt{range}^-(e) = \{b, k\}$.
4. $\mathtt{range}^-(h) = \{b, k, f\}$.

The following minimum hitting sets are candidate blocks:
1. $B_1 = \{b, i\}$.
2. $B_2 = \{b, j\}$.

These minimum hitting sets do not belong to any admissible set since they are not conflict-free. Hence the minimum blocks will have sizes of at least 3. Therefore, the following are minimum blocks (among others):
1. $B_3 = \{i, j, k\}$.
2. $B_4 = \{f, j, k\}$.
3. $B_5 = \{f, i, k\}$.
In the dialogue d_6, from Table 3, block B_3 has been constructed. We can see clearly that:
$\delta(\mathcal{H}, \mathcal{Q}) = 2 \times 3 = 6$.

A direct consequence of Fact 5 is the following.

Fact 6 *Let d be the shortest dialogue for the non-universal acceptance of $\mathcal{Q}$. The associated dialogue tree $\mathcal{T}(d)$ is a chain.*

This result is straightforward since OPP will attack all supporters without any need to retrace, hence there will be no branching in the associated dialogue tree.

Let us turn to the dispute complexity for universal acceptance. As we have seen in Subsection 3.2, proponent sets also have a strong relation with minimal hitting sets, in fact they are minimal hitting set over the reduct of extensions. We define similarly the proponent number and the attack degree of a query in an argumentation framework to be able to bound the dispute complexity of its universal acceptance.

Notation 4 *Let $\mathcal{Q}$ be a query and $\mathcal{H}$ an argumentation framework such that $\mathcal{Q}$ is universally accepted in $\mathcal{H}$.*

– *the proponent number is the size of the minimum proponent set:*
$\rho(\mathcal{H}, \mathcal{Q}) = min(|S| : S$ *is a proponent set of* $\mathcal{Q}$ *in* $\mathcal{H})$.
– *the attack degree of $\mathcal{Q}$ in $\mathcal{H}$:* $\deg(\mathcal{H}, \mathcal{Q}) = max(|\mathtt{range}^{-}(a)| : a \in \mathcal{P}(\mathcal{Q}))$*, where $\mathcal{P}(\mathcal{Q})$ is the set of all arguments that belong to at least one minimum proponent set.*

When PRO is engaged in a dialogue he always uses the set of all supporters $\mathtt{S}(\mathcal{Q})$. It is obvious that a proponent set (minimum or not) can replace $\mathtt{S}(\mathcal{Q})$ since it represents all what PRO needs to establish the universal acceptance of $\mathcal{Q}$.

Proposition 18. *Let P be a proponent set of a given query $\mathcal{Q}$ in an argumentation framework $\mathcal{H}$. Let us change the rules of the dialogue described earlier so that* PRO *plays only from P instead of* $\mathtt{S}(\mathcal{Q})$*. Then, the soundness and completeness are preserved.*

The proposition is straightforward, because if there is no proponent set, the dialogue will not start and the query will not be accepted. If there exists a proponent set, then there is no block that can attack all members of P. This result is important to determine the smallest dialogue to prove $\mathcal{Q}$, consequently to bound the dispute complexity.

It is obvious that dialogues where PRO plays with minimum proponent sets are shorter than all other dialogues, because in the former dialogues PRO will play only the support moves that are needed to terminate the dialogue.

Proposition 19. *Let $\delta(\mathcal{H}, \mathcal{Q})$ be the dispute complexity of the query $\mathcal{Q}$ in $\mathcal{H}$. Let $\Theta(\mathcal{H}, \mathcal{Q})$ be the size of the shortest dialogue where* PRO *plays only with a minimum proponent set. Then,*

$$\delta(\mathcal{H}, \mathcal{Q}) \leqslant \Theta(\mathcal{H}, \mathcal{Q}).$$

But this claim is not precise enough to serve as an upper bound. Actually, we need to bound the quantity $\Theta(\mathcal{H}, \mathcal{Q})$. To do so, one has to see what would be the worst-case size of such quantity.

If we imagine the associated dialogue tree of the shortest dialogue, the worst-case for such quantity would be that at each support node we have k children with k being the worst-case number of attackers, which corresponds to the attack degree of the query $\mathcal{Q}$ in $\mathcal{H}$. This tree is similar to the one in Figure 6 (called the worst-case associated dialogue tree). The proponent number is 4 (number of gray nodes) and $k = 3$, therefore every support node has exactly 3 children, whereas every counter move has exactly one child. It is clear that to compute the worst-case quantity that bounds $\Theta(\mathcal{H}, \mathcal{Q})$ one needs to compute the number of nodes of the worst-case optimal associated dialogue tree. Therefore, we get the following theorem:

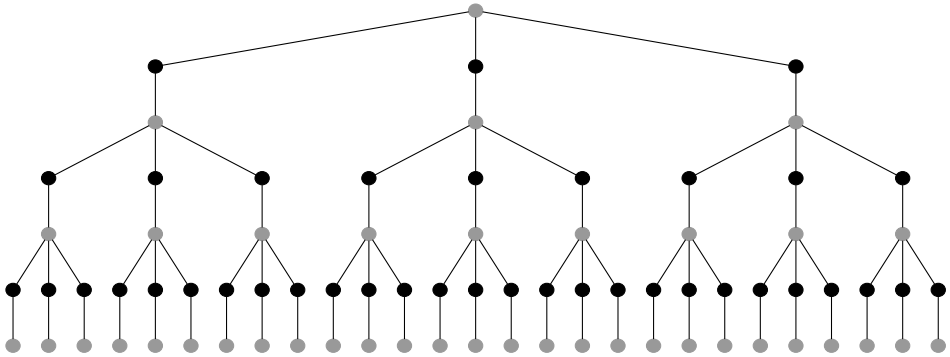


Fig. 6: A worst-case associated dialogue tree with $k = 3$. Gray-colored nodes are support move nodes while black-colored nodes are counter move nodes.

Theorem 8. *Let d be a dialogue about $\mathcal{Q}$ in an argumentation framework $\mathcal{H}$ where $\mathcal{Q}$ is universally accepted, and let $k = \deg(\mathcal{H}, \mathcal{Q})$. Then:*

$$\delta(\mathcal{H}, \mathcal{Q}) \leqslant 2 \cdot \frac{k^{\rho(\mathcal{H}, \mathcal{Q})} - 1}{k - 1} - 1$$

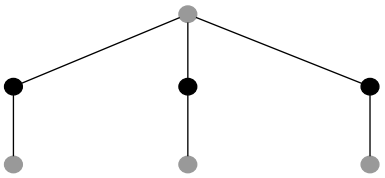
By applying the previous formula with $k = 3$ and $\rho = 4$, we conclude that there are 79 nodes in the tree from Figure 6.

Example 19. Consider again the example of Figure 5(a), but let us now consider another query $\mathcal{Q}''$ such that $\mathtt{S}(\mathcal{Q}'') = \{a, d, e, l, h\} \cup \{k\}$. Query $\mathcal{Q}''$ has two minimum

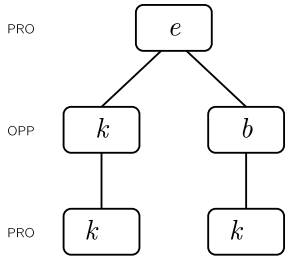
proponent sets: $\mathcal{P} = \{P_1, P_2\}$ and $P_1 = \{k, h\}$, $P_2 = \{k, e\}$. Let us compute the upper bound for the dispute complexity of $\mathcal{Q}''$. The query has two minimum proponent sets $\mathcal{P} = \{\{k, h\}, \{k, e\}\}$. We have that $k = 3$ and $\rho(\mathcal{H}, \mathcal{Q}'') = 2$. Therefore,

$$\delta(\mathcal{H}, \mathcal{Q}) \leqslant 7.$$

The worst-case associated dialogue tree is presented in Figure 7(a). The shortest dialogue is presented in Figure 7(b).



(a) The worst-case associated dialogue tree for $\mathcal{Q}$.



(b) The associated dialogue tree for the shortest dialogue.

Fig. 7: The two associated dialogue trees.

7 Discussion

This paper presents the first instantiation of Dung's abstract argumentation framework with the logical language expressed using existential rules. The benefit of using argumentation is that it allows to represent the data in a format that is easier to grasp by a user. It allows (by examining the support of an argument) to track the provenance of different pieces of information used to conclude a given formula and to see (by examining the attacks between arguments) which pieces of information are not compatible together.

We studied the properties of our system and showed that it satisfied rationality postulates from the literature. We also showed the link between the argumentation semantics and inconsistency-tolerant semantics. Those results show that the two theories, which were developed independently, yield very similar results.

We then focused on universal acceptance for which we provided a precise characterization with respect to the state of the art. We used the explanatory power of dialectical proofs to give an alternative dialectical proof procedure for universal acceptance. Important properties for this proof theory have been given: finiteness, completeness, soundness and dispute complexity.

Related work wrt existential rule based argumentation frameworks. The link between maximal consistent subsets of a knowledge base and stable extensions of the

corresponding argumentation system was shown by Cayrol [20]. That was the first work showing this type of connection between argument-based and non argument-based reasoning. This result was generalized [46] by studying the whole class of argumentation systems corresponding to maximal consistent subsets of the propositional knowledge base. The link between the ASPIC system [40] and the Argument Interchange Format (AIF) ontology [24] has recently been studied [13]. Another related paper comprises constructing an argumentation framework with ontological knowledge allowing two agents to discuss the answer to queries concerning their knowledge (even if it is inconsistent) without one agent having to copy all of their ontology to the other [15]. While those papers are in the area of our paper, none of them is related to the study of the links between different semantics for inconsistent ontological KB query answering and different argumentation semantics.

Related work wrt universal acceptance dialectical proofs. After the introduction of abstract argumentation framework by Dung [28], many attempts have been made to provide formal proof theories for abstract argumentation. They are often referred to as *dialectical proof theories* where the adjective "dialectical" is due to the conversational aspect of the proof. Jakobovits and Vermeir [34] and Prakken [42] define, similarly to dialogical logic, a dialectical proof theory as an argument game with a winning criterion alongside with a legal move function that decides the allowed moves to be played. Given an argumentation framework, a semantics x and an argument a, the objective is to prove whether the argument a is skeptically/credulously accepted under the semantics x.

The TPI (Two party Immediate Response) procedure [47, 30] is used for credulous and skeptical games in finite and coherent argumentation frameworks where two players exchange arguments (moves) until one of them cannot play. The justification status of the argument (skeptical/credulous) is decided with respect to the wining criterion. In TPI-disputes each move attacks the previous one. Their dialectical proof theories are shown to be sound and complete. Cayrol et al. [21] follow the same guideline but with a refinement on the size of the proof, thus producing shorter proofs than in the approach by Dunne and Bench-Capon [30]. Modgil and Caminada [38] proposed another dialectical proof theory for skeptical acceptance where, instead of exchanging arguments the proponent and the opponent exchange whole admissible sets. The goal is to construct a *block*, which is an admissible set of arguments that conflicts with all admissible sets around the argument in question [38]. Following the same idea, Doutre and Mengin [27] construct such a block in a *meta-argumentation framework* within a *meta-dialogue* where admissible sets are considered as moves, then the classical credulous proof theory [21] is used as a sub-procedure to proof skeptical acceptance. Thank et al. [45] defined a more general framework, which is sound for any argumentation frameworks and is complete for general classes of finitary argumentation frameworks (including the class of finite argumentation frameworks). Skeptical and grounded acceptance have also been studied by the authors [6] for existential rules.

To the best of our knowledge no existing work addressed the problem of dialectical approaches for universal acceptance.

Future work. It is important to point out that this dialectical proof theory can also be used in abstract settings like the one by Amgoud et al. [3]. In that work, Dung's abstract framework [28] is used in decision-support systems where arguments support

different options (or decisions) and the final decision is computed using Dung's semantics. The authors introduced the concept of universal acceptance for a given option and shown that skeptical and universal acceptance differ. In fact, the distinction is important and practical since in certain decision making situations we may opt for an option that is supported by different arguments from different extensions but not supported by skeptical arguments (as there may be none). It would be interesting to investigate how to change our dialectical theory so that it corresponds to the abstract setting.

We would like to investigate other argumentation semantics and inconsistency-tolerant semantics and to establish more links. Actually the recent work by Baget [9] provided a unified framework for inconsistency-tolerant semantics in ontological knowledge bases. Providing a representation theorem between all semantics proposed in this work and those of argumentation would be of big impact. This endeavor follows the general line of examining how the knowledge representation community could benefit from other results from argumentation theory and whether the argumentation community could use some open problems in the knowledge representation as inspiration for future work.

Acknowledgments

This paper is an expanded and revised version of three conference papers [26, 7, 6].

The authors would like to help Abdelraouf Hecham and the anonymous reviewers for their comments.

Abdallah Arioua and Madalina Croitoru benefited from the support of the projects Dur-Dur ANR-13-ALID-0002 and ASPIQ ANR-12-BS02-0003 of the French National Research Agency (ANR). The major part of the work in this paper was carried on while Abdallah Arioua was doing his PhD at the university of Montpellier and UMR IATE/LIRMM laboratories. Srdjan Vesic benefited from the support of the project AMANDE ANR-13-BS02-0004 of the French National Research Agency (ANR).

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Addison-Wesley Reading, 1995.
2. L. Amgoud. Postulates for logic-based argumentation systems. *International Journal of Approximate Reasoning*, 55(9):2028–2048, 2014.
3. L. Amgoud, Y. Dimopoulos, and P. Moraitis. Making decisions through preference-based argumentation. *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, 8:963–970, 2008.
4. L. Amgoud, D. Saint-Cyr, and F. Dupin. An axiomatic approach for persuasion dialogs. In *Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI'13)*, pages 618–625. IEEE, 2013.
5. M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 68–79. ACM, 1999.
6. A. Arioua and M. Croitoru. Dialectical characterization of consistent query explanation with existential rules. In *Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference (FLAIRS'16)*, pages 14–19, 2016.

7. A. Arioua and M. Croitoru. A dialectical proof theory for universal acceptance in coherent logic-based argumentation frameworks. In *Proceedings of the 22nd European Conference on Artificial Intelligence, (ECAI'16)*, 2016.
8. F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, 2005.
9. J. Baget, S. Benferhat, Z. Bouraoui, M. Croitoru, M. Mugnier, O. Papini, S. Rocher, and K. Tabia. A general modifier-based framework for inconsistency-tolerant query answering. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'16)*, pages 513–516, 2016.
10. J.-F. Baget, M. LeclÃĺre, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9âĂŞ10):1620 – 1654, 2011.
11. J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 712–717, 2011.
12. P. Besnard and A. Hunter. *Elements of argumentation*, volume 47. MIT press Cambridge, 2008.
13. F. J. Bex, S. J. Modgil, H. Prakken, and C. Reed. On logical specifications of the argument interchange format. *Journal of Logic and Computation*, page In Press, 2013.
14. M. Bienvenu and R. Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 2013.
15. E. Black, A. Hunter, and J. Z. Pan. An argument-based approach to using multiple ontologies. In *Proceedings of the 3rd International Conference on Scalable Uncertainty Management (SUM'09)*, pages 68–79. Springer-Verlag, 2009.
16. A. Bondarenko, F. Toni, and R. A. Kowalski. An assumption-based framework for non-monotonic reasoning. In *Proceedings of the International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR'93)*, volume 93, pages 171–189, 1993.
17. A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics*, 14:57–83, 2012.
18. 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. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
19. M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171 (5-6):286–310, 2007.
20. C. Cayrol. On the relation between argumentation and non-monotonic coherence-based entailment. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1443–1448, 1995.
21. C. Cayrol, S. Doutre, and J. Mengin. On decision problems related to the preferred semantics for argumentation frameworks. *Journal of logic and computation*, 13(3):377–403, 2003.
22. S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989.
23. M. Chein and M. Mugnier. *Graph-based Knowledge Representation - Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2009.
24. C. Chesnevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott. Towards an argument interchange format. *Knowledge Engineering Review*, 21(4):293–316, 2006.
25. J. Chomicki. Consistent query answering: Five easy pieces. In *International Conference on Database Theory (ICDT'07)*, pages 1–17. Springer, 2007.

26. M. Croitoru and S. Vesic. What can argumentation do for inconsistent ontology query answering? In *Proceedings of the International Conference on Scalable Uncertainty Management (SUM'13)*, pages 15–29. Springer, 2013.
27. S. Doutre and J. Mengin. On sceptical versus credulous acceptance for abstract argument systems. In *Logics in Artificial Intelligence*, pages 462–473. Springer, 2004.
28. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.
29. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence Journal*, 77:321–357, 1995.
30. P. E. Dunne and T. J. Bench-Capon. Two party immediate response disputes: properties and efficiency. *Artificial Intelligence*, 149(2):221–250, 2003.
31. T. Eiter and G. Gottlob. Hypergraph transversal computation and related problems in logic and ai. In *European Workshop on Logics in Artificial Intelligence*, pages 549–564. Springer, 2002.
32. R. Fagin. *Encyclopedia of Database Systems*, chapter Tuple-Generating Dependencies, pages 3201–3202. Springer US, 2009.
33. A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming*, 4(1+ 2):95–138, 2004.
34. H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *Proceedings of the 7th International Conference on Artificial Intelligence and Law*, pages 53–62. ACM, 1999.
35. M. W. Johnson, P. McBurney, and S. Parsons. When are two protocols the same? In *Communication in Multiagent Systems*, pages 253–268. Springer, 2003.
36. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proceedings of the International Conference on Web Reasoning and Rule Systems (RR'10)*, pages 103–117. Springer-Verlag, 2010.
37. B. Marnette. Generalized schema-mappings: from termination to tractability. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 13–22. ACM, 2009.
38. S. Modgil and M. Caminada. *Argumentation in Artificial Intelligence*, chapter Proof Theories and Algorithms for Abstract Argumentation Frameworks, pages 105–129. Springer US, 2009.
39. S. Modgil and H. Prakken. A general account of argumentation with preferences. *Artificial Intelligence*, 195:361–397, 2013.
40. S. J. Modgil and H. Prakken. A general account of argumentation with preferences. *Artificial Intelligence Journal*, page In Press, 2013.
41. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. In *Journal on data semantics X*, pages 133–173. Springer, 2008.
42. H. Prakken. Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese*, 127(1-2):187–219, 2001.
43. R. Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.
44. N. Rescher and R. Manor. On inference from inconsistent premisses. *Theory and decision*, 1(2):179–217, 1970.
45. P. M. Thang, P. M. Dung, and N. D. Hung. Towards a common framework for dialectical proof procedures in abstract argumentation. *Journal of Logic and Computation*, 19(6):1071–1109, 2009.
46. S. Vesic. Identifying the class of maxi-consistent operators in argumentation. *Journal of Artificial Intelligence Research*, 47:71–93, 2013.

47. G. A. Vreeswik and H. Prakken. Credulous and sceptical argument games for preferred semantics. In *Logics in Artificial Intelligence*, pages 239–253. Springer, 2000.

A Appendix

Example 20 (Pick two!). Complete set of arguments of Example 12.

$a_1 = (\{project(P), isfast(P), ischeap(P)\}, project(P) \wedge isfast(P) \wedge ischeap(P))$
$a_2 = (\{project(P), isfast(P), isgood(P)\}, project(P) \wedge isfast(P) \wedge isgood(P))$
$a_3 = (\{project(P), isgood(P), ischeap(P)\}, project(P) \wedge isgood(P) \wedge ischeap(P))$
$a_4 = (\{project(P), isfast(P), ischeap(P)\}, project(P) \wedge isfast(P))$
$a_5 = (\{project(P), isfast(P), ischeap(P)\}, project(P) \wedge ischeap(P))$
$a_6 = (\{project(P), isfast(P), ischeap(P)\}, isfast(P) \wedge ischeap(P))$
$a_7 = (\{project(P), isgood(P), ischeap(P)\}, project(P) \wedge isgood(P))$
$a_8 = (\{project(P), isgood(P), ischeap(P)\}, project(P) \wedge ischeap(P))$
$a_9 = (\{project(P), isgood(P), ischeap(P)\}, isgood(P) \wedge ischeap(P))$
$a_{10} = (\{project(P), isfast(P), isgood(P)\}, project(P) \wedge isfast(P))$
$a_{11} = (\{project(P), isfast(P), isgood(P)\}, project(P) \wedge isgood(P))$
$a_{12} = (\{project(P), isfast(P), isgood(P)\}, isfast(P) \wedge isgood(P))$
$a_{13} = (\{project(P), isfast(P), isgood(P)\}, project(P))$
$a_{14} = (\{project(P), isfast(P), isgood(P)\}, isfast(P))$
$a_{15} = (\{project(P), isfast(P), isgood(P)\}, isgood(P))$
$a_{16} = (\{project(P), isgood(P), ischeap(P)\}, project(P))$
$a_{17} = (\{project(P), isgood(P), ischeap(P)\}, isgood(P))$
$a_{18} = (\{project(P), isgood(P), ischeap(P)\}, ischeap(P))$
$a_{19} = (\{project(P), isfast(P), ischeap(P)\}, project(P))$
$a_{20} = (\{project(P), isfast(P), ischeap(P)\}, isfast(P))$
$a_{21} = (\{project(P), isfast(P), ischeap(P)\}, ischeap(P))$
$a_{22} = (\{project(P), isfast(P)\}, project(P) \wedge isfast(P))$
$a_{23} = (\{project(P), isgood(P)\}, project(P) \wedge isgood(P))$
$a_{24} = (\{isgood(P), ischeap(P)\}, isgood(P) \wedge ischeap(P))$
$a_{25} = (\{project(P), ischeap(P)\}, project(P) \wedge ischeap(P))$
$a_{26} = (\{isfast(P), ischeap(P)\}, isfast(P) \wedge ischeap(P))$
$a_{27} = (\{isfast(P), isgood(P)\}, isfast(P) \wedge isgood(P))$
$a_{28} = (\{project(P), isgood(P)\}, project(P))$
$a_{29} = (\{project(P), isgood(P)\}, isgood(P))$
$a_{30} = (\{isgood(P), ischeap(P)\}, isgood(P))$
$a_{31} = (\{isgood(P), ischeap(P)\}, ischeap(P))$
$a_{32} = (\{project(P), isfast(P)\}, project(P))$
$a_{33} = (\{project(P), isfast(P)\}, isfast(P))$
$a_{34} = (\{project(P), ischeap(P)\}, project(P))$
$a_{35} = (\{project(P), ischeap(P)\}, ischeap(P))$
$a_{36} = (\{isfast(P), ischeap(P)\}, isfast(P))$
$a_{37} = (\{isfast(P), ischeap(P)\}, ischeap(P))$
$a_{38} = (\{isfast(P), isgood(P)\}, isfast(P))$
$a_{39} = (\{isfast(P), isgood(P)\}, isgood(P))$
$a_{40} = (\{project(P)\}, project(P))$
$a_{41} = (\{isfast(P)\}, isfast(P))$
$a_{42} = (\{ischeap(P)\}, ischeap(P))$
$a_{43} = (\{isgood(P)\}, isgood(P))$
$a_{44} = (\{isfast(P)\}, isfast(P))$
$a_{45} = (\{project(P), isfast(P), isgood(P)\}, project(P) \wedge isfast(P) \wedge isgood(P) \wedge isexpensive(P))$
$a_{46} = (\{project(P), isfast(P), isgood(P)\}, isfast(P) \wedge isgood(P) \wedge isexpensive(P))$
$a_{47} = (\{project(P), isfast(P), isgood(P)\}, project(P) \wedge isexpensive(P))$
$a_{48} = (\{project(P), isfast(P), isgood(P)\}, isfast(P) \wedge isexpensive(P))$
$a_{49} = (\{project(P), isfast(P), isgood(P)\}, isgood(P) \wedge isexpensive(P))$
$a_{50} = (\{isfast(P), isgood(P)\}, isfast(P) \wedge isgood(P) \wedge isexpensive(P))$
$a_{51} = (\{project(P), isfast(P), isgood(P)\}, isexpensive(P))$
$a_{52} = (\{isfast(P), isgood(P)\}, isfast(P) \wedge isexpensive(P))$
$a_{53} = (\{isfast(P), isgood(P)\}, isgood(P) \wedge isexpensive(P))$
$a_{54} = (\{isfast(P), isgood(P)\}, isexpensive(P))$

Table 4: All arguments of Example 12.