On the Influence of Incoherence in Inconsistency-tolerant Semantics for Datalog$^{\pm}$

C. A. D. Deagustini
M. V. Martinez M. A. Falappa G. R. Simari

Artificial Intelligence Research and Development Laboratory (LIDIA)
Institute for Computer Science and Engineering
Universidad Nacional del Sur - Consejo Nacional de Investigaciones Científicas y Técnicas
(UNS) (CONICET)

Motivation

- The problem of inconsistency in ontologies has been extensively acknowledged in AI.
- Several of the most known inconsistency-tolerant semantics often assume that there is no *incoherence*, a problem related to internal conflicts on the set of constraints [Flouris *et al.*, 2006].
- As a result, since they were not designed to acknowledge incoherence, such semantics for query answering fail at computing good quality answers in the presence of incoherence.
- We argue that, in more general scenarios, we have to distinguish between those different conflicts, and possibly consider alternative semantics suitable for dealing with both incoherent and inconsistent knowledge.

Talk Outline

This talk comprises three different building blocks:

- First, we introduce the notion of incoherence for Datalog$^{\pm}$ ontologies.
- Second, we show how such notion affects most of well-known inconsistency-tolerant semantics.
- Finally, we propose a definition for incoherence-tolerant semantics, introducing an alternative semantics based on an argumentative reasoning process that falls under such definition.

Preliminaries in Datalog$^{\pm}$

Datalog$^{\pm}$ is a family of ontology languages that enables a modular rule-based style of knowledge representation, which is based on the combination of four different components.

- Database D: a database D is a finite set of atoms.

$$D : \{can\_sing(simone), rock\_singer(axl)\}$$

- TGDs: a tuple-generating dependency (TGD) σ is a (possibly existentially quantified) formula which can be used to complete the database.

$$rock\_singer(X) \rightarrow can\_sing(X),$$
$$musician(X) \rightarrow \exists Y\, plays\_in(X, Y)$$

Preliminaries in Datalog$^{\pm}$

- EGDs: equality-generating dependencies (EGDs) are formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow X_i = X_j$ which have a two-fold semantics: on the one hand, they can be used to "unify" a null value to a constant; on the other hand, they can be used to check if some constant terms in two atoms are equal.

$$manage(X, Y) \wedge manage(X, Z) \rightarrow Y = Z$$

- NCs: Negative constraints (NCs) are formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \perp$, where the body $\mathbf{X}$ is a conjunction of atoms (without nulls) and the head is the truth constant *false*, denoted $\perp$. Intuitively, the atoms in the body of a NC cannot be true altogether.

$$unknown(X) \wedge famous(X) \rightarrow \perp$$

Datalog$^{\pm}$ ontologies and consistency

- A *Datalog$^{\pm}$ ontology* $KB = (D, \Sigma)$, where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a finite database D of ground atoms, a set of TGDs Σ_T, a set of separable EGDs Σ_E, and a set of negative constraints Σ_{NC}.
- We use the classical notion for consistency in Datalog$^{\pm}$, which states that consistent ontologies are those that have some models (supersets of the component D that satisfy every formula in Σ).

Definition (Consistency)

A Datalog$^{\pm}$ ontology $KB = (D, \Sigma)$ is *consistent* iff $mods(D, \Sigma) \neq \emptyset$. We say that KB is inconsistent otherwise.

Incoherence in Datalog$^{\pm}$

- From an operational point of view, inconsistencies appear in a Datalog$^{\pm}$ ontology whenever a NC or an EGD is violated (their bodies can be obtained either in D or by applying TGDs).
- A different kind of conflict appears when the TGDs in Σ_T cannot be applied without always leading to the violation of the NCs or EGDs.
- This issue is related to that of *unsatisfiability of a concept* in an ontology and it is known in the Description Logics community as *incoherence*[Flouris *et al.*, 2006].

Relevant atoms

- Before formalizing the notion of *incoherence* we need to identify the set of atoms relevant to a given set of TGDs.
- Intuitively, a set of atoms A is relevant to a set T of TGDs iff it holds that A triggers the application of every TGD in T.

Definition (Relevant Set of Atoms for a Set of TGDs)

Let $\mathcal{R}$ be a relational schema, T be a set of TGDs, and A a non-empty set of ground atoms, both over $\mathcal{R}$. We say that A is *relevant* to T iff for all $\sigma \in T$ of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$ it holds that $\mathit{chase}(A, T) \models \exists \mathbf{X} \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$.

Relevant atoms

Example (Relevant Set of Atoms)

Consider the following constraints:

$$\Sigma_T = \{\sigma_1 : \mathit{supervises}(X, Y) \rightarrow \mathit{supervisor}(X),$$
$$\sigma_2 : \mathit{supervisor}(X) \wedge \mathit{take\_decisions}(X) \rightarrow \mathit{leads\_department}(X, D),$$
$$\sigma_3 : \mathit{employee}(X) \rightarrow \mathit{works\_in}(X, D)\}$$

The set
$A_1 = \{\mathit{supervises}(\mathit{walter}, \mathit{jesse}), \mathit{take\_decisions}(\mathit{walter}), \mathit{employee}(\mathit{jesse})\}$ is relevant to Σ_T, since σ_1 and σ_3 are directly applicable to A_1 and σ_2 becomes applicable when we apply σ_1.
However, the set $A_2 = \{\mathit{supervises}(\mathit{walter}, \mathit{jesse}), \mathit{take\_decisions}(\mathit{gus})\}$ is not relevant to Σ_T. Note that even though σ_1 is applicable to A_2, the TGDs σ_2 and σ_3 are never applied in $\mathit{chase}(A_2, \Sigma_T)$, since the atoms in their bodies are never generated in $\mathit{chase}(A_2, \Sigma_T)$.

Satisfiability

- Our conception of (in)coherence is based on the notion of satisfiability of a set of TGDs *w.r.t.* a set of constraints.

Definition

(Satisfiability of a set of TGDs) Let $T \subseteq \Sigma_T$ be a set of TGDs, and $N \subseteq \Sigma_{NC} \cup \Sigma_E$. The set T is *satisfiable w.r.t.* N iff there is a set A of atoms such that A is relevant to T and $\mathit{mods}(A, T \cup N) \neq \emptyset$. We say that T is *unsatisfiable w.r.t.* N iff T is not satisfiable *w.r.t.* N.

- Intuitively, a set of dependencies is satisfiable when there is a relevant set of atoms that does not produce the violation of any constraint in $\Sigma_{NC} \cup \Sigma_E$, *i.e.*, the TGDs can be satisfied along with the NCs and EGDs in KB.

Satisfiability

Example (Satisfiable sets of dependencies)

$\Sigma^1_{NC} = \{\tau : risky\_job(P) \wedge unstable(P) \rightarrow \bot\}$
$\Sigma^1_{T} = \{\sigma_1 : dangerous\_work(W) \wedge works\_in(W, P) \rightarrow risky\_job(P),$
$\sigma_2 : in\_therapy(P) \rightarrow unstable(P)\}$

The set Σ^1_T is a satisfiable set of TGDs, for instance consider the set

$D_1 = \{dangerous\_work(police), works\_in(police, marty), in\_therapy(rust)\}.$

D_1 is a relevant set for Σ^1_T, however, as we have that no constraint is violated when we apply Σ^1_T to D_1 then Σ^1_T is satisfiable.

Satisfiability

Example (Unsatisfiable sets of dependencies)

$\Sigma^2_{NC} = \{\tau_1 : \mathit{sore\_throat}(X) \wedge \mathit{can\_sing}(X) \rightarrow \bot\}$

$$\begin{aligned}\Sigma^2_{T} = \{&\sigma_1 : \mathit{rock\_singer}(X) \rightarrow \mathit{sing\_loud}(X),\\ &\sigma_2 : \mathit{sing\_loud}(X) \rightarrow \mathit{sore\_throat}(X),\\ &\sigma_3 : \mathit{rock\_singer}(X) \rightarrow \mathit{can\_sing}(X)\}\end{aligned}$$

The set Σ^2_T is an unsatisfiable set of dependencies, as the application of TGDs $\{\sigma_1, \sigma_2, \sigma_3\}$ on any relevant set of atoms will cause the violation of τ_1.
For instance, consider the relevant atom $\mathit{rock\_singer}(axl)$: we have that $\mathit{mods}(\{\mathit{rock\_singer}(axl)\}, \Sigma^2_T \cup \Sigma^2_{NC} \cup \Sigma^2_E) = \emptyset$, since τ_1 is violated. Note that *any* set of relevant atoms will cause the violation of τ_1.

Coherence in Datalog$^{\pm}$

Based on satisfiability we define coherence for a Datalog$^{\pm}$ ontology. Intuitively, an ontology is coherent if there is no subset of their TGDs that is unsatisfiable *w.r.t.* the constraints in the ontology.

Definition (Coherence)

Let $KB = (D, \Sigma)$ be a Datalog$^{\pm}$ ontology. Then, KB is *coherent* iff Σ_T is satisfiable *w.r.t.* $\Sigma_{NC} \cup \Sigma_E$, and incoherent otherwise.

Example (Coherence)

Consider the sets of dependencies and constraints defined in the previous example and an arbitrary database instance D. Clearly, the Datalog$^{\pm}$ ontology $KB_1 = (D, \Sigma_T^1 \cup \Sigma_{NC}^1 \cup \Sigma_E^1)$ is coherent, while $KB_2 = (D, \Sigma_T^2 \cup \Sigma_{NC}^2 \cup \Sigma_E^2)$ is incoherent.

Incoherence and classic inconsistency-tolerant semantics

- Classic inconsistency-tolerant techniques do not account for coherence issues since they assume that such kind of problems will not appear.
- Nevertheless, if we consider that both components in the ontology evolve then certainly incoherence is prone to arise.
- Moreover, note that an incoherent KB will induce an inconsistent KB when the database instance contains any set of atoms that is relevant to the unsatisfiable sets of TGDs.
- Then, it may be important for inconsistency-tolerant techniques to consider incoherence as well, since as we will show if not treated appropriately an incoherent set of TGDs may produce meaningless answers for relevant atoms in D (in the worst case, it could produce an empty set of answers).

Repairs and inconsistency-tolerant semantics

- A basic notion in classic inconsistency-tolerant semantics is that off *repair*, which is a model of the set of integrity constraints that is maximally close, *i.e., "as close as possible"* to the original database.
- Depending on how repairs are obtained we can have different semantics.
- For instance, in *AR* semantics [Flouris *et al.*, 2010]an atom a is entailed from a Datalog$^\pm$ ontology KB, denoted $KB \models_{AR} a$, iff a is classically entailed from every ontology that can be built from every possible repair (a maximally consistent subset of the D component that after its application to Σ_T respects every constraint in $\Sigma_E \cup \Sigma_{NC}$).

Repairs and incoherence

- Incoherence has a great influence when calculating repairs, as can be seen in the following result: independently of the semantics (*i.e.*, AR or variants like CAR) no atom that is relevant to an unsatisfiable set of TGDs belongs to a repair of an incoherent KB.

Lemma

Let $KB = (D, \Sigma)$ be an incoherent Datalog$^{\pm}$ ontology where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$ and $\mathcal{R}(KB)$ be the set of (A-Box or Closed A-Box) repairs of KB. If $A \subseteq D$ is relevant to some unsatisfiable set $U \in \mathcal{U}(KB)$ then $A \not\subseteq R$ for every $R \in \mathcal{R}(KB)$.

Repairs and incoherence

Example

Consider the atom $rock\_singer(axl)$ and the set $U \subset \Sigma_T = \{\sigma_1 : rock\_singer(X) \rightarrow sing\_loud(X), \sigma_2 : sing\_loud(X) \rightarrow sore\_throat(X), \sigma_4 : rock\_singer(X) \rightarrow can\_sing(X)\}$.

It is easy to show that this atom does not belong to any repair. Consider the A-Box repairs adapted to Datalog$^{\pm}$ (maximally *consistent* subsets of the component D). We have that $mods(rock\_singer(axl), \Sigma) = \emptyset$, as the NC $\tau_1 : sore\_throat(X) \wedge can\_sing(X) \rightarrow \perp$ is violated.

Moreover, clearly this violation happens for every set $A \subseteq D$ such that $rock\_singer(axl) \in A$, and thus we have that $mods(A, \Sigma) = \emptyset$, *i.e.*, $rock\_singer(axl)$ cannot be part of any A-Box repair for the KB.
We can show an analogous example for CAR-semantics.

Incoherence and answers in AR/CAR

- Then, every atom that is relevant to an unsatisfiable set of TGDs cannot be *AR*-consistently (resp, *CAR*-consistently) entailed.

Proposition

If $A \subseteq D$ *is relevant to some unsatisfiable set* $U \subseteq \Sigma_T$ *then* $KB \nvDash_{AR} A$ *and* $KB \nvDash_{CAR} A$*.*

- In the limit case that every atom in the database instance is relevant to some unsatisfiable subset of the TGDs in the ontology then the set of *AR*-answers, denoted $\mathcal{A}_{AR}$, (resp, *CAR*-answers - $\mathcal{A}_{CAR}$) is empty.
- Both results can be straightforwardly extended to other repair based inconsistency-tolerant semantics such as ICAR and ICR [Lembo *et al.*, 2010].

Incoherence-tolerant semantics

- Since they were not develop to consider such kind of issues, incoherence greatly affects classic inconsistency-tolerant semantics.
- Notice that in our example $rock\_singer(axl)$ should be an answer; we do not know whether or not Axl can sing or has a sore throat, but we can at least agree that he is a rock singer.
- Nevertheless, such atom is not part of the answers of repair-based semantics such as AR or CAR.

Incoherence-tolerant semantics

- Intuitively, we say that a query answering semantics is tolerant to incoherence if it is possible for it to entail atoms that trigger incoherent sets of TGDs as answers.

Definition (Incoherence-tolerant semantics)

Let $KB = (D, \Sigma)$ be a Datalog$^{\pm}$ ontology where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$. A query answering semantics S is said to be *tolerant to incoherence* (or incoherency-tolerant) iff there exists $A \subseteq D$ and $U \in \mathcal{U}(KB)$ such that A is relevant to U and it holds that $KB \models_S A$.

- *AR* and *CAR* semantics are not incoherence-tolerant semantics.

Defeasible Datalog$^{\pm}$

- Defeasible Datalog$^{\pm}$ is a variation of Datalog$^{\pm}$ that enables argumentative reasoning in Datalog$^{\pm}$.
- To do this, a Datalog$^{\pm}$ ontology is extended with a set of *defeasible atoms* and *defeasible TGDs*; thus, a Defeasible Datalog$^{\pm}$ ontology contains both (classical) strict knowledge and defeasible knowledge.

- **Defeasible Datalog$^{\pm}$ Ontologies.** A *defeasible* Datalog$^{\pm}$ *ontology* KB consists of a finite set F of *ground atoms*, called *facts*, a finite 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 constraints $\Sigma_E \cup \Sigma_{NC}$.

Defeasible Datalog$^{\pm}$ ontologies

Example

The information in our running example can be better represented with the defeasible ontology $KB = (F, D, \Sigma'_T, \Sigma_D, \Sigma_{NC})$, where $F = \{can\_sing(simone), sing\_loud(ronnie), has\_fans(ronnie)\}$ and $D = \{rock\_singer(axl), manage(band_1, richard)\}$. For instance, we change the fact stating that *richard* manages $band_1$ to a defeasible one, since reports indicates that $band_1$ is looking for a new manager.
Also, we change some of the TGDs into defeasible TGDs to make clear that the connection between the head and body is weaker.

$$\Sigma_{T'} = \{sing\_loud(X) \rightarrow sore\_throat(X), rock\_singer(X) \rightarrow can\_sing(X)$$
$$\Sigma_D = \{rock\_singer(X) \succ sing\_loud(X), has\_fans(X) \succ famous(X)\}$$

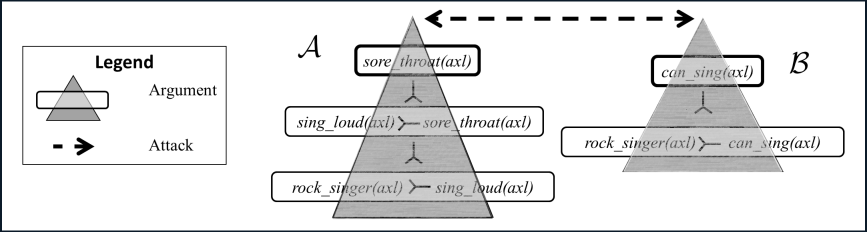
Conflicts in Defeasible Datalog$^{\pm}$

- Based on the information encoded in a defeasible Datalog$^{\pm}$ ontology, conflicting information can be derived.
- Conflicts in defeasible Datalog$^{\pm}$ ontologies come, as in classical Datalog$^{\pm}$, from the violation of NCs or EGDs.
- Intuitively, two atoms are in conflict whenever they can both be derived from the ontology and together map to the body of a NC or they violate an EGD.
- Conflicts in classical argumentation are inherently binary, since they are based on contrariness, *i.e.*, a contrary to b and b contrary to a means that they are in conflict. Here, we restrict NCs and EGDs to binary ones to mirror such kind of conflicts.

Arguments in Defeasible Datalog$^{\pm}$

- When conflicts arise we use a dialectical process to decide which piece of information is such that no acceptable reasons 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.
- It is possible to build arguments for conflicting atoms, and so arguments can *attack* each other.

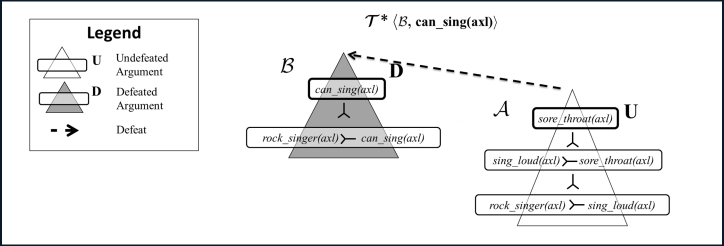
Example


Legend
Argument
Attack
$\mathcal{A}$
sore_throat(axl)
sing_loud(axl) ≻— sore_throat(axl)
rock_singer(axl) ≻— sing_loud(axl)
$\mathcal{B}$
can_sing(axl)
rock_singer(axl) ≻— can_sing(axl)

Warranting and answers

- The combination of arguments, attacks and a comparison criterion $\succ$ (used to establish whether and argument defeats another one in conflict with it) gives raise to Datalog$^\pm$ argumentation frameworks, denoted $\mathfrak{F}$.
- An atom is *warranted* in $\mathfrak{F}$ iff there exists an undefeated argument in favor of the atom.

Example



Legend
U Undefeated Argument
D Defeated Argument
Defeat
$\mathcal{T}^* \langle \mathcal{B}, \textbf{can\_sing(axl)} \rangle$
$\mathcal{B}$
D
can_sing(axl)
rock_singer(axl) ≻— can_sing(axl)
$\mathcal{A}$
U
sore_throat(axl)
sing_loud(axl) ≻— sore_throat(axl)
rock_singer(axl) ≻— sing_loud(axl)

Warranting and answers

- We define a semantics, denoted as $\mathbf{D}^2$ (**D**efeasible **D**atalog$^{\pm}$), based on the use of argumentative inference.
- Such semantics relies on the transformation of classic Datalog$^{\pm}$ ontologies to defeasible ones and then obtaining answers from the transformed one by means of an argumentation-based process.
- Intuitively, the transformation of a classic ontology to a defeasible one involves transforming every atom and every TGD in the classic ontology to its defeasible version.
- Finally, a literal is an answer for a classical Datalog$^{\pm}$ ontology KB under the $\mathbf{D}^2$ semantics iff it is warranted in the transformation of KB to a defeasible one.

Influence of incoherence in Defeasible Datalog$^{\pm}$

- We can show that one relevant atom L to an unsatisfiable set is warranted (and thus an answer), provided that the comparison criterion $\succ$ is such that it warrants some argument in its favor.

Proposition

Let KB be a Datalog$^{\pm}$ ontology defined over a relational schema $\mathcal{R}$, and KB' be a Defeasible Datalog$^{\pm}$ ontology such that $\mathcal{D}(KB) = KB'$. Finally, let $L \in D$ and $U \in \mathcal{U}(KB)$ such that L is relevant to U. Then, it holds that there exists $\succ$ such that $KB \vDash_{\mathbf{D}^2_{\succ}} L$.

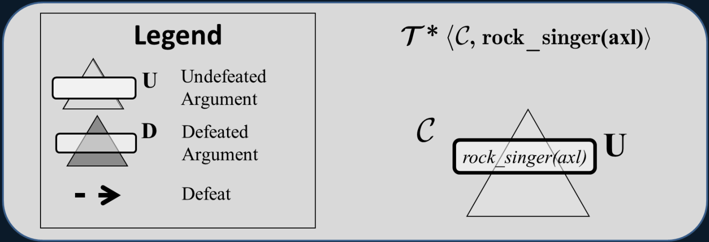
- *Such comparison criterion can always be found.*

Corollary

Given a Datalog$^{\pm}$ ontology KB there exists $\succ$ such that $\mathbf{D}^2_{\succ}$ applied to KB is tolerant to incoherence.

Influence of incoherence in Defeasible Datalog$^{\pm}$

Example



Then, clearly $KB' \models_{\mathfrak{F}} rock\_singer(axl)$, and thus $KB \vDash_{\mathbf{D}^2_{\succ}} rock\_singer(axl)$.
Note that the atom $rock\_singer(axl)$ is warranted under **any** criterion comparison $\succ$, and thus we have not needed to perform any restriction on the criterion.

Conclusions

- Incoherence is an important problem in knowledge representation and reasoning, but most of the works in query answering for Datalog$^{\pm}$ ontologies and DLs either completely ignore the possibility of conflicts or have focused on consistency issues, assuming that no conflict arise in the constraints.
- We have introduced the concept of incoherence for Datalog$^{\pm}$ ontologies, relating it to the presence of sets of TGDs such that their application inevitably yield the violation in the set of negative constraints and equality-generating dependencies.
- We have shown how incoherence affects classic inconsistency-tolerant semantics to the point that for some incoherent ontologies these semantics may produce no useful answer.
- Finally, we have introduced the concept of incoherency-tolerant semantics, and shown a particular semantics satisfying that property.

References I

Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache.
Inconsistencies, negations and changes in ontologies.
In *AAAI*, pages 1295–1300. AAAI Press, 2006.

D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo.
Inconsistency-tolerant semantics for description logics.
In *Proc. of RR*, pages 103–117, 2010.

Maria Vanina Martinez, Cristhian Ariel David Deagustini, Marcelo A. Falappa, and Guillermo Ricardo Simari.
Inconsistency-tolerant reasoning in datalog$\pm$ ontologies via an argumentative semantics.
In *proc. of IBERAMIA 2014*, pages 15–27, 2014.

The end

- Comments? Questions?

The end

- Comments? Questions?
- Thank you!