Journal of Artificial Intelligence Research 56 (2016) 613-656 Submitted 03/16; published 08/16

Datalog$^{\pm}$ Ontology Consolidation

Cristhian Ariel D. Deagustini CADD@CS.UNS.EDU.AR
María Vanina Martínez MVM@CS.UNS.EDU.AR
Marcelo A. Falappa MFALAPPA@CS.UNS.EDU.AR
Guillermo R. Simari GRS@CS.UNS.EDU.AR
AI R&D Lab., Institute for Computer Science and Engineering (ICIC)
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Universidad Nacional del Sur (UNS), Alem 1253,
(B8000CPB) Bahía Blanca, Argentina.

Abstract

Knowledge bases in the form of ontologies are receiving increasing attention as they allow to clearly represent both the available knowledge, which includes the knowledge in itself and the constraints imposed to it by the domain or the users. In particular, Datalog$^{\pm}$ ontologies are attractive because of their property of decidability and the possibility of dealing with the massive amounts of data in real world environments; however, as it is the case with many other ontological languages, their application in collaborative environments often lead to inconsistency related issues. In this paper we introduce the notion of incoherence regarding Datalog$^{\pm}$ ontologies, in terms of satisfiability of sets of constraints, and show how under specific conditions incoherence leads to inconsistent Datalog$^{\pm}$ ontologies. The main contribution of this work is a novel approach to restore both consistency and coherence in Datalog$^{\pm}$ ontologies. The proposed approach is based on kernel contraction and restoration is performed by the application of incision functions that select formulas to delete. Nevertheless, instead of working over minimal incoherent/inconsistent sets encountered in the ontologies, our operators produce incisions over non-minimal structures called clusters. We present a construction for consolidation operators, along with the properties expected to be satisfied by them. Finally, we establish the relation between the construction and the properties by means of a representation theorem. Although this proposal is presented for Datalog$^{\pm}$ ontologies consolidation, these operators can be applied to other types of ontological languages, such as Description Logics, making them apt to be used in collaborative environments like the Semantic Web.

1. Introduction

The integration of different systems, and the interaction resulting from this integration, led to a host of pervasive practical problems and challenging research opportunities; some of the most interesting ones occurs in the Web's collaborative environments, e.g., e-commerce, and with the arrival of the Semantic Web, such as ontology engineering. However, the collaboration among systems brings along the problem of conflicting pieces of information that are likely to appear as knowledge repositories evolve. Admittedly, the management of conflicting information is an important and challenging issue that has to be faced (Gómez, Chesñevar, & Simari, 2010; Haase, van Harmelen, Huang, Stuckenschmidt, & Sure, 2005; Huang, van Harmelen, & ten Teije, 2005; Bell, Qi, & Liu, 2007), specially when integrating

knowledge coming from different sources (Black, Hunter, & Pan, 2009; Baral, Kraus, & Minker, 1991; Amgoud & Kaci, 2005), or when such knowledge is expected to be exploited by a reasoning process. In this context, knowledge bases in the form of ontologies are becoming a useful device that provide a convenient way to represent both the intensional and extensional knowledge of the application domain. Moreover, the expressive power of ontologies allows to perform important tasks on data integration (Lenzerini, 2002), and also plays a role of great importance in the aforementioned Semantic Web (Berners-Lee, Hendler, & Lassila, 2001). In this work we adopt Datalog$^{\pm}$ ontologies, a family of rule-based ontology languages (Calì, Gottlob, & Lukasiewicz, 2012). Datalog$^{\pm}$ enables a modular rule-based style of knowledge representation, and it can represent syntactical fragments of first-order logic (FOL) so that answering a Boolean Conjunctive Query (BCQs) Q under a set Σ of Datalog$^{\pm}$ rules for an input database D is equivalent to the classical entailment check $D \cup \Sigma \models Q$. Tractable fragments of Datalog$^{\pm}$ guarantee termination of query answering procedures in polynomial time in the data complexity and first-order rewritability. Moreover, ontologies described using existential rules generalize several well-known Description Logics (DLs); in particular, linear and guarded Datalog$^{\pm}$ (two basic tractable fragments of this family) are strictly more expressive than the whole DL-Lite family (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2005), and guarded Datalog$^{\pm}$ is strictly more expressive than $\mathcal{EL}$ (Brandt, 2004; Baader, Brandt, & Lutz, 2005). Therefore, the results presented in this paper extend directly to these DLs as well. These properties of Datalog$^{\pm}$ together with its expressive power, and the fact that it keeps a syntax closer to that used in relational databases for greater readability, make it very useful in modeling real applications, such as ontology querying, Web data extraction, data exchange, ontology-based data access, and data integration.

We focus on two particular problems that arise from the integration and/or evolution of information systems: inconsistency and incoherence. Inconsistency refers to the lack of models of a theory. On the other hand, in ontological settings, incoherence refers to a set of ontological rules that cannot be applied without leading to violations of the constraints imposed on the knowledge, making them unsatisfiable. Incoherence and inconsistency, which can arise from automated procedures such as data integration and ontology matching, may be serious issues in real world applications. Since standard ontology languages adhere to the classical FOL semantics, classical inference semantics fails in the presence of this kind of problems. Thus, it is important to focus on the formalization of methods to address both inconsistency and incoherence in ontologies that are able to cope with the users' expectations in terms of effectiveness of the procedures for query answering and the meaning of these answers when potential conflict exists.

This paper addresses the problem of handling inconsistencies and incoherences that may appear in Datalog$^{\pm}$ ontologies. In this regard, we propose a general framework that aims at the consolidation of Datalog$^{\pm}$ ontologies (i.e., solving every conflict of coherence and consistency in them). That is, a consolidation operator takes as input a (possibly) incoherent and inconsistent Datalog$^{\pm}$ ontology and returns another Datalog$^{\pm}$ ontology where all conflicts are amended, thus ensuring that it is both coherent and consistent. As it is usual in this setting, an assumption of minimal change is made, that is to say, it is expected that the consolidation process changes the original ontology as little as possible. The approach presented is based on the use of incision functions (Hansson, 1993, 1994, 1997,

2001) from the Belief Revision literature. Instead of operators that only account for the information included in the conflicts in a knowledge base, in this work we aim to capture consolidation operators that can consider all the information included in a KB when solving conflicts. The main contributions of this work are the following:

- We introduce a notion of incoherence tailored for Datalog$^{\pm}$. To achieve this we adapt to this setting similar notions from Description Logics. Also, we look into the relationship of incoherence and inconsistency and how it impacts the consolidation process.
- We provide a set of properties expected to be satisfied by consolidation operators for Datalog$^{\pm}$ ontologies by means of postulates. These postulates provide a formal characterization of a consolidation operator without focusing on how the consolidation process is actually performed, thus providing a formal comparison framework for consolidation operators. The postulates consider some intuitions in classic Belief Revision; nevertheless, they are adapted to the Datalog$^{\pm}$ ontological setting (and could be also adapted to suit other ontological languages), meaning that they have two versions (one addressing incoherence and another one for inconsistency).
- We present a complete construction of consolidation operators that take a (possibly) incoherent and inconsistent Datalog$^{\pm}$ ontology and gives as a result a consistent and coherent one. A noteworthy characteristic of such operators is that it involves a two steps approach, first considering incoherence conflicts, and then solving inconsistency conflicts as a latter step, helping to improve the final result in terms of the information that needs to be deleted to solve conflicts.
- We study the relationship between the formal properties of the operator and the construction we propose, demonstrating that they are equivalent; thus, this shows that any consolidation operator satisfying the properties corresponds to the construction introduced in this work.

The paper is organized as follows: in Section 2 we introduce the necessary notions from Datalog$^{\pm}$ and Belief Revision. Next, though inconsistency and incoherence are related, they are also two very distinct problems in the setting of ontological knowledge bases in particular, where there is a clear separation of the intensional and the extensional knowledge. Therefore, in Section 3, we discuss the two notions in Datalog$^{\pm}$ ontologies, how they relate to each other, and the reasons why they need to be treated in combination but separately. Then, in Section 4 we present the properties that an ontology consolidation operator must satisfy, and in Section 5 we introduce the process used to restore consistency and coherence of Datalog$^{\pm}$ ontologies, and relate the presented process to the given properties by means of a representation theorem. Next, we present a complete example depicting the entire consolidation process. Finally, in Sections 7 and 8 we discuss related work from different areas in Artificial Intelligence and Database Theory, and provide conclusions and future lines of research, respectively.

2. Preliminaries and Background

To facilitate the reading, we begin by introducing the notions from Datalog$^{\pm}$ and Belief Revision that will be needed in the rest of the paper.

2.1 Preliminaries on Datalog$^{\pm}$

First, we recall the basic notions of Datalog$^{\pm}$ ontologies that will be used in the paper (see Calì et al., 2012 for more details). Datalog$^{\pm}$ extends Datalog by allowing existential quantification in the rule heads, together with other extensions that we enumerate below, but limiting the interaction of these elements in order to achieve tractability.

We will assume that the domain of discourse of a Datalog$^{\pm}$ ontology consists of a countable set of *data constants* Δ, a countable set of nulls Δ_N (as place holders for unknown values), and a countable set of variables $\mathcal{V}$. We also assume that different constants represent different values (unique names assumption). To distinguish constants from variables, we adopt the standard notation from logic programming, where variable names begin with uppercase letters, while constants and predicate symbols begin with lowercase letters.

We assume a *relational schema* $\mathcal{R}$ that is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, a null, or a variable. An *atom* 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; an atom is ground iff all terms in it are constants. Let $\mathcal{L}$ be a first-order language such that $\mathcal{R} \subset \mathcal{L}$; then $\mathcal{L}_{\mathcal{R}}$ denotes the sublanguage generated by $\mathcal{R}$. A *database (instance)* of $\mathcal{R}$ is a finite set of atoms with predicates in $\mathcal{R}$ and terms in $\Delta \cup \Delta_N$. A *homomorphism* on constants, nulls and variables is a mapping $h : \Delta \cup \Delta_N \cup \mathcal{V} \longrightarrow \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $h(c) = c$, (ii) $c \in \Delta_N$ implies $h(c) \in \Delta \cup \Delta_N$, and (iii) h is naturally extended to atoms, sets of atoms, and conjunctions of atoms.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$ where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ called the *body* (denoted *body*(σ)) and the *head* (denoted *head*(σ)), respectively. Consider a database D for a relational schema $\mathcal{R}$, and a TGD σ on $\mathcal{R}$ of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms in D. Let σ be applicable to D, and h' be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h'(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h'(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* σ on D adds to D the atom $h'(\Psi(\mathbf{X}, \mathbf{Z}))$ if it is not already in D. After the application we say that σ is satisfied by D. The *Chase* for a database D and a set of TGDs $\Sigma_{_T}$, denoted *chase*$(D, \Sigma_{_T})$, is the exhaustive application of the TGDs (Calì et al., 2012) in a breadth-first (level-saturating) fashion, which leads to a (possibly infinite) chase for D and Σ. It is important to remark that BCQs Q over D and $\Sigma_{_T}$ can be evaluated on the chase for D and $\Sigma_{_T}$, i.e., $D \cup \Sigma_{_T} \models Q$ is equivalent to *chase*$(D, \Sigma_{_T}) \models Q$ (Calì et al., 2012).

Negative constraints (NCs) are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (without nulls) and the head is the truth constant *false*, denoted $\bot$. An NC τ is satisfied by a database D under a set of TGDs $\Sigma_{_T}$ iff there does not exist a homomorphism h that maps the atoms of $\Phi(\mathbf{X})$ to D, where D is such that every TGD in $\Sigma_{_T}$ is satisfied, i.e., the atoms in the body cannot all be true together.

Equality-generating dependencies (EGDs) are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$ is a conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. An EGD σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. In this work we

will focus on a particular class of EGDs, called *separable* (Calì et al., 2012); intuitively, separability of EGDs *w.r.t.* a set of TGDs states that, if an EGD is violated, then atoms contained in D are the reason of the violation (and not the application of TGDs); i.e., if an EGD in Σ_{E} is violated when we apply the TGDs in Σ_{T} for a database D, then the EGD is also violated in D. Separability is a standard assumption in Datalog$^{\pm}$ ontology, as one of the most important features of this family of languages is the focus on decidable (Calì, Lembo, & Rosati, 2003) (actually tractable) fragments of Datalog$^{\pm}$.

NCs and EGDs play an important role in the matter of conflicts in Datalog$^{\pm}$ ontologies. In fact, the approach that we present in this work ensure that neither NCs nor EGDs are violated in the resulting ontology. Also, as an important remark, note that the restriction of using only separable EGDs makes that certain cases of conflicts are not considered in our proposal. The treatment of such cases, though interesting from a technical point of view, are outside the scope of this work since we focus on tractable fragments of Datalog$^{\pm}$.

As is the usual case in the literature, in general the universal quantifiers in TGDs, negative constraints and EGDs are omitted, and the sets of dependencies and constraints are assumed to be finite. Now that we have presented the different ways of expressing knowledge in Datalog$^{\pm}$, we are ready to formally define Datalog$^{\pm}$ ontologies.

Definition 1 (Datalog$^{\pm}$ Ontology) *A Datalog$^{\pm}$ ontology $KB = (D, \Sigma)$, where $\Sigma = \Sigma_{T} \cup \Sigma_{E} \cup \Sigma_{NC}$, consists of a database instance D that is a finite set of ground atoms (without nulls), a set of TGDs Σ_{T}, a set of separable EGDs Σ_{E} and a set of NCs Σ_{NC}.*

Otherwise explicitly said, through the paper when it is clear from context we will refer to the component Σ in KB as the set of constraints in the ontology, without distinguishing between dependencies and constraints. Given a database D for $\mathcal{R}$ and a set of constraints $\Sigma = \Sigma_{T} \cup \Sigma_{E} \cup \Sigma_{NC}$, the set of *models* of D and Σ, denoted $\text{mods}(D, \Sigma)$, is the set of all databases B such that $D \subseteq B$ and every formula in Σ is satisfied. The following example shows a simple Datalog$^{\pm}$ ontology; the ontology describes knowledge about the therapy/psychology domain.

Example 1 (Datalog$^{\pm}$ Ontology)

$$KB = \left\{ \begin{array}{rl} \text{D:} & \{a_1 : \mathit{in\_therapy}(\mathit{charlie}), a_2 : \mathit{dating}(\mathit{kate}, \mathit{charlie}), \\ & \ a_3 : \mathit{therapist}(\mathit{kate}), a_4 : \mathit{belongs\_to}(g_1, \mathit{charlie}), \\ & \ a_5 : \mathit{in\_therapy}(\mathit{patrick}), a_6 : \mathit{belongs\_to}(g_2, \mathit{ed}), \\ & \ a_7 : \mathit{belongs\_to}(g_1, \mathit{kate})\} \\ \Sigma_{NC} : & \{\tau_1 : \mathit{treating}(T, P) \wedge \mathit{dating}(T, P) \rightarrow \bot\} \\ \Sigma_{E} : & \{\nu_1 : \mathit{treating}(T, P) \wedge \mathit{treating}(T', P) \rightarrow T = T'\} \\ \Sigma_{T} : & \{\sigma_1 : \mathit{in\_therapy}(P) \rightarrow \mathit{patient}(P), \\ & \ \sigma_2 : \mathit{therapist}(T) \wedge \mathit{belongs\_to}(G, T) \rightarrow \mathit{leads}(T, G), \\ & \ \sigma_3 : \mathit{leads}(T, G) \wedge \mathit{belongs\_to}(G, P) \rightarrow \mathit{treating}(T, P), \\ & \ \sigma_4 : \mathit{treating}(T, P) \rightarrow \mathit{therapist}(T)\} \end{array} \right\}$$

The set Σ_T of TGDs expresses dependencies such as: TGD σ_1 states that if a person P is in therapy then P is a patient, σ_2 establishes that a therapist T that belongs to a group

G is the leader of that group. The only NC τ_1 states that a patient cannot be dating his therapist, and EGD ν_1 states that every patient is in treatment with at most one therapist.

Following the classical notion of consistency, we say that a consistent Datalog$^{\pm}$ ontology has a non-empty set of models.

Definition 2 (Consistency) *A Datalog$^{\pm}$ ontology $KB = (D, \Sigma)$ is* consistent *iff* $\text{mods}(D, \Sigma) \neq \emptyset$. *We say that KB is inconsistent otherwise.*

Example 2 *Consider the Datalog$^{\pm}$ ontology from the example above; this ontology is clearly inconsistent. Database instance D is clearly not a model in itself since at least TGD σ_2 is applicable to D, but there is no superset of D such that it satisfies all TGDs and constraints in Σ at the same time. For instance TGDs σ_2 is applicable in D creating the atom $leads(kate, g_1)$ making now σ_3 applicable and resulting in the new atom $treating(kate, charlie)$, which together with $dating(kate, charlie)$ (that was already in D) violate the NC τ_1, as a therapist is dating one of her patients.*

For the rest of the paper, otherwise explicitly stated $KB = (D, \Sigma)$ will denote a Datalog$^{\pm}$ ontology with $\Sigma = \Sigma_{_T} \cup \Sigma_{_E} \cup \Sigma_{_{NC}}$, where D is a database instance, $\Sigma_{_T}$ is the set of all TGDs, $\Sigma_{_E}$ the set of all separable EGDs and $\Sigma_{_{NC}}$ being the set of all NCs in Σ.

2.2 Background in Belief Revision

Establishing the origins of scientific ideas is a difficult task that sometimes can be controversial; nevertheless, it could be argued that the origins of belief change theory go back to the work of Isaac Levi (1977), who discussed the problems concerning this field of research, and to William Harper's proposal of a rational way to interrelate belief change operators (Harper, 1975). However, the main advances on belief change theory came during the 1980's when Carlos Alchourrón and David Makinson studied changes in legal codes (Alchourrón & Makinson, 1981), and Peter Gärdenfors's introduced rational postulates for change operators (Gärdenfors, 1982). After that, the three authors produced a foundational paper containing what became known as the AGM model (Alchourrón, Gärdenfors, & Makinson, 1985). The core contribution of the AGM model is the presentation of a new and more general formal framework for the study of belief change; today, this work is considered as the cornerstone from which belief change theory evolved.

Since the introduction of the AGM model, different frameworks for belief dynamics and their respective epistemic models have been proposed. The epistemic model corresponds to the formalism in which beliefs are represented, providing the framework in which different kinds of operators can be defined. The AGM model is conceived as an idealistic theory of rational change in which epistemic states are represented by belief sets (sets of sentences closed under logical consequence, commonly denoted in boldface), and the epistemic input is represented by a sentence. In the AGM model, three basic change operators are defined: *expansion*, *contraction*, and *revision*. In the rest of this section, whenever we use the term consistent or inconsistent, we refer to the traditional notion of inconsistency of a knowledge base that has no models. Let $\mathbf{K}$ be a belief set, the change operations are as follows:

- **Expansions**: the result of expanding $\mathbf{K}$ by a sentence α is a possibly larger set which infers α; intuitively, belief α, hopefully consistent with the given epistemic state, is directly added to $\mathbf{K}$.

- **Contractions**: the result of contracting $\mathbf{K}$ by α is a possibly smaller set which does not infer α, unless α is a tautology;

- **Revisions**: the result of revising $\mathbf{K}$ by α is a set that neither extends nor is part of the set $\mathbf{K}$. In general, if α is not a fallacy then α is consistently inferred from the revision of $\mathbf{K}$ by α.

The great importance of AGM comes from providing axiomatic characterizations of contraction and revision in terms of rationality postulates. Such rationality postulates regard the operators as black boxes, characterizing *what* they do, but not explaining *how* they do it. In other words, their behavior is constrained with regard to inputs in basic cases, without describing the internal mechanisms used for achieving that behavior, so it is crucial to say that contraction and revision operators can also be obtained via more constructive approaches. AGM contractions can be realized by *partial meet contractions*, which are based on a selection among (maximal) subsets of $\mathbf{K}$ that do not imply α. Via the Levi's identity (Gärdenfors, 1988), associated revision operations called *partial meet revisions* are obtained. Another possible approach for contraction is based on a selection among the (minimal) subsets of $\mathbf{K}$ that contribute to make $\mathbf{K}$ imply α, as in *safe contraction* (Alchourrón & Makinson, 1985). A more general variant of the same approach, known as *kernel contraction*, was introduced later (Hansson, 1994). It has been shown that both safe contractions and kernel contractions are equivalent to partial meet contractions, and hence to the AGM approach to contraction (Hansson, 1994, 2001).

A particularly interesting characteristic of kernel contraction is that it may be concerned with changes at the *symbolic level* since it is suitable of being applied to *belief bases* (set of sentences *not* closed under a consequence relation) as well as belief sets. Thus, it matters how the beliefs are actually represented. This does not happen in the AGM approach, as it studies the changes at the *knowledge level* since it uses belief sets. The distinction between knowledge and symbolic level was proposed by Allen Newell (1982). According to Newell, the knowledge level lies above the symbolic level, and the latter is used to somehow represent the former. Because of this, belief bases with different symbolic content may represent the same knowledge. The importance of this is that, although they are statically equivalent (they represent the same beliefs), equivalent belief bases could be dynamically different if we choose to use an approach working directly with them, as with kernel contraction.

Besides the three basic operations mentioned, through the years additional operations where developed in Belief Revision to achieve different behaviors. For instance, when a belief base is inconsistent, the removal of enough sentences from it can lead to a consistent state. This additional operation is called *consolidation*, and the consolidation of a belief base K is denoted $K!$ (see Hansson, 1991, 2001). Here we will focus on this last operation, which is inherently different from contraction and revision, since its ultimate goal is to obtain a consistent belief base from a possibly inconsistent one (without being given any epistemic input), rather than revising the knowledge base by a specific formula or by removing a particular formula from it. The consolidation of K can be obtained in a natural way in belief

bases by contracting them by *falsum*, i.e., $K! = K \div \bot$, where $\div$ represents a contraction operator; this process restores consistency attending every conflict in K (Hansson, 1991).

3. Incoherence and Inconsistency Problems Related to Datalog$^{\pm}$ Ontology Consolidation

The problem of obtaining consistent knowledge from an inconsistent knowledge base is natural in many computer science fields. As knowledge evolves, contradictions are likely to appear, and these inconsistencies have to be handled in a way that they do not affect the quality of the information obtained from the database.

In the setting of Consistent Query Answering (CQA), repairing of relational databases, and inconsistency-tolerant query answering in ontological languages (Arenas, Bertossi, & Chomicki, 1999; Lembo, Lenzerini, Rosati, Ruzzi, & Savo, 2010; Lukasiewicz, Martinez, & Simari, 2012), often the assumption is made that the set Σ expresses the semantics of the data in the component D, and as such there is no internal conflict on the set of constraints and these constraints are not subject to changes over time. This means first, that the set Σ is always satisfiable, in the sense that their application do not *inevitably* yield a consistency problem. Second, as a result of this assumption, it must be the case that the conflicts come from the data contained in the database instance, and that is the part of the ontology that must be modified in order to restore consistency. Although this is a reasonable assumption to make, specially in the case of a single ontology, in this work we will focus on a more general setting, and consider that both data and constraints can change through time and become conflicting. In this more general scenario, as knowledge evolves (and so the ontology that represents it) not only data related issues can appear, but also constraint related ones.

We argue that it is also important to identify and separate the sources of conflicts in Datalog$^{\pm}$ ontologies. In the previous section we defined inconsistency of a Datalog$^{\pm}$ ontology based on the lack of models. From an operational point of view, conflicts appear in a Datalog$^{\pm}$ ontology whenever a NC or an EGD is violated, that is, whenever the body of one such constraint can be mapped to either atoms in D or atoms that can be obtained from D by the application of the TGDs in $\Sigma_T \subseteq \Sigma$. Beside these conflicts, we will also focus on the relationship between the set of TGDs and the set of NCs and EGDs, as it could happen that (a subset of) the TGDs in Σ_T cannot be applied without leading always to the violation of the NCs or EGDs. Note that in this case clearly the data in the database instance is not the problem, as any database in which these TGDs are applicable will inevitable produce an inconsistent ontology. This issue is related to the *unsatisfiability problem of a concept* in an ontology, and it is known in the Description Logics community as *incoherence* (Flouris, Huang, Pan, Plexousakis, & Wache, 2006; Schlobach & Cornet, 2003; Borgida, 1995; Beneventano & Bergamaschi, 1997; Kalyanpur, Parsia, Sirin, & Hendler, 2005; Schlobach, Huang, Cornet, & van Harmelen, 2007; Qi & Hunter, 2007). Incoherence can be particularly important when combining multiple ontologies since the constraints imposed by each one of them over the data could (possibly) represent conflicting models of the application at hand. Clearly, the notions of incoherence and inconsistency are highly related; in fact, Flouris et al.'s (2006) work establish a relation between incoherence and inconsistency, considering incoherence as a particular form of inconsistency.

Later in this section we present a complete definition of incoherence in Datalog$^{\pm}$, based on the concept of unsatisfiability of sets of TGDs. Nevertheless, for now it is sufficient to know that our proposed notion of incoherence states that given a set of unsatisfiable constraints Σ it is not possible to find a set of atoms D such that $KB = (D, \Sigma)$ is a consistent ontology and at the same time all TGDs in $\Sigma_T \subseteq \Sigma$ are applicable in D. This means that a Datalog$^{\pm}$ ontology can be consistent even if the set of constraints is incoherent, as long as the database instance does not make those dependencies applicable. On the other hand, a Datalog$^{\pm}$ ontology can be inconsistent even when the set of constraints is satisfiable, e.g., $KB = (\{tall(peter), small(peter)\}, \{tall(X) \wedge small(X) \rightarrow \bot\})$, where the (empty) set of dependencies is trivially satisfiable and thus the ontology coherent; the ontology is, nevertheless, inconsistent.

Before formalizing the notion of *incoherence* that we use in our Datalog$^{\pm}$ setting we need to identify the set of atoms relevant to a given set of TGDs. Intuitively, we say that a set of atoms A is relevant to a set T of TGDs if the atoms in the set A are such that the application of T over A generates the atoms that are needed to apply all TGDs in T, i.e., A triggers the application of every TGD in T.

Definition 3 (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 (possibly existentially closed) non-empty set of 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 $chase(A, T) \models \exists \mathbf{X} \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$.*

When it is clear from the context, if a singleton set $A = \{a\}$ is relevant to $T \subseteq \Sigma_T$ we just say that atom a is relevant to T.

Example 3 (Relevant Set of Atoms) Consider the following constraints:

$$\begin{aligned}\Sigma_T = \{&\sigma_1 : supervises(X, Y) \rightarrow supervisor(X),\\ &\sigma_2 : supervisor(X) \wedge makes\_decisions(X) \rightarrow leads\_department(X, D),\\ &\sigma_3 : employee(X) \rightarrow works\_in(X, D)\}\end{aligned}$$

Consider set $A_1 = \{supervises(walter, jesse), makes\_decisions(walter), employee(jesse)\}$. This set is a relevant set of atoms to the set of constraints $\Sigma_T = \{\sigma_1, \sigma_2, \sigma_3\}$, since σ_1 and σ_3 are directly applicable to A_1 and σ_2 becomes applicable when we apply σ_1 (i.e., the chase entails the atom $supervisor(walter)$, which together with $makes\_decisions(walter)$ triggers σ_2).

However, the set $A_2 = \{supervises(walter, jesse), makes\_decisions(gus)\}$ 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 $chase(A_2, \Sigma_T)$, since the atoms in their bodies are never generated in $chase(A_2, \Sigma_T)$. For instance, consider the TGD $\sigma_2 \in \Sigma_T$. In the chase of Σ_T over D we create the atom $supervisor(walter)$, but nevertheless we still cannot trigger σ_2 since we do not have and cannot generate the atom $makes\_decisions(walter)$, and the atom $makes\_decisions(gus)$ that is already in A_2 does not match the constant value.

We now present the notion of coherence for Datalog$^{\pm}$, which adapts efforts made for DLs such as Schlobach and Cornet's (2003) and Flouris et al.'s (2006). Our conception

of (in)coherence is based on the notion of satisfiability of a set of TGDs *w.r.t.* a set of constraints. Intuitively, a set of dependencies is satisfiable when there is a relevant set of atoms that triggers the application of all dependencies in the set and 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 the KB.

Definition 4 (Satisfiability of a Set of TGDs *w.r.t.* a Set of Constraints) *Let $\mathcal{R}$ be a relational schema, $T \subseteq \Sigma_T$ be a set of TGDs, and $N \subseteq \Sigma_{NC} \cup \Sigma_E$, both over $\mathcal{R}$. The set T is* satisfiable w.r.t. *N iff there is a set A of (possibly existentially closed) atoms over $\mathcal{R}$ such that A is relevant to T and* $\text{mods}(A, T \cup N) \neq \emptyset$. *We say that T is* unsatisfiable w.r.t. *N iff T is not satisfiable* w.r.t. *N. Furthermore, Σ_T is satisfiable* w.r.t. *$\Sigma_{NC} \cup \Sigma_E$ iff there is no $T \subseteq \Sigma_T$ such that T is unsatisfiable* w.r.t. *some N with $N \subseteq \Sigma_{NC} \cup \Sigma_E$.*

In the rest of the paper sometimes we write that a set of TGDs is (un)satisfiable omitting the set of constraints, we do this in the context of a particular ontology where we have a fixed set of constraints $\Sigma_{NC} \cup \Sigma_E$ since any set of TGDs that is satisfiable *w.r.t.* $\Sigma_{NC} \cup \Sigma_E$ is satisfiable *w.r.t.* any subset of it and, on the other hand, any set of TGDs that is unsatisfiable *w.r.t.* a subset of $\Sigma_{NC} \cup \Sigma_E$ is also unsatisfiable *w.r.t.* the whole set of constraints.

Example 4 (Unsatisfiable Sets of Dependencies) *Consider the following constraints.*

$$\Sigma^1_{NC} = \{\tau : \mathit{risky\_job}(P) \wedge \mathit{unstable}(P) \rightarrow \bot\}$$

$$\Sigma^1_T = \{\sigma_1 : \mathit{dangerous\_work}(W) \wedge \mathit{works\_in}(W, P) \rightarrow \mathit{risky\_job}(P),$$
$$\sigma_2 : \mathit{in\_therapy}(P) \rightarrow \mathit{unstable}(P)\}$$

The set Σ^1_T is a satisfiable set of TGDs, and even though the simultaneous application of σ_1 and σ_2 may violate some formula in $\Sigma^1_{NC} \cup \Sigma^1_E$, that does not hold for every relevant set of atoms. Consider as an example the relevant set $D_1 = \{\mathit{dangerous\_work}(\mathit{police}), \mathit{works\_in}(\mathit{police}, \mathit{marty}), \mathit{in\_therapy}(\mathit{rust})\}$; D_1 is a relevant set for Σ^1_T, however, as we have that $\text{mods}(D_1, \Sigma^1_T \cup \Sigma^1_{NC} \cup \Sigma^1_E) \neq \emptyset$ *then Σ^1_T is satisfiable.*

On the other hand, as an example of unsatisfiability consider the following constraints:

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

$$\Sigma^2_T = \{\sigma_1 : \mathit{rock\_singer}(X) \rightarrow \mathit{sing\_loud}(X), \sigma_2 : \mathit{sing\_loud}(X) \rightarrow \mathit{sore\_throat}(X),$$
$$\sigma_3 : \mathit{rock\_singer}(X) \rightarrow \mathit{can\_sing}(X)\}$$

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 $\mathit{rock\_singer}(\mathit{axl})$: we have that the application of Σ^2_T over $\{\mathit{rock\_singer}(\mathit{axl})\}$ causes the violation of τ_1 when considered together with Σ^2_T, and therefore we have that $\text{mods}(\{\mathit{rock\_singer}(\mathit{axl})\}, \Sigma^2_T \cup \Sigma^2_{NC} \cup \Sigma^2_E) = \emptyset$. *Note that* any *set of relevant atoms will cause the violation of τ_1.*

We are now ready to formally 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 5 (Coherence) *An ontology KB is* coherent *if and only if* Σ_T *is satisfiable* w.r.t. $\Sigma_{NC} \cup \Sigma_E$. *Also, KB is said to be incoherent iff it is not coherent.*

Example 5 (Coherence) *Consider the sets of dependencies and constraints defined in Example 4 and an arbitrary database instance D. We can see that the Datalog$^{\pm}$ ontology* $KB_1 = (D, \Sigma^1_T \cup \Sigma^1_{NC} \cup \Sigma^1_E)$ *is coherent, while* $KB_2 = (D, \Sigma^2_T \cup \Sigma^2_{NC} \cup \Sigma^2_E)$ *is incoherent.*

Considering incoherence of a set of TGDs is important in the consolidation process of Datalog$^{\pm}$ ontologies, since if not treated appropriately within the consolidation process, an incoherent set of TGDs may lead to the trivial solution of removing every single relevant atom in D (in the worst case, the entire database instance). This may be adequate for some particular domains, but does not seem to be a desirable outcome in the general case.

Looking into Definitions 4 and 5 we can see that there is a close relationship between the concepts of incoherence and inconsistency. In fact, it can be inferred from those definitions 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. This result is captured in the following proposition (proofs of results are presented in Appendix A).

Proposition 1 *If KB is incoherent and there exists* $A \subseteq D$ *such that A is relevant to some unsatisfiable set* $U \subseteq \Sigma_T$ *then* $KB = (D, \Sigma)$ *is inconsistent.*

As an instance of this relationship, consider the following representative example.

Example 6 (Relating Incoherence and Inconsistency) Consider the following ontology.

$$KB = \left\{ \begin{array}{ll} D: & \{a_1 : \mathit{can\_sing}(\mathit{simone}), a_2 : \mathit{rock\_singer}(\mathit{axl}), a_3 : \mathit{sing\_loud}(\mathit{ronnie}), \\ & a_4 : \mathit{has\_fans}(\mathit{ronnie}), a_5 : \mathit{rock\_singer}(\mathit{ronnie}), a_6 : \mathit{rock\_singer}(\mathit{roy}), \\ & a_7 : \mathit{manage}(\mathit{band}_1, \mathit{richard})\} \\ \\ \Sigma_{NC}: & \{\tau_1 : \mathit{sore\_throat}(X) \wedge \mathit{can\_sing}(X) \rightarrow \bot, \\ & \tau_2 : \mathit{has\_private\_life}(X) \wedge \mathit{famous}(X) \rightarrow \bot\} \\ \\ \Sigma_E: & \{\nu_1 : \mathit{manage}(X, Y) \wedge \mathit{manage}(X, Z) \rightarrow Y = Z\} \\ \\ \Sigma_T: & \{\sigma_1 : \mathit{rock\_singer}(X) \rightarrow \mathit{sing\_loud}(X), \\ & \sigma_2 : \mathit{sing\_loud}(X) \rightarrow \mathit{sore\_throat}(X), \\ & \sigma_3 : \mathit{has\_fans}(X) \rightarrow \mathit{famous}(X), \\ & \sigma_4 : \mathit{rock\_singer}(X) \rightarrow \mathit{can\_sing}(X), \\ & \sigma_5 : \mathit{has\_fans}(X) \rightarrow \mathit{has\_private\_life}(X)\} \end{array} \right\}$$

As hinted previously in Example 4, there we have the set $A \subset D = \{\mathit{rock\_singer}(\mathit{axl})\}$ and the unsatisfiable set of TGDs $U \subset \Sigma_T = \{\sigma_1 : \mathit{rock\_singer}(X) \rightarrow \mathit{sing\_loud}(X), \sigma_2 : \mathit{sing\_loud}(X) \rightarrow \mathit{sore\_throat}(X), \sigma_4 : \mathit{rock\_singer}(X) \rightarrow \mathit{can\_sing}(X)\}$. Since A is relevant to U the conditions in Proposition 1 are fulfilled, and indeed the ontology $KB = (D, \Sigma)$ is inconsistent since $\tau_1 \in \Sigma_T$ is violated.

A set of constraints such as the one presented in Example 6 may appear when we consider scenarios where both components of an ontology evolve (perhaps being collaboratively

maintained by a pool of users). As long as new constraints are added, incoherence problems may arise. In this particular scenario it would seem more sensible to identify and modify, somehow, the set of incoherent constraints to make them satisfiable, instead of deleting all the information from the ontology; and only then proceed to solve remaining inconsistencies, if any. That is, it could be beneficial to define consolidation processes in which the changes performed to achieve coherence are given higher priority than the changes needed for consistency when possible. To address this we present a twofold proposal for consolidation of Datalog$^{\pm}$ ontologies: that is, to obtain the new KB' we begin by addressing issues in the component Σ_T *w.r.t.* the components Σ_E and Σ_{NC} in the original ontology to obtain a new coherent set of constraints, giving up some TGDs in Σ_T if necessary. Then, we address the problems arising from the component D, obtaining a new one D' that is consistent with $\Sigma'_T \cup \Sigma_E \cup \Sigma_{NC}$. In the next section we characterize, by means of a set of postulates, a consolidation operator that takes into account these considerations.

4. Characterizing the Consolidation: Postulates for Datalog$^{\pm}$ Ontology Consolidation Operators

Belief Revision is one of the main areas that deals with defined principled methods to solve incoherences and inconsistencies; as explained in Section 2, it is common to characterize change operators by means of *postulates*, which are properties that the operators must satisfy. In this section we introduce a set of postulates with the objective of characterizing consolidation operators for Datalog$^{\pm}$ ontologies. We start by briefly defining the scenario underlying the consolidation process and introducing the characteristics of the sets of formulas that we focus on (Friedman & Halpern, 2001).

4.1 Defining the Consolidation Environment

Depending on the type of knowledge base, we find two main streams of work in Belief Revision. On one hand, some works are based on sets of formulas that are closed under some consequence relation, called *belief sets* (Alchourrón et al., 1985). This is known in the Belief Revision literature as the *coherence model*. On the other hand, the option is to choose *belief bases* (Katsuno & Mendelzon, 1991, 1992; Fuhrmann, 1991; Hansson, 1994, 1997, 2001; Falappa, Kern-Isberner, & Simari, 2002), i.e., non-closed sets of formulas; this is referred to as the *foundational model*.

Opposite to the traditional closed world assumption found in other established areas like relational databases, one important characteristic of Datalog$^{\pm}$ is that of an open world assumption, unknown data is represented by means of null values. As a consequence, the generation of new information in the language by the application of rules is susceptible of being infinite (Calì, Gottlob, & Kifer, 2008, 2013), which seems to make the foundational model a more appealing choice when working in this setting. Therefore, for the consolidation of Datalog$^{\pm}$ ontologies we have chosen to follow the foundational model. In this model, the epistemic state is a (possibly incoherent and inconsistent) Datalog$^{\pm}$ ontology.

4.2 Expected Properties for the Consolidation Operator: Postulates

We present now the set of properties that a consolidation operator for Datalog$^{\pm}$ ontologies must satisfy. We use the following notation through the rest of the paper. Let $KB = (D, \Sigma)$ be the original Datalog$^{\pm}$ ontology being consolidated, where $\Sigma = \Sigma_{_T} \cup \Sigma_{_E} \cup \Sigma_{_{NC}}$. Also, $KB!$ denotes the Datalog$^{\pm}$ ontology $KB! = (D!, \Sigma!)$ resulting from the consolidation of KB, with $D!$ and $\Sigma!$ being the consolidated components D and Σ in $KB!$, respectively. When necessary we will differentiate KBs by using subscripts. In such cases, given KB_i we denote its consolidation by $KB_i! = (D_i!, \Sigma_i!)$.

We are ready now to introduce the Ontology Consolidation Postulates (OCP) expected to be satisfied by the consolidation operators. Let Θ be the set of all Datalog$^{\pm}$ ontologies. Then, a Datalog$^{\pm}$ ontology consolidation operator $! : \Theta \longrightarrow \Theta$ is a function that must satisfy the following properties:

OCP 1. **(Inclusion)** $\Sigma! \subseteq \Sigma$ and $D! \subseteq D$.

The consolidation process only includes in the resulting ontology formulas belonging to the original ontology.

OCP 2. **(Consistency)** $KB!$ is consistent.

The ontology obtained by the consolidation process must be consistent, i.e., there are no negative constraints or equality-generating dependencies that are violated when we apply all TGDs in $\Sigma!$ to the atoms in $D!$, and therefore $\text{mods}(\{D!, \Sigma!\}) \neq \emptyset$.

OCP 3. **(Coherence)** $KB!$ is coherent.

The ontology obtained by the consolidation process must be coherent, i.e., $\Sigma_{_T}$ in $\Sigma!$ must be satisfiable with respect to $\Sigma_{_{NC}} \cup \Sigma_{_E}$ in $\Sigma!$.

OCP 4. **(Minimality)**: If $KB' \subseteq KB$ is coherent and consistent, then it holds that $KB! \not\subset KB'$.

There is no coherent and consistent ontology obtained from the original ontology that strictly contains the consolidated ontology.

Some of the postulates presented are inspired by the properties proposed by Hansson (1994) and by Konieczny and Pino-Pérez (2002). Nevertheless, they are adapted to suit the particularities of the ontological setting of Datalog$^{\pm}$; in particular, they take into account the distinction between incoherence and inconsistency. For instance, **Inclusion** is a direct adaptation of Hansson's homonymous postulate (Hansson, 1994), which states that the contraction of a knowledge base should be a (not necessarily proper) subset of the original one. **Consistency** and **Coherence**, on the other hand, result from adapting to our setting Konieczny and Pino-Pérez's postulate **IC1** (2002), which intuitively ask that the resulting merging must be consistent; in here we ask that the resulting consolidation is not only consistent but also coherent. **Minimality** is a postulate added to ensure the quality of the consolidation (*w.r.t.* a loss of information aspect), and is not adapted from any particular work, but rather as a general notion in Belief Revision, where as noted by

Hansson (2001) it has been given many names such as conservatism (Harman, 2008), conservativity (Gärdenfors, 1988), minimum mutilation (Quine, 1986) and minimal change (Rott, 1992).

The proposed postulates capture the notion that changes made with respect to the original ontology are those that are necessary, and that the resulting ontology is, as expected, both coherent and consistent. That is, given the original ontology the consolidation process only removes constraints (TGDs) and atoms if they are somehow involved in making the original ontology incoherent/inconsistent, and makes it in such a way that no unnecessary removal is made.

5. A Datalog$^{\pm}$ Ontology Consolidation Operator

In previous sections we have presented examples of incoherences and inconsistencies that can arise in Datalog$^{\pm}$ ontologies. Additionally, we stated the properties that the consolidation operator should satisfy in order to make *adequate* changes in the original ontology regaining coherence and consistency. Now, we propose a construction for the consolidation operator that addresses such incoherence and inconsistency problems in Datalog$^{\pm}$ ontologies.

5.1 A Possible Construction for the Consolidation Operator

In the literature of Belief Revision several constructions for revision and contraction operators have been studied. Hansson (1994) presents how a contraction operation on belief bases can be modeled by means of the application of *incision functions*. These functions contract a belief base by a formula α by taking minimal sets that entail α (called α-*kernels*) and producing "incisions" on these sets so they no longer entail α. The resulting belief base is then conformed by the union of all formulas that are not removed by any function. This approach is known as *kernel contraction*; the task of restoring consistency is also known in the belief revision literature as contraction by *falsum* (Hansson, 1991). In this work, we define the consolidation process as the application of incision functions. Nevertheless, instead of directly considering minimal inconsistent subsets of formulas from the different components of the ontology (which are equivalent to $\bot$-kernels), in this work we perform incisions over structures called *clusters* (Martinez, Pugliese, Simari, Subrahmanian, & Prade, 2007; Lukasiewicz et al., 2012) that groups together related kernels. More specifically, to solve incoherence we begin by establishing the dependency kernels; in an analogous way, we define the data kernels to solve inconsistencies in D *w.r.t.* Σ, then, based on them, we obtain the dependency clusters and data clusters by exploiting an *overlapping* relation.

5.1.1 Identifying the Relation among Conflicts

The first step towards conflict resolution in our framework is to calculate the minimal coherence and consistency conflicts, and identify possible relations among such conflicts, if any. Dependency kernels are sets of TGDs which are unsatisfiable *w.r.t.* the set of NCs and EGDs in a Datalog$^{\pm}$ ontology and are minimal under set inclusion. These sets are known as Minimal unsatisfiability-preserving sub-TBoxes (MUPS) and Minimal incoherence-preserving sub-TBoxes (MIPS) (Schlobach & Cornet, 2003) in the DL community.

Definition 6 (Dependency Kernels) *The set of dependency kernels of KB, denoted with $\prod_{KB}$, is the set of all $X \subseteq \Sigma_T$ such that X is an unsatisfiable set of dependencies* w.r.t. $\Sigma_{NC} \cup \Sigma_E$ *and every proper subset X' of X ($X' \subsetneq X$) is satisfiable* w.r.t. $\Sigma_{NC} \cup \Sigma_E$.

Example 7 (Dependency Kernels) Consider the following sets of constraints in a Datalog$^{\pm}$ ontology KB:

$$KB = \left\{ \begin{array}{ll} \Sigma_{NC}: & \{\tau_1 : counselor(X) \wedge regent(X) \rightarrow \bot, \\ & \ \tau_2 : cannot\_rule(X) \wedge heir(X) \rightarrow \bot\} \\ \\ \Sigma_E: & \{\nu_1 : advise(X,K) \wedge advise(X,K') \rightarrow K = K'\} \\ \\ \Sigma_T: & \{\sigma_1 : advise(X,K) \rightarrow counselor(X), \\ & \ \sigma_2 : propose\_law(X,K) \rightarrow regent(X), \\ & \ \sigma_3 : prince(P) \rightarrow heir(P), \\ & \ \sigma_4 : son(P,K) \wedge king(K) \rightarrow prince(P), \\ & \ \sigma_5 : counselor(C) \rightarrow regent(C), \\ & \ \sigma_6 : bastard\_son(X,Y) \rightarrow son(X,Y), \\ & \ \sigma_7 : bastard\_son(X,K) \wedge king(K) \rightarrow cannot\_rule(X)\} \end{array} \right\}$$

For this KB there exist two dependency kernels, i.e.,

$$\prod_{KB} = \{\{\sigma_3, \sigma_4, \sigma_6, \sigma_7\}, \{\sigma_5\}\}.$$

It is easy to show that the dependency kernels for a Datalog$^{\pm}$ ontology are independent from the particular component D in the ontology, and thus they can be obtained by looking only into the component Σ. That is, even if we replace the component D in an ontology with an empty set of atoms, the dependency kernels for the ontology with the empty database are the same than those in the original one.

Lemma 1 *Let $KB_1 = (D_1, \Sigma_1)$ and $KB_2 = (\emptyset, \Sigma_2)$ be two Datalog$^{\pm}$ ontologies such that $\Sigma_1 = \Sigma_2$. Then, $\prod_{KB_1} = \prod_{KB_2}$.*

In addition to the removal of the TGDs that make a set Σ unsatisfiable (thus making an ontology incoherent), to solve inconsistencies we may need to remove atoms from components D in order to address data inconsistency as well. Analogously to the definition of the dependency kernels, we define now *data kernels* as the minimal subset of atoms in D that makes a $KB = (D, \Sigma)$ inconsistent.

Definition 7 (Data Kernels) *The set of data kernels of KB, denoted with $\coprod_{KB}$, is the set of all $X \subseteq D$ such that* $\mathrm{mods}(X, \Sigma) = \emptyset$ *and for every $X' \subsetneq X$ it holds that* $\mathrm{mods}(X', \Sigma) \neq \emptyset$.

Example 8 (Data Kernels) Consider the following coherent but inconsistent KB, proposed by Lukasiewicz et al. (2012).

$$KB = \left\{ \begin{array}{ll} D: & \{directs(john, d_1), directs(tom, d_1), directs(tom, d_2), \\ & \ supervises(tom, john), works\_in(john, d_1), works\_in(tom, d_1)\} \\ \\ \Sigma_{NC}: & \{supervises(X, Y) \wedge manager(Y) \rightarrow \bot, \\ & \ supervises(X, Y) \wedge works\_in(X, D) \wedge directs(Y, D) \rightarrow \bot\} \\ \\ \Sigma_{E}: & \{directs(X, D) \wedge directs(X, D') \rightarrow D = D'\} \\ \\ \Sigma_{T}: & \{\sigma_1 : works\_in(X, D) \rightarrow employee(X), \\ & \ \sigma_2 : directs(X, D) \rightarrow employee(X), \\ & \ \sigma_3 : directs(X, D) \wedge works\_in(X, D) \rightarrow manager(X)\} \end{array} \right\}$$

For this KB, the set of data kernels is

$$\coprod_{KB} = \left\{ \begin{array}{l} \{supervises(tom, john), directs(john, d_1), works\_in(john, d_1)\}, \\ \{supervises(tom, john), directs(john, d_1), works\_in(tom, d_1)\}, \\ \{directs(tom, d_1), directs(tom, d_2)\} \end{array} \right\}$$

Once we know the minimal conflicts in the ontology we identify relations among them, if such relation exists. To do this, we group related kernels together in a new structure called *cluster*, which makes possible to achieve an optimal solution in related kernels. Clusters are obtained through an overlapping relation defined as follows.

Definition 8 (Overlapping, Equivalence) *Let $\mathcal{L}$ be a first order language, $\mathcal{R} \subset \mathcal{L}$ be a relational schema, and $\mathcal{L}_{\mathcal{R}}$ the sublanguage generated by $\mathcal{R}$. Given $A \subset \mathcal{L}_{\mathcal{R}}$ and $B \subset \mathcal{L}_{\mathcal{R}}$, we say they overlap, denoted $A\ \theta B$, iff $A \bigcap B \neq \emptyset$. Furthermore, given a multi-set of first order formulas $\mathcal{M} \subset 2^{\mathcal{L}_{\mathcal{R}}}$ we denote as $\theta^*_{\mathcal{M}}$ the equivalence relation obtained over $\mathcal{M}$ through the reflexive and transitive closure of θ.*

By exploiting the overlapping among dependency kernels and data kernels we can define *dependency clusters* and *data clusters*, respectively.

Definition 9 (Dependency Clusters) *Let $\prod_{KB}$ be the set of Dependency Kernels for KB. Let θ be the overlapping relation, and $\mathcal{K} = \prod_{KB} / \theta^*_{\prod_{KB}}$ the quotient set for the equivalence relation obtained over $\prod_{KB}$. A Constraint Cluster is a set $C = \bigcup_{\kappa \in [\kappa]} \kappa$, where $[\kappa] \in \mathcal{K}$. We denote by $\mathbb{\prod}_{KB}$ the set of all Constraint Clusters for KB.*

Definition 10 (Data Clusters) *Let $\coprod_{KB}$ be the set of Data Kernels for KB. Let θ be the overlapping relation, and $\mathcal{K} = \coprod_{KB} / \theta^*_{\coprod_{KB}}$ the quotient set for the equivalence relation obtained over $\coprod_{KB}$. A Data Cluster is a set $C = \bigcup_{\kappa \in [\kappa]} \kappa$, where $[\kappa] \in \mathcal{K}$. We denote by $\mathbb{\coprod}_{KB}$ the set of all Data Clusters for KB.*

Intuitively, a dependency cluster groups dependency kernels that have some TGD in common, in a transitive fashion; data clusters groups data kernels in an analogous way.

Example 9 (Dependency Clusters and Data Clusters) *Assume we have KB such that* $\prod_{KB} = \{\{\sigma_1, \sigma_2\}, \{\sigma_1, \sigma_3\}, \{\sigma_4, \sigma_5\}\}$ *and* $\coprod_{KB} = \{\{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_4, a_5\}\}$. *Then, we have two dependency clusters based on those kernels, grouping the first two kernels (due to* σ_1*) and the remaining kernel in another cluster; i.e.,*

$$\amalg_{KB} = \{\{\sigma_1, \sigma_2, \sigma_3\}, \{\sigma_4, \sigma_5\}\}.$$

On the other hand, for the case of data clusters we have that

$$\amalg_{KB} = \{\{a_1, a_2, a_3, a_4, a_5\}\}.$$

The following proposition states that, since clusters are based on equivalence classes, every kernel is included in one and only one cluster.

Proposition 2 *$Y \in \prod_{KB}$ is such that $Y \subseteq X$ for some $X \in \amalg_{KB}$ if and only if $Y \nsubseteq X'$ for all $X' \in \amalg_{KB}$ such that $X \neq X'$. Analogously, $Y \in \coprod_{KB}$ is such that $Y \subseteq X$ for some $X \in \amalg_{KB}$ if and only if $Y \nsubseteq X'$ for all $X' \in \amalg_{KB}$ such that $X \neq X'$.*

As a corollary of Proposition 2 we have that a formula in a kernel is included in only one cluster.

Corollary 1 (Corollary from Proposition 2) *Let $\alpha \in Y$ for some $Y \in \prod_{KB}$ and $\beta \in Y'$ for some $Y' \in \coprod_{KB}$. Then, $\alpha \in X$ for some $X \in \amalg_{KB}$ if and only if $\alpha \notin X'$ for all $X' \in \amalg_{KB}$ such that $X \neq X'$. Analogously, $\beta \in Y'$ is such that $\beta \in X$ for some $X \in \amalg_{KB}$ if and only if $\beta \notin X'$ for all $X' \in \amalg_{KB}$ such that $X \neq X'$.*

The following lemma that we shall use further in the paper shows an example of how, in the ontological setting of Datalog$^{\pm}$, Leibniz's *indiscernibility of identicals* (von Leibniz, 1976) holds *w.r.t.* the clusters in Datalog$^{\pm}$ ontologies, as when two *KBs* are equivalent they have the same set of clusters.

Lemma 2 *Let KB_1 and KB_2 be two Datalog$^{\pm}$ ontologies such that $KB_1 = KB_2$. Then, $\amalg_{KB_1} = \amalg_{KB_2}$ and $\amalg_{KB_1} = \amalg_{KB_2}$.*

5.1.2 Solving Conflicts: Incision Functions

Once we have identified the clusters, we have to establish how the incoherences and inconsistencies are solved. An incision function selects which formulas should be deleted from the data and dependency clusters.

Definition 11 (General Incision Function) *A* General Incision Function *for KB is a function $\delta : (2^{\mathcal{L}_\mathcal{R}}, 2^{\mathcal{L}_\mathcal{R}}) \longrightarrow 2^{\mathcal{L}_\mathcal{R}}$ such that all following conditions holds:*

1. *$\delta(KB) \subseteq \bigcup(\amalg_{KB}) \cup \bigcup(\amalg_{KB})$.*
2. *For all $X \in \amalg_{KB}$ and $Y \in \prod_{KB}$ such that $Y \subseteq X$ it holds that $(Y \cap \delta(KB)) \neq \emptyset$.*
3. *For all $X \in \amalg_{KB}$ and $Y \in \coprod_{KB}$ such that $Y \subseteq X$ it holds that $(Y \cap \delta(KB)) \neq \emptyset$.*

4. *For all* $X \in \mathbb{III}_{KB}$ *it holds that* $T = (X \cap \delta(KB))$ *is such that there not exists* $R \subset X$ *where* R *satisfies conditions 1 and 2, and* $R \subsetneq T$.

5. *For all* $X \in \mathbb{III}_{KB}$ *it holds that* $T = (X \cap \delta(KB))$ *is such that there not exists* $R \subset X$ *where* R *satisfies conditions 1 and 3, and* $R \subsetneq T$.

Definition 11 states that a general incision function selects from each dependency (data, respectively) cluster TGDs (atoms, respectively) for deletion in order to restore coherence (consistency). Any incision function that complies with Definition 11 can be used as a base for a consolidation operator. However, note that such an operator may not differentiate between restoring coherence and consistency. This is not a problem in the classic literature of Belief Revision since there is no notion of incoherence, and there is no distinction between rules and facts in languages like propositional logic; thus, only consistency conflicts can appear, avoiding the need to treat incoherences. Nevertheless, in the ontological setting of Datalog$^{\pm}$ we have the opportunity of exploiting the fact that we have two different although related kinds of conflicts to address them separately with the goal of finding a solution that better suits the needs of applications that rely on this kind of knowledge bases.

The point this paper is trying to make is that, for knowledge bases in the form of Datalog$^{\pm}$ ontologies it is important to differentiate, and adequately handle, incoherence from inconsistency as the quality of the consolidated ontology heavily depends on that — assuming we strive for minimal loss in the process. This, more complex setting, needs a careful definition of what constitutes a kernel. To see what could happen if this is not done properly, consider the following example.

Example 10 (Influence of Incoherence on Consolidation) Consider KB from Example 6. There we have $\Sigma = \Sigma^2_T \cup \Sigma^2_E \cup \Sigma^2_{NC}$ such that Σ^2_T is unsatisfiable. As explained in Example 6, for the singleton set $\{rock\_singer(axl)\}$ we have that the NC $\tau_1 : sore\_throat(X) \wedge can\_sing(X) \rightarrow \bot$ is violated, making $\{rock\_singer(axl)\}$ inconsistent with Σ. Then, $\{rock\_singer(axl)\}$ is a data kernel (and a data cluster, since it cannot overlap with any other kernel) and the same is verifiable for every singleton set in D relevant to some dependency cluster. Thus, we have that

$$\mathbb{III}_{KB} = \left\{ \begin{array}{l} \{rock\_singer(axl)\}, \\ \{rock\_singer(ronnie)\}, \\ \{rock\_singer(roy)\}, \\ \{has\_fans(ronnie)\} \end{array} \right\}$$

Consider the cluster $\{rock\_singer(axl)\}$; for this cluster we have that

$$\delta(\{rock\_singer(axl)\}) = rock\_singer(axl).$$

This same situation holds for every cluster in $\mathbb{III}_{KB}$, and thus $\delta(\mathbb{III}_{KB}) = \bigcup(\mathbb{III}_{KB})$.

The problem in this example is that the data kernels (and hence so the data clusters) are computed *w.r.t.* the original Σ component, which, in this case, contain unsatisfiable sets of constraints. As can be seen in Example 10, this becomes of utter importance when we have atoms relevant to unsatisfiable sets: in that case, any general incision function (and any inconsistency management technique based on deletion that does not treat incoherence conflicts) will necessarily delete such atoms.

Proposition 3 *Let δ be a general incision function. If $\alpha \in D$ is relevant to some $X \in \prod_{KB}$ then $\alpha \in \delta(KB)$.*

Clearly, as a corollary of Proposition 3 we have that if every atom in D is relevant to some unsatisfiable set then we have to remove every atom in D to restore consistency.

Corollary 2 (Corollary from Proposition 3) *Let δ be a general incision function. If for all $\alpha \in D$ it holds that α is relevant to some $X \in \prod_{KB}$ then $D \subseteq \delta(KB)$.*

As seen, incoherence can have great influence in consolidation if not treated properly (that is, previously to the consistency restoration). It would seem better to compute the data clusters based only on the retained satisfiable part of the Σ components. In Lemma 1 we show that the dependency kernels can be obtained independently of the D component from the original ontology, because unsatisfiable sets are such that they violate a negative constraint or equality-generating dependency for *any* relevant set of atoms. Therefore, we can first obtain $\mathbb{III}_{KB}$, and use the incision function on the dependency clusters to select which TGDs will be deleted. Then, we calculate $\mathbb{III}_{KB}$ based on the result of the application of the incision function on $\mathbb{III}_{KB}$, in this way only paying attention to the constraints that will prevail in the consolidation process.

Next, we define both *constraint incision functions* and *data incision functions* which are used to select candidates for deletion (from the original ontology) to restore coherence and consistency, respectively. First, we define an incision function on dependency clusters that helps to solve incoherence on the constraints.

Definition 12 (Constraints Incision Function) *A* Constraint Incision Function *for KB is a function $\rho : (2^{\mathcal{L}_\mathcal{R}}, 2^{\mathcal{L}_\mathcal{R}}) \longrightarrow 2^{\mathcal{L}_\mathcal{R}}$ such that all following conditions hold:*

1. *$\rho(KB) \subseteq \bigcup(\mathbb{III}_{KB})$.*
2. *For all $X \in \mathbb{III}_{KB}$ and $Y \in \prod_{KB}$ such that $Y \subseteq X$ it holds that $(Y \cap \rho(KB)) \neq \emptyset$.*
3. *For all $X \in \mathbb{III}_{KB}$ it holds that $T = (X \cap \rho(KB))$ is such that there not exists $R \subset X$ where R satisfies conditions 1 and 2, and $R \subsetneq T$.*

Intuitively, a constraint incision function takes dependency clusters and removes TGDs from each of them in such a way that the resulting KB is coherent. Analogously to the constraints incision functions, we define data incision functions that solve inconsistencies in $\mathbb{III}_{KB}$.

Definition 13 (Data Incision Function) *A* Data Incision Function *for D is a function $\varrho : (2^{\mathcal{L}_\mathcal{R}}, 2^{\mathcal{L}_\mathcal{R}}) \longrightarrow 2^{\mathcal{L}_\mathcal{R}}$ such that all following conditions hold:*

- *$\varrho(KB) \subseteq \bigcup(\mathbb{III}_{KB})$.*
- *For all $X \in \mathbb{III}_{KB}$ and $Y \in \coprod_{KB}$ such that $Y \subseteq X$ it holds that $(Y \cap \varrho(KB)) \neq \emptyset$.*
- *For all $X \in \mathbb{III}_{KB}$ it holds that $T = (X \cap \varrho(KB))$ is such that there not exists $R \subset X$ where R satisfies conditions 13 and 13, and $R \subsetneq T$.*

Finally, it is necessary to make a significant remark regarding our usage of incision functions. For that, let us first consider the following excerpt quoted from Hansson's (2001, cf. p. 122) regarding the possible parameters passed to selection functions (which in our case are incision functions) and how this choice affects the possible outcomes.

> *"[...] the proof of uniformity makes essential use of the fact that selection functions have been defined on remainder sets of the form $A\perp\alpha$, not on pairs of the form $\langle A\perp\alpha, \alpha\rangle$. If we had instead defined selection functions as follows:*
>
> - *$\gamma(A,\alpha)$ is a non-empty subset of $\gamma(A\perp\alpha,\alpha)$ if $A\perp\alpha$ is non-empty.*
> - *$\gamma(A,\alpha) = \{A\}$ if $A\perp\alpha$ is empty.*
>
> *then $\bigcap\gamma(A,\alpha)$ would have been an operation very similar to partial meet contraction in other respects, but it would have been possible for $\gamma(A,\alpha) \neq \gamma(A,\beta)$ to hold if $A\perp\alpha = A\perp\beta$, which the standard definition does not allow [...]"*

Thus, extending Hansson's observation to incision functions and their use on consolidation, we have that if we take only the sets of conflicts as arguments for incisions then the formulas to be removed from two different ontologies having the same set of conflicts by an operator using the incision function are identical. The reason for this is that the operator could not tell the difference between the ontologies since its parameter is only the conflicts, which are exactly the same. However, here we have chosen not to restrict our family of operators to such behaviors; instead, we model operators whose behavior could select for removal the same formula from equal conflicts, but that are not restricted to that choice. To achieve this, we have chosen to take ontologies as parameters; so, if it fits the application domain in which the operators are exploited, formulas that are not in any conflict could affect the outcome of the consolidation.

In the approach presented here, an incision function not only should consider the TGD's effect over a cluster, but its global effect over the whole knowledge base. The reason for this requirement is that unlike the classic models of belief revision, the language used has greater expressivity and the fact that a TGD generates multiple inferences. For instance, in our framework from a TGD of the form $\forall\mathbf{X}\forall\mathbf{Y}\Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\Psi(\mathbf{X},\mathbf{Z})$ it is possible to infer multiple instances of $\Psi(\mathbf{X},\mathbf{Z})$.

To see the reason behind our choice more clearly consider the following example.

Example 11 *Consider the following ontologies.*

$$KB_1 = \left\{ \begin{array}{ll} D: & \{p(a), q(a)\} \\ \Sigma_{NC}: & \{p(X) \wedge r(X) \rightarrow \bot\} \\ \Sigma_{T}: & \{\sigma_1 : q(X) \rightarrow r(X)\} \end{array} \right\} \quad KB_2 = \left\{ \begin{array}{ll} D: & \{p(a), q(a)\} \\ \Sigma_{NC}: & \{p(X) \wedge r(X) \rightarrow \bot\} \\ \Sigma_{T}: & \{\sigma_1 : q(X) \rightarrow r(X), \\ & \sigma_2 : p(X) \rightarrow s(X), \\ & \sigma_3 : p(X) \rightarrow t(X)\} \end{array} \right\}$$

For these KB, the set of data clusters are equal, as we have

$$\text{Ⅲ}_{KB_1} = \text{Ⅲ}_{KB_2} = \{\ \{p(a), q(a)\}\ \}$$

If we use the standard approach and take clusters as arguments for incisions, then we must remove the same formula in both ontologies, because as it was explained the incision is a function and therefore it cannot choose differently for the same argument.

Nevertheless, suppose that in our particular scenario we want to remove the atoms based on the information they help to infer. If that is the case, then from KB_1 we should remove $p(a)$, but from KB_2 we should take out $q(a)$, since in KB_2 the formula $p(a)$ triggers more TGDs, thus inferring more atoms. To achieve this type of behavior, it is necessary to pass ontologies as parameters, since it is that what provides the adequate context.

5.1.3 Cluster Contraction-Based Consolidation Operator

Lastly, we define the consolidation operator for Datalog$^{\pm}$ ontologies that represents the two different parts in the consolidation. First, the coherence restoration of the component Σ is obtained based on the dependency clusters in the D component in the original ontology. Second, the restoration of consistency in the D component is obtained based on the data clusters *w.r.t.* the $\Sigma!$ component obtained by applying a constraint incision function on the original Σ. In this way we achieve the behavior stated earlier in the paper; in a sense, we give the incoherence resolution higher priority, since if we can retain atoms by addressing unsatisfiable sets of TGDs instead, we choose to follow that path. The cluster contraction-based consolidation operator is formally defined as follows:

Definition 14 (Cluster Contraction-based Consolidation Operator)
Let KB be a Datalog$^{\pm}$ ontology, ρ be a Constraint Incision Function and ϱ a Data Incision Function. Also, let $KB^{\star} = (D, \Sigma \setminus \rho(KB))$ be the Datalog$^{\pm}$ ontology resulting from deleting from KB the TGDs selected by ρ. The Cluster contraction-based consolidation operator $KB!$, is defined as follows:

$$KB! = (D\ \setminus \varrho(KB^{\star}), \Sigma \setminus \rho(KB))$$

The result of $KB!$ is the Datalog$^{\pm}$ ontology obtained by removing, first, the TGDs (selected by ρ from Ⅲ_{KB}) and then atoms (selected by ϱ from $\text{Ⅲ}_{KB^{\star}}$) from the original ontology KB. It is important to note that, on one hand only TGDs are removed from Σ, as dependency clusters do not contain EGDs or NCs. On the other hand, as the Data Incision Function uses $KB^{\star}$ instead of KB then only atoms from D that are in conflicts with $\Sigma \setminus \rho(KB)$ are removed; this is because data clusters are calculated based on the constrains obtained after the consolidation of Σ.

5.2 Relation between Postulates and Construction: Representation Theorem

In Section 4 we have introduced the properties that a Datalog$^{\pm}$ consolidation operator must satisfy. By means of the following representation theorem we can now establish the relationship between the set of postulates for a Datalog$^{\pm}$ ontology consolidation operator and the cluster contraction-based consolidation operator that we proposed in the previous section. In what follows we denote with ! a consolidation operator defined as in Definition 14 where ρ and ϱ correspond to arbitrary constraint and data incision functions, respectively.

Theorem 1 (Representation Theorem) *The operator consolidation* ! *is a* Cluster Contraction-based Datalog$^{\pm}$ Ontology Consolidation Operator *for a Datalog$^{\pm}$ ontology KB iff it satisfies* ***Inclusion***, ***Coherence***, ***Consistency***, *and* ***Minimality***.

6. A Complete Example of Datalog$^{\pm}$ Ontologies Consolidation

We have introduced an operator that allows us to consolidate Datalog$^{\pm}$ ontologies that satisfies the set of expected properties expressed by the postulates in Section 4. In this section, the complete process for the consolidation of Datalog$^{\pm}$ ontologies is depicted in the following example.

Example 12 (Consolidation of Datalog$^{\pm}$ Ontologies) Suppose that we have the (incoherent and inconsistent) ontology KB shown in Figure 1, which expresses the information we have collected about certain company.

$$KB = \left\{ \begin{array}{ll} D: & \{a_1 : boss(walter), a_2 : supervises(walter, jesse), \\ & \ a_3 : makes\_decisions(walter), a_4 : makes\_decisions(jesse), \\ & \ a_5 : supervises(skyler, walter), a_6 : employee(walter), \\ & \ a_7 : in\_charge\_of(jesse, distribution), \\ & \ a_8 : in\_charge\_of(walter, cooking), \\ & \ a_9 : on\_strike(mike)\} \\ \\ \Sigma_{NC}: & \{\tau_1 : follows\_orders(X) \wedge makes\_decisions(X) \rightarrow \bot, \\ & \ \tau_2 : supervises(Y, X) \wedge supervisor(X) \rightarrow \bot, \\ & \ \tau_3 : absent(X) \wedge on\_strike(X) \rightarrow \bot\} \\ \\ \Sigma_{E}: & \{\nu_1 : in\_charge\_of(X, Y) \wedge in\_charge\_of(X, Y') \rightarrow Y = Y'\} \\ \\ \Sigma_{T}: & \{\sigma_1 : employee(X) \rightarrow is\_supervised(X), \\ & \ \sigma_2 : is\_supervised(X) \rightarrow follows\_orders(X), \\ & \ \sigma_3 : boss(X) \rightarrow makes\_profit(X), \\ & \ \sigma_4 : supervises(Y, X) \rightarrow supervisor(Y), \\ & \ \sigma_5 : supervises(Y, X) \rightarrow employee(X), \\ & \ \sigma_6 : is\_supervised(X) \rightarrow makes\_decisions(X), \\ & \ \sigma_7 : is\_supervised(X) \rightarrow has\_work(X), \\ & \ \sigma_8 : has\_work(X) \rightarrow get\_paid(X), \\ & \ \sigma_9 : has\_work(X) \rightarrow \exists Y\ in\_charge\_of(X, Y), \\ & \ \sigma_{10} : on\_strike(X) \rightarrow absent(X)\} \end{array} \right\}$$

Figure 1: The original ontology to be consolidated.

Now, to begin with the first part of the consolidation process (i.e., solving incoherences by making the set Σ_T satisfiable) we obtain, as the first step towards obtaining the dependency clusters, the dependency kernels for KB:

$$\prod_{KB} = \{\{\sigma_2, \sigma_6\}, \{\sigma_{10}\}\},$$

and based on the kernels, we calculate the set of dependency clusters for KB

$$\amalg_{KB} = \{\{\sigma_2, \sigma_6\}, \{\sigma_{10}\}\}.$$

Note that, as there is no overlap among dependency kernels, we have $\prod_{KB} = \iiint_{KB}$. Next, we use a cluster incision function to solve incoherency problems. For the sake of the example assume that we will guide the contraction process by means of a quantitative criterion, i.e., choosing among the possible incisions the ones that removes fewer formulas, and using the plausibility among formulas when the cardinality of the incisions is the same. In the following we show the possible incisions, i.e., those satisfying the conditions in Definition 12. These sets are

- For cluster $\{\sigma_2, \sigma_6\}$ we could either remove σ_2 or σ_6. Since the two incisions remove the same number of atoms assume for this example that σ_2 is more plausible than σ_6, and thus we prefer to retain the former.
- For cluster $\{\sigma_{10}\}$ we can only remove σ_{10}.

Then, the particular incision in this example will be as follows:

$$\rho(\{\sigma_2, \sigma_6\}) = \{\sigma_6\}$$
$$\rho(\{\sigma_{10}\}) = \{\sigma_{10}\}$$

Now, we move on to the next part in the consolidation process: the consistency recovery. As explained before, for this part the operator only considers TGDs that will effectively be included in the consolidation. In this particular example this is $\Sigma_T! = \Sigma_T \setminus \{\sigma_6, \sigma_{10}\}$. From now on then let $KB^\star = (D, \Sigma!)$; based on $KB^\star$ we calculate the data kernels.

$$\prod_{KB^\star} = \{\{a_2, a_4\}, \{a_3, a_5\}, \{a_3, a_6\}, \{a_2, a_5\}\}$$

Then, we obtain the data clusters, which are:

$$\iiint_{KB^\star} = \{\{a_2, a_3, a_4, a_5, a_6\}\}$$

Now, to solve inconsistencies we need to consider those sets such that the intersection with all kernels included in the clusters is not empty, using $\Sigma_T!$ instead of Σ_T when doing this. Once again, we analyze the possible incisions (the sets respecting all conditions in Definition 13) in the light of the number of atoms deleted and the plausibility of the formulas in them. The different possible incisions for the cluster are:

- To remove $\{a_2, a_3\}$.
- To remove $\{a_2, a_3, a_6\}$.
- To remove $\{a_2, a_5, a_6\}$.
- To remove $\{a_4, a_5, a_6\}$.

Once again, each of the sets presented is such that its removal induce an operator that satisfies the postulates, and thus is captured by our framework. Nonetheless, as explained before for this example we will choose to remove as few atoms as possible. That is, we choose to remove $\{a_2, a_3\}$), and so we have

$$\varrho(\{\{a_2, a_3, a_4, a_5, a_6\}\}) = \{a_2, a_3\})$$

Then, using a Datalog$^{\pm}$ ontology consolidation operator based on the contraction of clusters like the one introduced in Definition 14 we can obtain the coherent and consistent ontology shown in Figure 2.

$$KB! = \left\{ \begin{array}{ll} D! : & \{boss(walter), makes\_decisions(jesse), \\ & \ supervises(skyler, walter), employee(walter), \\ & \ in\_charge\_of(jesse, distribution), \\ & \ in\_charge\_of(walter, cooking), \\ & \ on\_strike(mike)\} \\ \\ \Sigma_{NC}! : & \{follows\_orders(X) \wedge makes\_decisions(X) \rightarrow \bot, \\ & \ supervises(Y, X) \wedge supervisor(X) \rightarrow \bot, \\ & \ absent(X) \wedge on\_strike(X) \rightarrow \bot\} \\ \\ \Sigma_{E}! : & \{in\_charge\_of(X, Y) \wedge in\_charge\_of(X, Y') \rightarrow Y = Y'\} \\ \\ \Sigma_{T}! : & \{employee(X) \rightarrow is\_supervised(X), \\ & \ is\_supervised(X) \rightarrow follows\_orders(X), \\ & \ boss(X) \rightarrow makes\_profit(X), \\ & \ supervises(Y, X) \rightarrow supervisor(Y), \\ & \ supervises(Y, X) \rightarrow employee(X), \\ & \ is\_supervised(X) \rightarrow has\_work(X), \\ & \ has\_work(X) \rightarrow get\_paid(X), \\ & \ has\_work(X) \rightarrow \exists Y\ in\_charge\_of(X, Y)\} \end{array} \right\}$$

Figure 2: The ontology resulting from the consolidation.

7. Related Work

The most closely related work to ours is the work by Croitoru and Rodriguez (2015). In that work the authors present consolidation operators that are used as the basis for the definition of semantics for inconsistency tolerant ontology query answering for Datalog+ (a more expressive language than Datalog$^{\pm}$, Calì et al., 2012). As for the case with our work, the work by Croitoru and Rodriguez (2015) is based on the use of Hansson's incision functions (Hansson, 1994) to solve conflicts. Nevertheless, there are some remarkable differences between the works as well. Among the most important ones is that the operators presented by Croitoru and Rodriguez only deal with inconsistent ontologies, but no acknowledgment of the incoherence problem is made. As we have shown through this work, this can have a significant impact in the quality of the consolidation when analysed with respect to minimal loss of information. Moreover, this fact makes that, even though the set of postulates in both works are similar in spirit, the family of operators characterized by Croitoru and Rodriguez is a subset of the ones characterized here. This is due to the fact that the setting that we consider here (i.e., both inconsistent and incoherent ontologies) is more general, since for instance our operators remove not only facts but also TGDs, which Croitoru and Rodriguez's operators do not since they only focus on inconsistency.

Another closely related work is the one by Lukasiewicz et al. (2012). There, the authors define a general framework for inconsistency-tolerant query answering in Datalog$^{\pm}$ ontologies based on the notion of incision functions. Nevertheless, their work is focused on enforcing consistency at query time obtaining (lazy) consistent answers from an inconsistent ontology instead of consolidating one. Clearly, such process must be carried on for every query posed to the system, while with our approach we obtain a new knowledge base

in an *offline* manner, and such knowledge base can be queried without considering inconsistency issues; both approaches can prove useful, depending on the application domain. Additionally, as only one KB is used the rational assumption that there are no conflicts in the constraints in Σ is also made, and therefore there are no notions of unsatisfiability and incoherence. As stated before, in order to gain generality we have chosen to drop that assumption, and treat incoherence problems as well as inconsistency ones. In addition to the works by Croitoru and Rodriguez and Lukasiewicz et al., there are several other works that solve inconsistency or incoherence by means of adapting approaches based on Belief Revision techniques in other knowledge representation formalism.

7.1 Propositional Knowledge Bases

There are numerous works in revision and merging of propositional knowledge bases (see, for instance, Konieczny & Pérez, 2002; Katsuno & Mendelzon, 1992; Lin & Mendelzon, 1999; Liberatore & Schaerf, 1998; Everaere, Konieczny, & Marquis, 2008; Konieczny & Pérez, 2011; Delgrande, Dubois, & Lang, 2006; Booth, Meyer, Varzinczak, & Wassermann, 2010; Delgrande, 2011; Delgrande & Jin, 2012; Falappa, Kern-Isberner, Reis, & Simari, 2012), which had provided the foundations to further work on (fragments of) first order logics. As expected, those works have deep connections with ours, but also has some remarkable differences, as we shall see.

As we have mentioned throughout the paper, the work by Sven Ove Hansson (1994) provides the inspiration and foundations for our work: we follow an approach akin to Kernel Contraction and several intuitions from it, adapted to an ontological language, Datalog$^{\pm}$. As a consequence, besides treating incoherence we also provide a complete inconsistency resolution process which takes advantage of the ontological setting, exploiting the relation between the components of the ontology to define how coherence and consistency should be restored. Also, the classic incision functions introduced by Hansson produce their incision over minimal conflicts. In our approach, however, we work over clusters, which are groupings of kernels, and thus they are not always minimal. Then, we propose a particularization over Hansson's incision functions, focusing on those incision functions that successfully work over clusters.

Konieczny and Pino-Pérez (2002) made one of the main contributions on the merging of conflicting information. In our work we follow some of the intuitions proposed by them. Nevertheless, the main difference between their approach and ours (besides the obvious one on the aims of the works, merging vs. consolidation) is that they state that the final merging will be consistent only in presence of a consistent (or, in our terminology, coherent) set of integrity constraints, and they do not analyze the alternative case.

With respect to the work by Lin and Mendelzon (1999), besides the difference in the focus (once again merging vs. consolidation), the main difference in the inconsistency management strategy chosen with our work is that their conflict solving strategy relies on the "votes" of the majority to establish which formulas is retained in the merging. Instead, we have chosen not to introduce a particular strategy. Nevertheless, it is possible to adapt our framework to the use of preference relations to choose between possible incisions (in a similar way to what we have shown in Example 12). Such relations can indeed be designed to comply with the majority intuition (providing that we have the votes, which does not

apply to an ontology consolidation environment since there is only one ontology), thus obtaining a similar strategy.

In the work by Katsuno and Mendelzon (1992) the problem of knowledge base revision for the propositional case is addressed. As in our approach, the same language is used to express both the facts about the world and the constraints imposed to them in some KB. Nevertheless, once again the difference between the (in this case) update of a KB and the consolidation of a KB arises in the treatment of the integrity constraints: in their work integrity constraints are considered invariant and the updates to restore consistency are restricted to facts.

In they works by Delgrande (2011), Delgrande and Jin (2012) the authors present an approach for revising a propositional knowledge base by a set of sentences, where every sentence in the set can be independently accepted but there can be inconsistencies when considering the whole set. The main idea follows from the AGM theory, but differs in that, it is necessary to alter the Success postulate so it suits the intuition that not every sentence in the set have to be in the final revision (since this set can be inconsistent). Guided by the principle of informational economy, they characterize the revision as the most plausible worlds among the various maximally consistent subsets of the set of sentences. In a parallel with our Datalog$^{\pm}$ ontology environment, this is as revising the D component in the ontology to solve inconsistencies. Being the set of sentences inconsistent, the union of it with the original KB will be inconsistent. Nonetheless, there is an important difference between these works and ours. In those works the authors first solve inconsistencies in the set of sentences, so they can decide which subset of it will characterize the revision. Our approach is different, as we directly consider an inconsistent KB. Then, in order to solve the same problem in our setting, it is necessary to consider the union of the KB with the entire set of sentences, and then apply a consolidation operator.

7.2 Knowledge Expressed in Description Logics Ontologies, Logic Programs and Relational Databases

We now focus on other knowledge representation formalisms that are more closely related to Datalog$^{\pm}$, mainly in the family of Description Logics (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003) and Logic Programming (Lloyd, 1987; Nilsson & Maluszynski, 1995; Gelfond, 2008). A remarkable work using belief revision to solve conflicts in DLs is the one by Qi, Liu, and Bell (2006), which is based on the AGM theory (Alchourrón et al., 1985; Gärdenfors, 1988). What makes this work stand out is that they do not only introduce the generalizations of the AGM postulates to the case of DLs, but also define two operators on knowledge bases, based on formulas weakening, that satisfy such postulates. The main difference with our approach is that they only take into account consistency problems in the ontologies, and no incoherence treatment is provided. As we pointed out earlier, incoherence can lead to extreme a weakening of the information, where they may have to take out every individual name from some general concept inclusion.

As we have previously mentioned, our notion of incoherence was inspired by Schlobach and Cornet's work (2003), among others. In that paper the authors focus on the definition of processes capable of detecting unsatisfiabilities and incoherences in DLs ontologies, introducing complete algorithms along with an empirical analysis of the approach. Nevertheless,

as it is not in the main focus of their work, the authors set aside the issue of how to recover coherence once a conflict has been detected, and also do not consider inconsistencies. In our work we presented a consolidation process that treats both incoherence and inconsistency, based on the use of Belief Revision techniques. Thus, the approach presented by Schlobach and Cornet could potentially be useful regarding the implementation of the operators presented in this work, providing an effective way of obtaining the set of kernels in which the set of clusters will be based.

Black et al. (2009) propose an approach that is capable of using information coming from several DL ontologies in order to answer queries, taking care in the process of both incoherence and inconsistency. Their approach is based on agents with argumentative capabilities, each one with a personal knowledge base in the form of a DL ontology. These agents use dialogue games to interchange arguments until they reach an agreement about the answer to a certain query. Thus, the agents can use the (possible incoherent/inconsistent) union of the ontologies without merging them, and still obtain an answer influenced by every ontology in play. Moreover, this approach has the advantage that no information is lost, as no formula is deleted from the ontologies, and as a result the inferences obtained by this approach are a superset of those that can be obtained in the ontology resulting from the consolidation of the union of DL ontologies. Even though the authors argue that one advantage of the proposed approach is that they do not need to waste time and effort in performing the consolidation of the KB, one disadvantage is the computational complexity associated with argumentative reasoning (Parsons, Wooldridge, & Amgoud, 2003; Dunne & Wooldridge, 2009; Cecchi, Fillottrani, & Simari, 2006) as this process has to be conducted for each query issued and in an online manner. Even though a consolidation process can also be computationally expensive, it is only necessary to perform it once and it can be done offline before the query answering system becomes available. The choice of one approach over the other depends highly on the environment in which they will be used, i.e., on the size of the ontologies that will be used, how often updates are issued over the KB or how critical time consumption is for the system, among other considerations; of course the set of inferences that can be obtained from every approach may differ and this should also be taken into account. A consolidation-based approach could be more suitable for time-dependant systems like real-time systems or query intensive systems where the data tractability associated with (a consolidated) Datalog$^{\pm}$ ontology may be proven handy.

Another work worth mentioning is that by Kalyanpur, Parsia, Horridge, and Sirin's (2007). This work verses on how to find all justifications of entailments over a Description Logics ontology. A justification is simply the precise set of axioms in an ontology responsible for a particular entailment (Kalyanpur et al., 2007). In other words, it is a minimal set of axioms sufficient to produce an entailment, which is related to our use of kernels as a mean to obtain clusters as part of the consolidation strategy used. Moreover, Horridge, Parsia, and Sattler (2009) state that justifications are important for repairing inconsistent ontologies. Thus, they could be important for the definition of consolidation processes similar to our cluster-based consolidation, as "If at least one of the axioms in each of the justifications for an entailment is removed from the ontology, then the corresponding entailment no longer holds."(Kalyanpur et al., 2007, p. 269). One of the main contributions of that work is the definition of practical black-box (i.e.,, reasoner independent) techniques that allows us to find justifications for entailments in the ontology in an efficient way. As such, is evident

that while our work verses in a different direction we still can benefit from those findings. In particular, it may be possible to use the developed algorithms as part of our implementation strategy for our consolidation operators, adapting them to be used in Datalog$^{\pm}$ and our dual incoherence/inconsistency setting.

Regarding Logic Programming, there are also several works that address the problem of merging knowledge bases expressed as logic programs, solving inconsistency issues in the process. For instance, Hué, Papini, and Würbel (2009) introduce a merging process based on stable model semantics, using the logic of Here-and-There (Turner, 2003). Hué et al. consider the merging strategy based on pre-orders among deletion candidates called *potential removed sets* and they do not establish any particular way to obtain these pre-orders. Instead, they assume that for any strategy P there is a given pre-order that defines P. As for the case with Lin and Mendelzon's work (1999), although it falls out of the scope of the present work we certainly can adapt our framework to use similar techniques when choosing which incision prevails.

Another notorious work in the Logic Programming field is the one by Delgrande, Schaub, Tompits, and Woltran (2009). In that work two different approaches are proposed. The first one follows an arbitration approach, selecting the models of a program that differs the least *w.r.t.* the models of the other programs. In this work the case of unsatisfiable programs is studied, similar to the way we consider incoherence leaded by unsatisfiable sets of TGDs. Nevertheless, they consider unsatisfiability of a certain program, and not of some concept in the union of the programs. Furthermore, the strategy to solve unsatisfiability is simply leaving the unsatisfiable program out of consideration for the merging, instead of trying to solve the conflict somehow. The second approach is based on the selection of the models of a special program P_0, which can be thought as the constraints guiding the merging process, that has the least variations *w.r.t.* the programs for the merging. This approach can be seen as a particular instance of the approach proposed by Konieczny and Pérez (2002).

In the area of databases, one of the most influential works is the one by Arenas et al. (1999) on *Consistent Query Answering*, there the authors propose a model theoretic definition of consistent answers to a query in a relational database potentially inconsistent with a set of integrity constraints. Intuitively, the consistent answers to a query is the set of atoms that are (classical) answers to the query in *every repair* of the inconsistent database; a repair is a set of atoms that satisfy the set of constraints and is "as close as possible" to the original database. Different notions of repairs have been studied in the literature, as well as different notions of what it means for a set of atoms to be as close as possible to the original database. Most of the proposals are based on repairing by inserting and/or deleting tuples to/from the database (actually, the possible actions depend on the form of the integrity constraints and their expressiveness) and the notion of closeness is defined via set inclusion or cardinality. The work by Arieli, Denecker, and Bruynooghe (2007), however, proposes a uniform framework for representing and implementing different approaches to database repairing based on minimizing domain dependent distances. The main idea of that work is to show how thinking in terms of (different) distances to express preferences among repairs leads to different preferences that can be applied in different scenarios. The authors show that the set of repairs obtained using the proposed distance functions deviate from those that can be obtained using set-inclusion. Furthermore, besides insertion and deletion of entire tuples there are other several domain independent approaches, e.g., based on cardinality

or more complex objective functions. In the approach proposed by Wijsen (2005) updates are considered a primitive in the theoretical framework; Bohannon et al. (2005) present a cost-based framework that allows finding "good" repairs for databases that exhibit inconsistencies in the form of violations to either functional or inclusion dependencies, allowing also updates to attribute values. In that work, two heuristics are defining for constructing repairs both based on equivalence classes of attribute values; the algorithms presented are based on greedy selection of least repair cost, and a number of performance optimizations are also explored. A quite different semantics for repairing is proposed by Caroprese, Greco, and Zumpano (2009), Caroprese and Truszczynski (2011) through Active Integrity Constraints (AICs for short); an AIC is a production rule where the body is a conjunction of literals, which should be false for the database to be consistent, whereas the head is a disjunction of update atoms that have to be performed if the body is true (that is the constraint is violated). Repairs are then defined as minimal sets (under set inclusion) of update actions (tuple deletions/insertions) and AICs specify the set of update actions that are used to restore data consistency. Hence, among the set of all possible repairs, only the subset of founded repairs consisting of update actions supported by AICs is considered. Other works in this area propose different semantics for repairing by either explicitly or implicitly considering a preference relation among the set of repairs (cf. Andritsos, Fuxman, & Miller, 2006; Staworko, Chomicki, & Marcinkowski, 2012; Greco & Molinaro, 2012).

More recently, in the area of ontology-based data access (OBDA), Lembo et al. (2010) study the adaptation of CQA for *DL-Lite* ontologies, called AR (ABox semantics). In that work, also the intersection (IAR) semantics is presented as a sound approximation of consistent answers; this semantics consists of computing the intersection of all repairs and answers are obtained from there, though (possibly many) AR answers cannot be obtained from under the IAR semantics, the latter are computationally easy to obtain for the *DL-Lite* family, i.e., it is not necessary to compute the whole set of repairs in order to compute their intersection. The data and combined complexity of these and other semantics were studied (Rosati, 2011) for a wider spectrum of DLs. Also, Rosati (2011) presents intractability results for query answering for $\mathcal{EL}_{\bot}$ under the intersection semantics, and a non-recursive segment of that language was proved to be computable in polynomial time. More recently, Bienvenu and Rosati (2013) propose another family of approximations to CQA, also for the *DL-Lite* family. The k-support semantics allows to (soundly) approximate the set of queries entailed under the CQA semantics, based on k subsets of the database that consistently entail q; on the other hand, the k-defeater semantics approximates complete approximations seeking sets that contradict the supporters for q. Both semantics are FO-rewritable for any ontological language for which standard CQ answering is FO-rewritable as well, and can be used in conjunction to over- and under-approximate consistent answers.

Much like Black et al. (2009), the treatment of inconsistencies proposed by all these semantics is related to particular queries instead of the inconsistency of the whole database. Thus, they do not attempt to obtain a final consistent database that can be queried without considering restrictions. Furthermore, they do not address the issues of incoherence and inconsistency together; instead most of the approaches either assume that the set of integrity constraint correctly defines the semantics of the database instance, so there is no room for incoherence, or they treat constraints and data alike at the moment of removing or ignoring information, which leads to the type of problems that we discuss in Example 10. While

these techniques may be suitable for the case of one single database, in the presence of incoherence in the set of ICs, as can be the case when we consider several databases together, this approach would lead to meaningless empty answers, since no subset of the database could satisfy the constraints— as would also be the case for the approach by Lukasiewicz et al. (2012).

Also related to the databases field is the work by Lin and Mendelzon (1998). There, a database is viewed as a first-order theory without rules, and ICs are used to ensure consistency in the final result as in the work by Konieczny and Pérez (2002), presenting ways to solve known database merging problems like synonyms and homonyms. Nonetheless, like Konieczny and Pino-Pérez's work, they do not consider problems related to the set of ICs. Instead, the set of ICs used in the merging process is unique, and the choice of such set is expected to be performed by a merge designer. Unlike Lin and Mendelzon, we do not made an assumption for our consolidation environment on the set of ICs being conflict-free.

Cholvy (1998) introduces another approach that can be used to reason with contradictory information. The framework is represented through a set of axioms and inference rules. Additionally, in the paper several applications for the framework are introduced, e.g., the solving of conflicts among beliefs represented by first order databases, where facts are ground literals and there are rules that can be integrity constraints or deduction rules. In that scenario, contradiction is obtained by the application of the constraints when considering several databases together. This establishes a certain parallel with the case of inconsistency in a Datalog$^{\pm}$ ontology. However, the main difference with our work lies in how the strategy for the inconsistency management process is defined. In that work, a preference order between databases is assumed. Instead, we have chosen not to restrict how to achieve the consolidation, thus presenting a general approach. Nevertheless, as stated before we can adapt incision functions to suit the intuition that no every formula is equally desirable, choosing for instance preferences between ontologies as a guideline (if we are using the approach for other tasks rather than consolidation of a single ontology), obtaining an inconsistency management strategy akin to the one introduced by Cholvy.

Finally, Meyer, Lee, and Booth (2005) use two well-known techniques for knowledge integration for the propositional case, adapted and refined to the expressiveness of DLs. The proposed approach takes the knowledge bases and produces a disjunctive knowledge base (DKB) as the result of the integration. One disadvantage of DKBs is that they state the possible options we can take when the conflicting knowledge is expected to be exploited by a reasoning process rather than choosing one of them. Thus, contrary to our approach where a final consolidated ontology is given, in theirs there is no definitive final merging; moreover, they set aside for further research problems related to incoherence in the integration process.

8. Conclusions and Future Work

Collaborative work and information exchange are becoming key aspects of almost any system; thus, it is of the uttermost importance to have automatic and adequate ways to solve conflicts: as knowledge evolves in a collaborative environment incoherence and inconsistency are prone to arise. This knowledge is often represented by ontologies that can be collaboratively built, and often shared among entities that use and modify them. One particular way to deal with the conflicts that can appear in such application environments is

to try to modify the information contained in the ontology in order to regain coherence and consistency. In this paper we have shown how to achieve the consolidation of Datalog$^{\pm}$ ontologies. We introduced the concept of incoherence in Datalog$^{\pm}$ ontologies in terms of unsatisfiability of sets of TGDs, and showed the relationship with the classical notion of inconsistency of a logical theory that lacks models.

We also proposed a construction for consolidation operators. The construction is inspired by kernel contraction, and uses incision functions on groupings of minimal unsatisfiable/inconsistent sets called clusters to solve conflicts. Finally, we stated the properties that the Datalog$^{\pm}$ ontology consolidation operator is expected to satisfy. We showed that our operators satisfy the respective properties, obtaining as the result of the consolidation a new Datalog$^{\pm}$ ontology that is always coherent and consistent while minimizing the changes that are made in the conflict resolution.

As a final remark, notice that these operators take care of all incoherences in the ontology. However, there are rare cases when the ontology designer introduce some unsatisfiable concepts in an ontology on purpose, to model some particular feature of the application domain. If that is the case then we should not remove the incoherence, and rather we have to delete the atoms triggering it, if any. Clearly, since they were not defined with that setting in mind this behavior cannot be achieved by the operators presented here. Nevertheless, to modify our present approach to suit such setting is almost straightforward, provide that we can identify whether or not some unsatisfiable set of TGDs is made on purpose or not.

As for future work, we intend to study new constructions for Datalog$^{\pm}$ consolidation operators. To do this, first we plan to change the general approach, i.e., operators based on formalisms other than kernel contraction, mainly from the AGM theory (Alchourrón & Makinson, 1985; Alchourrón et al., 1985); and then, while the proposed framework for cluster contraction based consolidation operators is fully constructive, depending on the application domain it may certainly be difficult to asses how to effect the incisions, i.e., it may be hard to decide among the family of possible incisions which one to select. From a design point of view, it may be easier to select how to perform the consolidation if we have some additional information about the formulas in the knowledge base, such as a preference relation that can, for example, be elicited from domain experts. In general, it could be easier for an expert to provide guidelines and information about the application domain at hand that could be then modeled into a preference relation on the formulas in an ontology rather than trying to single out the desired incisions. In this direction we want to explore constructions based on exploiting preference relations among the formulas in the ontologies to define different strategies to choose which formulas to delete, possibly tailored for particular scenarios. Mainly, we plan to analyze two different aspects: the relation between these operators based on preference relations with respect to the ones presented in this work, and how different strategies affect their behavior.

Also, in this work we make a point in differentiating the concept of inconsistency from that of incoherence; therefore, we need to focus on languages that separate extensional from intensional knowledge, otherwise the two notions are indistinguishable (as it is the case in propositional logic). In that sense, the choice of Datalog$^{\pm}$ is due to its desirable property of generalizing several popular languages such as classical Datalog, DL-Lite, ELH, F-Logic-Lite, etc. Even though in this paper we do not perform a particular analysis of the effects of nulls in the proposed solutions to consolidation, the Datalog$^{\pm}$ family of languages

was chosen because it offers a wide variety of languages with high computational tractability (some are FO rewritable and others have PTIME inference algorithms). The results in this work pave the way to continue the research line into the next natural step, which is to show how (or whether) the different syntactic and semantic properties that yield tractability for query answering allow us to obtain tractability results also for the consolidation problem, much in the same way as it has happened already in the area of consistent query answering (where only repairs over the extensional part of the KB are considered). It is, for example, in the rewriting algorithms where the capability of value invention plays an important role: the value invention process should be controlled (in general with syntactic restrictions) in order to keep a low complexity for the reasoning tasks. With this in mind, in the future we will further look into the role of processes like value invention in the consolidation of Datalog$^{\pm}$ ontologies, and their impact both on how conflicts should be solved and computational efficiency.

We are currently working on the implementation of our operators; we plan to study different techniques that can be used in order to produce an efficient implementation, possibly tailored for specific fragments of Datalog$^{\pm}$. As explained before, the algorithms introduced by Schlobach and Cornet (2003) can be proven useful regarding this aspect since they may provide a way to calculate the kernels in a Datalog$^{\pm}$ ontology, thus providing the first step towards incoherence resolution. Another important work regarding the implementation of our consolidation operators is the one by Wassermann (2000), where the author shows that the minimal incision functions of a knowledge base can be obtained from the kernels of a KB by using any algorithm for finding minimal hitting sets (Reiter, 1987). Several works in the area of ontology debugging and repairs, (e.g., Halaschek-Wiener & Katz, 2006, or Horridge et al., 2009 as a way to find the justifications for an inconsistency) have exploited Reiter's algorithms in order to implement their frameworks. Among others, we plan to study how adequate this techniques are for our operators, as there is an almost direct relation between minimal incision functions and Reiter's minimal hitting sets; in this way, it may be possible to adapt Reiter techniques to attend incoherences and inconsistencies, moreover, as we already discussed, we plan to analyze the relation between cluster incision functions and preference relations. Regarding implementation, we hold the conjecture that such relations can be exploited to further refine the implementation of the operators: Reiter's algorithm is based on the expansion of a directed acyclic graph, and such expansion is made in a breadth first fashion, which in the end generates all possible values for minimal incision functions. As acknowledged by Wassermann, if some kind of ordering among the formulas is present, this ordering can be used to choose which branch to expand; in other words, not only it may be possible to implement the construction for operators proposed in this work by means of exploiting Reiter's hitting sets algorithm, but also we can use the preference relation equivalent to the incision (if any) to guide the consolidation process. That is, it may be possible to adapt the algorithm so it chooses to expand the branch that has the less preferred set of formulas, thus guiding the graph expansion process.

Appendix A. Proofs

Proof for Proposition 1

Proof Consider some $U \subseteq \Sigma_T$ such that U is an unsatisfiable set of dependencies *w.r.t.* $\Sigma_{NC} \cup \Sigma_E$, and $A \subseteq D$ a set of atoms relevant to U.

It follows from the definition of satisfiability of a set of dependencies *w.r.t.* a set of constraints that if U is unsatisfiable then there does not exist a relevant set of atoms A' that makes $\text{mods}(A', U \cup \Sigma_E \cup \Sigma_{NC}) \neq \emptyset$, because otherwise U is satisfiable.

Then, $\text{mods}(A, U \cup \Sigma_E \cup \Sigma_{NC}) = \emptyset$. Moreover, since $A \subseteq D$ and $U \subseteq \Sigma_T$ we have that $\mathit{chase}(A, U) \subseteq \mathit{chase}(D, \Sigma_T)$, and thus any NC or EGD that is violated in $\mathit{chase}(A, U)$ is also violated in $\mathit{chase}(D, \Sigma_T)$. Thus, $\text{mods}(D, \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}) = \emptyset$, i.e., KB is inconsistent. ∎

Proof for Lemma 1

Proof Let $KB_1 = (D_1, \Sigma_1)$ with $\Sigma_1 = \Sigma_{1_T} \cup \Sigma_{1_E} \cup \Sigma_{1_{NC}}$, and $KB_2 = (\emptyset, \Sigma_2)$ with $\Sigma_2 = \Sigma_{2_T} \cup \Sigma_{2_E} \cup \Sigma_{2_{NC}}$ be two Datalog$^{\pm}$ ontologies such that $\Sigma_1 = \Sigma_2$, $\prod_{KB_1}$ be the dependency kernels of KB_1, and $\prod_{KB_2}$ be the dependency kernels of KB_2, respectively.

Consider any $X \in \prod_{KB_1}$. Then, by Definition 6 we have that $X \subset \Sigma_{1_T}$ is an unsatisfiable set of dependencies *w.r.t.* $\Sigma_{1_E} \cup \Sigma_{1_{NC}}$ and every $X' \subsetneq X$ is satisfiable *w.r.t.* $\Sigma_{1_E} \cup \Sigma_{1_{NC}}$. Since $\Sigma_1 = \Sigma_2$, then $\Sigma_{1_T} = \Sigma_{2_T}$, $\Sigma_{1_E} = \Sigma_{2_E}$ and $\Sigma_{1_{NC}} = \Sigma_{2_{NC}}$, and thus it holds that $X \subset \Sigma_{2_T}$ is an unsatisfiable set of dependencies *w.r.t.* $\Sigma_{2_E} \cup \Sigma_{2_{NC}}$ and every $X' \subsetneq X$ is satisfiable *w.r.t.* $\Sigma_{2_E} \cup \Sigma_{2_{NC}}$.

Then, by Definition 6 we have that $X \in \prod_{KB_2}$, and since this holds for any arbitrary kernel in $\prod_{KB_1}$ we have that $\prod_{KB_1} = \prod_{KB_2}$.

∎

Proof for Proposition 2

Proof We will focus on the case of dependency clusters, omitting the proof for data clusters, as they are analogous to each other. Consider any arbitrary $Y \in \prod_{KB}$.

$\Rightarrow$) We begin by showing that if a kernel is part of a cluster then it is not part of any other cluster, i.e., if $Y \subseteq X$ for some $X \in \coprod_{KB}$ then $Y \nsubseteq X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$.

This is obtained directly from the definition of clusters: we have that $X = \bigcup_{Y \in [\kappa]} Y$ where $[\kappa]$ is an equivalence class in the equivalence relation θ^* obtained from $\prod_{KB}$. Then, clearly for Y we have that if $Y \subseteq X$ then $Y \in [\kappa]$. Therefore, since by definition two equivalence classes are either equal or disjoint then it holds that $Y \notin [\kappa']$ for all $[\kappa']$. Let $X' = \bigcup_{Y' \in [\kappa']} Y'$. Then it holds that $X \neq X'$ and that $Y \nsubseteq X'$. Since this holds for any arbitrary equivalence class $[\kappa']$ then it holds that if $Y \subseteq X$ for some $X \in \coprod_{KB}$ then $Y \nsubseteq X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$.

$\Leftarrow$) Now we show that there not exist any kernel that does not belong to a cluster, i.e., if $Y \nsubseteq X'$ for all $X \in \coprod_{KB}$ such that $X \neq X'$ then $Y \subseteq X$ for $X \in \coprod_{KB}$. Again, this arise from the use of equivalence classes in Definitions 9 and 10. If $Y \nsubseteq X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$, then it holds that $Y \notin [\kappa']$ for all $[\kappa'] \neq [\kappa]$. So, since equivalence classes form a partition it must holds that $Y \in [\kappa]$. Therefore, as $X = \bigcup_{Y \in [\kappa]} Y$ we have that $Y \nsubseteq X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$ then $Y \subseteq X$. ∎

Proof for Corollary 1

Proof Consider $\alpha \in Y$ for some $Y \in \prod_{KB}$. By Proposition 2 we have that $Y \subseteq X$ for some $X \in \coprod_{KB}$ if and only if $Y \nsubseteq X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$. Thus, we have that $\alpha \in X$ for some $X \in \coprod_{KB}$ if and only if $\alpha \notin X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$.

Analogously, we can show that $\beta \in Y'$ for some $Y' \in \coprod_{KB}$ is such that $\beta \in X$ for some $X \in \coprod_{KB}$ if and only if $\beta \notin X'$ for all $X' \in \coprod_{KB}$ such that $X \neq X'$. ∎

Proof for Lemma 2

Proof Consider any $X \in \prod_{KB_1}$. Then, X is a minimal unsatisfiable set of TGDs *w.r.t.* $\Sigma_{1_{NC}} \cup \Sigma_{1_E}$. Since $KB_1 = KB_2$, then it holds that $X \subset KB_2$, $\Sigma_{1_E} = \Sigma_{2_E}$, $\Sigma_{1_{NC}} = \Sigma_{2_{NC}}$ and X is an unsatisfiable set of TGDs *w.r.t.* $\Sigma_{2_{NC}} \cup \Sigma_{2_E}$. Also, there does not exist $X' \subsetneq X$ such that X' is an unsatisfiable set of TGDs *w.r.t.* $\Sigma_{2_{NC}} \cup \Sigma_{2_E}$, since otherwise we would contradict our hypothesis that $X \in \prod_{KB_1}$, as $\Sigma_{1_{NC}} = \Sigma_{2_{NC}}$ and $\Sigma_{1_E} = \Sigma_{2_E}$. Then, $X \in \prod_{KB_2}$; and since this holds for any arbitrary $X \in \prod_{KB_1}$, then we have that $\prod_{KB_1} = \prod_{KB_2}$.

Consider any arbitrary $X, Y \in \prod_{KB_1}$ such that $X \theta Y$. Since $\prod_{KB_1} = \prod_{KB_2}$, then $X, Y \in \prod_{KB_2}$. Thus, $\theta^*_{\prod_{KB_1}}$ is equivalent to $\theta^*_{\prod_{KB_2}}$, and then $\coprod_{KB_1} = \coprod_{KB_2}$.

Likewise, consider any arbitrary $X', Y' \in \coprod_{KB_1}$ such that $X' \theta Y'$. Since $\coprod_{KB_1} = \coprod_{KB_2}$, then $X', Y' \in \coprod_{KB_2}$. Therefore, $\theta^*_{\coprod_{KB_1}}$ is equivalent to $\theta^*_{\coprod_{KB_2}}$, and thus $\coprod_{KB_1} = \coprod_{KB_2}$.

∎

Proof for Proposition 3

Proof Consider $\alpha \in D$ and $X \in \prod_{KB}$ such that α is relevant to X. From Definition 6 we have that X is unsatisfiable *w.r.t.* $N \subseteq \Sigma_E \cup \Sigma_{NC}$, and then from Definition 4 and the fact that α is relevant to X we have that $\text{mods}(\{\alpha\}, X \cup N) = \emptyset$ (1). Also, since $\{\alpha\}$ is singleton then the only $A \subsetneq \{\alpha\}$ is $A = \emptyset$, and clearly $\text{mods}(\emptyset, X \cup N) \neq \emptyset$ (2). Then, from (1), (2) and Definition 7 it follows that $\{\alpha\} \in \coprod_{KB}$. Also, from Definition 9 we have that $\{\alpha\} \in \coprod_{KB}$, since $\{\alpha\}$ cannot overlap with any other kernel, being singleton.

Consider the incision over $\{\alpha\}$. From Definition 11 it follows that $\delta(KB) \cap \{\alpha\} \neq \emptyset$. Then, we have that $\delta(KB) \cap \{\alpha\} = \alpha$, and thus $\alpha \in \delta(KB)$.

∎

Proof for Corollary 2

Proof Consider any arbitrary $\alpha \in D$. Since α is relevant to some $X \in \prod_{KB}$, then by Proposition 3 it holds that $\alpha \in \delta(KB)$. Thus, since this holds for any arbitrary $\alpha \in D$ we have that $D \subseteq \delta(KB)$. ∎

Proof for Theorem 1

Proof Let $KB_1 = (D_1, \Sigma_1)$ and $KB_2 = (D_2, \Sigma_2)$ be two Datalog$^{\pm}$ ontologies such that $KB_1 = KB_2$.

$\Rightarrow$) *Construction to postulates*

Consider an operator ! defined as in Definition 14; we have to prove that ! satisfies every postulate in Theorem 1. Let $KB_1! = (D_1!, \Sigma_1!)$ and $KB_2! = (D_2!, \Sigma_2!)$ be the two Datalog$^{\pm}$ ontologies resulting from the consolidation of KB_1 and KB_2 by means of !, respectively. Furthermore, let $KB_1^\star = (D_1, \Sigma_1 \setminus \rho(KB_1))$ and $KB_2^\star = (D_2, \Sigma_2 \setminus \rho(KB_2))$ be the ontology resulting from removing the TGDs selected by ρ from KB_1 and KB_2. Let $\prod_{KB_1}$ and $\coprod_{KB_1^\star}$

be the set of dependency and data kernels for KB_1 and $KB_1^\star$ respectively, $\prod_{KB_2}$ and $\coprod_{KB_2^\star}$ be the sets of dependency and data kernels for KB_2 and $KB_2^\star$. Finally, let $\mathbb{\Pi}_{KB_1}$ and $\mathbb{\Pi}_{KB_1^\star}$ be the set of dependency and data clusters for KB_1 and $KB_1^\star$ respectively, $\mathbb{\Pi}_{KB_2}$ and $\mathbb{\Pi}_{KB_2^\star}$ be the sets of dependency and data clusters for KB_2 and $KB_2^\star$.

- **Inclusion**: $\Sigma_1! \subseteq \Sigma_1$ and $D_1! \subseteq D_1$.

  By definition of $KB_1!$ we have that $D_1! = D_1 \setminus \varrho(KB_1^\star)$, and thus $D_1! \subseteq D_1$.

  In a similar way, by definition of $KB_1!$ we have that $\Sigma_1! = \Sigma_1 \setminus \rho(KB_1)$, and thus $\Sigma_1! \subseteq \Sigma_1$.

- **Coherence**: $KB_1!$ is coherent.

  To prove that $KB_1!$ is coherent we have to show that $\Sigma_{_T} \in \Sigma_1!$ is satisfiable for $\Sigma_{_E} \cup \Sigma_{_{NC}} \in \Sigma_1!$. To do this it is sufficient to show that all minimal conflicts are attended to by the operator, i.e., that no dependency kernel is included in $\Sigma_1!$,

  Consider any arbitrary $X \in \prod_{KB_1}$. From Proposition 2 we have that there exists $Y \in \mathbb{\Pi}_{KB_1}$ such that $X \subseteq Y$. By definition of ρ it holds that for all $Y \in \mathbb{\Pi}_{KB_1}$ and $X \in \prod_{KB_1}$ where $X \subseteq Y$ it holds that $(\rho(KB_1) \cap\ X) \neq \emptyset$. Then, there exists some $\alpha \in X$ such that $\alpha \in (\rho(KB_1) \cap\ X)$, and thus $\alpha \notin \Sigma_1!$. Therefore, $X \nsubseteq \Sigma_1!$, i.e., the conflict was solved. Since this holds for any arbitrary $X \in \prod_{KB_1}$ then every unsatisfiable set in Σ_1 is not included in $\Sigma_1!$, and thus $\Sigma_{_T} \in \Sigma_1!$ is satisfiable for $\Sigma_{_E} \cup \Sigma_{_{NC}} \in \Sigma_1!$, i.e., $KB_1!$ is coherent.

- **Consistency**: Proof is analogous to that for **Coherence**.

- **Minimality**: If $KB' \subseteq KB_1$ is coherent and consistent, then it holds that $KB_1! \not\subset KB'$.

  Let $KB' \subseteq KB_1$ be such that KB' is coherent and consistent, and let $\mathcal{CF}_{\Sigma_1} = \Sigma_1 \setminus \bigcup(\mathbb{\Pi}_{KB})$ and $\mathcal{CF}_{D_1} = D_1 \setminus \bigcup(\mathbb{\Pi}_{KB})$ be the set of formulas that do not belong to any kernel in Σ_1 and D_1, respectively.

  Suppose *by reductio* that $KB_1! \subset KB'$. By definition of $KB_1!$ we have that $\rho(KB_1) \subseteq \bigcup(\mathbb{\Pi}_{KB})$, and that $\varrho(KB_1) \subseteq \bigcup(\mathbb{\Pi}_{KB})$. Then, $\mathcal{CF}_{\Sigma_1} \subseteq KB_1!$ and $\mathcal{CF}_{D_1} \subseteq KB_1!$. Therefore, we have that $\mathcal{CF}_{\Sigma_1} \subset KB'$ and that $\mathcal{CF}_{D_1} \subset KB'$.

  Then, since $KB_1! \subset KB'$ there must exist $\alpha \in \mathbb{\Pi}_{KB} \cup \mathbb{\Pi}_{KB}$ such that $\alpha \in KB'$ but $\alpha \notin KB_1!$, while KB' is coherent and consistent all the same. That is, there exists a dependency cluster or a data cluster where the removal is not optimal, since α could be included in the consolidation. For the rest of the proof and for simplicity reasons, we consider the case where α belongs to a dependency cluster. This is made without loss of generality, since the proof for the case where α is included in a data cluster is analogous to the one presented here.

  Let us consider then $\alpha \in \mathbb{\Pi}_{KB}$ such that $\alpha \in KB'$. By Corollary 1 we have that $\alpha \in X$ where $X \in \mathbb{\Pi}_{KB}$. Let $T = (X \cap \rho(KB))$ be the incision performed over the cluster, and let $R = (X \cap \{KB \setminus KB'\})$ be those formulas removed from X when obtaining KB'. Clearly, since KB' is coherent then for all $Y \subseteq X$ where $Y \in \prod_{KB}$ it

holds that $R \cap Y \neq \emptyset$, because otherwise $Y \subset KB'$, which will make KB' incoherent. Besides, since $R \subseteq Y$ then $R \subseteq \text{Ⅲ}_{KB}$, and thus R satisfies the first two conditions in Definition 12.

By Definition 12 we have that T is such that there not exists a set of TGDs that satisfies the first two conditions in the definition and at the same time it holds that (1) $T \subset R$.

Since $\alpha \notin KB_1!$ and $\alpha \in X$ then $\alpha \in \rho(KB)$, and thus $\alpha \in T$. However, we know that $\alpha \in X$ and $\alpha \in KB'$, and thus $\alpha \notin R$. Therefore we have that (2) $T \not\subset R$.

From (1) and (2) we have that $T \subset R$ and that $T \not\subset R$, an absurd coming from our original assumption that $KB_1! \subset KB'$, and it holds that if $KB' \subseteq KB_1$ is coherent and consistent then $KB_1! \not\subset KB'$.

$\Leftarrow$) *Postulates to Construction*

For the second part of the proof, consider an operator ! that satisfies all postulates in Theorem 1. Let $\rho_{(!)_\Sigma}$ be a function based on ! defined as follows:

$$\rho_{(!)_\Sigma}(KB_1) = \{x \,|\, x \in X \text{ for some } X \in \text{Ⅲ}_{KB_1} \text{ and } x \notin \{\Sigma_1 \cap KB_1!\}\}$$

Let $KB_1^\star = (D_1, \Sigma_1 \setminus \rho_{(!)_\Sigma}(KB_1))$ be the ontology resulting of removing from KB_1 the TGDs selected by $\rho_{(!)_\Sigma}$. Then, let $\varrho_{(!)_D}$ be another function based on ! defined as follows:

$$\varrho_{(!)_D}(KB_1^\star) = \{x \,|\, x \in X \text{ for some } X \in \text{Ⅲ}_{KB_1^\star} \text{ and } x \notin \{D_1 \cap KB_1!\}\}$$

Based on $\varrho_{(!)_D}$ and $\rho_{(!)_\Sigma}$ we define a new operator as follows:

$$KB_1!' = (D_1 \setminus \varrho_{(!)_D}(KB_1^\star), \Sigma_1 \setminus \rho_{(!)_\Sigma}(KB_1))$$

We have to show that $!'$ is a *Datalog*$^\pm$ *ontology consolidation operator based on Cluster Contraction*. To do this, we first prove that $\varrho_{(!)_D}$ is a well-defined *data incision function* and that $\rho_{(!)_\Sigma}$ is a well-defined *constraint incision function*. That is, given $\rho_{(!)_\Sigma}$ we have to prove that:

- $\rho_{(!)_\Sigma}$ is well-defined, i.e., if $KB_1 = KB_2$, then $\rho_{(!)_\Sigma}(KB_1) = \rho_{(!)_\Sigma}(KB_2)$.

  By definition of $\rho_{(!)_\Sigma}$ we have that $\rho_{(!)_\Sigma}(KB_1) = \{x \,|\, x \in X \text{ for some } X \in \text{Ⅲ}_{KB_1} \text{ and } x \notin \Sigma_1 \cap KB_1!\}$.

  Consider any arbitrary $x \in \rho_{(!)_\Sigma}(KB_1)$. Since $KB_1 = KB_2$, then by Lemma 2 we have that $\text{Ⅲ}_{KB_1} = \text{Ⅲ}_{KB_2}$. Since $x \in \rho_{(!)_\Sigma}(KB_1)$, then $x \in X \in \text{Ⅲ}_{KB_1}$, and thus it holds that $x \in X \in \text{Ⅲ}_{KB_2}$(1).

  Besides, since $x \in X \in \text{Ⅲ}_{KB_1}$ then $x \in \Sigma_1$. Thus, since $x \notin \Sigma_1 \cap KB_1!$, then $x \notin KB_1!$. Since $KB_1 = KB_2$, from the fact that ! is a function we have that $KB_1! = KB_2!$, and then it also holds that $x \notin KB_2!$. Thus, $x \notin \Sigma_2 \cap KB_2!$(2).

  From (1) and (2) it follows that for any $x \in \rho_{(!)_\Sigma}(KB_1)$ it holds that $x \in \{y \,|\, y \in Y \text{ for some } Y \in \text{Ⅲ}_{KB_2} \text{ and } y \notin \Sigma_2 \cap KB_2!\}$. By definition of $\rho_{(!)_\Sigma}$ this is $\rho_{(!)_\Sigma}(KB_2)$, and thus we have that if $KB_1 = KB_2$, then $\rho_{(!)_\Sigma}(KB_1) = \rho_{(!)_\Sigma}(KB_2)$.

- $\rho_{(!)_\Sigma}(KB_1) \subseteq \bigcup(\text{Ⅲ}_{KB_1})$.

  This follows directly from the definition of $\rho_{(!)_\Sigma}$, since for every $x \in \rho_{(!)_\Sigma}(KB_1)$ it holds that $x \in X$ for some $X \in \text{Ⅲ}_{KB_1}$ because of the first condition in the definition.

- If $X \in \text{Ⅲ}_{KB_1}$ e $Y \in \prod_{KB_1}$ are such that $Y \neq \emptyset$ and $Y \subseteq X$, then $(Y \cap \rho_{(!)_\Sigma}(KB_1)) \neq \emptyset$.

  Suppose *by reductio* that there exists some $X \in \text{Ⅲ}_{KB_1}$ and $Y \in \prod_{KB_1}$ such that $Y \neq \emptyset$, $Y \subseteq X$ and $(Y \bigcap \rho_{(!)_\Sigma}(KB_1)) = \emptyset$.

  Then, for all $\alpha \in Y$ it holds that $\alpha \notin \rho_{(!)_\Sigma}(KB_1)$, i.e., $\alpha \notin X \in \text{Ⅲ}_{KB_1}$ or $\alpha \in \{\Sigma_1 \cap KB_1!\}$. By our hypothesis we have that $\alpha \in Y \in \prod_{KB_1}$ and $Y \subseteq X$. Thus $\alpha \in X$, and therefore it must hold that $\alpha \in \{\Sigma_1 \cap KB_1!\}$, and by extension $\alpha \in KB_1!$.

  Since this holds for any arbitrary $\alpha \in Y$ then we have that $Y \subset \Sigma_1!$. From Definition 6 it holds that Y is a minimal unsatisfiable set of TGDs *w.r.t.* $\Sigma_{_E} \cup \Sigma_{_{NC}} \subset \Sigma_1$. Then, for any relevant set of atoms A it holds that $\text{mods}(A, Y \cup \Sigma_{_E} \cup \Sigma_{_{NC}}) = \emptyset$. Then, since $Y \subset \Sigma_1!$ then for any relevant set A' it holds that $\text{mods}(A', \Sigma_1! \cup \Sigma_{_E} \cup \Sigma_{_{NC}}) = \emptyset$, because the TGDs in Y are triggered by A'. Then, $\Sigma_1!$ is an unsatisfiable set of TGDs *w.r.t.* $\Sigma_{_E} \cup \Sigma_{_{NC}} \subset \Sigma_1$.

  However, from **Coherence** we have that $KB_1!$ is coherent, and thus $\Sigma_1!$ is satisfiable *w.r.t.* $\Sigma_{_E} \cup \Sigma_{_{NC}} \subset \Sigma_1$.

  Then we have that $\Sigma_1!$ is satisfiable *w.r.t.* $\Sigma_{_E} \cup \Sigma_{_{NC}} \subset \Sigma_1$ and $\Sigma_1!$ is unsatisfiable *w.r.t.* $\Sigma_{_E} \cup \Sigma_{_{NC}} \subset \Sigma_1$, an absurd coming from our initial supposition that there exists some $X \in \text{Ⅲ}_{KB_1}$ and $Y \in \prod_{KB_1}$ such that $Y \neq \emptyset$, $Y \subseteq X$ and $(Y \bigcap \rho_{(!)_\Sigma}(KB_1)) = \emptyset$, and it holds that for all $X \in \text{Ⅲ}_{KB_1}$ and $Y \in \prod_{KB_1}$ such that $Y \subseteq X$, if $Y \neq \emptyset$ then $(Y \bigcap \rho_{(!)_\Sigma}(KB_1)) \neq \emptyset$.

- For all $X \in \text{Ⅲ}_{KB_1}$ it holds that $T = (X \cap \rho_{(!)_\Sigma}(KB_1))$ is such that there not exists $R \subset X$ where R satisfies the two previous conditions and $R \subsetneq T$.

  To prove this is sufficient to show that, being clusters disjoint sets, the election in each cluster is optimal, because otherwise if should exists any cluster where the incision function does not choose in an optimal way then **Minimality** would not be satisfied. So, suppose *by reductio* that there exists $X \in \text{Ⅲ}_{KB_1}$ where $T = (X \cap \rho_{(!)_\Sigma}(KB_1))$ is such that there does exist $R \subset X$ where R satisfies the two previous conditions and $R \subsetneq T$.

  Let us consider $KB' = (\Sigma', D')$ such that for all $Y \in \text{Ⅲ}_{KB_1}$ such that $Y \neq X$ it holds that $T' = (Y \cap \rho_{(!)_\Sigma}(KB_1))$ and $R' = (Y \cap \{KB \setminus KB'\})$ (those formulas removed from Y when obtaining KB') are such that $T' = R'$. Since $T' = R'$ then R' is such that the two conditions in Definition 12 are satisfied. Besides, let $\mathcal{CF}_{\Sigma_1} = \Sigma_1 \setminus \text{Ⅲ}_{KB}$ and $\mathcal{CF}_{D_1} = D_1 \setminus \text{Ⅲ}_{KB}$ be the set of formulas that do not belong to any kernel in Σ_1 and D_1, respectively; and let KB' be such that $\mathcal{CF}_{\Sigma_1} \subset \Sigma'$ and $\mathcal{CF}_{D_1} \subset D'$.

  The fact that every formula that is not in any conflict belongs to KB' and that KB' is built in such a way that the election in each cluster different than X is the same both in KB' and $\rho_{(!)_\Sigma}(KB_1)$ makes that $KB' \setminus (X \cap \{KB \setminus KB'\}) = KB_1! \setminus (X \cap \rho_{(!)_\Sigma}(KB_1))$. This is, if there is any difference between KB' and $KB_1!$ that difference arise from the election on which formulas to remove from X.

Finally, from our supposition we have that there exists $R \subset X$ where R satisfies the two previous conditions and $R \subsetneq T$. Let KB' and $R \subsetneq T$ be such that $R = (X \cap \{KB \setminus KB'\})$ is the set of formulas removed from X when obtaining KB'. Then, we have that KB' is coherent and consistent, since every conflict in clusters in KB_1 where solved, whether by removing R (for cluster X) or the sets R' (for every cluster different than X). Besides, since we have that $KB' \setminus (X \cap \{KB \setminus KB'\}) = KB_1! \setminus (X \cap \rho_{(!)_\Sigma}(KB_1))$ and that $R \subsetneq T$, then for $KB_1! = KB_1 \setminus \rho_{(!)_\Sigma}(KB_1)$ and $KB' = KB_1 \setminus \{\bigcup_{Y \in \{\amalg_{KB_1} \setminus X\}} R' \cup R\}$ where $(R' = Y \cap \{KB \setminus KB'\})$ and $R = (X \cap \{KB \setminus KB'\})$ it holds that $KB_1! \subset KB'$ (1). That is, if all formulas that are not involved in conflicts belong to both $KB_1!$ and KB', in each cluster different than X the same formulas are removed, and the set of formulas removed from X to obtain KB' are an strict subset of those removed by $\rho_{(!)_\Sigma}(KB_1)$ to obtain $KB_1!$, then $KB_1!$ is an strict subset of KB', i.e., we have removed more formulas by deleting T than by deleting R.

On the other hand, since KB' is coherent and consistent, then by **Minimality** we have that $KB_1! \not\subset KB'$ (2).

Therefore, from (1) and (2) we have that $KB_1! \subset KB'$ and $KB_1! \not\subset KB'$, and absurd coming from our initial supposition that there exists $X \in \amalg_{KB_1}$ where $T = (X \cap \rho_{(!)_\Sigma}(KB_1))$ is such that there exists $R \subset X$ where R satisfies the two previous conditions and $R \subsetneq T$, and we have that for all $X \in \amalg_{KB_1}$ it holds that $T = (X \cap \rho_{(!)_\Sigma}(KB_1))$ is such that there not exists $R \subset X$ where R satisfies the two previous conditions and $R \subsetneq T$.

We omit the proof that $\varrho_{(!)_D}$ is a well-defined *data incision function* using **Consistency** and **Minimality** since it is analogous to the proof that $\rho_{(!)_\Sigma}$ is a well-defined *constraint incision function* using **Coherence** and **Minimality**.

Now that we have shown that $\varrho_{(!)_D}$ and $\rho_{(!)_\Sigma}$ are well-defined *data incision functions* and *constraint incision functions*, respectively, to conclude this second part of the proof we have to show that $!'$ coincides with $!$. From **Inclusion** it follows that $D_1! \subseteq D_1$ and $\Sigma_1! \subseteq \Sigma_1$ (1). Also, from our definition of $\varrho_{(!)_D}$ it follows that $\varrho_{(!)_D}(KB_1^\star) = D_1 \setminus D_1!$, and from our definition of $\rho_{(!)_\Sigma}$ it follows that $\rho_{(!)_\Sigma}(KB_1) = \Sigma_1 \setminus \Sigma_1!$ (2). Then, from (1) and (2) we have that $D_1! = D_1 \setminus \varrho_{(!)_D}(KB_1^\star)$ and $\Sigma_1! = \Sigma_1 \setminus \rho_{(!)_\Sigma}(KB_1)$. Thus, $! = (D_1 \setminus \varrho_{(!)_D}(KB_1^\star), \Sigma_1 \setminus \rho_{(!)_\Sigma}(KB_1))$, and therefore $!'$ coincides with $!$. ∎

References

Alchourrón, C., Gärdenfors, P., & Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, *50*(2), 510–530.

Alchourrón, C., & Makinson, D. (1981). Hierarchies of Regulation and Their Logic. *New Studies in Deontic Logic*, 125–148.

Alchourrón, C., & Makinson, D. (1985). On the Logic of Theory Change: Safe Contraction. *Studia Logica*, *44*, 405–422.

Amgoud, L., & Kaci, S. (2005). An argumentation framework for merging conflicting knowledge bases: The prioritized case. In *Proc. of 8th European Conferences on Symbolic*

and Quantitative Approaches to Reasoning with Uncertainty (ECSQUARU '05), pp. 527–538.

Andritsos, P., Fuxman, A., & Miller, R. J. (2006). Clean answers over dirty databases: A probabilistic approach. In *Proc. of 22nd International Conference on Data Engineering (ICDE '06)*, p. 30.

Arenas, M., Bertossi, L. E., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. In *Proc. of 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '99)*, pp. 68–79.

Arieli, O., Denecker, M., & Bruynooghe, M. (2007). Distance semantics for database repair. *Annals of Mathematics and Artificial Intelligence*, *50*(3-4), 389–415.

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the EL envelope. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI '05)*, pp. 364–369.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baral, C., Kraus, S., & Minker, J. (1991). Combining multiple knowledge bases. *Transactions on Knowledge and Data Engineering*, *3*(2), 208–220.

Bell, D. A., Qi, G., & Liu, W. (2007). Approaches to inconsistency handling in description-logic based ontologies. In *Proc. of On the Move to Meaningful Internet Systems (OTM) Workshops (2)*, pp. 1303–1311.

Beneventano, D., & Bergamaschi, S. (1997). Incoherence and subsumption for recursive views and queries in object-oriented data models. *Data and Knowledge Engineering*, *21*(3), 217–252.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284(5):3443*.

Bienvenu, M., & Rosati, R. (2013). Tractable approximations of consistent query answering for robust ontology-based data access. In *Proc. of 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*, pp. 775–781.

Black, E., Hunter, A., & Pan, J. Z. (2009). An argument-based approach to using multiple ontologies. In *Proc. of 3rd International Conference on Scalable Uncertainty Management (SUM '09)*, pp. 68–79.

Bohannon, P., Flaster, M., Fan, W., & Rastogi, R. (2005). A cost-based model and effective heuristic for repairing constraints by value modification. In *Proc. of 24th ACM SIGMOD International Conference on Management of Data / Principles of Database Systems (PODS '05)*, pp. 143–154.

Booth, R., Meyer, T. A., Varzinczak, I. J., & Wassermann, R. (2010). Horn belief change: A contraction core. In *Proc. of 19th European Conference on Artificial Intelligence (ECAI '10)*, pp. 1065–1066.

Borgida, A. (1995). Description logics in data management. *Transactions on Knowledge and Data Engineering*, *7*(5), 671–682.

Brandt, S. (2004). Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else?. In *Proc. of 16th European Conference on Artificial Intelligence (ECAI '04)*, pp. 298–302.

Calì, A., Gottlob, G., & Kifer, M. (2008). Taming the infinite chase: Query answering under expressive relational constraints. In Brewka, G., & Lang, J. (Eds.), *Proc. of 11th International Conference on Principles of Knowledge Representation and Reasoning (KR '08)*, pp. 70–80. AAAI Press.

Calì, A., Gottlob, G., & Kifer, M. (2013). Taming the infinite chase: Query answering under expressive relational constraints. *Journal of Artificial Intelligence Research*, *48*, 115–174.

Calì, A., Gottlob, G., & Lukasiewicz, T. (2012). A general Datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantic*, *14*, 57–83.

Calì, A., Lembo, D., & Rosati, R. (2003). On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of 22nd ACM SIGMOD Symposium on Principles of database systems (PODS '03)*, pp. 260–271. ACM.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2005). DL-Lite: Tractable description logics for ontologies. In *AAAI*, pp. 602–607.

Caroprese, L., Greco, S., & Zumpano, E. (2009). Active integrity constraints for database consistency maintenance. *Transactions on Knowledge and Data Engineering*, *21*(7), 1042–1058.

Caroprese, L., & Truszczynski, M. (2011). Active integrity constraints and revision programming. *Theory and Practice of Logic Programming*, *11*(6), 905–952.

Cecchi, L., Fillottrani, P., & Simari, G. R. (2006). On the Complexity of DeLP through Game Semantics. In Dix, J., & Hunter, A. (Eds.), *Proc. of 11th International Workshop on Non-Monotonic Reasoning (NMR '06)*, pp. 386–394.

Cholvy, L. (1998). Reasoning about merged information. In *Belief Change*, Vol. 3, pp. 233–263. Springer Netherlands.

Croitoru, M., & Rodriguez, R. O. (2015). Using kernel consolidation for query answering in inconsistent OBDA. In *Proc. of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology*.

Delgrande, J. P. (2011). Revising by an inconsistent set of formulas. In *Proc. of 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 833–838.

Delgrande, J. P., Dubois, D., & Lang, J. (2006). Iterated revision as prioritized merging. In *Proc. of 10th International Conference on Principles of Knowledge Representation and Reasoning (KR '06)*, pp. 210–220.

Delgrande, J. P., & Jin, Y. (2012). Parallel belief revision: Revising by sets of formulas. *Artificial Intelligence*, *176*(1), 2223–2245.

Delgrande, J. P., Schaub, T., Tompits, H., & Woltran, S. (2009). Merging logic programs under answer set semantics. In *Proc. of 25th International Conference on Logic Programming (ICLP '09)*, pp. 160–174.

Dunne, P., & Wooldridge, M. (2009). *Argumentation in Artificial Intelligence*, chap. Complexity of Abstract Argumentation, pp. 85–104. Springer.

Everaere, P., Konieczny, S., & Marquis, P. (2008). Conflict-based merging operators. In *Proc. of 11th International Conference on Principles of Knowledge Representation and Reasoning (KR '08)*, pp. 348–357.

Falappa, M. A., Kern-Isberner, G., Reis, M. D. L., & Simari, G. R. (2012). Prioritized and non-prioritized multiple change on belief bases. *Journal of Philosophical Logic*, *41*(1), 77–113.

Falappa, M. A., Kern-Isberner, G., & Simari, G. R. (2002). Belief Revision, Explanations and Defeasible Reasoning. *Artificial Intelligence*, *141*, 1–28.

Flouris, G., Huang, Z., Pan, J. Z., Plexousakis, D., & Wache, H. (2006). Inconsistencies, negations and changes in ontologies. In *Proc. of 21st National Conference on Artificial Intelligence (AAAI '06)*, pp. 1295–1300.

Friedman, N., & Halpern, J. Y. (2001). Belief revision: A critique. *Computer Research Repository (CoRR)*, *cs.AI/0103020*.

Fuhrmann, A. (1991). Theory contraction through base contraction. *Journal of Philosophical Logic*, *20*, 175–203.

Gärdenfors, P. (1982). Rule for rational changes of belief. *Philosophical Essay Dediccated To Lennart Åqvist on his Fiftieth Birthday*, 88–101.

Gärdenfors, P. (1988). *Knowledge in Flux: Modeling the dynamics of epistemic states*. MIT Press.

Gelfond, M. (2008). Answer sets. In *Handbook of Knowledge Representation*, chap. 7, pp. 285–316. Elsevier.

Gómez, S. A., Chesñevar, C. I., & Simari, G. R. (2010). Reasoning with inconsistent ontologies through argumentation. *Applied Artificial Intelligence*, *24*(1&2), 102–148.

Greco, S., & Molinaro, C. (2012). Probabilistic query answering over inconsistent databases. *Annals of Mathematics and Artificial Intelligence (AMAI)*, *64*(2-3), 185–207.

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., & Sure, Y. (2005). A framework for handling inconsistency in changing ontologies. In *Proc. of 4th International Semantic Web Conference (ISWC '05)*, pp. 353–367.

Halaschek-Wiener, C., & Katz, Y. (2006). Belief base revision for expressive description logics. In *Proc. of International Workshop on OWL: Experiences and Directions (OWLED '06)*.

Hansson, S. O. (1991). *Belief Base Dynamics*. Ph.D. thesis, Uppsala University, Department of Philosophy, Uppsala, Sweden.

Hansson, S. O. (1993). Theory contraction and base contraction unified. *Journal of Symbolic Logic*, *58*(2), 602–625.

Hansson, S. O. (1994). Kernel contraction. *Journal of Symbolic Logic*, *59*(3), 845–859.

Hansson, S. O. (1997). Semi-revision. *Journal of Applied Non-Classical Logics*, *7*(1-2), 151–175.

Hansson, S. O. (2001). *A Textbook of Belief Dynamics: Solutions to Exercises.* Kluwer Academic Publishers, Norwell, MA, USA.

Harman, G. (2008). *Change in view: Principles of reasoning.* Cambridge University Press.

Harper, W. (1975). Rational Belief Change, Popper Functions and Counterfactuals. *Synthese*, *30*, 221–262.

Horridge, M., Parsia, B., & Sattler, U. (2009). Explaining inconsistencies in OWL ontologies. In *Scalable Uncertainty Management*, pp. 124–137. Springer.

Huang, Z., van Harmelen, F., & ten Teije, A. (2005). Reasoning with inconsistent ontologies. In *Proc. of 19th International Joint Conference on Artificial Intelligence (IJCAI '05)*, pp. 454–459.

Hué, J., Papini, O., & Würbel, E. (2009). Merging belief bases represented by logic programs. In *Proc. of 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU '09)*, pp. 371–382.

Kalyanpur, A., Parsia, B., Horridge, M., & Sirin, E. (2007). *Finding all justifications of OWL DL entailments.* Springer.

Kalyanpur, A., Parsia, B., Sirin, E., & Hendler, J. A. (2005). Debugging unsatisfiable classes in owl ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, *3*(4), 268–293.

Katsuno, H., & Mendelzon, A. O. (1991). On the difference between updating a knowledge base and revising it. In *Proc. of 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 387–394.

Katsuno, H., & Mendelzon, A. O. (1992). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, *52*(3), 263–294.

Konieczny, S., & Pérez, R. P. (2002). Merging information under constraints: A logical framework. *Journal of Logic and Computation*, *12*(5), 773–808.

Konieczny, S., & Pérez, R. P. (2011). Logic based merging. *Journal of Philosophical Logic*, *40*(2), 239–270.

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2010). Inconsistency-tolerant semantics for description logics. In *Proc. of 4th International Conference on Web Reasoning and Rule Systems (RR '10)*, pp. 103–117.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proc, of 21st ACM SIGMOD Symposium on Principles of Database Systems (PODS '02)*, pp. 233–246.

Levi, I. (1977). Subjunctives, Dispositions and Chances. *Synthese*, *34*(4), 423–455.

Liberatore, P., & Schaerf, M. (1998). Arbitration (or how to merge knowledge bases). *Knowledge and Data Engineering*, *10*(1), 76–90.

Lin, J., & Mendelzon, A. O. (1998). Merging databases under constraints. *International Journal of Cooperative Information Systems*, *7*(1), 55–76.

Lin, J., & Mendelzon, A. O. (1999). Knowledge base merging by majority. *Applied Logic Series*, *18*, 195–218.

Lloyd, J. W. (1987). *Foundations of Logic Programmming*. Springer-Verlag.

Lukasiewicz, T., Martinez, M. V., & Simari, G. I. (2012). Inconsistency handling in datalog+/- ontologies. In *Proc. of 20th European Conference on Artificial Intelligence (ECAI '12)*, pp. 558–563.

Martinez, M., Pugliese, A., Simari, G., Subrahmanian, V., & Prade, H. (2007). How dirty is your relational database? an axiomatic approach. In Mellouli, K. (Ed.), *Proc. of 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU '07)*, Vol. 4724 of *Lecture Notes in Computer Science*, pp. 103–114. Springer.

Meyer, T., Lee, K., & Booth, R. (2005). Knowledge integration for description logics. In Veloso, M., & Kambhampati, S. (Eds.), *Proceedings of AAAI05, Twentieth National Conference on Artificial Intelligence*, pp. 645–650. AAAI Press.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, *18*, 87–127.

Nilsson, U., & Maluszynski, J. (1995). *Logic, Programming and Prolog (2ed)*. John Wiley & Sons Ltd.

Parsons, S., Wooldridge, M., & Amgoud, L. (2003). Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation*, *13*(3), 347–376.

Qi, G., & Hunter, A. (2007). Measuring incoherence in description logic-based ontologies. In *Proc. of 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC/ASWC '07)*, pp. 381–394.

Qi, G., Liu, W., & Bell, D. A. (2006). Knowledge base revision in description logics. In *Proc. of 10th European Conference in Logics in Artificial Intelligence (JELIA '06)*, pp. 386–398.

Quine, W. V. O. (1986). *Philosophy of logic*. Harvard University Press.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, *32(1)*, 57–95.

Rosati, R. (2011). On the complexity of dealing with inconsistency in description logic ontologies. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 1057–1062.

Rott, H. (1992). Modellings for belief change: Prioritization and entrenchment. *Theoria*, *58*(1), 21–57.

Schlobach, S., & Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies.. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 355–362.

Schlobach, S., Huang, Z., Cornet, R., & van Harmelen, F. (2007). Debugging incoherent terminologies. *Journal of Automated Reasoning*, *39*(3), 317–349.

Staworko, S., Chomicki, J., & Marcinkowski, J. (2012). Prioritized repairing and consistent query answering in relational databases. *Annals of Mathematics and Artificial Intelligence*, *64*(2-3), 209–246.

Turner, H. (2003). Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, *3*(4-5), 609–622.

von Leibniz, G. W. F. (1976). *Philosophical Papers and Letters: a selection*, Vol. 1. Springer.

Wassermann, R. (2000). An algorithm for belief revision. In *Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR '00)*, pp. 345–352.

Wijsen, J. (2005). Database repairing using updates. *ACM Transaction on Database Systems*, *30*(3), 722–768.