Inconsistency-Tolerant Reasoning in Datalog$^{\pm}$ Ontologies via an Argumentative Semantics

Maria Vanina Martinez[1], Cristhian Ariel David Deagustini[2,3(✉)], Marcelo A. Falappa[2,3], and Guillermo Ricardo Simari[2]

[1] Department of Computer Science, University of Oxford, Oxford, UK
[2] Artificial Intelligence Research and Development Laboratory Department of Computer Science and Engineering, Universidad Nacional del Sur, Alem 1253, 8000 Bahía Blanca, Buenos Aires, Argentina
caddeagustini@gmail.com
[3] Consejo Nacional de Investigaciones Científicas y Técnicas, Buenos Aires, Argentina

Abstract. The Semantic Web provides an effective infrastructure that allows data to be easily shared and reused across applications. At its core is the description of ontological knowledge using ontological languages which are powerful knowledge representation tools with good decidability and tractability properties; Datalog$^{\pm}$ is one of these tools.The problem of inconsistency has been acknowledged in both the Semantic Web and Database Theory communities. Here we introduce elements of defeasible argumentative reasoning in Datalog$^{\pm}$, consequences to represent statements whose truth can be challenged leading to a better handling of inconsistency in ontological languages.

Keywords: Defeasible Argumentation · Inconsistency-tolerant reasoning · Datalog$^{\pm}$

1 Introduction and Motivation

The Semantic Web provides an effective infrastructure that allows data to be shared and reused across applications. At its core is the description of ontological knowledge using ontological languages which are powerful knowledge representation tools, since their decidability and tractability properties makes them attractive for handling practical applications. In particular, Datalog$^{\pm}$ [9] is a family of ontology languages which enables a modular rule-based style of knowledge representation. Datalog$^{\pm}$ provides the capability of representing fragments of first-order logic so that answering a Boolean Conjunctive query Q under a set Σ of Datalog$^{\pm}$ rules for an input database I is equivalent to checking whether Q is classically entailed from $I \cup \Sigma$. Furthermore, its properties of decidability of query answering and good query answering complexity in the data complexity allow to realistically assume that the database I is the only really large object in the input. These properties and its expressive power make Datalog$^{\pm}$ very useful in scenarios such as Ontology Querying, Web Data Extraction, and Ontology-based Data Access.

The problem of inconsistency in ontologies has been acknowledged in both the Semantic Web and Database Theory communities, and several methods have been developed to deal with it. The most widely accepted semantics for querying inconsistent

A.L.C. Bazzan and K. Pichara (Eds.): IBERAMIA 2014, LNAI 8864, pp. 15–27, 2014.
DOI: 10.1007/978-3-319-12027-0_2

databases is that of *consistent answers* [1] (or *AR* semantics in [19]), which yields the set of atoms that can be derived despite all possible ways of repairing the inconsistency. This semantics is based on the *"when in doubt, throw it out"* principle. We argue that the process of conflict resolution could be carried out through logical reasoning, using as much information as possible to weigh out conflicting pieces of information.

We introduce in Datalog$^{\pm}$ elements of defeasible reasoning that have already shown practical results [12,17], allowing consequences to represent statements whose acceptance can be challenged. Section 3 presents *defeasible* Datalog$^{\pm}$ ontologies extending classical Datalog$^{\pm}$ ontologies with defeasible atoms and defeasible *tuple-generating dependencies* (or TGDs) which are used as inference rules in Datalog$^{\pm}$. Defeasible atoms represent statements that can be challenged, while defeasible TGDs represent a weaker connection between pieces of information. Conflicts among derived atoms are resolved through an argumentative dialectical process. The argumentation-based reasoning mechanism described in Section 4 consider reasons for and against potential conclusions and decides which are the ones that can be obtained (warranted) from the knowledge base. In Section 5, we show that this extension allows to entail atoms that are not yielded by several inconsistency-tolerant semantics from the literature, including *AR* and several others designed to be sound approximations of *AR*; yet, we are guaranteed that a very important property holds: no conflicting atoms can be entailed – we call this the *NCE property*. We then go on to show how to obtain sound approximations (that also enjoy the NCE property) to a family of semantics called k-defeaters [7].

2 Preliminaries on Datalog$^{\pm}$ Ontologies

First, we briefly recall some basics on Datalog$^{\pm}$ [9], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple-generating dependencies (TGDs), negative constraints, the chase, and ontologies in Datalog$^{\pm}$.

We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathscr{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geq 0$. We assume a *relational schema* $\mathscr{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, \ldots, t_n)$, where P is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. A *database (instance)* I for a relational schema $\mathscr{R}$ is a (possibly infinite) set of atoms with predicates from $\mathscr{R}$ and arguments from Δ.

Given a relational schema $\mathscr{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathscr{R}$ (without nulls), called the *body* and the *head* of σ, respectively. Satisfaction of TGDs are defined via *homomorphisms*, which are mappings $\mu \colon \Delta \cup \Delta_N \cup \mathscr{V} \rightarrow \Delta \cup \Delta_N \cup \mathscr{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and

conjunctions of atoms. A TGD σ is satisfied in a database I for $\mathscr{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X},\mathbf{Y})$ to atoms of I, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X},\mathbf{Z})$ to atoms of I. A TGD σ is *guarded* iff an atom in its body contains all universally quantified variables of σ. Since TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has without loss of generalization a single atom in its head.

A *conjunctive query (CQ)* over $\mathscr{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X},\mathbf{Y})$, where $\Phi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but without nulls. A *Boolean CQ (BCQ)* over $\mathscr{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. The set of *answers* for a CQ Q to I and Σ, denoted $ans(Q,I,\Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(I,\Sigma)$. The *answer* for a BCQ Q to I and Σ is *Yes*, denoted $I \cup \Sigma \models Q$, iff $ans(Q,I,\Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [3], even when the schema and TGDs are fixed [8]. Decidability of query answering for the guarded case follows from a bounded tree-width property. The data complexity of query answering in this case is P-complete (see [9] for details).

The chase algorithm for a database I and a set of TGDs Σ consists of an exhaustive application of the TGDs [9] in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) chase for I and Σ. The (possibly infinite) chase relative to TGDs is a *universal model*, i.e., there exists a homomorphism from $chase(I,\Sigma)$ onto every $B \in mods(I,\Sigma)$ [9]. This implies that BCQs Q over I and Σ can be evaluated on the chase for I and Σ, i.e., $I \cup \Sigma \models Q$ is equivalent to $chase(I,\Sigma) \models Q$.

A *negative constraint* (or simply *constraint*) γ is a first-order formula of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the *body* of γ) is a conjunction of atoms over $\mathscr{R}$ (without nulls). Under the standard semantics of query answering of BCQs in Datalog$^{\pm}$ with TGDs, adding negative constraints is computationally easy, as for each constraint $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, we only have to check that the BCQ $\exists \mathbf{X}\, \Phi(\mathbf{X})$ evaluates to false in I under Σ; if one of these checks fails, then the answer to the original BCQ Q is true, otherwise the constraints can simply be ignored when answering the BCQ Q. In this work we restrict our attention to binary denial constraints. As we will show later, this class of constraints suffices for the formalization of the concept of conflicting atoms.

As another component, the Datalog$^{\pm}$language has special types of *equality-generating dependencies (EGDs)*. Without loss of generality we do not consider EGDs in this work, since for our purposes they can also be modeled via negative constraints (see [9] for details). We usually omit the universal quantifiers in TGDs, negative constraints, and we implicitly assume that all sets of dependencies and/or constraints are finite.

Datalog$^{\pm}$ Ontologies. A *Datalog$^{\pm}$ ontology* $KB = (I, \Sigma)$, where $\Sigma = \Sigma_T \cup \Sigma_{NC}$, consists of a database I, a set of TGDs Σ_T, and a set of negative constraints Σ_{NC}. We say KB is *guarded* (resp., *linear*) iff Σ_T is guarded (resp., linear). Example 1 (used in the sequel as a running example) illustrates a simple Datalog$^{\pm}$ ontology.

Example 1. Consider the following simple Datalog$^{\pm}$ ontology $KB = (I, \Sigma_T \cup \Sigma_{NC})$:
$I = \{collaborates(will,fbi), security\_agency(fbi), psychiatrist(hannibal,will),$
$\quad victim(abigail)\}$
$\Sigma_{NC} = \{risky\_job(P) \wedge unstable(P) \rightarrow \bot\}$

$$\Sigma_T = \{collaborates(P,A) \rightarrow works\_in(A,P), in\_therapy(P) \rightarrow unstable(P),$$
$$lives\_depend\_on(A) \wedge works\_in(A,P) \rightarrow risky\_job(P),$$
$$psychiatrist(S,P) \rightarrow in\_therapy(P), security\_agency(A) \rightarrow lives\_depend\_on(A)\}.$$

3 Defeasible Datalog$^{\pm}$ Ontologies

Here we extend Datalog$^{\pm}$ ontologies to allow defeasible reasoning. First, to represent statements whose acceptance can be challenged, we consider the existence of a set of *defeasible atoms*; thus, the database instance of a defeasible Datalog$^{\pm}$ ontology consists of two parts, a set of facts (*i.e.,* strict knowledge) and a set of defeasible atoms.

Second, we also want to add defeasibility to express weaker connections between pieces of information than in TGDs; thus, we extend the language accordingly. *Defeasible TGDs* are rules of the form $\Upsilon(\mathbf{X},\mathbf{Y}) \sim\!\!\succ \exists \mathbf{Z}\, \Psi(\mathbf{X},\mathbf{Z})$, where $\Upsilon(\mathbf{X},\mathbf{Y})$ and $\Psi(\mathbf{X},\mathbf{Z})$ are conjunctions of atoms. As in DeLP's defeasible rules [15], defeasible TGDs are used to represent weaker connections between the body and the head of a rule. Unlike strict (traditional) TGDs, acceptance of the body of a defeasible rule does not always lead to the acceptance of the head, which means that consequences of such rule can be challenged. For our running Example 1, we can represent the information that if A is a security agency then it is the case that the lives of people depend on A as a defeasible TGD instead of a strict one, reflecting that the connection between the two atoms holds in general but is weak in nature. Defeasible TGDs are written using the symbol "$\sim\!\!\succ$", while the classical (right) arrow "$\rightarrow$" is reserved to *strict* TGDs and NCs.

Then, a *defeasible* Datalog$^{\pm}$ *ontology KB* consists of a (finite) set F of *ground atoms*, called *facts*, a set D of *defeasible atoms*, a finite set of TGDs Σ_T, a finite set of defeasible TGDs Σ_D, and a finite set of binary negative constraints Σ_{NC}. The following example shows a defeasible Datalog$^{\pm}$ ontology that encodes the knowledge from Example 1 changing some of the facts and TGDs to defeasible ones.

Example 2. The information from the ontology presented in Example 1 can be better represented by the following defeasible Datalog$^{\pm}$ ontology $KB = (F, D, \Sigma'_T, \Sigma_D, \Sigma_{NC})$, where $F = \{collaborates(will, fbi), security\_agency(fbi), psychiatrist(hannibal, will)\}$ and $D = \{victim(abigail)\}$. Note that we have changed the fact stating that *abigail* is a victim to a defeasible atom since some suspicious actions from her indicate that she may be an accomplice instead. The sets of TGDs, and defeasible TGDs are now given by the following sets; note that we have changed some of the TGDs into defeasible TGDs to make clear that the connection between the head and body is weaker.

$$\Sigma_{T'} = \{collaborates(P,A) \rightarrow works\_in(A,P), psychiatrist(S,P) \rightarrow in\_therapy(P)\}$$
$$\Sigma_D = \{in\_therapy(P) \sim\!\!\succ unstable(P), lives\_depend\_on(A) \wedge works\_in(A,P) \sim\!\!\succ risky\_job(P),$$
$$security\_agency(A) \sim\!\!\succ lives\_depend\_on(A)\}$$

As in classical Datalog$^{\pm}$, derivations from a defeasible Datalog$^{\pm}$ ontology rely in the application of (strict or defeasible) TGDs. Given a defeasible Datalog$^{\pm}$ ontology *KB* the classical application of a TGD applies almost directly to defeasible TGDs and ontologies. The difference is that for a (strict or defeasible) TGD σ to be applicable

there must exist a homomorphism mapping the atoms in the body of σ into $F \cup D$. The *application of* σ on KB generates a new atom from the head of σ if it is not already in $F \cup D$, in the same way as explained in Section 2. The following definitions follow similar ones first introduced in [22]. Here we adapt the notions to defeasible Datalog$^{\pm}$ ontologies.

Definition 1. Let $KB = (F, D, \Sigma_T, \Sigma_D, \Sigma_{NC})$ be a defeasible Datalog$^{\pm}$ ontology and L an atom. An *annotated derivation* ∂ of L from KB consists of a finite sequence $[R_1, R_2, \ldots, R_n]$ such that R_n is L, and each atom R_i is either: (*i*) R_i is a fact or defeasible atom, *i.e.,* $R_i \in F \cup D$, or (*ii*) there exists a TGD $\sigma \in \Sigma_T \cup \Sigma_D$ and a homomorphism h such that $h(head(\sigma)) = R_i$ and σ is applicable to the set of all atoms and defeasible atoms that appear before R_i in the sequence. When no defeasible atoms and no defeasible TGDs are used in a derivation, we say the derivation is a *strict derivation*, otherwise it is a *defeasible derivation*.

Note that there is non-determinism in the order in which the elements in a derivation appear. Syntactically distinct derivations are, however, equivalent for our purposes. When no confusion is possible, we assume that a unique selection has been made.

We say an atom a is strictly derived from KB iff there exists a strict derivation for a from KB, denoted with $KB \vdash a$, and a is defeasibly derived from KB iff there exists a defeasible derivation for a from KB and no strict derivation exists, denoted with $KB \mathrel{|\!\sim} a$.

A derivation ∂ for L is *minimal* if no proper sub-derivation ∂' of ∂ (every member of ∂' is a member of ∂) is also an annotated derivation of L. Considering minimal derivations in a defeasible derivation avoids the insertion of unnecessary elements that will weaken its ability to support the conclusion by possibly introducing unnecessary points of conflict. Given a derivation ∂ for L, there exists at least one minimal sub-derivation $\partial' \subseteq \partial$ for an atom L. Thus, we only consider minimal derivations.

Example 3. From the defeasible Datalog$^{\pm}$ ontology in Example 2, we can get the following (minimal) annotated derivation for atom $unstable(will)$:
$\partial = [psychiatrist(hannibal, will), psychiatrist(S,P) \rightarrow in\_therapy(P),$
$in\_therapy(will), in\_therapy(P) \mathrel{|\!\sim} unstable(P), unstable(will)]$
Then, we have $KB \vdash in\_therapy(will)$ (following from the fact that Will has a psychiatrist) and $KB \mathrel{|\!\sim} unstable(will)$ (by means of the defeasible rule that says that a person in therapy generally is unstable).

We now show that classical query answering in defeasible Datalog$^{\pm}$ ontologies is equivalent to query answering in Datalog$^{\pm}$ ontologies.

Proposition 1. *Let L be a ground atom, $KB = (F, D, \Sigma_T, \Sigma_D, \Sigma_{NC})$ be a defeasible Datalog$^{\pm}$ ontology, $KB' = (F \cup D, \Sigma'_T \cup \Sigma_{NC})$ is a classical Datalog$^{\pm}$ ontology where $\Sigma'_T = \Sigma_T \cup \{\Upsilon(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z}) \mid \Upsilon(\mathbf{X}, \mathbf{Y}) \mathrel{|\!\sim} \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})\}$. Then, $KB' \models L$ iff $KB \vdash L$ or $KB \mathrel{|\!\sim} L$.*

Proposition 1 states the equivalence between derivations from defeasible Datalog$^{\pm}$ ontologies and entailment in traditional Datalog$^{\pm}$ ontologies whose database instance corresponds to the union of facts and defeasible atoms, and the set of TGDs corresponds to the union of the TGDs and the strict version of the defeasible TGDs. As a

direct consequence, all the existing work done for Datalog$^{\pm}$ directly applies to defeasible Datalog$^{\pm}$. In particular, it is easy to specify a defeasible Chase procedure over defeasible Datalog$^{\pm}$ ontologies, based on the revised notion of application of (defeasible) TGDs, whose result is a *universal model*. Therefore, a (B)CQ Q over a defeasible Datalog$^{\pm}$ ontology can be evaluated by verifying that Q is a classical consequence of the chase obtained from the defeasible Datalog$^{\pm}$ ontology. Despite this equivalence, the main reason, for defining a defeasible extension of Datalog$^{\pm}$ ontologies, is that defeasible knowledge and reasoning allows the possibility of managing conflicts in a more sensible way; *i.e.,* aspects of the nature of the different pieces of knowledge in conflict and/or the way they are derived from previous knowledge can be considered in the process of conflict resolution. For this reason, in the following section we propose an argumentation-based procedure to answer queries in defeasible Datalog$^{\pm}$ ontologies.

4 Argumentation-Based Reasoning in Defeasible Datalog$^{\pm}$

Conflicts in defeasible Datalog$^{\pm}$ ontologies come, as in classical Datalog$^{\pm}$, from the violation of negative constraints. Intuitively, two atoms are in conflict relative to a defeasible Datalog$^{\pm}$ ontology whenever they are both derived from the ontology (either strictly o defeasible) and together map to the body of a negative constraint.

Definition 2. Given a set of negative constraints Σ_{NC}, two ground atoms (possibly with nulls) a and b are said to be *in conflict* relative to Σ_{NC} iff there exists an homomorphism h such that $h(body(\upsilon)) = a \wedge b$ for some $\upsilon \in \Sigma_{NC}$.

In what follows, we say that a set of atoms is a *conflicting* set of atoms relative to Σ_{NC} if and only if there exist at least two atoms in the set that are in conflict relative to Σ_{NC}, otherwise will be called *non-conflicting*. Whenever is clear from the context we omit the set of negative constraints.

Example 4. Consider the set $\Sigma_{NC} = \{risky\_job(P) \wedge unstable(P) \rightarrow \bot\}$ of negative constraints from Example 3. In this case, the set of atoms $\{unstable(will), risky\_job(will)\}$ is a conflicting set relative to Σ_{NC}. However, this is not the case for the set $S = \{collaborates(will,fbi), psychiatrist(hannibal,will), security\_agency(fbi)\}$.

Given a defeasible Datalog$^{\pm}$ ontology $KB = (F, D, \Sigma_T, \Sigma_D, \Sigma_{NC})$, sets F and Σ_T are used to represent non-defeasible information, as it is the case with facts and strict rules in DeLP [15,16]. Therefore, we require F to be *representationally coherent*, that is F must be non-conflicting with respect to Σ_{NC} and furthermore given KB there cannot be strict derivations for conflicting atoms.

Whenever defeasible derivations of conflicting atoms exist, we use, as in DeLP, a dialectical process to decide which information prevails, *i.e.,* which piece of information is such that no acceptable reason can be put forward against it. Reasons are supported by arguments. An argument is an structure that supports a claim from evidence through the use of a reasoning mechanism. Maintaining the intuition that led to the classic definition of arguments in [27], given a defeasible Datalog$^{\pm}$ ontology, an argument $\mathscr{A}$ for a claim L is a minimal (under $\subseteq$) set of facts, defeasible atoms, TGDs, and defeasible TGDs

contained in KB, such that L is derived from it and no conflicting atoms can be derived from it, and $\mathbb{A}_{KB}$ denotes the set of all arguments that can be built from KB.

Answers to atomic queries are supported by arguments built from the ontology. However, it is possible to build arguments for conflicting atoms, and so arguments can *attack* each other. We now adopt the definitions of counter-argument and attacks for defeasible Datalog$^{\pm}$ ontologies from [15]. First, an argument $\langle \mathscr{B}, L' \rangle$ is a sub-argument of $\langle \mathscr{A}, L \rangle$ if $\mathscr{B} \subseteq \mathscr{A}$. Argument $\langle \mathscr{A}_1, L_1 \rangle$ counter-argues, rebuts, or attacks $\langle \mathscr{A}_2, L_2 \rangle$ at literal L, iff there exists a sub-argument $\langle \mathscr{A}, L \rangle$ of $\langle \mathscr{A}_2, L_2 \rangle$ such that L and L_1 conflict.

Example 5. Consider derivation ∂ from Example 3 and let $\mathscr{A}$ be the set of (defeasible) atoms and (defeasible) TGDs used in ∂. $\mathscr{A}$ is an argument for $unstable(will)$. Also, we can obtain a minimal derivation ∂' for $risky\_job(will)$ where $\mathscr{B}$, the set of (defeasible) atoms and (defeasible) TGDs used in ∂', is such that no conflicting atoms can be defeasibly derived from $\mathscr{B} \cup \Sigma_T$. As $\{unstable(will), risky\_job(will)\}$ is conflicting relative to Σ_{NC}, we have that $\langle \mathscr{A}, unstable(will) \rangle$ and $\langle \mathscr{B}, risky\_job(will) \rangle$ attack each other.

Once the attack relation is established between arguments, it is necessary to analyze whether the attack is strong enough so one of the arguments can *defeat* the other. Given an argument $\mathscr{A}$ and a counter-argument $\mathscr{B}$, a comparison criterion is used to determine if $\mathscr{B}$ is preferred to $\mathscr{A}$ and, therefore, *defeats* $\mathscr{A}$. Different preference criteria can be applied for this purpose; *specificity* [28] is often used in the defeasible reasoning and argumentation literature. In the presence of defeasible atoms, specificity might not always return the intended results — other preference criteria have been developed for such cases [15,22]. For our defeasible Datalog$^{\pm}$ framework, unless otherwise stated, we assume an arbitrary preference criterion $\succ$ among arguments.

Let $\langle \mathscr{A}_1, L_1 \rangle$ and $\langle \mathscr{A}_2, L_2 \rangle$ be two arguments. We say that argument $\langle \mathscr{A}_1, L_1 \rangle$ is a defeater of $\langle \mathscr{A}_2, L_2 \rangle$ iff there exists a sub-argument $\langle \mathscr{A}, L \rangle$ of $\langle \mathscr{A}_2, L_2 \rangle$ such that $\langle \mathscr{A}_1, L_1 \rangle$ counter-argues $\langle \mathscr{A}, L \rangle$ at L, and either $\langle \mathscr{A}_1, L_1 \rangle \succ \langle \mathscr{A}, L \rangle$ (it is a proper defeater) or $\langle \mathscr{A}_1, L_1 \rangle \not\succ \langle \mathscr{A}, L \rangle$, and $\langle \mathscr{A}, L \rangle \not\succ \langle \mathscr{A}_1, L_1 \rangle$ (it is a blocking defeater).

Definition 3. Given a defeasible Datalog$^{\pm}$ ontology KB defined over a relational schema $\mathscr{R}$, a *Datalog$^{\pm}$ argumentation framework* $\mathfrak{F}$ is a tuple $\langle \mathscr{L}_{\mathscr{R}}, \mathbb{A}_{KB}, \succ \rangle$, where $\succ$ specifies a preference relation defined over $\mathbb{A}_{KB}$.

To decide whether an argument $\langle \mathscr{A}_0, L_0 \rangle$ is undefeated within a Datalog$^{\pm}$ argumentation framework, all its defeaters must be considered, and there may exist defeaters for their counter-arguments as well. An *argument line* for $\langle \mathscr{A}_0, L_0 \rangle$ is defined as a sequence of arguments that starts at $\langle \mathscr{A}_0, L_0 \rangle$, and every element in the sequence is a defeater of its predecessor in the line [15]. Note that for defeasible Datalog$^{\pm}$ ontologies arguments in an argumentation line can contain both facts and defeasible atoms.

Different argumentation systems can be defined by setting a particular criterion for proper attack or defining the admissibility of argumentation lines. Here, we adopt the one from [15], which states that an argumentation line has to be finite, and no argument is a sub-argument of an argument used earlier in the line; furthermore, when an argument $\langle \mathscr{A}_i, L_i \rangle$ is used as a blocking defeater for $\langle \mathscr{A}_{i-1}, L_{i-1} \rangle$ during the construction of an argumentation line, only a proper defeater can be used for defeating $\langle \mathscr{A}_i, L_i \rangle$.

The dialectical process considers all possible admissible argumentation lines for an argument, which together form a dialectical tree. Dialectical trees for defeasible

Datalog$^{\pm}$ ontologies are defined following [15], and we adopt the notion of coherent dialectical tree from [22], which ensures that the use of defeasible atoms is *coherent* in the sense that conflicting defeasible atoms are not used together in supporting (or attacking) a claim. We denote with $Args(\mathscr{T})$ the set of arguments in $\mathscr{T}$.

Argument evaluation, *i.e.,* determining whether the root node of the tree is defeated or undefeated, is done by means of a *marking* or *labelling* criterion, similar to the grounded semantics in abstract argumentation frameworks [2,13]. Each node in an argument tree is labelled as either defeated (D) or undefeated (U). We denote the root of $\mathscr{T}(\langle \mathscr{A}, L\rangle)$ with $root(\mathscr{T}(\langle \mathscr{A}, L\rangle))$, and $marking(N)$, where N is a node in a dialectical tree, denotes the value of the marking for node N (either U or D). Deciding whether a node is defeated or undefeated depends on whether or not all its children are defeated: (1) if node N is a leaf then $marking(N) = U$, (2) node N is such that $marking(N) = D$ iff at least one of its children that is marked with U, and (3) node N is such that $marking(N) = U$ iff all its children are marked with D.

Definition 4. Let KB be a defeasible Datalog$^{\pm}$ ontology and $\mathfrak{F}$ the corresponding Datalog$^{\pm}$ argumentation framework. An atom L is *warranted* in $\mathfrak{F}$ (through $\mathscr{T}$) iff there exists an argument $\langle \mathscr{A}, L\rangle$ such that $marking(root(\mathscr{T}(\langle \mathscr{A}, L\rangle))) = U$. We say that L is entailed from KB (through $\mathfrak{F}$), denoted with $KB \models_{\mathfrak{F}} L$, iff it is *warranted* in $\mathfrak{F}$.

Example 6. Continuing with KB from Example 2, consider its corresponding Datalog$^{\pm}$ argumentation framework $\mathfrak{F}$, the atom $unstable(will)$ is warranted through $\mathfrak{F}$ under the assumption that arguments $\mathscr{A}$ and $\mathscr{B}$ from Example 5 are such that $\mathscr{A} \succ \mathscr{B}$.

The following proposition establishes that no conflicting sets of atoms can be entailed/ warranted from a Datalog$^{\pm}$ argumentation framework.

Proposition 2. *Let $KB = (F, D, \Sigma_T, \Sigma_D, \Sigma_{NC})$ be a defeasible Datalog$^{\pm}$ ontology. No two atoms L_1 and L_2 that are warranted in $\mathfrak{F}$ are conflicting relative to Σ_{NC}.*

We regard this property as desirable for inconsistency-tolerant reasoning mechanisms which is related to a similar one introduced in [10]. We formalize it as follows.

Non-Conflicting Entailment (NCE). Given a knowledge base $\mathscr{K}$ and a set of binary negative constraints Σ_{NC}, any entailment operator $\models$ satisfies the NCE property iff for any two atoms L_1 and L_2 $\mathscr{K} \models L_1$ and $\mathscr{K} \models L_2$ are non-conflicting relative to Σ_{NC}.

5 A Comparison with Inconsistency-Tolerant Semantics

Although query answering in Datalog$^{\pm}$ does not contemplate the possibility of returning meaningful answers in the presence of conflicts, a variety of inconsistency-tolerant semantics have been developed in the last decade for ontological languages, including lightweight Description Logics (DLs), such as $\mathscr{EL}$ and *DL-Lite* [7,19], and several fragments of Datalog$^{\pm}$ [20]. In this section we analyze entailment in defeasible Datalog$^{\pm}$ ontologies in relation to several inconsistency-tolerant semantics for ontological languages: *AR* semantics [19], *CAR* semantics [19], *IAR*, *k*-support [7], and *ICAR* semantics that are sound approximations of *AR* and of *CAR*, respectively,

and finally, the k-defeater semantics [6] that comprises a family of complete approximation of *AR*.

We present the basic concepts needed to understand the different semantics on Datalog$^{\pm}$ ontologies and then show how entailment under such semantics compare to entailment on defeasible Datalog$^{\pm}$. We first recall the notion of *repair*; in relational databases a repair is a model of the set of integrity constraints that is maximally close, *i.e., "as close as possible"* to the original database. Different notions of repairs have been developed depending on the meaning of "closeness" used and on the type of constraints. For a Datalog$^{\pm}$ ontology $KB = (I, \Sigma_T \cup \Sigma_{NC})$, repairs are maximal subsets of I such that their consequences with respect to Σ_T are non-conflicting relative Σ_{NC}.

AR Semantics. The *AR* semantics corresponds to the notion of *consistent answers* in relational databases [1]. Intuitively, an atom L is said to be *AR*-consistently entailed from a Datalog$^{\pm}$ ontology KB, denoted $KB \models_{AR} L$ iff L is classically entailed from every ontology that can be built from every possible repair.

CAR Semantics. Another definition of repairs was also proposed in [19] that includes knowledge that comes from the closure of the database instance with respect to the set of TGDs. Since the closure of an inconsistent ontology yields the whole language, they define the *consistent closure* of an ontology $KB = (I, \Sigma_T \cup \Sigma_{NC})$ as the set $CCL(KB) = \{\alpha \mid \alpha \in \mathscr{H}(\mathscr{L}_{\mathscr{R}})\, s.t.\ \exists S \subseteq I \text{ and } mods(S, \Sigma_T \cup \Sigma_{NC}) \neq \emptyset \text{ and } (S, \Sigma_T) \models \alpha\}$. A *Closed ABox repair* of a Datalog$^{\pm}$ ontology KB is a consistent subset I' of $CCL(KB)$ such that it maximally preserves the database instance [19]. It is said that an atom L is *CAR*-consistently entailed from a Datalog$^{\pm}$ ontology KB, denoted by $KB \models_{CAR} L$ iff L is classically entailed from every ontology built from each possible closed ABox repair.

The following result shows that every atom that is *AR*-consistently (resp., *CAR*-consistently) entailed from a Datalog$^{\pm}$ ontology $KB = (I, \Sigma_T \cup \Sigma_{NC})$ is also entailed from the defeasible Datalog$^{\pm}$ ontology $KB' = (\emptyset, I, \Sigma_T, \emptyset, \Sigma_{NC})$ constructed from KB. This transformation from Datalog$^{\pm}$ to defeasible Datalog$^{\pm}$ is without loss of generality; the inconsistency-tolerant semantics that we study here assume that the knowledge contained in I is somehow *challengeable* as it can be in conflict once considered together with the set of constraints. Defeasible Datalog$^{\pm}$ ontologies allow to express both strict and defeasible knowledge, however the strict part is assumed to be consistent in itself (in particular, it could be empty), therefore, to make a fair comparison between approaches we need to translate the data contained in I to defeasible atoms.

Theorem 1. *Let $KB = (I, \Sigma_T \cup \Sigma_{NC})$ be a Datalog$^{\pm}$ ontology, $KB' = (\emptyset, I, \Sigma_T, \emptyset, \Sigma_{NC})$ be a defeasible Datalog$^{\pm}$ ontology and $\mathfrak{F} = \langle \mathscr{L}_{\mathscr{R}}, \mathbb{A}_{KB'}, \succ \rangle$. Then, (i) if $KB \models_{AR} L$ then $KB' \models_{\mathfrak{F}} L$, and (ii) if $KB \models_{CAR} L$ then $KB' \models_{\mathfrak{F}} L$.*

The converse does not hold. In our running example, $unstable(will)$ is not entailed by *AR* neither by *CAR* since every (closed) ABox repair either entails $unstable(will)$ or $risky\_job(will)$, but not both. However, as shown in Example 6, it is entailed from KB as $\mathscr{A} \succ \mathscr{B}$. Depending on the preference criterion, it could be the case that $unstable(will)$ would not be entailed, in which case $risky\_job(will)$ could be, or neither would be. The results from Theorem 1 directly extends to the sound approximations for *AR* and *ICAR* defined in [19] and the family of k-support semantics from [7].

Finally, we analyze the k-defeater semantics from [7]. Given a Datalog$^{\pm}$ ontology $KB = (I, \Sigma_T \cup \Sigma_{NC})$, an atom L is entailed from KB under the k-defeater semantics, $KB \models_{k\text{-}def} L$, for some $k \geq 0$, iff no set of facts with size smaller than k is such that it *contradicts* every minimal set from I that yields L. From an argumentation point of view, this semantics looks for counter-arguments for L up to a certain size – the size of an argument being the number of (defeasible) atoms used. If no such argument can be found, L is entailed from KB. Conflicting atoms could be entailed from KB for any k (except for that in which converges to AR), therefore it does not enjoy the NCE property.

In the following, consider $\succ_{k\text{-}def}$, a preference criterion such that $\mathscr{A} \succ_{k\text{-}def} B$ iff the number of facts and defeasible atoms used in $\mathscr{A}$ is less than or equal to k, for any $k \geq 0$.

Theorem 2. *Let $KB = (I, \Sigma_T \cup \Sigma_{NC})$ be a Datalog$^{\pm}$ ontology, $KB' = (\emptyset, I, \Sigma_T, \emptyset, \Sigma_{NC})$ be a defeasible Datalog$^{\pm}$ ontology, $\mathfrak{F} = \langle \mathscr{L}_{\mathscr{R}}, \mathbb{A}_{KB'}, \succ \rangle$, and $\mathfrak{F}' = \langle \mathscr{L}_{\mathscr{R}}, \mathbb{A}_{KB'}, \succ_{k\text{-}def} \rangle$, Then, (i) if $KB' \models_{\mathfrak{F}} L$ then $KB \models_{0\text{-}def} L$, and (ii) for any $0 \leq k < k'$ if $KB' \models_{\mathfrak{F}'} L$ then $KB \models_{k\text{-}def} L$, where k' is the point where AR and k-defeater semantics coincide.*

Statement (i) from Theorem 2 shows, unsurprisingly, that independently of the preference criterion defined over $\mathbb{A}_{KB'}$, $\models_{\mathfrak{F}}$ is a sound approximation to 0-defeaters, which corresponds to the *brave* semantics in which anything that can be classically entailed from some repair is *consistently* entailed from the ontology. Furthermore, given Proposition 2, this approximation is not only sound but it also satisfies the NCE property.

More interesting is statement (ii) showing that, using $\succ_{k\text{-}def}$, argumentation-based entailment on defeasible Datalog$^{\pm}$ is a sound approximation of the k-defeater semantics for every k (up to the point where k-defeater coincides with AR). Furthermore, since Proposition 2 ensures the NCE property independently of the preference criterion, we have obtained a family of semantics that soundly approximate the k-defeater semantics and ensure that no conflicting atoms can be entailed from a defeasible ontology.

6 Related Work

Within Artificial Intelligence, many efforts to deal with inconsistent information have been developed in the last four decades. Frameworks such as default logic [26] can be used to represent a database DB with integrity constraints IC as a default logic theory where the background theory consists of the IC and the facts in D constitutes the defaults rules, *i.e.,* a fact in D is assumed to be true if it can be assumed to be true. Argumentation [13,15,23,27] has been used for handling uncertainty and inconsistency by means of reasoning about how contradictory arguments defeat each other.

The most widely accepted semantics for querying a possibly inconsistent database is that of *consistent answers*, which yields the set of tuples (atoms) that appear in the answer to the query over *every* possible repair, which we have discussed in more detail in Section 5. More recently, Ontology-based Data Access approach to data integration, has led to a resurgence of interest in this area, specially focusing on the development of efficient inconsistency-tolerant reasoning and query answering in DLs and other ontology languages. Lately, several works have focused on inconsistency handling for different classes of DLs, adapting and specializing general techniques previously considered for traditional logics [18,21,24]. In [19], the adaptation of CQA for *DL-Lite*

ontologies and several sound and complete approximation are studied. Computing consistent answers is an inherently hard problem, [19] shows co-NP completeness for ground atomic queries in *DL-Lite*, though some works identify cases for very simple ontologies and restricted queries (within the *DL-Lite* family) for which tractable results can be obtained [6]. In [20], an alternative semantics called *k*-lazy is proposed, which relaxes the notion of repairs by adopting a compromise between quality of answers and tractability for fragments of Datalog$^{\pm}$. Section 5 presents some preliminary comparison between these semantics and our approach, a more detailed analysis is leave for future work.

Finally, more recently, there have been several developments in combining argumentation methods with description logics in order to overcome inconsistency and incoherence in such ontological languages, such as the work of [29] and [11]. In [29] an argumentation framework for reasoning and management in (inconsistent or incoherent) description logic ontologies, based Besnard and Hunter's framework [5], is proposed. In [11], on the other hand, Dung's abstract frameworks are used instead, and consistent answers under de *ICR* semantics [6] are obtained by means of obtaining the sets of accepted answers under some of the classical argumentation semantics. The main difference with our approach is that these frameworks are based on classical logic consequences and therefore, in the end answers and repairs are equivalent to those obtained for particular semantics of CQA, whose relationship we have discussed in Section 5.

7 Conclusions

We have introduced the idea of defeasible reasoning over Datalog$^{\pm}$ ontologies by introducing the construction of arguments, the definition of conflict using the negative constrains available in the system, and completing the argumentative infrastructure by characterizing defeat employing a preference criterion that has remained as an abstract element to be instantiated. The dialectical process to decide what arguments are warranted is done applying the classic techniques of argumentation theory. Furthermore, we show how such approach ensures the reasonableness of the answers given by it, as no conflicting atoms can be entailed/warranted in it [4,25].

We have also shown that atoms entailed from a Datalog$^{\pm}$ ontology, under well-known inconsistency-tolerant semantics, namely *AR* and *CAR* semantics, and sound approximations of these, are also entailed from the corresponding defeasible Datalog$^{\pm}$ ontology that includes the database instance of the ontology as defeasible atoms. Moreover, we have shown that the converse property does not hold in general, and therefore argumentation-based query answering for defeasible Datalog$^{\pm}$ ontologies allows to produce answers that though are involved in conflicts, and therefore are not consistent answers, the ontology contains enough information in order to warrant them. Furthermore, we show how to construct a Datalog$^{\pm}$ argumentation framework that yields a semantics that is a sound approximation to the *k*-defeaters semantics from [6], that enjoys the property of never entailing conflicting atoms.

Future work will involve a full complexity study of argumentation-based entailment for defeasible Datalog$^{\pm}$ ontologies to complete the comparison with the different inconsistent-tolerant semantics and better understand possible implementation problems for the framework [14].

Acknowledgments. This work was partially supported by CONICET Argentina, SeGCyT Universidad Nacional del Sur in Bahia Blanca, UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/J008346/1 (PrOQAW).

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. of PODS, pp. 68–79 (1999)
2. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. Knowledge Eng. Review **26**(4), 365–410 (2011)
3. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Even, S., Kariv, O. (eds.) Automata, Languages and Programming. LNCS, vol. 115, pp. 73–85. Springer, Heidelberg (1981)
4. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press (2008)
5. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artif. Intell. **128**(1–2), 203–235 (2001)
6. Bienvenu, M.: On the complexity of consistent query answering in the presence of simple ontologies. In: Proc. of AAAI (2012)
7. Bienvenu, M., Rosati, R.: Tractable approximations of consistent query answering for robust ontology-based data access. In: Proc. of IJCAI (2013)
8. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. of KR, pp. 70–80 (2008)
9. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. **14**, 57–83 (2012)
10. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artif. Intell. **171**(5–6), 286–310 (2007)
11. Croitoru, M., Vesic, S.: What Can Argumentation Do for Inconsistent Ontology Query Answering? In: Liu, W., Subrahmanian, V.S., Wijsen, J. (eds.) SUM 2013. LNCS, vol. 8078, pp. 15–29. Springer, Heidelberg (2013)
12. Deagustini, C.A.D., Dalibón, S.E.F., Gottifredi, S., Falappa, M.A., Chesñevar, C.I., Simari, G.R.: Relational databases as a massive information source for defeasible argumentation. Knowl. Based Syst. **51**, 93–109 (2013)
13. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. Artif. Intell. **77**, 321–357 (1995)
14. Dunne, P., Wooldridge, M.: Complexity of Abstract Argumentation. In: Argumentation in Artificial Intelligence, pp. 85–104. Springer (2009)
15. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. TPLP **4**(1–2), 95–138 (2004)
16. García, A.J., Simari, G.R.: Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers. Argument & Computation **5**(1), 63–88 (2014)
17. Gómez, S.A., Chesñevar, C.I., Simari, G.R.: Ontoarg: A decision support framework for ontology integration based on argumentation. Expert Syst. Appl. **40**(5), 1858–1870 (2013)
18. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proc. of IJCAI, pp. 354–359 (2005)
19. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 103–117. Springer, Heidelberg (2010)
20. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Inconsistency handling in Datalog+/- ontologies. In: Proc. of ECAI, pp. 558–563 (2012)

21. Ma, Y., Hitzler, P.: Paraconsistent Reasoning for OWL 2. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 197–211. Springer, Heidelberg (2009)
22. Martinez, M.V., García, A.J., Simari, G.R.: On the use of presumptions in structured defeasible reasoning. In: Proc. of COMMA, pp. 185–196 (2012)
23. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. J. Appl. Non-Classical Logics 7(1) (1997)
24. Qi, G., Du, J.: Model-based revision operators for terminologies in description logics. In: Proc. of IJCAI, pp. 891–897 (2009)
25. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence. Springer (2009)
26. Reiter, R.: A logic for default reasoning. Artif. Intel. **13**(1–2), 81–132 (1980)
27. Simari, G.R., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. Artif. Intell. **53**(2–3), 125–157 (1992)
28. Stolzenburg, F., García, A.J., Chesñevar, C.I., Simari, G.R.: Computing generalized specificity. J. of Applied Non-Classical Logics **13**(1), 87–113 (2003)
29. Zhang, X., Lin, Z.: An argumentation framework for description logic ontology reasoning and management. J. Intell. Inf. Syst. **40**(3), 375–403 (2013)