The Knowledge Engineering Review

RESEARCH ARTICLE

Open Access

https://doi.org/10.48130/ker-0026-0006

The Knowledge Engineering Review **2026**, 41: e005

Inconsistent databases and argumentation frameworks with collective attacks

Yasir Mahmood[1], Jonni Virtema[2,3], Timon Barlag[4*] and Axel-Cyrille Ngonga Ngomo[1]

[1] *Data Science Group, Heinz Nixdorf Institute, Paderborn University, Paderborn, Germany*
[2] *School of Computer Science, University of Sheffield, Sheffield, UK*
[3] *School of Computing Science, University of Glasgow, Glasgow, UK*
[4] *Institut für Theoretische Informatik, Leibniz Universität Hannover, Hannover, Germany*
* Correspondence: barlag@thi.uni-hannover.de (Barlag T)

Abstract

The connection between subset-maximal repairs for inconsistent databases involving various integrity constraints (ICs) and acceptable sets of arguments within argumentation frameworks (AFs) has recently drawn growing interest. In this paper, we contribute to this domain by establishing a new connection when ICs include denial constraints and local-as-view tuple-generating dependencies. It turns out that set-based AFs (SETAFs), an extension of Dung's AFs allowing collective attacks, is needed. It is known that subset-maximal repairs under denial constraints correspond to the naive extensions, which also coincide with the preferred and stable extensions in the resulting SETAFs. Our main findings establish that repairs under the considered fragment of tuple-generating dependencies correspond to the preferred extensions. Moreover, for these dependencies, an additional pre-processing allows us to compute a unique extension that is stable and naive. Allowing both types of constraints breaks this relationship, and even the pre-processing does not help, as only preferred semantics captures these repairs. Finally, while it is known that functional dependencies do not require set-based attacks, we prove the same regarding inclusion dependencies. Thus, one can translate inconsistent databases under these restricted classes of ICs to plain AFs with attacks only between arguments.

Citation: Mahmood Y, Virtema J, Barlag T, Ngomo ACN. 2026. Inconsistent databases and argumentation frameworks with collective attacks. *The Knowledge Engineering Review* 41: e005 https://doi.org/10.48130/ker-0026-0006

1 Introduction

In real-world applications, the provenance of data can be very diverse and include non-trustworthy sources. Thus, databases are often inconsistent in practice when the data do not conform to the imposed integrity constraints (ICs). A rich theory has been developed to deal with inconsistent databases.

Formally, ICs describe fundamental structural properties of data that should be satisfied for a database to be considered consistent. ICs are often formulated in some logic, most notably in fragments of first-order logic. Two of the most fundamental classes of ICs are key constraints (indicating that the value of a given collection of attributes uniquely determines the whole record in the database table) and inclusion dependencies (which express that all values for a given attribute occur as values of another given attribute). Together, key constraints and inclusion dependencies (IDs) can express foreign key constraints. The literature for database dependencies considers a whole hierarchy of dependencies in increasing generality (see, e.g., Abiteboul[1]). Within this hierarchy, key constraints are special cases of functional dependencies (FDs), which are special cases of equality-generating dependencies, which again generalize to denial constraints (DCs). Likewise, IDs generalize to local-as-view tuple-generating dependencies (LTGDs), which are special cases of tuple-generating dependencies (TGDs). All the aforementioned dependencies can be defined as particular syntactic fragments of first-order logic (see the section "Preliminaries" for formal definitions). In this paper, we examine expressive ICs, specifically DCs and LTGDs, alongside their less expressive counterparts, FDs and IDs.

One of the main approaches for handling inconsistency is *database repairing*. The goal is to identify and *repair* inconsistencies in data to obtain a consistent database that satisfies the imposed constraints. In the usual approaches, one would search for a database that satisfies the given constraints and differs minimally from the original database; the obtained database is called a *repair* of the original. Some of the most prominent notions of repairs are set-based repairs[2,3], attribute-based repairs[4], and cardinality-based repairs[5].

Dung's abstract argumentation framework (AF)[6] has been specifically designed to model conflicts and support relationships among arguments. An abstract AF represents arguments and their conflicts through directed graphs and allows for a convenient exploration of the conflicts at an abstract level. Furthermore, it is often necessary to consider conflicts not only between individual arguments but also between sets of arguments. This necessity has given rise to the so-called *set-based AFs* (SETAFs)[7], which allow modeling of collective attacks.

AFs have been explored extensively for representing and reasoning with inconsistent knowledge bases (KBs), covering Datalog$^{\pm}$ and description logics (see, e.g., studies[8–13] and Arieli[14] for an overview). The common goal in each of these works is to formally establish a connection between inconsistent KBs and AFs such that the argumentation machinery then outputs extensions equivalent to the set of repairs of the KB. Nevertheless, in the setting of relational databases and ICs, there is still a gap with respect to how or whether a connection between inconsistent databases and AFs can be established. To the best of our knowledge, only FDs (or their generalization, denial constraints) have been investigated in the context of AFs (resp., SETAFs) in Bienvenu & Bourgaux[13]. We expand this area of research by establishing further connections between repairs and abstract argumentation frameworks when further ICs are allowed.

In this paper, we focus on subset repairs of relational databases when the ICs are DCs and LTGDs. The first problem of interest to us is establishing a connection between repairs for the inconsistent

© The Author(s)

databases (DBs) and extensions in SETAFs. We show how subset-maximal repairs under a set of DCs and LTGDs can be obtained by computing the naive, preferred, or stable extensions (see the section "Preliminaries" for definitions) in the related SETAFs. Repairs under DCs correspond to the naive extensions, which also coincide with the preferred and stable extensions in the resulting SETAFs. For LTGDs, the correspondence requires preferred semantics. However, a polynomial-time pre-processing on the resulting SETAFs results in its unique extension that is also stable and naive. Allowing both types of ICs breaks this relationship between extensions, and only preferred semantics captures the repairs. Our second contribution highlights that when one restricts DCs to FDs and LTGDs to IDs, respectively, a similar connection as for the expressive ICs holds; however, now the resulting SETAFs are in fact AFs (with attacks only between arguments, rather than collective attacks). We are also interested in the computational problems of deciding the existence of a repair and of determining whether a given fact belongs to some (or every) repair and consider their complexity. See Table 1 for the complexity results.

By employing Dung's AFs (and their generalizations, SETAFs) to model repairs of a relational database, one can effectively abstract away from the detailed content of individual entries in the database and focus solely on their relationships with other entries. This approach provides a clearer understanding of why specific records either appear or do not appear in a repair, as well as the reasons certain values may be absent from query answers. Furthermore, this modeling approach allows for the incorporation of additional information about records, such as priorities among them, directly at an abstract level.

Related work The problem of computing subset-maximal repairs and its complexity has been explored extensively in the database setting[15–20] (see studies[21,22] for an overview). The notions of conflict graphs and hypergraphs have been introduced before for certain families of ICs[23–25]. In particular, a correspondence between repairs and subset-maximal independent sets of the conflict graphs for FDs[16] and DCs[17] has been established. Moreover, a recent work defines conflict (hyper)graphs for a richer setting of universal constraints considering symmetric difference repairs[26]. Notice that the connection between repairs and independent sets also yields a correspondence between repairs and the naive extensions when the conflict graph is seen as an AF. Nevertheless, to the best of our knowledge, no work has considered a similar graph representation when LTGDs (or even IDs) are taken into account. Hannula and Wijsen[18] addressed the problem of consistent query answering with respect to primary and foreign keys. Their setting allows the insertion of new tuples to fulfill foreign key constraints rather than only deleting.

The case of unirelational databases is also connected to the team-semantics literature[27]. Team semantics is a logical framework where formulae are evaluated over unirelational databases (teams in their terminology). In this setting, the complexity of finding maximal satisfying subteams has been studied by Hannula & Hella[28] for inclusion logic formulas and by Mahmood[29] for propositional dependence logic. In the team-semantics literature, FDs are known as *dependence* atoms and IDs as *inclusion* atoms, denoted, respectively, as $\mathrm{dep}(\boldsymbol{x};\boldsymbol{y})$ and $\boldsymbol{x} \subseteq \boldsymbol{y}$ (We borrow this notation and write $\mathrm{dep}(\boldsymbol{x};\boldsymbol{y})$ and $\boldsymbol{x} \subseteq \boldsymbol{y}$ for FDs and IDs, respectively).

Our work differs from the previous work, since it combines denial constraints (a subclass of universal constraints) and LTGDs (a subclass of tuple-generating dependencies). Moreover, one of our main contributions lies in connecting repairs under DCs (resp., FDs) and LTGDs (IDs) to the extensions of set-based (plain) AFs in Dung's setting[6]. A similar connection between (SET)AFs and preferred repairs has been explored in the context of prioritized description logic[13] and Datalog$^{\pm}$ KBs[30–33]. Employing abstract argumentation, we utilize facts in a database as arguments, aligning with the approach presented in Bienvenu & Bourgaux[13]. In fact, the results for DCs and FDs regarding preferred and stable semantics follow as a corollary from the work by Bienvenu & Bourgaux[13] since repairs in the prioritized setting generalize subset repairs. In contrast, Croitoru et al.[31,32] take an orthogonal approach in the datalog setting, employing structured argumentation to construct arguments from a given KB.

Regarding the expressivity comparison between various formalisms, König el al.[34] provided several translations between different argumentation formalisms, namely assumption-based argumentation (ABA)[35], claim-augmented frameworks (CAF)[36], and argumentation frameworks with collective attacks (SETAF)[7]. This puts our work in a broader context by presenting novel translations from databases with expressive ICs to SETAFs. Furthermore, recently, Mahmood et al.[37] presented translations from AFs to inconsistent databases, which is the converse direction to what we aimed to achieve in this paper. Their main findings indicate that one can translate argument interaction in an AF via FDs and IDs alone.

Prior work A preliminary version of this work was published in FoIKS 2024[38]. The current paper expands the earlier work to more

Table 1. Overview of our main contributions.

Equivalence between DBs with **expressive ICs** and SETAFs, along with **data complexity** results.

ICs	SETAF-equivalent semantics for repairs	Complexity results		
		REP_B	$\exists$-REP_B	$\forall$-REP_B
DCs	$\sigma \in \{\mathrm{naive}, \mathrm{pref}, \mathrm{stab}\}^\star$	(trivial)	(trivial)	$\in \mathbf{P}$
LTGDs	pref (Thm. 3.9)	$\in \mathbf{P}$[15]	$\in \mathbf{P}$[15]	$\in \mathbf{P}$[15]
DCs+LTGDs	pref (Thm. 3.13)	**NP** (Thm. 3.14)	**NP** (Thm. 3.14)	$\mathbf{\Pi}_2^{\mathbf{P}}$ (Thm. 3.15)

Equivalence between DBs with **less expressive ICs** and AFs, along with **combined complexity** results.

ICs	AF-equivalent semantics for repairs	Complexity Results		
		REP	$\exists$-REP	$\forall$-REP
FDs	$\sigma \in \{\mathrm{naive}, \mathrm{pref}, \mathrm{stab}\}^\star$	(trivial)	(trivial)	$\in \mathbf{P}$
IDs	pref (Cor. 4.7)	$\in \mathbf{P}$[15]	$\in \mathbf{P}$[15]	$\in \mathbf{P}$[15]
FDs+IDs	pref (Cor. 4.12)	**NP** (Thm. 4.13)	**NP** (Thm. 4.13)	$\mathbf{\Pi}_2^{\mathbf{P}}$ (Thm. 4.16)

The table at the top indicates results for expressive ICs involving a fixed set B of DCs and LTGDs, whereas the table at the bottom depicts results for less expressive ICs including FDs and IDs. The lower bounds for less expressive ICs also hold for data complexity since the involved reductions use fixed sets of constraints. The second column in each table indicates the (SET)AF-semantics corresponding to repairs for ICs in the first column, and the last three columns present the complexity of each problem. The **P**-results are already known in the literature, whereas the remaining results are new. Finally, the results marked by * follow from the earlier work in studies[13,17].

expressive ICs. To be precise, we utilize SETAFs to simulate denial constraints and LTGDs using the same ideas as for FDs and IDs. The conference version focused on FDs and IDs in connection to plain AFs. The results for DCs follow from the work by Bienvenu & Bourgaux[13]; thus, the task is to extend them to LTGDs. The results of the conference version can now be derived as special cases from the more general results of this paper, when the involved ICs are FDs and IDs.

2 Preliminaries

We assume that the reader is familiar with basics of complexity theory and first-order logic. In particular, we will encounter (1) the complexity classes $\mathbf{P}, \mathbf{NP}$, and $\mathbf{\Pi}_2^{\mathbf{P}}$, and (2) closed first-order formulas having $\forall.\exists.$ as the quantifier prefixes. In the following, we shortly recall the necessary definitions from databases and argumentation.

Databases A (relational) *schema* τ is a finite set of relation names. Each relation $T \in \tau$ is associated with a positive integer, called the *arity* of T. We consider a countable set C of constants to act as domain elements, from which the values can appear in a database. A *term* is either a constant from C or a variable. Let $t_1, \ldots, t_k$ be terms and $T \in \tau$ be a relation of arity k, the expression $T(t_1, \ldots, t_k)$ is called a *relational atom*. A relational atom composed of only constants is called a *fact*. Given a schema τ, a *database* $\mathcal{T}$ over τ is a finite set of facts using relation symbols from τ. The *active domain* of a database $\mathcal{T}$ is denoted as $\text{dom}(\mathcal{T})$ and defined as the collection of all the constant values from C that occur in facts of $\mathcal{T}$.

A fact can also be seen as a database tuple. In this notation, the fact $T(a_1, \ldots, a_k)$ corresponds to stating that the tuple $(a_1, \ldots, a_k)$ is a record in the *(database) table* T of arity k—which is a collection of such tuples, each with an associated unique identifier. For analogy, a table corresponds to a relation name T in a schema τ and can be seen as a grouping of facts together that use the same relation symbol (namely, T). For a table T of arity k, the positions $\{1, \ldots, k\}$ in the columns of T are usually given explicit names, resulting in the set $\text{att}(T)$ of attributes of T. The active domain $\text{dom}(T)$ of T in the tabular view is the collection of all the values that occur in the tuples of T. We denote individual attributes (or variables) by small letters (e.g., x, y) and reserve boldface letters (e.g., $\boldsymbol{x}, \boldsymbol{y}$) for sequences of attributes. For an attribute $x \in \text{att}(T)$ and a tuple $s \in T$, $s(x)$ denotes the value taken by s for the attribute x. For a sequence $\boldsymbol{x} = (x_1, \ldots, x_k)$, $s(\boldsymbol{x})$ denotes the sequence of values $(s(x_1), \ldots, s(x_k))$. Given this tabular view, a database over schema $\tau = \{T_1, \ldots, T_m\}$ is seen as a collection $\mathcal{T} = (T_1, \ldots, T_m)$ of tables corresponding to relation symbols in τ. Again, the active domain $\text{dom}(\mathcal{T})$ of a database $\mathcal{T}$ is the union of active domains of each table $T \in \mathcal{T}$. We find it convenient to write a database without explicitly mentioning its schema and active domain always.

We next present an example database.

Example 2.1 *Consider the schema* $\{E, D\}$. *The database tables over this schema (Table 2) contain data about employees and departments, respectively. Here,* E*(E1, D1, Paderborn) is a fact and* (*E1, D1, Paderborn*) *is a tuple in* E *with identifier* e_1. *For presentation of examples, we refer to tuple identifiers also as facts. Hence, we write* e_1 *for the fact* E*(E1, D1, Paderborn*). ◁

Integrity constraints. ICs are closed first-order formulas, i.e., all the variables are bound to quantification. In the following, we first specify different types of ICs.

Let τ be a database schema. Recall that a database atom is an application of any relation name $T \in \tau$ over variables or constants. Let φ and ψ_i be conjunctions of database atoms and let β be a quantifier-free formula using only built-in predicates (e.g., equality or inequality). We call a formula of the form

$$\forall \boldsymbol{x}(\varphi(\boldsymbol{x}) \wedge \beta(\boldsymbol{x}) \rightarrow \bigvee_{i=1}^{n} \exists \boldsymbol{y}_i \psi_i(\boldsymbol{x}, \boldsymbol{y}_i))$$

a generic *first-order IC* (FO). All other types of constraint languages arise from restrictions on the generic ICs, as we illustrate next. A generic IC is called *full* or a *universal constraint* (UC) if it contains no existential quantifiers. A *DC* has the form $\forall \boldsymbol{x} \neg(\varphi(\boldsymbol{x}) \wedge \beta(\boldsymbol{x}))$, which can also be thought of as a UC with empty right hand side. An *equality-generating dependency* (EGD) has the form $\forall \boldsymbol{x}(\varphi(\boldsymbol{x}) \rightarrow x_i = x_j)$, which can be thought of as a DC where β is a single inequality. Let T be an n-ary relation and $I, J \subseteq \{1, \ldots, n\}$ be two sets with pairwise distinct variables $x_1, \ldots, x_n, y_1, \ldots, y_n$. Then, an FD over T is an EGD of the form $\forall \boldsymbol{x} \forall \boldsymbol{y}(T(\boldsymbol{x}) \wedge T(\boldsymbol{y}) \wedge \bigwedge_{i \in I} x_i = y_i \rightarrow \bigwedge_{j \in J} x_j = y_j)$. A *key constraint* is a special type of FD requiring $I \cup J = \{1, \ldots, n\}$.

A *disjunctive tuple-generating dependency* ($\vee$-TGD) is a generic IC that has empty β and an ordinary *tuple-generating dependency* (TGD) additionally requires $n = 1$. An LTGD is a TGD where φ is a single atom and an ID is an LTGD where ψ_i is also a single atom. Figure 1 (adapted from study by Arming[39]) presents an overview of the hierarchy of different ICs together with their syntactic forms.

The semantics for ICs is defined similarly to first-order (FO) formulas. A database $\mathcal{T}$ over schema τ gives rise to an FO-structure $\mathcal{A}$ over the vocabulary τ. The domain of $\mathcal{A}$ is the active domain $\text{dom}(\mathcal{T})$ of $\mathcal{T}$ and each relation symbol $T \in \tau$ is interpreted as the corresponding database table T in $\mathcal{T}$, i.e., it contains all the tuples of constants $(a_1, \ldots, a_k)$ over $\text{dom}(\mathcal{T})$ such that $T(a_1, \ldots, a_k) \in \mathcal{T}$. Then, for an IC α, we write $\mathcal{T} \models \alpha$ iff $\mathcal{A} \models \alpha$ under the classical Tarski semantics. Moreover, one often uses a more convenient notation for FDs and IDs as we illustrate next. Let T be an n-ary relation, $I, J \subseteq \{1, \ldots, n\}$ be two sets and let $x_1, \ldots, x_n, y_1, \ldots, y_n$ be pairwise distinct variables. Then, for sequences $\boldsymbol{X} = (x_i \mid i \in I)$ and $\boldsymbol{Y} = (y_j \mid j \in J)$, the FD $\forall \boldsymbol{x} \forall \boldsymbol{y}(T(\boldsymbol{x}) \wedge T(\boldsymbol{y}) \wedge \bigwedge_{i \in I} x_i = y_i \rightarrow \bigwedge_{j \in J} x_j = y_j)$ can be equivalently expressed as $\text{dep}(\boldsymbol{X}; \boldsymbol{Y})$ or $\boldsymbol{X} \rightarrow \boldsymbol{Y}$ (as done usually in the DBs setting). Moreover, let T_i and T_j be two relations, and $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ be sequences of variables. The ID $\forall \boldsymbol{x}_1 \forall \boldsymbol{x}_2(T_i(\boldsymbol{x}_1, \boldsymbol{x}_2) \rightarrow \exists \boldsymbol{x}_3 T_j(\boldsymbol{x}_2, \boldsymbol{x}_3))$ can be equivalently expressed as $T_i[\boldsymbol{Y}] \subseteq T_j[\boldsymbol{X}]$ where $\boldsymbol{X}$ (respectively, $\boldsymbol{Y}$) is the set of attributes of T_j (T_i) corresponding to $\boldsymbol{x}_2$. We find it convenient to write $\boldsymbol{Y} \subseteq \boldsymbol{X}$ instead of $T[\boldsymbol{Y}] \subseteq T[\boldsymbol{X}]$ for an ID involving a single relation T. We will use this notation throughout our paper.

The following example depicts ICs over the database from Example 2.1.

Example 2.2 *Consider a DC* dc *and an LTGD* lav, *as given below.*

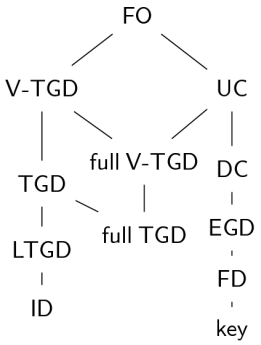
$$\text{dc} := \forall x_1, x_2, x_3, x_4, x_5 \neg(E(x_1, x_2, x_3) \wedge D(x_2, x_4, x_5) \wedge x_3 \neq x_5)$$

Table 2. A database with two tables over the schema $\{E, D\}$.

E	Emp_ID	Dept_ID	Location
e_1	E1	D1	Paderborn
e_2	E2	D2	Sheffield
e_3	E3	D2	Hanover

D	Dept_ID	Dept_NAME	Location
d_1	D1	Sales	Paderborn
d_2	D2	Marketing	Sheffield
d_3	D3	HR	Hanover

The first column indicates identifier for each fact/tuple.



Generic ICs (FO)	$\forall \boldsymbol{x}(\varphi(\boldsymbol{x}) \wedge \beta(\boldsymbol{x}) \rightarrow \bigvee_{i=1}^{n} \exists \boldsymbol{y}_i \psi_i(\boldsymbol{x}, \boldsymbol{y}_i))$
V-TGD	FO, where β is empty
TGD	V-TGD, where $n = 1$
LTGD	TGD, where φ is a single atom
ID	LTGD, where ψ_1 is a single atom
UC	FO with no existential quantifier
DC	UC, where ψ_i are empty $(n = 0)$
EGD	DC, where β is a single inequality
FD	DC, where φ uses one relation symbol

Fig. 1 Hierarchy of ICs and syntactic form for the most commonly studied constraints[39].

$$\text{lav} := \forall x_1, x_2, x_3 (D(x_1, x_2, x_3) \rightarrow \exists y_1, y_2 E(y_1, x_1, y_2))$$

The DC dc *states that the database should not contain any employees that work at a different location than where their department is located and the LTGD* lav *requires that every department has at least one employee. Both* dc *and* lav *are violated in the database from Example 2.1.* ◁

Let $\mathcal{T}$ be a database (a collection of facts) and B be a collection of ICs. Then $\mathcal{T}$ is *consistent* with respect to B, denoted as $\mathcal{T} \models B$, if $\mathcal{T} \models b$ for each $b \in B$. Moreover, $\mathcal{T}$ is *inconsistent* with respect to B if there is some $b \in B$ such that $\mathcal{T} \not\models b$. A *subset-repair* of $\mathcal{T}$ with respect to B is a subset $\mathcal{P} \subseteq \mathcal{T}$ that is consistent with respect to B, and maximal in the sense that no set $\mathcal{P}'$ exists such that it is consistent with respect to B and $\mathcal{P} \subset \mathcal{P}' \subseteq T$. In the tabular view, the subset of a DB $\mathcal{T} = (T_1, \ldots, T_m)$ is defined as $\mathcal{P} = (P_1, \ldots, P_m)$, where $P_i \subseteq T_i$ for each $i \leq m$. In the following, we simply speak of a repair when we intend to mean a subset-repair. Furthermore, we often consider a database $\mathcal{T}$ and a set B of ICs assuming that B only uses relation names from the schema of $\mathcal{T}$. In that case, we call $\mathcal{D} = \langle \mathcal{T}, B \rangle$ a *constrained database*. For a constrained database $\mathcal{D}$, repairs($\mathcal{D}$) denotes the set of all repairs for $\mathcal{D}$.

The first problem we are interested in (REP) asks whether there exists a repair for a constrained database $\mathcal{D}$. Since the empty database ($\emptyset$) satisfies each IC trivially, we are interested in the decision problem asking whether a non-empty repair exists.

Problem: REP

Input: a constrained database $\mathcal{D} = \langle \mathcal{T}, B \rangle$

Question: is there a repair $\mathcal{R} \in \text{repairs}(\mathcal{D})$ with $\mathcal{R} \neq \emptyset$

Two further problems of interest are *brave* and *cautious* reasoning for a given fact $s \in \mathcal{T}$, asking whether s belongs to some (every) repair for $\mathcal{D}$.

Problem: ∃-REP

Input: a constrained database $\mathcal{D} = \langle \mathcal{T}, B \rangle$ and a fact $s \in \mathcal{T}$

Question: does s belong to some repair for $\mathcal{D}$

Problem: ∀-REP

Input: a constrained database $\mathcal{D} = \langle \mathcal{T}, B \rangle$ and a fact $s \in \mathcal{T}$

Question: does s belong to all repairs for $\mathcal{D}$

We also consider the case when the set B of ICs is fixed and the input only involves a database $\mathcal{T}$ (and a fact s). This gives rise to separate decision problems REP_B, $\exists\text{-REP}_B$ and $\forall\text{-REP}_B$, for each B. We can then define the *data complexity* of REP, ∃-REP and ∀-REP, in the usual way. That is, for a complexity class C, the data complexity of REP is in C, if REP_B is in C, for every B, and C-hard, if REP_B is C-hard, for some B. Thus, the problems defined above (i.e., REP, ∃-REP and ∀-REP) then correspond to the setting of *combined complexity*.

Observe that our focus in this work lies on inconsistent databases, i.e., we assume that $\mathcal{T}$ is inconsistent with respect to B in a constrained database $\mathcal{D} = \langle \mathcal{T}, B \rangle$. This is not a restriction since the considered questions about repairs are trivial if the database is consistent with respect to its constraints. Moreover, all our translations to (SET)AFs and the connection between repairs and extensions still apply to the consistent case, although they do not provide much insight.

The following example presents a repair for the database from Example 2.1 with respect to the ICs from Example 2.2.

Example 2.3 *Reconsider the database from Example 2.1, which is inconsistent with the ICs from Example 2.2. Here, the fact d_3 violates* lav*, and the only way to resolve this violation (in our subset-repair setting) is to remove d_3. Moreover, the set $\{e_3, d_2\}$ also violate* dc*. This results in two ways to repair our database, by removing either of those facts. Thus, we have two subset-maximal repairs, namely $\{e_1, e_2, d_1, d_2\}$ and $\{e_1, e_2, e_3, d_1\}$.* ◁

Abstract argumentation We use Dung's argumentation framework[6] and consider only non-empty and finite sets of arguments A. An *(argumentation) framework (AF)* is a directed graph $\mathcal{F} = (A, R)$, where A is a set of arguments and the relation $R \subseteq A \times A$ represents direct attacks between arguments. If $S \subseteq A$, we say that an argument $s \in A$ is *defended by S in $\mathcal{F}$*, if for every $(s', s) \in R$ there exists $s'' \in S$ such that $(s'', s') \in R$.

In abstract argumentation, one is interested in computing the so-called *extensions*, which are subsets $S \subseteq A$ of the arguments that have certain properties. The set S of arguments is called *conflict-free in $\mathcal{F}$* if $(S \times S) \cap R = \emptyset$. Let S be conflict-free, then S is

1. *naive in $\mathcal{F}$* if no $S' \supset S$ is *conflict-free* in $\mathcal{F}$;
2. *admissible in $\mathcal{F}$* if every $s \in S$ is *defended by S in $\mathcal{F}$*.

Further, let S be admissible. Then, S is

3. *preferred in $\mathcal{F}$*, if there is no $S' \supset S$ that is *admissible in $\mathcal{F}$*;
4. *stable in $\mathcal{F}$* if every $s \in A \setminus S$ is *attacked* by some $s' \in S$.

We denote each of the mentioned semantics by abbreviations: conf, naive, adm, pref and stab, respectively. For a semantics $\sigma \in \{\text{conf}, \text{naive}, \text{adm}, \text{pref}, \text{stab}\}$, we write $\sigma(\mathcal{F})$ for the set of *all extensions* of semantics σ in $\mathcal{F}$. Now, we are ready to define the corresponding decision problem asking for extension existence with respect to a semantics σ.

Problem: Extσ

Input: an argumentation framework $\mathcal{F}$

Question: is it true that $\sigma(\mathcal{F}) \neq \emptyset$

Finally, for an AF $\mathcal{F}$=(A, R) and $a \in A$, we define the problems Cred_σ and Skep_σ, which ask whether a is in some σ-extension of $\mathcal{F}$

("*credulously* accepted") or every σ-extension of $\mathcal{F}$ ("*skeptically* accepted"), respectively.

Problem: Cred_σ

Input: an AF $\mathcal{F} = (A, R)$ and an argument $a \in A$

Question: is it true that $a \in E$ for some $E \in \sigma(\mathcal{F})$

Problem: Skep_σ

Input: an AF $\mathcal{F} = (A, R)$ and an argument $a \in A$

Question: is it true that $a \in E$ for all $E \in \sigma(\mathcal{F})$

Example 2.4 *Consider the argumentation framework $\mathcal{AF} = \{a, b, c, d\}$ with the attack relation as depicted in Fig. 2. Then extensions for $\mathcal{F}$ are also depicted in Fig. 2 for the mentioned semantics.* ◁

The complexity of reasoning in argumentation is well understood, see Table 1 in Dvorák & Dunne[40] for an overview. In particular, $\text{Cred}_{\text{naive}}$ and $\text{Skep}_{\text{naive}}$ are in **P**, whereas, $\text{Cred}_{\text{pref}}$ and $\text{Skep}_{\text{pref}}$ are **NP**-complete and $\mathbf{\Pi}_2^{\mathbf{P}}$-complete, respectively. Moreover, the problem to decide whether there is a non-empty extension is in **P** for naive and **NP**-complete for pref-semantics. This makes naive-semantics somewhat easier and pref the hardest among the considered semantics in this work.

Set-based argumentation. Dung's AFs can be generalized by allowing attacks not only from single arguments, but from collections of arguments[7]. A *B-hypergraph* (backwards hypergraph) is a directed hypergraph (V, E) where each edge goes from a set of nodes to a single node, i.e., $E \subseteq 2^V \times V$. A *set-based argumentation framework (SETAF)* is a B-hypergraph $\mathcal{F} = (A, R)$ where A is the set of arguments and $R \subseteq 2^A \times A$ represents the attacks from sets of arguments to individual arguments. A subset $S \subseteq A$ of arguments is *conflict-free*, if $(2^S \times S) \cap R = \emptyset$ and S *defends* an argument $s \in A$ if for every $(S', s) \in R$ there exists $S'' \subseteq S$ such that $(S'', s') \in R$ for some $s' \in S'$. For $S = \{s\}$ a singleton set, we prefer denoting the attack $(\{s\}, s')$ by (s, s'). With this definition of conflict-freeness, the notions of *naivety*, *admissibility*, *preferredness* and *stability* are defined analogously to plain AFs.

Problem: $\text{Ext}\sigma^{SET}$

Input: a SETAF $\mathcal{F}$

Question: is it true that $\sigma(\mathcal{F}) \neq \emptyset$

For a SETAF $\mathcal{F}$ and argument $a \in A$, we similarly define the problems Cred_σ and Skep_σ asking whether a is in some σ-extension of $\mathcal{F}$ or every σ-extension of $\mathcal{F}$, respectively.

Example 2.5 *Consider the SETAF $\mathcal{S} = (A, R)$, where $A = \{a, b, c, d, e\}$ and $R = \{(\{a, b\}, d), (\{b, c\}, e), (\{e\}, a)\}$. In $\mathcal{S}$, the extension $\{a, b, c\}$ is conflict-free, preferred and stable. Note that $\{b, c\}$ defends a against $\{e\}$. This example is visualized in Fig. 3.* ◁

The corresponding decision problem for the extensions existence is defined as follows.

3 Translating databases to argumentation frameworks

3.1 Simulating DCs via SETAFs

We first formalize the notion of conflicts for a DB with respect to a set of DCs. Intuitively, a *conflict* in a constrained database $\mathcal{D}$ is a

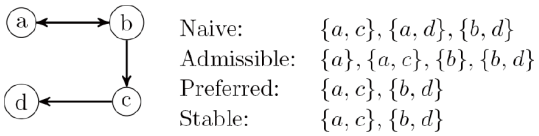


Fig. 2 Framework $\mathcal{F}$ from Example 2.4 (left) and extensions for given semantics (right).

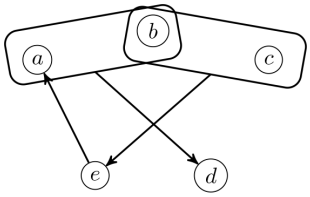


Fig. 3 The SETAF $\mathcal{S}$ from Example 2.5.

minimal (under set inclusion) inconsistent set of facts from $\mathcal{T}$ with respect to some DC in $\mathcal{D}$.

Definition 3.1 *Let $\mathcal{D} = \langle \mathcal{T}, D \rangle$ be a constrained database with a set D of DCs. A collection $C \subseteq \mathcal{T}$ of facts is a conflict in $\mathcal{D}$ if there exists some $d \in D$ such that (i) $C \not\models d$, and (ii) $C' \models d$ for each proper subset $C' \subsetneq C$.*

Given $\mathcal{D}$, then by $\text{Conf}(\mathcal{D})$ we denote the set of all conflicts in $\mathcal{D}$. It is known that $\text{Conf}(\mathcal{D})$ can be represented as a *conflict hypergraph*[20], where the vertices are the facts in $\mathcal{D}$ and hyperedges are conflicts in $\mathcal{T}$. We follow Bienvenu & Bourgaux[13] and transform an instance $\mathcal{D} = \langle \mathcal{T}, D \rangle$ with database $\mathcal{T}$ and a collection D of DCs to a SETAF $\mathcal{S}_\mathcal{D}$.

Definition 3.2 (Bienvenu & Bourgaux[13]) *Let $\mathcal{D} = \langle \mathcal{T}, D \rangle$ be a constrained database including a database $\mathcal{T}$ and a collection D of DCs. Then, $\mathcal{S}_\mathcal{D}$ denotes the following SETAF.*

- $A := \mathcal{T}$, *i.e, each fact $s \in \mathcal{T}$ is seen as an argument.*
- $R := \{(C \setminus \{t\}, t) \mid C \in \text{Conf}(\mathcal{D}), t \in C\}$.

We call $\mathcal{S}_\mathcal{D}$ the SETAF generated by $\mathcal{D}$.

It is worth highlighting that given a set D of DCs as input, then checking whether "a database $\mathcal{T}'$ is a repair of a database $\mathcal{T}$ w.r.t. D" is **DP**-complete even if $\mathcal{T}$ and $\mathcal{T}'$ are both fixed[39], Lemma 6. The complexity of the repair checking problem drops to **P** if we consider a fixed set of DCs. As a result, one normally considers a fixed set D of DCs. In our encoding, this has the effect that one can construct the argumentation framework from a constrained database $\mathcal{D}$ in polynomial time. Furthermore, given the conflict hypergraph $\mathcal{G}_\mathcal{D}$ for a constrained database $\mathcal{D}$, the framework $\mathcal{S}_\mathcal{D}$ can still be constructed in polynomial time in the size of $\mathcal{G}_\mathcal{D}$.

Example 3.3 *Consider the following database tables C and O, containing the data about customers and orders, respectively.*

C	*Cus_ID*	*Location*
s_1	*C1*	*Paderborn*
s_2	*C2*	*Sheffield*
s_3	*C3*	*Hanover*

O	*Ord_ID*	*Cus_ID*	*Prod_ID*	*Ship_ID*
t_1	*O1*	*C1*	*P1*	*S1*
t_2	*O2*	*C2*	*P2*	*S2*
t_3	*O3*	*C2*	*P1*	*S1*
t_4	*O5*	*C1*	*P1*	*S4*

Here, Cus_ID, Ord_ID, Prod_ID and Ship_ID are the IDs of individual customers, orders, products and shipments, respectively. Let $D = \{\text{dc}_1, \text{dc}_2\}$ be a set of DCs where

$$\text{dc}_1 := \forall x_1, \ldots, x_6 \neg(O(x_1, x_2, x_3, x_4) \wedge O(x_5, x_2, x_3, x_6) \wedge x_1 \neq x_5)$$

and

$$\text{dc}_2 := \forall x_1, \ldots, x_9 \neg(O(x_1, x_2, x_3, x_4) \wedge O(x_5, x_6, x_7, x_4) \wedge C(x_2, x_8) \wedge C(x_6, x_9) \wedge x_8 \neq x_9).$$

The DC dc_1 essentially says that no customer should have more than one order for the same product and dc_2 requires that no orders to different locations can be transported in the same shipment.

Both constraints are violated in the tables C and D. The conflicts in the DB include $\{t_1,t_4\},\{t_1,t_3,s_1,s_2\}$. *The respective SETAF* $\mathcal{S}_{\langle\{C,O\},D\rangle}$ *is depicted in Fig. 4.* ◁

It is known that the repairs for $\mathcal{D}$ correspond precisely to the independent sets of its conflict hypergraph $\mathcal{G}_\mathcal{D}$. Observe that the independent sets in $\mathcal{G}_\mathcal{D}$ are exactly the naive extensions of $\mathcal{S}_\mathcal{D}$. Moreover, using the observations from Bienvenu & Bourgaux[13], Theorem 28, it is easy to see that every naive extension is also preferred as well as stable. Therefore, we have the following equivalence between the extensions and repairs. The equivalence regarding the stable and preferred semantics is a direct consequence of the results in Bienvenu & Bourgaux[13].

Theorem 3.4. *Let* $\mathcal{D}=\langle\mathcal{T},D\rangle$ *be a constrained database where* D *is a set of DCs and let* $\mathcal{S}_\mathcal{D}$ *denote the SETAF generated by* $\mathcal{D}$. *Then for every* $\mathcal{P}\subseteq\mathcal{T}$, $\mathcal{P}\in\text{repairs}(\mathcal{D})$ *iff* $\mathcal{P}\in\sigma(\mathcal{S}_\mathcal{D})$ *for* $\sigma\in\{\text{naive},\text{stab},\text{pref}\}$.

Proof Given a repair $\mathcal{P}\subseteq\mathcal{T}$, then $\mathcal{P}\models d$ for every $d\in D$ and hence there is no $C\in\text{Conf}(\mathcal{D})$ with $C\subseteq\mathcal{P}$. So, the set of arguments in $\mathcal{P}$ is conflict-free since every attack in R has the form $(E\setminus\{t\},t)$ where $E\cup\{t\}\in\text{Conf}(\mathcal{D})$.

The converse holds similarly. A naive extension $P\subseteq A$ does not contain all the arguments in any attack (E,t). Thus, for every conflict $C\in\text{Conf}(\mathcal{D})$, such that $C=E\cup\{t\}$, we have that $C\not\subseteq P$ and therefore P (seen as a subset of $\mathcal{T}$) is a repair for $\mathcal{D}$. Finally, the subset-maximality remains the same for both cases since each fact is seen as an argument and vice versa. This establishes the correctness of our theorem. □

It is worth remarking that the presence of DCs with a single database atom yields singleton conflicts which results in self-contradictory facts (or self-attacking arguments in the translation). However, such self-contradictory facts do not occur in any repair and can be removed using existing reasoning algorithms, as highlighted by Bienvenu & Bourgaux[13]. Therefore, a usual assumption is that every DC contains at least two database atoms. This has the effect that each conflict contains at least two arguments. Consequently, in the presence of DCs, every fact in a conflict belongs to at least one repair and no fact in any conflict belongs to every repair. This holds because for each conflict $C\in\text{Conf}(\mathcal{D})$, there is a repair containing facts in $C\setminus\{t\}$ as well as another one containing t. As a result, we have the following observation regarding the acceptability of facts with respect to $\mathcal{D}$. Specifically, we can decide if a given fact $s\in\mathcal{T}$ is in some (or all) repairs, in polynomial time for a fixed set of DCs.

Remark 3.5 *Let* D *be a set of DCs over the schema* τ. *Then, for every constrained database* $\mathcal{D}=\langle\mathcal{T},D\rangle$ *involving a database* $\mathcal{T}$ *over* τ: ∃-REP *is true for every* $s\in\mathcal{T}$ *and* ∀-REP *is true iff* $s\notin C$ *for any* $C\in\text{Conf}(\mathcal{D})$. *Moreover, both problems are decidable in polynomial time in data complexity.*

In this section, we prove that repairs under expressive families of ICs (involving DCs and LTGDs) can be equivalently seen as extensions in SETAFs. In the first two subsections, we consider instances containing only one type of ICs to SETAFs and the third subsection combines both constraints (DCs and LTGDs). Given an instance $\mathcal{D}=\langle\mathcal{T},B\rangle$ comprising a database $\mathcal{T}$ and a set B of ICs, the goal is to capture all the subset-repairs for $\mathcal{D}$ by σ-extensions of the resulting SETAF $\mathcal{S}_\mathcal{D}$ for some semantics σ.

In the section "Simulating DCs via SETAFs", we encode a constrained database $\mathcal{D}=\langle\mathcal{T},D\rangle$ with database $\mathcal{T}$ and a collection D of DCs into a SETAF $\mathcal{S}_\mathcal{D}$. This is achieved by letting each fact $s\in\mathcal{T}$ be an argument. Then the attack relation between arguments simulates the violation of some $d\in D$ by sets of facts. This allows to establish a connection between repairs for $\mathcal{D}$ and the extensions for the resulting SETAF under preferred, stable and naive semantics. This construction

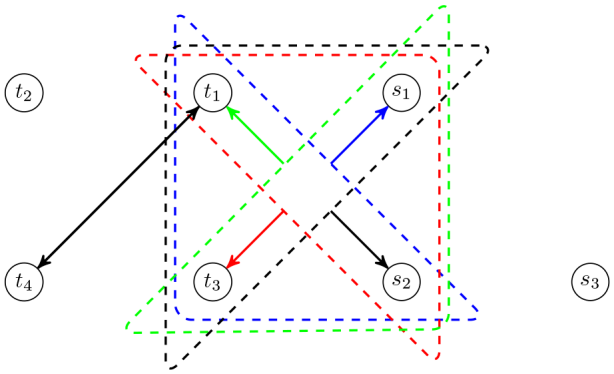


Fig. 4 SETAF $\mathcal{S}_{\langle\{C,O\},D\rangle}$ for Example 3.3. The attacker in each set-attack is depicted as a triangle of different color (for better presentation) and the attack is presented in the same color. For example, the red triangle and the arrow depict the attack $(\{t_1,s_1,s_2\},t_3)$.

for DCs is a special case of the results in Bienvenu & Bourgaux[13]. In our case, we do not consider priorities among facts in the database, which allows a weaker SETAF-semantics (naive extensions) to capture repairs.

In the section "Simulating LTGDs via SETAFs", we simulate a constrained database $\mathcal{D}=\langle\mathcal{T},L\rangle$ including a collection L of LTGDs via SETAFs. The first observation is that the semantics for LTGDs requires the notion of *support* or *defense* rather than *conflict* between facts. Then, we depict each fact as an argument as well as use auxiliary arguments to simulate the dependency $\ell\in L$ (i.e., to model the semantics for LTGDs). Further, we add self-attacks for these auxiliary arguments to prohibit them from appearing in any extension. Consequently, we establish a connection between repairs for $\mathcal{D}$ and the extensions for SETAFs under preferred semantics. Finally, we present a polynomial-time processing of the resulting SETAFs that computes its unique preferred extension, which also yields a unique repair for $\mathcal{D}$. Interestingly, the computed preferred extension is also stable and naive, and hence coincides with the only repair for $\mathcal{D}$. Our construction for LTGDs relies on self-attacking arguments. This allows us to avoid taking auxiliary arguments in any extension and establish a connection between repairs of a DB and extensions of the resulting SETAF. Toward the end of our paper, in the section "Simulating FDs and IDs via AFs", we discuss the role played by the self-attacking arguments and a possibility for a different translation from DBs to (SET)AFs.

Having established that both DCs and LTGDs can be modeled in SETAFs via attacks, the section "Simulating DCs and LTGDs via SETAFs" generalizes this approach by allowing both types of constraints. We observe that no prior work has handled the case of tuple-generating dependencies (to our knowledge).

3.2 Simulating LTGDs via SETAFs

Let $\mathcal{D}=\langle\mathcal{T},L\rangle$ be a constrained database with a collection $\mathcal{T}$ of facts and a collection L of LTGDs. We recall the semantics for LTGDs via variable assignments. For a set V of variables, an *assignment* of V into a database $\mathcal{T}$ is a mapping $h\colon V\to\text{dom}(\mathcal{T})$. For a relational atom $T(\boldsymbol{t})$ with terms $\boldsymbol{t}=(t_1,\ldots,t_k)$, we denote by $h(T(\boldsymbol{t}))$ the result of substituting every t_i in $T(\boldsymbol{t})$ by $h(t_i)$, where we set $h(t_i)=t_i$ when t_i is a constant. In other words, an assignment maps a relational atom $T(\boldsymbol{t})$ to a candidate fact $h(T(\boldsymbol{t}))$, which may or may not be present in the database $\mathcal{T}$. The notation extends to a conjunction of atoms by considering them as a set. Precisely, for a conjunction $\phi(\boldsymbol{x}):=\bigwedge_{1\le i\le n}T_i(\boldsymbol{x}_i)$ with $\boldsymbol{x}_i\subseteq\boldsymbol{x}$ and $T_i\in\tau$, and an assignment $h\colon\boldsymbol{x}\to\text{dom}(\mathcal{T})$, we define $h(\phi(\boldsymbol{x}))=\{h(T_i(\boldsymbol{x}_i))\mid 1\le i\le n\}$. Recall that

every LTGD $\ell \in L$ has the form $\forall \boldsymbol{x}(T(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}\psi(\boldsymbol{x},\boldsymbol{y}))$ where $\psi := \bigwedge_{1\le i\le n_\ell} \psi_i(\boldsymbol{z}_i)$ is a conjunction of relational atoms over variables $\boldsymbol{z}_i \subseteq \boldsymbol{x}\cup\boldsymbol{y}$ using relation names $\psi_i \in \tau$. Then, $\mathcal{T} \models \ell$ if for each assignment h such that $h(T(\boldsymbol{x})) \in \mathcal{T}$, there exists an assignment $h' : \boldsymbol{x}\cup\boldsymbol{y} \rightarrow \mathrm{dom}(\mathcal{T})$ such that $h' \restriction_{\boldsymbol{x}} = h \restriction_{\boldsymbol{x}}$ and $h'(\psi(\boldsymbol{x},\boldsymbol{y})) \subseteq \mathcal{T}$, where $h \restriction_{\boldsymbol{x}}$ is the restriction of h on $\boldsymbol{x}$. If $h(T(\boldsymbol{x})) \in \mathcal{T}$ is true and there indeed exists such an assignment h' then we say that the set of facts $h'(\psi(\boldsymbol{x},\boldsymbol{y}))$ *supports* the fact $h(T(\boldsymbol{x}))$ for the LTGD ℓ. For such an LTGD ℓ, we call T the *source* of ℓ, denoted as source(ℓ). For a source(ℓ)-fact s, we denote by $\boldsymbol{S}_\ell(s)$ the collection of all the sets supporting s for the LTGD ℓ. It is worth remarking that, for a constrained database $\mathcal{D} = \langle \mathcal{T}, L\rangle$ and $\ell \in L$, $\mathcal{D} \models \ell$ if and only if $\boldsymbol{S}_\ell(s) \neq \emptyset$ for each source(ℓ)-fact s.

Interestingly, we can encode the sets supporting a fact by using SETAFs, as formalized next.

Definition 3.6 *Let $\mathcal{D} = \langle \mathcal{T}, L\rangle$ be a constrained database including a database $\mathcal{T}$ and a collection L of LTGDs. Then $\mathcal{S}_\mathcal{D}$ is the following SETAF.*

- $A := \{s \mid s \in \mathcal{T}\} \cup \{s_\ell \mid s \text{ is a source}(\ell)-\text{fact for } \ell \in L\}$,
- $R := \{(s_\ell, s), (s_\ell, s_\ell) \mid s \text{ is a source}(\ell)-\text{fact for } \ell \in L\} \cup \{(S, s_\ell) \mid S \in \boldsymbol{S}_\ell(s), \ell \in L\}$.

Intuitively, for each $\ell \in L$ and {a source(ℓ)-fact s}, the attack (s_ℓ, s) models that the LTGD $\ell \in L$ must be satisfied for the fact s. Moreover, the self-attacks (s_ℓ, s_ℓ) enforce that the extensions only contain arguments corresponding to facts in the database. Then, each $S \in \boldsymbol{S}_\ell(s)$ attacks s_ℓ and consequently, defends s against s_ℓ. The whole idea captured in this translation is that for each $\ell \in L$ of the form $\forall \boldsymbol{x}(T(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \bigwedge_{1\le i\le n_\ell} \psi_i(\boldsymbol{z}_i))$ where $\boldsymbol{z}_i \subseteq \boldsymbol{x}\cup\boldsymbol{y}$: a T-fact s is in a repair $\mathcal{P}$ of $\mathcal{D}$ if and only if there are ψ_i-facts t_i for $1 \le i \le n_\ell$, such that $\{t_1, \ldots, t_{n_\ell}\} \in \boldsymbol{S}_\ell(s)$ if and only if the set $S = \{t_1, \ldots, t_{n_\ell}\}$ attacks s_ℓ in $\mathcal{S}_\mathcal{D}$ if and only if the argument $s \in A$ is defended against s_ℓ in $\mathcal{S}_\mathcal{D}$.

Example 3.7. *The database tables E, D and P contain data about employees, departments and projects, respectively, and are defined as follows:*

E	*Emp_ID*	*Dept_ID*
s_1	*E1*	*D1*
s_2	*E2*	*D2*
s_3	*E3*	*D3*

D	*Dept_ID*	*Dept_Name*
t_1	*D1*	*Accounting*
t_2	*D3*	*Sales*

P	*Prod_ID*	*Dept_ID*
u_1	*P1*	*D1*
u_2	*P2*	*D3*

Here, Emp_ID, Dept_ID and Prod_ID are the IDs of individual employees, departments and products, respectively. Let $L = \{\mathrm{lav}_1, \mathrm{lav}_2\}$ be a set of LTGDs where

$$\mathrm{lav}_1 := \forall x_1, x_2(E(x_1, x_2) \rightarrow \exists y, z(D(x_2, y) \wedge P(z, x_2)))$$

and

$$\mathrm{lav}_2 := \forall x_1, x_2(D(x_1, x_2) \rightarrow \exists y, z(E(y, x_1) \wedge P(z, x_1))).$$

The constraint lav_1 essentially says that if an employee is in a department, then the department must exist and be associated with a project. The constraint lav_2 says that each department has at least one employee and at least one project associated with it.

In the tables E, D and P, lav_1 is violated while lav_2 is satisfied. The source of each TGD ℓ, and the supporting facts for source(ℓ)*-fact s are presented as follows:*

- source(lav_1) = E,
- source(lav_2) = D,
- $\{t_1, u_1\}$ *supports* s_1 *for* lav_1,
- s_2 *does not have any supporter for* lav_1
- $\{t_2, u_2\}$ *supports* s_3 *for* lav_1,
- $\{s_1, u_1\}$ *supports* t_1 *for* lav_2,
- $\{s_3, u_2\}$ *supports* t_2 *for* lav_2.

The respective SETAF $\mathcal{S}_{\langle\{E,D,P\},L\rangle}$ is depicted in Fig. 5. ◁

Similar to the case of DCs, the repair checking problem under LTGDs is **DP**-complete if the set L of ICs is considered as input (see Table 3 in Arming et al.[39]). Therefore, we consider a fixed set L of LTGDs as the complexity drops to **P** in this case. Then, for a constrained database $\mathcal{D} = \langle \mathcal{T}, L\rangle$, the framework $\mathcal{S}_\mathcal{D}$ can be constructed in polynomial time. An algorithm basically goes through every $\ell \in L$ in turn: for each source(ℓ)-fact s, it computes the set $\boldsymbol{S}_\ell(s)$ in polynomial time since the size of each LTGD in L is constant.

For a set L of LTGDs, a minimal violation of $\ell \in L$ is caused by a single source(ℓ)-fact s for which $\boldsymbol{S}_\ell(s) = \emptyset$. Since we only allow subset repairs (deleting facts), the only way to eliminate a violation is to remove such a fact s. However, this removal of facts may trigger other violations for facts, all of whose supporting sets contained some deleted facts. This triggers a chain reaction and the repair process iteratively removes any fact that has no remaining supporting set. Since the database is finite, this process eventually terminates (in the worst case, there are no facts remaining and $\emptyset$ is the only repair). The remaining facts are exactly those that have at least one full support left, so no violations remain. The key observation is that this process always leads to the same final set of facts, no matter in which order we remove unsupported facts. Therefore, the final set of facts is consistent and subset-maximal. Observe that in some cases $\emptyset$ might be the only repair for a database $\mathcal{D}$ involving LTGDs.

Remark 3.8 *A constrained database $\mathcal{D} = \langle \mathcal{T}, L\rangle$ involving a set L of LTGDs admits a unique repair.*

Now we prove that the repair of an instance $\mathcal{D}$ actually corresponds to the unique preferred extension of the resulting SETAF $\mathcal{S}_\mathcal{D}$.

Theorem 3.9. *Let $\mathcal{D} = \langle \mathcal{T}, L\rangle$ be a constrained database where L is a set of LTGDs and let $\mathcal{S}_\mathcal{D}$ denote the SETAF generated by $\mathcal{D}$. Then for $\mathcal{P} \subseteq \mathcal{T}$, $\mathcal{P} \in$ repairs($\mathcal{D}$) iff $\mathcal{P} \in$ pref($\mathcal{S}_\mathcal{D}$). Moreover, $\mathcal{S}_\mathcal{D}$ has exactly one preferred extension.*

Proof We first prove the reverse direction. Let $P \subseteq A$ be a preferred extension in $\mathcal{S}_\mathcal{D}$, then P must not contain an auxiliary argument s_ℓ corresponding to any LTGD $\ell \in L$ since P is conflict-free. This implies that $P \subseteq \mathcal{T}$, which, together with the fact that P is admissible (hence every $s \in P$ is defended against each $s_\ell \in A$) and maximal under set inclusion, yields the proof of the claim.

Conversely, let $\mathcal{P} \subseteq \mathcal{T}$ denote a repair for $\mathcal{D} = \langle \mathcal{T}, L\rangle$. Then $\mathcal{P}$ is conflict-free in $\mathcal{S}_\mathcal{D}$ since each attack in R contains at least one argument among the auxiliary arguments (s_ℓ) which are not in $\mathcal{P}$ (as $\mathcal{P} \subseteq \mathcal{T}$). Moreover, for each $s \in \mathcal{P}$ and $\ell \in L$, there are $t_1, \ldots, t_{n_\ell} \in \mathcal{P}$,

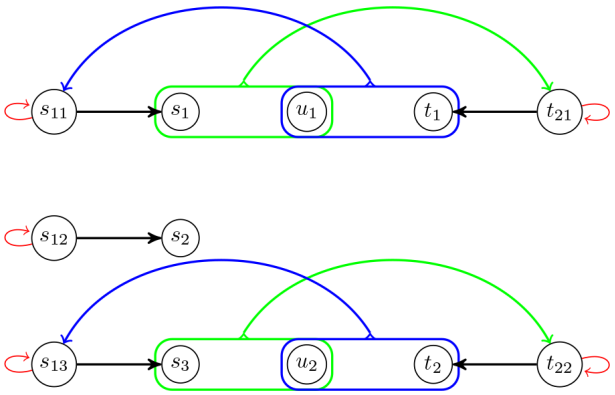


Fig. 5 SETAF $\mathcal{S}_{\langle\{E,D,P\},L\rangle}$ for Example 3.7. For brevity, we rename the LTGDs $\{\mathrm{lav}_1, \mathrm{lav}_2\}$ to be $\{1, 2\}$. Moreover, the auxiliary arguments for source(1)-facts s_i are renamed to s_{1i} and those for source(2)-facts t_j to t_{2j}.

such that $\{s, t_1, \ldots, t_{n_\ell}\} \models \ell$. This implies that each $s \in \mathcal{P}$ is defended against the attack $s_\ell \in A$ since $(\{t_1, \ldots, t_{n_\ell}\}, s_\ell) \in R$. Consequently, $\mathcal{P}$ is admissible. To prove that $\mathcal{P}$ is also preferred in $\mathcal{S}_\mathcal{D}$, assume the contrary that there is an admissible $\mathcal{P}' \supset \mathcal{P}$ in $\mathcal{S}_\mathcal{D}$. Since $\mathcal{P}'$ is also conflict-free, using the same argument as for $\mathcal{P}$ we notice that $\mathcal{P}' \subseteq \mathcal{T}$. Now, $\mathcal{P}'$ being admissible (together with the claim in reverse direction) implies that $\mathcal{P}'$ is a repair for $\mathcal{D}$ contradicting the fact that $\mathcal{P}$ is a subset-maximal repair for $\mathcal{D}$. As a consequence, $\mathcal{P}$ is preferred in $\mathcal{S}_\mathcal{D}$.

This proves the correctness of the theorem. The uniqueness of the extension follows due to Remark 3.8. □

Notice that a framework $\mathcal{S}_\mathcal{D}$ may not have stable extensions for certain constrained databases $\mathcal{D}$ including databases $\mathcal{T}$ and LTGDs L. This holds because some arguments can neither be accepted in an extension (e.g., when $\boldsymbol{S}_\ell(s) = \emptyset$ for some source(ℓ)-fact s and $\ell \in L$), nor attacked by arguments in an extension (since arguments in A only attack auxiliary arguments). Mahmood et al.[38] presented a pre-processing over the argumentation framework generated by a constrained database $\mathcal{D}$ involving IDs, which computes its unique preferred extension, which is also naive and stable. It is easy to observe that a similar pre-processing can be implemented for the case of LTGDs. This pre-processing also allows us to answer credulous and skeptical acceptance for a fact $s \in \mathcal{T}$ in polynomial time considering the data complexity.

A pre-processing algorithm for $\mathcal{S}_\mathcal{D}$. Observe that an undefended argument in a SETAF $\mathcal{S} = (A, R)$ cannot belong to any preferred extension of $\mathcal{S}$. The intuition behind pre-processing is to remove such arguments, which are not defended against some of their attacks in $\mathcal{S}_\mathcal{D}$. This corresponds to (recursively) removing those facts $s \in \mathcal{T}$, for which there is $\ell \in L$ such that s is a source(ℓ)-fact and $\boldsymbol{S}_\ell(s) = \emptyset$. The pre-processing (denoted PRE($\mathcal{S}_\mathcal{D}$)) applies the following procedure as long as possible.

∗ For each $s_\ell \in A$ such that s_ℓ is not attacked by any $S \subseteq A \setminus \{s_\ell\}$: remove s, s_j for each $j \in L$, as well as each attack to and from s and s_j.

We repeat this procedure until convergence. Once a fixed point has been reached, the remaining arguments in A are all defended. Interestingly, after the pre-processing, removing the arguments with self-attacks results in a unique naive extension, which is also stable and preferred. In fact, this naive extension corresponds to the unique repair of the instance $\mathcal{D}$ (see Remark 3.8). In the following, we also denote by PRE($\mathcal{S}_\mathcal{D}$) the SETAF obtained after applying the pre-processing on $\mathcal{S}_\mathcal{D}$. Notice that PRE is basically an adaptation to the SETAFs of the previously presented procedure for AFs and originates from the well-known algorithm for finding a maximal satisfying subteam for inclusion logic formulas[28], Lem. 12.

Lemma 3.10. *Let L be a fixed set of LTGDs, $\mathcal{D} = \langle \mathcal{T}, L \rangle$ be a constrained database, and $\mathcal{S}_\mathcal{D}$ denote the SETAF generated by $\mathcal{D}$. Then* PRE($\mathcal{S}_\mathcal{D}$) *can be computed from $\mathcal{S}_\mathcal{D}$ in polynomial time. Moreover,* PRE($\mathcal{S}_\mathcal{D}$) *has a unique naive extension which is also stable and preferred.*

Proof The procedure PRE($\mathcal{S}_\mathcal{D}$) removes recursively all the arguments corresponding to facts s such that $\boldsymbol{S}_\ell(s) = \emptyset$ for some $\ell \in L$ and s is a source(ℓ)-fact. Notice that $\boldsymbol{S}_\ell(s)$ can be computed for each $\ell \in L$ and source(ℓ)-fact s in polynomial time for a fixed set L of LTGDs. Then, PRE stores in a data structure (such as a queue) all the arguments s for which $\boldsymbol{S}_\ell(s) = \emptyset$. Finally, each argument s in this queue can be processed turn by turn, adding possibly new arguments when PRE triggers the removal of certain arguments from A and hence from $\boldsymbol{S}_\ell(t)$ for some $t \in A$. A fixed point is reached when every element in the queue has been processed, this gives the size of A as the total number of iterations. Consequently, PRE runs in polynomial time in the size of $\mathcal{S}_\mathcal{D}$.

Let PRE($\mathcal{S}_\mathcal{D}$) $= (A', R')$ denote the SETAF generated by the pre-processing. To prove the equivalence between extensions, notice that the set of arguments S without self-attacks in A' forms a naive extension since S is conflict-free and every argument in $A' \setminus S$ contains self-attacks. Moreover, S is also admissible since $\boldsymbol{S}_\ell(s) \neq \emptyset$ for every argument $s \in S$ corresponding to a source(ℓ)-fact s. Furthermore, since $A' \setminus S$ only includes auxiliary arguments, those are all attacked by S and therefore S is stable. Finally, S is the only naive extension in the reduced AF since S is the maximal conflict-free in PRE($\mathcal{S}_\mathcal{D}$) and arguments in $A' \setminus S$ contain self-attacks.

This establishes the correctness of the lemma together with Theorem 3.9. □

As a consequence of Lemma 3.10, we can also determine ∃-REP and ∀-REP for each $s \in \mathcal{T}$, once the pre-processing has terminated, resulting in PRE($\mathcal{S}_\mathcal{D}$).

Remark 3.11 *Let $\mathcal{D} = \langle \mathcal{T}, L \rangle$ be a constrained database involving a set L of LTGDs and $\mathcal{S}_\mathcal{D}$ denote the SETAF generated by $\mathcal{D}$. Then,* ∃-REP *and* ∀-REP *is true for every $s \in \mathcal{T}$ iff $s \in$* PRE($\mathcal{S}_\mathcal{D}$). *As a consequence, both problems are decidable in polynomial time in data complexity.*

3.3 Simulating DCs and LTGDs via SETAFs

Consider an instance $\mathcal{D} = \langle \mathcal{T}, B \rangle$ with a database $\mathcal{T}$ and a collection $B = D \cup L$ of DCs (D) and LTGDs (L). We first understand the role of applying pre-processing in the presence of both ICs. Removing those facts from $\mathcal{T}$ failing some LTGD reduces the number of arguments in the resulting SETAF without affecting the connection between extensions and repairs. The removed facts are limited to those that cannot belong to any repair since we are in the setting of subset-repairs. However, one still cannot directly answer ∃-REP or ∀-REP (as done in Remark 3.11) for all facts $s \in \mathcal{T}$ even after applying this pre-processing. Moreover, in the presence of both types of ICs, one may want to see the effect of the failure of LTGDs and DCs separately for certain facts. In other words, one can distinguish a fact violating an LTGD since it does not have a supporting fact in $\mathcal{T}$ from another fact violating an LTGD since all of its supporting facts also violate some DCs. In this case, the pre-processing allows to distinguish these facts even if they are not removed from the resulting SETAF.

In the following, we do not apply pre-processing and instead consider all the arguments for facts in the database. The framework $\mathcal{S}_\mathcal{D} := (A, R_\mathrm{D} \cup R_\mathrm{L})$ generated by $\mathcal{D}$ is specified as below.

Definition 3.12 *Let $\mathcal{D} = \langle \mathcal{T}, B \rangle$ be a constrained database including a database $\mathcal{T}$ and a collection $B = D \cup L$ of DCs D and LTGDs L. Then, we construct the SETAF $\mathcal{S}_\mathcal{D} = (A, R_D \cup R_L)$ as follows.*

- $A := \{s \mid s \in \mathcal{T}\} \cup \{s_\ell \mid s$ *is a* source(ℓ) − fact for $\ell \in L\}$,
- $R_D := \{(C \setminus \{t\}, t) \mid C \in \mathrm{Conf}(\mathcal{D}), t \in C\}$,
- $R_L := \{(s_\ell, s), (s_\ell, s_\ell) \mid s$ *is a* source(ℓ) − fact for $\ell \in L\} \cup \{(S, s_\ell) \mid S \in \boldsymbol{S}_\ell(s)$ for $\ell \in L\}$.

As before, we call $\mathcal{S}_\mathcal{D}$ the SETAF generated by $\mathcal{D}$.

A consequence of allowing both types of ICs (DCs and LTGDs) is that the preferred and naive extensions do not coincide in general. Moreover, both ∃-REP and ∀-REP are non-trivial and distinct (cf. Remarks 3.5 and 3.11). Furthermore, in the presence of both types of ICs, the repairs correspond to somewhat costly (that is, preferred) SETAF-semantics.

Theorem 3.13. *Let $\mathcal{D} = \langle \mathcal{T}, B \rangle$ be a constrained database where B includes DCs and LTGDs. Further, let $\mathcal{S}_\mathcal{D}$ denote the SETAF generated by $\mathcal{D}$. Then for every subset $\mathcal{P} \subseteq \mathcal{T}$, $\mathcal{P} \in$* repairs($\mathcal{D}$) *iff $\mathcal{P} \in$* pref($\mathcal{S}_\mathcal{D}$).

Proof The correctness follows from the proofs of Theorems 3.4 and

3.9. The conflict-freeness and admissibility of $\mathcal{P}$ implies that each DC and LTGD in B, respectively, is true in $\mathcal{P}$. The converse follows the same line of argument. Finally, the maximality of repairs in $\mathcal{D}$ corresponds to the maximality of extensions in the resulting SETAF $\mathcal{S}_{\mathcal{D}}$. □

Next, we establish that allowing DCs and LTGDs renders the data complexity for repair existence (REP), as well as credulous ($\exists$-REP) and skeptical ($\forall$-REP) reasoning for facts, same as, respectively, the existence, credulous, and skeptical reasoning for preferred semantics for AFs. These lower bounds are proven for the case of FDs and IDs (see Theorems 4.13 and 4.16). Here, we prove that the upper bounds hold even if we extend FDs to DCs and IDs to LTGDs, respectively. However, the set of ICs has to be fixed.

Theorem 3.14. *The problems* REP *and* $\exists$-REP *are both* **NP***-complete in the data complexity for DCs and LTGDs.*

Proof We prove that REP_{B} and $\exists\text{-REP}_{\text{B}}$ are both in **NP** for any fixed set B of DCs and LTGDs. Let $\mathcal{D} = \langle \mathcal{T}, B \rangle$ be a constrained database involving a database $\mathcal{T}$. For membership, one needs to guess a repair for $\mathcal{D}$ that is non-empty, respectively contains s. The verification can be done in **P** since B is a fixed set[39]. Notice that we do not need to check the maximality, since if there is a non-empty subset of $\mathcal{T}$ satisfying every IC in B (resp., a subset containing s) then there is also a non-empty repair for $\mathcal{D}$ (containing s).

Hardness follows from the case of FDs and IDs proved in Theorem 4.13. To be precise, we construct a fixed set B (resp., B') of FDs and IDs for which the problem REP_{B} ($\exists\text{-REP}_{B'}$) is **NP**-hard. □

Theorem 3.15. *The data complexity of* $\forall$-REP *is* $\mathbf{\Pi}_2^{\mathbf{P}}$*-complete for DCs and LTGDs.*

Proof For membership, we prove that $\forall\text{-REP}_{\text{B}}$ is in $\mathbf{\Pi}_2^{\mathbf{P}}$ for any fixed set B of DCs and LTGDs. To this aim, we present the following $\mathbf{\Sigma}_2^{\text{P}}$ procedure for the complement of $\forall\text{-REP}_{\text{B}}$. Let $\mathcal{D} = \langle \mathcal{T}, B \rangle$ be a constrained database involving a database $\mathcal{T}$. One can guess a set $\mathcal{P}$ of facts in $\mathcal{T}$ satisfying each IC in B as a counter-example for s, that is, $s \notin \mathcal{P}$ and $\mathcal{P}$ is a (candidate) repair for $\mathcal{D}$, which can be decided in **P** since B is fixed[39]. However, one has to use oracle calls to determine whether $\mathcal{P}$ is a repair (hence maximal). Then, $\forall$-REP is true for s if there exists no $\mathcal{P} \subseteq \mathcal{T}$ such that (i) $\mathcal{P}$ satisfies each IC in B, (ii) $s \notin \mathcal{P}$, and (iii) $\mathcal{P}$ is a repair. This gives the stated upper bound of $\mathbf{\Pi}_2^{\mathbf{P}}$.

For hardness, we prove that there exists a set B of DCs and LTGDs such that the problem $\forall\text{-REP}_{\text{B}}$ is $\mathbf{\Pi}_2^{\mathbf{P}}$-hard. We prove this claim for the case of FDs and IDs in Theorem 4.16. □

We conclude this section by noting that the combined complexity of REP, $\exists$-REP, and $\forall$-REP remains open for now, although the hardness transfer from the data complexity.

4 The case of less expressive ICs

In this section, we consider FDs and IDs, which are restricted classes of ICs inside DCs and LTGDs, respectively. The two sub-cases are interesting due to the fact that one only requires binary conflicts/supports. As a result, the classical Dung's AFs with binary attacks suffice for encoding repairs into extensions.

Following the same theme as in the section "Simulating DCs and LTGDs via SETAFs", the first two subsections translate instances containing only one type of ICs to AFs. Then, we combine both FDs and IDs in the third subsection. As before, we adopt the construction for FDs from Bienvenu & Bourgaux[13], but without fact priorities, thus resulting in the need for a weaker AF-semantics to capture repairs.

Having established that both FDs and IDs can be modeled in AFs via binary attacks, the section "The role of self-attacking arguments" generalizes this approach by allowing both types of dependencies. The primary result of this section is noteworthy: it demonstrates that the integrity constraints can be restricted from (1) DCs to FDs and (2) LTGDs to IDs, while the corresponding encoding into argumentation frameworks preserves the exact semantics, modulo the transition from SETAFs to AFs.

4.1 Simulating FDs via AFs

We transform an instance $\mathcal{D} = \langle \mathcal{T}, D \rangle$ with database $\mathcal{T}$ and a collection D of FDs to an AF $\mathcal{F}_{\mathcal{D}}$. In the case of FDs, each conflict involves two facts failing some FD $d \in D$. As a result, this conflict can be modeled in an AF by drawing a bi-directional attack between the two corresponding arguments. For this reason, we adhere to AFs rather than SETAFs. Recall that each FD is defined over a single relation T in the schema of $\mathcal{D}$.

Definition 4.1 *Let $\mathcal{D} = \langle \mathcal{T}, D \rangle$ be a constrained database including a database $\mathcal{T}$ and a collection D of FDs. Then, $\mathcal{F}_{\mathcal{D}}$ denotes the following AF.*

- $A := \mathcal{T}$*, that is, each $s \in \mathcal{T}$ is seen as an argument,*
- $R := \{(s,t),(t,s) \mid s,t \in \mathcal{T}$ *and* $\{s,t\} \not\models d$ *for some* $d \in D\}$.

We call $\mathcal{F}_{\mathcal{D}}$ the argumentation framework generated by $\mathcal{D}$ and call R the conflict graph for $\mathcal{D}$.

Note that, for a given constrained database $\mathcal{D}$, the framework $\mathcal{F}_{\mathcal{D}}$ can be generated in polynomial time. The attack relation R is constructed for each $d \in D$ over T in the schema of $\mathcal{D}$ by taking each pair s,t of T-facts in turn and checking whether $\{s,t\} \models d$ or not. Moreover, the set D of FDs does not have to be fixed, as in the case of DCs.

Example 4.2 *Consider $\mathcal{D} = \langle T, D \rangle$ with database $T = \{s,t,u,v\}$ as depicted inside table in Fig. 6 and FDs* $\{\text{dep}(Emp_ID; Dept), \text{dep}(Sup_ID; Building)\}$ *over T. Informally, each employee is associated with a unique department, and employees supervised by the same supervisor work in the same building. Observe that,* $\{s,t\} \not\models \text{dep}(Emp_ID; Dept)$, $\{u,v\} \not\models \text{dep}(Emp_ID; Dept)$, *and* $\{t,v\} \not\models \text{dep}(Sup_ID; Building)$. *The resulting AF $\mathcal{F}_{\mathcal{D}}$ is depicted on the right side of Fig. 6. The preferred (as well as naive and stable) extensions of $\mathcal{F}_{\mathcal{D}}$ include $\{s,v\}, \{t,u\}$ and $\{s,u\}$. Clearly, these three are the only repairs for $\mathcal{D}$.* ◁

It is easy to observe that any subset $\mathcal{P} \subseteq \mathcal{T}$ that satisfies each $d \in D$ contains precisely those facts in $\mathcal{T}$ that are not pairwise conflicting. As a result, such subsets correspond to the naive extensions (maximal conflict-free sets) of $\mathcal{F}_{\mathcal{D}}$. Moreover, since the attack relation in $\mathcal{F}_{\mathcal{D}}$ is symmetric, i.e., $(s,t) \in R$ iff $(t,s) \in R$, the preferred, stable and naive extensions coincide[41], Prop. 4 & 5.

Observe that every FD is also a DC and hence the AF $\mathcal{F}_{\mathcal{D}}$ coincides with the SETAF $\mathcal{S}_{\mathcal{F}}$ for the constrained database $\mathcal{D}$. As a consequence, we have the following corollary due to Theorem 3.4. In particular, each conflict set C (see proof of Theorem 3.4) has size two for FDs.

Corollary 4.3. *Let $\mathcal{D} = \langle \mathcal{T}, D \rangle$ be a constrained database where D is a set of FDs and let $\mathcal{F}_{\mathcal{D}}$ denote the argumentation framework generated by $\mathcal{D}$. Then for every subset $\mathcal{P} \subseteq \mathcal{T}$, $\mathcal{P} \in \text{repairs}(\mathcal{D})$ iff $\mathcal{P} \in \sigma(\mathcal{F}_{\mathcal{D}})$ for $\sigma \in \{\text{naive}, \text{stab}, \text{pref}\}$.*

Corollary 4.3 reproves that a subset-repair for $\mathcal{D}$ can be computed in polynomial time[40]. Moreover, similar to the case of DCs, we can decide if a given fact $s \in \mathcal{T}$ is in some (or all) repairs, in polynomial time. In fact, it follows from the basic properties of FDs that $\exists$-REP is true for every $s \in \mathcal{T}$, and $\forall$-REP is true for a fact $s \in \mathcal{T}$ iff $\{s,t\} \models d$ for each $t \in \mathcal{T}$ and $d \in D$. Furthermore, the latter problem can be decided in polynomial time for an input $\mathcal{D}$ where the set D of FDs does not have to be fixed (see Remark 3.5).

We conclude this section by observing that adding a size restriction on repairs renders the REP problem **NP**-hard. Observe that the hard-

T	Emp_ID	Sup_ID	Dept.	Building
s	TimX3	JonX1	Marketing	B1
t	TimX3	AxeK4	Sales	B2
u	JonX1	JonX1	Production	B1
v	JonX1	AxeK4	Distribution	B4

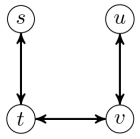


Fig. 6 Argumentation framework for modeling the FDs in Example 4.2.

ness follows due to a one-to-one correspondence between repairs and maximal independent sets of the conflict graph[5]. However, we strengthen this result and note that the hardness already holds for propositional databases involving a single relation, that is, for a database T with $\mathrm{dom}(T) = \{0,1\}$. The following result was proven in the context of team-semantics and maximal satisfying subteams for propositional dependence logic.

Proposition 4.4. *Mahmood[29], Theorem 3.32. There is a constrained database $\mathcal{D}$ including a propositional database T and FDs D, such that given $k \in \mathbb{N}$, the problem to decide whether there is a repair $P \subseteq T$ for $\mathcal{D}$ such that $|P| \geq k$ is* **NP***-complete.*

4.2 Simulating IDs via AFs

Let $\mathcal{D} = \langle \mathcal{T}, I \rangle$ be a constrained database with a database $\mathcal{T}$ and collection I of IDs. Similar to the case of LTGDs, for an ID $i \in I$ (say, $i := T[\boldsymbol{x}] \subseteq T'[\boldsymbol{y}]$), we call T the *source* of i. Moreover, we call T' the *target* of i. For $i \in I$ and a source(i)-fact s, let t be target(i)-fact such that $\{s,t\} \models i$. Then we say that t *supports* s for the ID $i \in I$, and $\boldsymbol{S}_i(s)$ denotes the set of all such supporting facts for s and an ID i. Notice that for each $i \in I$ and source(i)-fact s, we have that $\boldsymbol{S}_i(s) \subseteq \mathrm{target}(i)$. Thus, the elements in $\boldsymbol{S}_i(s)$ are facts instead of sets of facts as in the case of LTGDs. Clearly, $\mathcal{D} \models i$ if and only if $\boldsymbol{S}_i(s) \neq \emptyset$ for each source(i)-fact s. As for LTGDs, we create auxiliary argument s_i for each source(i)-fact s, which can be attacked by arguments corresponding to target(i)-facts in $\boldsymbol{S}_i(s)$. In the following, we formalize this notion and simulate the semantics for IDs via AFs.

Definition 4.5 *Let $\mathcal{D} = \langle \mathcal{T}, I \rangle$ be a constrained database including a database $\mathcal{T}$ and a collection I of IDs. Then $\mathcal{F}_\mathcal{D}$ is the following AF.*

- $A := \mathcal{T} \cup \{s_i \mid s \text{ is a source}(i)-\text{fact for } i \in I\}$,
- $R := \{(s_i, s), (s_i, s_i) \mid s \text{ is a source}(i)-\text{fact for } i \in I\} \cup \{(t, s_i) \mid t \in \boldsymbol{S}_i(s) \text{ for } i \in I\}$.

The intuition behind our encoding remains the same as for the case of LTGDs. That is, the attack (s_i, s) models that the ID $i \in I$ must be satisfied for the source(i)-fact s and the self-attacks (s_i, s_i) enforce extensions to only contain arguments corresponding to facts. However, we only need attacks between arguments (instead of attacks from sets of arguments) as $\boldsymbol{S}_i(s) \subseteq \mathrm{target}(i)$ for each $i \in I$ and source(i)-fact s.

Example 4.6 *Consider $\mathcal{D} = \langle T, I \rangle$ with database $T = \{s,t,u,v\}$ and IDs $I := \{Sup_ID \subseteq Emp_ID, Covers_For \subseteq Dept\}$. For brevity, we denote IDs by $I = \{1,2\}$. The database and the supporting facts $\boldsymbol{S}iw$ for each $i \in I, w \in T$ are depicted in the table inside Fig. 7. Informally, a supervisor is also an employee, and each employee is assigned a department to cover if that department is short on employees. For example, $s(Sup_ID) = t(Emp_ID)$, $s(Covers_For) = t(Dept) = u(Dept)$, and therefore $\boldsymbol{S}_1(s) = \{t\}$, $\boldsymbol{S}_2(s) = \{t,u\}$. Then we have the AF $\mathcal{F}_\mathcal{D}$ as depicted in Fig. 7. The AF $\mathcal{F}_\mathcal{D}$ has a unique preferred extension, given by $\{s,t\}$. Clearly, this is also the only repair for $\mathcal{D}$.* ◁

It is worth mentioning that $\{s,t,u,v\}$ constitutes a naive extension for $\mathcal{F}_\mathcal{D}$ in Example 4.6, although this is not a repair for $\mathcal{D}$. Clearly, the semantics for IDs in $\mathcal{F}_\mathcal{D}$ requires admissibility (defending against attacking arguments).

Similar to the case of FDs, we observe that every ID is also an LTGD and hence the AF $\mathcal{F}_\mathcal{D}$ coincides with the SETAF $\mathcal{S}_\mathcal{D}$ for the constrained database $\mathcal{D}$. As a consequence, the results for LTGDs transfer to IDs, modulo a translation from SETAFs to AFs. In particular, this allows us to apply Theorem 3.9, which establishes the following claim.

Corollary 4.7. *Let $\mathcal{D} = \langle \mathcal{T}, I \rangle$ be a constrained database where I is a set of IDs and let $\mathcal{F}_\mathcal{D}$ denote the argumentation framework generated by $\mathcal{D}$. Then for $\mathcal{P} \subseteq \mathcal{T}$, $\mathcal{P} \in \mathrm{repairs}(\mathcal{D})$ iff $\mathcal{P} \in \mathrm{pref}(\mathcal{F}_\mathcal{D})$. Moreover, $\mathcal{F}_\mathcal{D}$ has exactly one preferred extension.*

The observation for LTGDs that some frameworks may not have any stable extension also applies to the special case of IDs and their corresponding AFs. As before, this holds because some arguments can neither be accepted in an extension nor attacked by arguments in an extension. The argument corresponding to the fact v in Example 4.6 depicts such an argument. Regarding the pre-processing, we still obtain the unique preferred extension, which is also naive and stable. Importantly, the set I of IDs does not have to be fixed to require the polynomial-time pre-processing. Following Remark 3.11, $\exists$-REP and $\forall$-REP is true for every $s \in \mathcal{T}$ such that $s \in \mathrm{PRE}(\mathcal{F}_\mathcal{D})$.

Example 4.8 (Continued) *Reconsider the constrained database $\mathcal{D} = \langle T, I \rangle$ from Example 4.6. Observe that the argument v is not defended against v_2 and therefore cannot be in a repair. The pre-processing removes $\{v, v_1, v_2\}$ and all the edges to/from arguments in this set. This has the consequence that all the arguments which are only defended by v are no longer defended (e.g., u). Consequently, the arguments $\{u, u_1, u_2\}$ have to be removed as well. After repeating the same process for u, we notice that no further argument needs to be removed. Hence, the set $\{s,t\}$ yields a repair for $\mathcal{D}$ as well as a σ-extension in the reduced AF for $\sigma \in \{\mathrm{naive}, \mathrm{stab}, \mathrm{pref}\}$.* ◁

Our construction of AFs for IDs relies on self-attacking arguments. However, these self-attacks are restricted to only auxiliary arguments and allow us to avoid taking these arguments in any extension. This results in obtaining a precise connection between repairs of a DB and extensions of the resulting AF. Without self-attacking arguments, one gets extensions that may contain auxiliary arguments, and thus, a precise connection between repairs and extensions might be lost. This is further highlighted in the following example.

Example 4.9 *Consider a single binary relation $T(x,y)$ in the schema, a constrained database $\mathcal{D} = \langle T, I \rangle$ with database $T = \{s,t,u,v\}$ and a single ID $T[x] \subseteq T[y]$. The database table, as well as its resulting AF $\mathcal{F}_\mathcal{D}$ (without self-attacks for auxiliary arguments), is depicted in Fig. 8. Observe that $\mathcal{F}_\mathcal{D}$ admits the following four preferred extensions: $\{\{s,t,u_1,v\}, \{s,t,u_1,v_1\}, \{s_1,t_1,u_1,v\}, \{s_1,t_1,u_1,v_1\}\}$.*

T	Emp_ID	Sup_ID	Dept.	Covers_For	S_1	S_2
s	JonX1	AxeK4	Production	Marketing	t	t,u
t	AxeK4	AxeK4	Marketing	Production	t	s
u	TimX3	JonX1	Marketing	Distribution	s,v	v
v	JonX1	AxeK4	Distribution	R&D	t	-

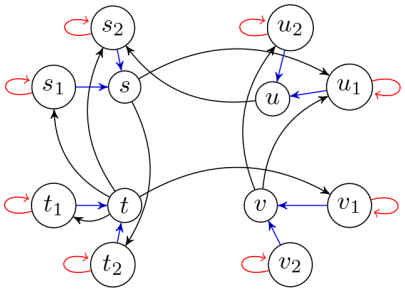


Fig. 7 The AF $\mathcal{F}_\mathcal{I}$ modelling $\mathcal{I}$ in Example 4.6: the red self-loops together with blue arcs depict the attacks for each fact $w \in T$ due to IDs $i \in I$ and the black arcs model the attacks due to the support set $\boldsymbol{S}_i(w)$.

However, there is a unique repair for $\mathcal{D}$ given as $\{s,t,v\}$.

Toward the end of our paper, in the next section, we will discuss how and whether we can model IDs via another translation that avoids self-attacking arguments.

4.3 Simulating FDs and IDs via AFs

Consider an instance $\mathcal{D} = \langle \mathcal{T}, B \rangle$ of a database $\mathcal{T}$ and a collection $B = D \cup I$ of FDs D and IDs I. We construct the AF $\mathcal{F}_\mathcal{D} = (A, R_\mathrm{D} \cup R_\mathrm{I})$ as follows.

- $A := \{s \mid s \in \mathcal{T}\} \cup \{s_i \mid s \text{ is a source}(i) - \text{fact for } i \in I\}$,
- $R_\mathrm{D} := \{(s,t),(t,s) \mid s,t \in \mathcal{T} \text{ and } \{s,t\} \not\models d \text{ for some } d \in D\}$,
- $R_\mathrm{I} := \{(s_i,s),(s_i,s_i) \mid s \text{ is a source}(i) - \text{fact for } i \in I\} \cup \{(t,s_i) \mid t \in \boldsymbol{S}_i(s) \text{ for } i \in I\}$.

As before, one may apply the pre-processing as a first step, thereby removing those facts from $\mathcal{T}$ failing some $i \in I$. However, the discussion regarding the case of LTGDs still applies for IDs. The following example illustrates that even if we apply pre-processing, some facts may not be accepted in combination with each other due to the presence of FDs.

Example 4.10 *Consider $\mathcal{D} = \langle T, B \rangle$ with database $T = \{s,t,u\}$ and ICs $B = D \cup I$ where $D = \{\mathrm{dep}(Sup_ID; Building)\}$ and $I = \{Covers_For \subseteq Dept\}$. Moreover, the database T and the support $\boldsymbol{S}_{Covers_For \subseteq Dept}(w)$ for each $w \in T$ is depicted in the table inside Fig. 9. Then, $\{s,t\} \not\models \mathrm{dep}(Sup_ID; Building)$, and $\{t,u\} \not\models \mathrm{dep}(Sup_ID; Building)$. The resulting AF $\mathcal{F}_\mathcal{D}$ is shown in Fig. 9, where the edges due to the IDs are depicted in red and blue. The only preferred extension for $\mathcal{F}_\mathcal{D}$ is $\{t\}$. Also, the only repair for $\mathcal{D}$ is $\{t\}$. Further, although $\{s,u\}$ is preferred for $\mathcal{F}_{\mathcal{D}'}$ where $\mathcal{D}' = \langle T, D \rangle$ (ignoring red and blue arcs), and $\{s,t,u\}$ is preferred for $\mathcal{F}_{\mathcal{D}''}$ where $\mathcal{D}'' = \langle T, I \rangle$ (ignoring black arcs), none of them is preferred for $\mathcal{F}_\mathcal{D}$.* ◁

Similar to their expressive counterparts, allowing both types of ICs (FDs and IDs) has the effect that the preferred and naive extensions do not coincide in general. Moreover, the repairs correspond to preferred semantics for AFs.

Example 4.11 (Cont.) *Reconsider the constrained database $\mathcal{D}$ from Example 4.10. Then, $\{s,u\}$ is a naive extension for $\mathcal{F}_\mathcal{D}$ but not preferred. Moreover, t is the only fact for which $\exists$-REP and $\forall$-REP is true.* ◁

Corollary 4.12. *Let $\mathcal{D} = \langle \mathcal{T}, B \rangle$ be a constrained database where B includes FDs and IDs. Further, let $\mathcal{F}_\mathcal{D}$ denote the argumentation framework generated by $\mathcal{D}$. Then for every subset $\mathcal{P} \subseteq \mathcal{T}$, $\mathcal{P} \in \mathrm{repairs}(\mathcal{D})$ iff $\mathcal{P} \in \mathrm{pref}(\mathcal{F}_\mathcal{D})$.*

Next, we establish the data and combined complexity results for REP, $\exists$-REP and $\forall$-REP. Observe that a fixed collection of integrity constraints suffices for establishing the hardness results (see Table 1). Whereas, the membership in each case also applies to the case when the set of constraints is given as input (hence not fixed).

Theorem 4.13. *The problems* REP *and* $\exists$-REP *are in* **NP** *for constraints that consist of FDs and IDs. Moreover, there exists a set B of FDs and IDs such that the problems* REP_B *and* $\exists$-REP_B *are both* **NP**-*hard.*

Proof The membership follows analogous to the proof of Theorem 3.14 in both cases. Let $\mathcal{D} = \langle \mathcal{T}, B \rangle$ be a constrained database where B consists of FDs and IDs. Further, let $s \in \mathcal{T}$ be a fact. Observe that for constraints involving FDs and IDs, the verification that a guessed set of facts satisfies B can be performed in polynomial time given $\mathcal{D}$ as input. Therefore, the **NP**-membership applies to REP and $\exists$-REP.

For hardness, we reduce from SAT to $\exists$-REP_B and REP_B simultaneously for a fixed set B of constraints. Let $\varphi := \{c_i \mid i \le m\}$ be a propositional formula in conjunctive normal form over propositions $X = \{p_1, \ldots, p_n\}$. We construct a database with two relations (We find it convenient to write explicit attribute names for each relation.): (i) a 4-ary relation $F(t_0, u_0, t_1, u_1)$, and (ii) a binary relation $C(t_2, u_2)$. Intuitively, F encodes the positive and negative participation of propositions in the clauses of φ. Precisely, a fact $F(x, v, c_i, \mathrm{sat})$ encodes that the proposition x appears positively in clause c_i when $v = 1$, and negatively when $v = 0$. The value "sat" appears in each fact to enforce a non-empty set of facts in a repair. The relation C is used to encode the set of clauses. With the help of IDs, we aim to enforce that if any C-fact belong to a repair, then all of the C-facts (a "chain") must belong to such a repair, which in turn forces (via further IDs) a repair of $\mathcal{T}$ to include F-facts in such a way that their corresponding literals satisfy every clause.

Formally, we will define F and G such that $\mathrm{dom}(F) \subseteq \{c_i \mid i \le m\} \cup \{p_1, \ldots, p_n\} \cup \{0, 1, \mathrm{sat}\}$ and $\mathrm{dom}(C) = \{c_i \mid i \le m\} \cup \{\mathrm{sat}\}$. Our database consists of the following facts:

$$\begin{aligned} \mathcal{T} = \; & \{F(x, 1, c_i, \mathrm{sat}) \mid x \in c_i, c_i \in \varphi\} \cup \{F(x, 0, c_i, \mathrm{sat}) \mid \neg x \in c_i, c_i \in \varphi\} \\ & \cup \{F(\mathrm{sat}, \mathrm{sat}, \mathrm{sat}, \mathrm{sat})\} \cup \{C(\mathrm{sat}, c_1), C(c_m, \mathrm{sat})\} \\ & \cup \{C(c_i, c_{i+1}) \mid 1 \le i < m\}. \end{aligned}$$

As ICs, we consider a collection B of FDs and IDs over the schema $\{F, C\}$ using $V := \{t_j, u_j \mid 0 \le j \le 2\}$ as the set of attributes (Recall that our notation for FDs and IDs uses attributes as names for the corresponding argument positions of the relation (here $F(t_0, u_0, t_1, u_1)$ and $C(t_2, u_2)$)). Precisely, we let

$$B = \{\mathrm{dep}(t_0; u_0)\} \cup \{F[u_1] \subseteq C[u_2], C[u_2] \subseteq C[t_2], C[t_2] \subseteq F[t_1], F[u_1] \subseteq F[t_1]\}.$$

The FD $\mathrm{dep}(t_0; u_0)$ over F ensures that each proposition (corresponding to facts in a repair) takes at most one value in $\{0, 1\}$. The IDs in B ensure that each clause $c_i \in \varphi$ is satisfied by modeling the following intuition. (I) The ID $F[u_1] \subseteq C[u_2]$, together with a dummy fact $F(\mathrm{sat}, \mathrm{sat}, \mathrm{sat}, \mathrm{sat})$ (referred to as s_d), encodes that any repair for $\mathcal{T}$ containing s_d triggers a non-empty subset of C-facts in a repair. (II) The ID $C[u_2] \subseteq C[t_2]$ triggers a chain of C-facts in a repair and thus enforces that either $\emptyset$, or all C-facts are considered in a repair. (III) The ID $C[t_2] \subseteq F[t_1]$ encodes that all clauses must be satisfied if a repair contains any C-fact. Moreover, (IV) the ID $F[u_1] \subseteq F[t_1]$ is used in the reduction for REP_B to enforce that a valid repair must contain the dummy fact s_d. Notice that the collection B does not

T	x	y	S_1
s	b	a	t
t	b	b	t
u	d	c	-
v	e	e	v

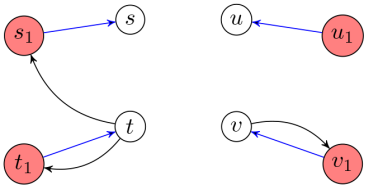


Fig. 8 Argumentation framework without self-attacks for modelling IDs in Example 4.9. The auxiliary arguments are highlighted in red for convenience.

T	Emp_ID	Sup_ID	Dept.	Building	Covers_for	S_i
s	JonX1	AxeK4	Production	B4	Sales	t
t	TimX3	AxeK4	Sales	B4	Sales	t
u	AxeK4	AxeK4	Marketing	B4	Production	s

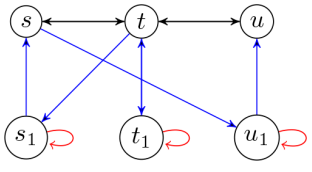


Fig. 9 Argumentation framework for modelling dependencies in Example 4.10. Black arcs depict conflicts due to functional, and blue ones due to inclusion dependency.

depend on the formula φ, hence it is fixed.

We next highlight some observations before delving into the correctness proof. Clearly, $\mathcal{D}$ is inconsistent due to the presence of at least one pair of facts $F(x,1,c_i,\text{sat}), F(x,0,c_j,\text{sat})$ and the FD $\text{dep}(t_0;u_0)$ (based on the fact that at least one variable x appears positively in one clause c_i and negatively in another clause c_j, since otherwise φ is trivially satisfiable). Then, a set P_X of F-facts satisfies $\text{dep}(t_0;u_0)$ if for each proposition $x \in X$: P_X contains facts of the form either $F(x,1,*,\text{sat})$ or $F(x,0,*,\text{sat})$ but not both (here $*$ indicates the occurrence of any value in a fact). For P_X to be a candidate repair among F-facts, it additionally has to contain facts of the form $F(x,*,*,*)$ for each $x \in X$. Moreover, a set P_c of C-facts satisfies the ID $C[u_2] \subseteq C[t_2]$ iff either $P_c = \emptyset$ or P_c contains all C-facts. Next, we consider a set P_X of F-facts containing the fact s_d and let $\mathcal{P} = P_X \cup P_c$. It follows that $\mathcal{P} \models F[u_1] \subseteq C[u_2]$ iff $P_c \neq \emptyset$ and hence we let P_c contain all C-facts. Finally, $\mathcal{P} \models C[t_2] \subseteq F[t_1]$ iff for each $c_i \in \varphi$, P_X contains at least one fact $F(x,v,c_i,\text{sat})$ such that $v = 1$ if $x \in c_i$ and $v = 0$ if $\neg x \in c_i$. This is guaranteed since $\mathcal{P}$ contains facts of the form $C(*,c)$ for each clause $c \in \varphi$ and thus there must exist F-facts in $\mathcal{P}$ also of the form $F(*,*,c,*)$ for each clause $c \in \varphi$. We prove the correctness via the following claim.

Claim 4.14. *φ is satisfiable if and only if* $\exists$-REP *is true for the fact s_d.*

Proof of claim "$\Longrightarrow$". Suppose φ is satisfiable and let θ be a satisfying assignment. We let $\mathcal{P} := \{F(x,1,c_i,\text{sat}) \mid \theta(x) = 1, \text{ for } x \in X \text{ and } i \leq m\} \cup \{F(x,0,c_i,\text{sat}) \mid \theta(x) = 0, \text{ for } x \in X \text{ and } i \leq m\} \cup \{s_d\} \cup P_c$ where P_c contains all C-facts in $\mathcal{T}$. Then $\mathcal{P} \models \text{dep}(t_0;u_0)$ because for any $x \in X$: $\mathcal{P}$ contains either facts of the form $F(x,1,*,*)$ or $F(x,0,*,*)$, since θ is an assignment over X. Next, we have that (I) $\mathcal{P} \models F[u_1] \subseteq C[u_2]$ since $P_c \neq \emptyset$, and (II) $\mathcal{P} \models C[u_2] \subseteq C[t_2]$ since $\mathcal{P}$ contains all C-facts. Finally, $\mathcal{P}$ contains at least one fact of the form $F(*,*,c,*)$ for each clause $c \in \varphi$ since $\theta(\ell) = 1$ for at least one literal $\ell \in c$ in each $c \in \varphi$. This holds because for every clause $c \in \varphi$, $\mathcal{P}$ contains a fact $F(x,v,c,\text{sat})$ for some variable x appearing in c. The value v in this fact depends on how the assignment θ satisfies c, that is, $v = 1$ if $\theta(x) = 1$ makes c true, and $v = 0$ if $\theta(x) = 0$ does. As a consequence, $\mathcal{P} \models C[t_2] \subseteq F[t_1]$. Observe that $\mathcal{P} \models F[u_1] \subseteq F[t_1]$ follows due to the fact $s_d \in \mathcal{P}$. To see why $\mathcal{P}$ is a repair (i.e., subset-maximal), we note that the only facts not in $\mathcal{P}$ are of the form $F(x,v,c,\text{sat})$ for some $v \in \{0,1\}$ and $c \in \varphi$ such that $F(x,1-v,c,\text{sat}) \in \mathcal{P}$ due to the way $\mathcal{P}$ is defined. However, adding such a fact would violate the FD. Therefore, $\mathcal{P}$ is indeed a repair. We conclude by observing that $s_d \in \mathcal{P}$.

"$\Longleftarrow$". Let $\mathcal{P}$ be a repair and $s_d \in \mathcal{P}$. From $\mathcal{P} \models F[u_1] \subseteq C[u_2]$, it follows that $\mathcal{P}$ must contain at least one C-fact as $s_d \in \mathcal{P}$, and from $\mathcal{P} \models F[u_1] \subseteq C[u_2]$, it follows that $\mathcal{P}$ actually contains all C-facts. We define an assignment θ over X by setting $\theta(x) = 1$ if $\mathcal{P}$ contains $F(x,1,*,*)$ and $\theta(x) = 0$ if it contains $F(x,0,*,*)$. Since $\mathcal{P} \models \text{dep}(t_0;u_0)$, the assignment θ is well-defined, i.e., there is no $x \in X$ for which we assign both $\theta(x) = 1$ and $\theta(x) = 0$. Now, θ is defined for all the propositions in X since $\mathcal{P}$ is subset-maximal (hence, it contains fact of the form either $F(x,0,*,*)$ or $F(x,1,*,*)$ for each $x \in X$). Now, we prove that $\theta \models \varphi$. To this aim, let $c \in \varphi$ be any clause. Since $\mathcal{P} \models C[t_2] \subseteq F[t_1]$, $\mathcal{P}$ contains at least one fact of the form $F(x,v,c,\text{sat})$ for some variable x and $v \in \{0,1\}$, such that $v = 1$ if $x \in c$ and $v = 0$ if $\neg x \in c$ (by definition of how F-facts are constructed). Therefore, we must have set $\theta(x) = 1$ if $v = 1$ (hence $x \in c$) and $\theta(x) = 0$ if $v = 0$ ($\neg x \in c$). As a result, $\theta \models c$ for each $c \in \varphi$. This proves that $\theta \models \varphi$ and our claim follows. ■

To reduce SAT into REP_B, we observe that $\mathcal{P} \models F[u_1] \subseteq F[t_1]$ iff either $\mathcal{P}$ contains no F-fact, or it contains the fact s_d. However, there cannot be a repair for $\mathcal{D}$ without any F-fact but still containing C-facts, since it violates the ID $C[t_2] \subseteq F[t_1]$. As a result, one cannot construct a non-empty repair for $\mathcal{D}$ by excluding s_d. Therefore, every repair $\mathcal{P}$ for $\mathcal{D}$ necessarily contains s_d, thereby proving the equivalence as before. In other words, there is a non-empty repair for $\mathcal{D}$ iff φ is satisfiable.

This completes the proof for both cases. □

We provide an example for a better understanding of the reductions from the proof of Theorem 4.13.

Example 4.15 (Example with fixed ICs) *Let $\varphi := \{x \vee y, \neg x \vee \neg y, \neg x \vee y\}$ be a propositional formula. Our reduction for* $\exists$-REP *gives a database $\mathcal{T}$ with two tables F and C for the collection $B := \{\text{dep}(t_0;u_0)\} \cup \{F[u_1] \subseteq C[u_2], C[u_2] \subseteq C[t_2], C[t_2] \subseteq F[t_1]\}$ of dependencies.*

Observe that the only satisfying assignment for φ is given by $\{x \mapsto 0, y \mapsto 1\}$, which corresponds to the repair $\{S, \bar{x}_2, \bar{x}_3, y_1, y_3\} \cup \{s_c, s_1, s_2, s_3\}$ for $\mathcal{D}$ containing S. Consequently $\exists$-REP *is true for the fact s_d.* ◁

Next, we prove that $\forall$-REP is even harder and $\mathbf{\Pi}_2^\mathbf{P}$-complete.

Table 3. The database corresponding to the formula φ from Example 4.15.

F	t_0	u_0	t_1	u_1
s_d	sat	sat	sat	sat
x_1	x	1	c_1	sat
$\bar{x}_2$	x	0	c_2	sat
$\bar{x}_3$	x	0	c_3	sat
y_1	y	1	c_1	sat
$\bar{y}_2$	y	0	c_2	sat
y_3	y	1	c_3	sat

C	t_2	u_2
s_c	sat	c_1
s_1	c_1	c_2
s_2	c_2	c_3
s_3	c_3	sat

Theorem 4.16. *The problem* $\forall$-REP *is contained in $\mathbf{\Pi}_2^\mathbf{P}$ for FDs and IDs. Moreover, there exists a set B of FDs and IDs such that the problem $\forall$-REP_B is $\mathbf{\Pi}_2^\mathbf{P}$-hard.*

Proof The membership follows analogously to the proof of Theorem 3.15. The only difference here is that one can check in polynomial time whether a set of facts satisfies each IC in the input. As a result, the membership holds for $\forall$-REP (i.e., in the combined complexity).

For hardness, we build on the same idea as in the proof of Theorem 4.13. Here, we reduce from an instance Φ of the $\mathbf{\Pi}_2^\mathbf{P}$-complete problem 2QBF. The problem 2QBF is defined as follows: given a propositional formula $\Phi = \forall Y \exists Z \varphi(Y,Z)$ where $\varphi := \{c_i \mid 1 \leq i \leq m\}$ is a CNF formula, determine whether Φ is true (i.e., whether for every assignment to the variables in Y, there exists an assignment to the variables in Z that satisfies φ). We let $X = Y \cup Z$ and construct a constrained database $\mathcal{D}$ with three relations: (i) a 4-ary relation $F(t_0,u_0,t_1,t_\exists)$, (ii) a binary relation $S(u_1,u_\exists)$, and another (iii) binary-relation $C(t_2,u_2)$. Moreover, we consider a collection B of FDs and IDs over the schema of $\mathcal{T}$ using a set $V := \{t_j,u_j \mid 0 \leq j \leq 2\} \cup \{t_\exists,u_\exists\}$ of attributes. B contains one FD $\text{dep}(t_0;u_0)$ similar to the proof of Theorem 4.13. Due to the change in database relations, we adapt the collection of IDs and consider the set $\{S[u_1] \subseteq C[u_2], C[u_2] \subseteq C[t_2], C[t_2] \subseteq F[t_1]\}$ to encode whether each clause $c_i \in \varphi$ is satisfied. Moreover, we require an additional ID $F[t_\exists] \subseteq S[u_\exists]$ to encode the existentially quantified variables Z. As a result, we obtain $B = \{\text{dep}(t_0;u_0)\} \cup \{S[u_1] \subseteq C[u_2], C[u_2] \subseteq C[t_2], C[t_2] \subseteq F[t_1], F[t_\exists] \subseteq S[u_\exists]\}$ as our fixed collection of ICs.

For the database relations, we let $\text{dom}(F) \subseteq \{c_i \mid i \leq m\} \cup X \cup \{0,1,\text{exists},d\}$, $\text{dom}(C) = \{c_i \mid i \leq m\} \cup \{d\}$ and $\text{dom}(S) = \{c_1,\text{exists},d\}$. Then our encoding works mostly similar to that in the proof of Theorem 4.13, except for the minor differences outlined below.

1. We include an S-fact to trigger a non-empty set of C-facts in any

repair. We obtain this by adding the fact $S(c_1, \text{exists})$ together with the fact $C(c_m, c_1)$. Here, we do not need the fact $C(\text{sat}, c_1)$ as in the proof of Theorem 4.13. The remaining C-facts $\{C(c_i, c_{i+1}) \mid i < m\}$ are added as before.

2. We use $s^d := S(d,d), y^d := F(d,d,d,d)$ and $c^d := C(d,d)$ as dummy facts to simulate the effect that any assignment over Y yields a collection of facts in $\mathcal{T}$ that *trivially* satisfies all the ICs. This is required to simulate the $\forall$ quantification over Y.

3. If a literal $\ell \in \text{Lit}(Y)$ does not appear in even a single clause, we still add its corresponding F-fact following the same intuition as before, i.e., for such a literal ℓ: we add $F(y,1,d,d)$ if $\ell = y$ and $F(y,0,d,d)$ if $\ell = \neg y$. Moreover, the dummy value d in this case encodes that ℓ does not satisfy any clause. As a result of this, we can connect repairs to all assignments over universally quantified variables in Y. For the ease of notation, we write $\ell \in c_0$ for a literal over Y that does not appear in any clause.

4. The attributes $\{t_\exists, u_\exists\}$ encode that the S-fact $s_{\text{sat}} := S(c_1, \text{exists})$ *supports* facts involving propositions $z \in Z$ via the inclusion dependency $F[t_\exists] \subseteq S[u_\exists]$. This is achieved by letting facts $F(z, *, *, \text{exists})$ for each $z \in Z$ together with the fact s_{sat}. However, all facts corresponding to variables $y \in Y$ take the form $F(y, *, *, d)$ for the dummy value d.

Observe that the construction in (3) and (4) requires us to distinguish facts based on variables in Y and Z. Formally, our database consists of the following facts:

$$\begin{aligned}\mathcal{T} = \; & \{F(y,1,c_i,d) \mid y \in Y \cap c_i, c_i \in \varphi\} \cup \{F(y,0,c_i,d) \mid y \in Y, \neg y \in c_i, c_i \in \varphi\} \\ & \cup \{F(y,1,d,d) \mid \text{ no clause in } \varphi \text{ contains the literal } y \in Y\} \\ & \cup \{F(y,0,d,d) \mid \text{ no clause in } \varphi \text{ contains the literal } \neg y \text{ for } y \in Y\} \\ & \cup \{F(z,1,c_i,\text{exists}) \mid z \in Z \cap c_i, c_i \in \varphi\} \cup \{F(z,0,c_i,\text{exists}) \mid \\ & z \in Z, \neg z \in c_i, c_i \in \varphi\} \\ & \cup \{F(d,d,d,d)\} \cup \{S(c_1,\text{exists}), S(d,d))\} \\ & \cup \{C(c_i,c_{i+1}) \mid 1 \le i < m\} \cup \{C(c_m,c_1), C(d,d)\}.\end{aligned}$$

For correctness, notice that every assignment I_Y over Y (seen as a subset of Y) corresponds to a set $P_Y = \{F(y,1,c_i,d) \mid y \in I_Y \text{ and } y \in c_i \text{ for } 0 \le i \le m\} \cup \{F(y,0,c_i,d) \mid y \notin I_Y \text{ and } \neg y \in c_i \text{ for } 0 \le i \le m\}$ of F-facts in $\mathcal{T}$. Moreover, $P_Y \models \text{dep}(t_0; u_0)$ since I_Y is an assignment over Y and thus for each $i \ne j \le m$, P_Y includes facts of the form either $F(y,1,c_i,d)$ or $F(y,0,c_j,d)$ but not both, for each variable $y \in Y$ and clause c_i, c_j. Furthermore, it is easy to observe that taking the dummy facts, i.e., $P_D = \{y^d, s^d, c^d\}$ the collection $P_Y \cup P_D$ satisfies the IDs $S[u_1] \subseteq C[u_2]$ and $C[u_2] \subseteq C[t_2]$. Now, we let $\mathcal{P}_Y = P_Y \cup P_D$. We have that $s_{\text{sat}} \notin \mathcal{P}_Y$. In order to extend $\mathcal{P}_Y$ by adding $F(z,1,c_i,\text{exists})$ or $F(z,0,c_i,\text{exists})$ for any $z \in Z$ and $i \le m$, s_{sat} must be added as well due to the ID $F[t_\exists] \subseteq S[u_\exists]$. However, in order to include s_{sat} in any repair, we have to find I_Y and I_Z that together satisfy φ due to the remaining IDs, in particular due to $C[t_2] \subseteq F[t_1]$. As a result, for any interpretation I_Y over Y, there is an interpretation I_Z over Z such that: $I_Y \cup I_Z \models \varphi$ if and only if $\mathcal{P}_Y$ is not a repair for $\mathcal{D}$ (since it can be extended by adding facts $P_Z = \{F(z,1,c_i,d) \mid z \in I_Z \text{ and } z \in c_i \text{ for } 0 \le i \le m\} \cup \{F(z,0,c_i,d) \mid z \notin I_Z \text{ and } \neg z \in c_i \text{ for } 0 \le i \le m\}$ for such an interpretation I_Z in this case). Equivalently, there is a repair for $\mathcal{D}$ not containing s_{sat} if and only if the formula Φ is false. We conclude by observing that Φ is true if and only if every repair for $\mathcal{D}$ contains s_{sat} if and only if $\forall$-REP is true for s_{sat}. □

We provide an example for a better understanding of the reduction from the proof of Theorem 4.16.

Example 4.17 *Let* $\Phi = \forall x \forall y \exists z \exists w((x \vee y \vee z) \wedge (y \vee \neg z \vee \neg w) \wedge (y \vee z \vee w))$ *be a* 2QBF. *Then, our reduction yields a database* $\mathcal{T}$ *with three database tables as depicted in Table 4 and the (fixed) collection* $B := \{\text{dep}(t_0; u_0)\} \cup \{S[u_1] \subseteq C[u_2], C[u_2] \subseteq C[t_2], C[t_2] \subseteq F[t_1], F[t_\exists] \subseteq S[u_\exists]\}$ *of dependencies. The reader can verify that the formula* Φ *is true and that for* s_{sat}, $\forall$-REP *is true as well. For instance, the assignment* $\{x \mapsto 0, y \mapsto 0\}$ *results in* $I_Y = \emptyset$ *and thus* $P_Y = \{\bar{x}_0, \bar{y}_0\}$. *Then, although* $P_Y \cup \{y^d, s^d, c^d\}$ *satisfies each dependency in* B, *it is not a repair since it is not subset-maximal. A repair is obtained by adding further the facts* $\{z_1, z_3, \bar{w}_2\} \cup \{S, s_1, s_2, s_3\}$. ◁

5 Concluding remarks

Overview. We simulated the problem of finding repairs of an inconsistent database under various families of ICs by Dung's (set-based) argumentation frameworks. Our main results (see Table 1) indicate that subset-maximal repairs correspond to naive extensions when only one type of dependency is allowed, whereas only preferred extensions yield all the repairs when both types are allowed. Note that the case of LTGDs (or IDs) requires a polynomial-time pre-processing to eliminate all the non-accepted arguments from the resulting (SET)AFs. The result of this yields a unique preferred extension that is also stable and naive.

Moreover, the following interesting facts can be derived: "FDs can be modeled via AFs" in a similar way in which "DCs can be modeled via SETAFs". Analogous results hold when we replace FDs and DCs by (1) IDs and LTGDs, respectively, or (2) FDs + IDs and DCs + LTGDs, respectively. Furthermore, for the problem to determine whether a tuple is in some (resp., every) repair, we establish the same complexity bounds as the complexity of credulous (skeptical) reasoning for preferred semantics in AFs. Interestingly, the combined complexity with FDs+IDs remains the same as the data complexity with DCs+LTGDs for the considered problems.

5.1 The role of self-attacking arguments

Observe that our construction of (SET)AFs for LTGDs and IDs relies on self-attacking auxiliary arguments. Our motivation lies in avoiding those arguments in any extension of the resulting (SET)AF and connecting repairs to extensions directly. In the absence of self-attacking arguments, extensions may contain auxiliary arguments and therefore a precise connection between repairs and extensions cannot be established (see Example 4.9).

We find it worth highlighting that considerable thought has been given to how LTGDs/IDs can be modeled in an argumentation framework. Intuitively, the semantics of inclusion dependencies closely resemble the notion of "necessary support" in the argumentation literature[42,43,37]. If a DB instance contains only one inclusion dependency (i) per relation, we can simply model the set of supporting facts in

Table 4. The database corresponding to the 2QBF instance Φ from Example 4.17.

S	u_1	$u_\exists$
s_{sat}	c_1	exists
s^d	d	d

F	t_0	u_0	t_1	$t_\exists$
y^d	d	d	d	d
x_1	x	1	c_1	d
$\bar{x}_0$	x	0	d	d
y_1	y	1	c_1	d
y_2	y	1	c_2	d
y_3	y	1	c_3	d
$\bar{y}_0$	y	0	d	d
z_1	z	1	c_1	exists
$\bar{z}_2$	z	0	c_2	exists
z_3	z	1	c_3	exists
$\bar{w}_2$	w	0	c_2	exists
w_3	w	1	c_3	exists

C	t_2	u_2
s_1	c_1	c_2
s_2	c_2	c_3
s_3	c_3	c_1
c^d	d	d

$\boldsymbol{S}_i(s)$ for a fact s via the *support* relation between arguments. However, we observed that this does not apply to the case when an instance $\mathcal{D}$ contains multiple IDs with the same relation as their source. Modeling this requires as many (distinct) support relations as the number of IDs. This holds since for each fact s, one would have to distinguish the satisfaction of individual IDs (i.e., the sets $\boldsymbol{S}_i(s)$ and $\boldsymbol{S}_j(s)$ for IDs $i \neq j$) which a single support relation cannot offer. Our translation captures the same essence of supporting arguments via auxiliary attackers, with the benefit that one can distinguish supporters and attackers corresponding to each inclusion dependency. As future work, we aim to think further in this direction and explore ideas that might lead to improved translations for IDs and LTGDs.

5.2 Discussion and future work

We would like to point out that, although a correspondence between subset-maximal repairs in the presence of functional dependencies (resp., denial constraints) and extensions for (SET)AFs is known[13], the main contributions of our work establish the correspondence when inclusion dependencies or LTGDs are also allowed. This novel contribution opens up several directions for future work. First and foremost, the authors believe that the connection between repairs in the setting of inconsistent databases and extensions in AFs is stronger than what is established here. Intuitively, one can model the attack relationship via functional dependencies and defense/support via inclusion dependencies. A transformation based on this intuition was recently presented in ref.[37], showing that AFs can be simulated via inconsistent databases considering FDs and IDs.

Further future work may consider whether the connection between inconsistent databases and argumentation can be generalized to other well-known types of expressive ICs. This is particularly interesting, keeping in mind that the current paper extends an earlier work for FDs and IDs[38] to certain expressive ICs toward *both sides* of Fig. 1. Currently, the authors believe that to model arbitrary (or full) tuple-generating dependencies, one needs the so-called hyper argumentation frameworks[44], allowing attacks between two sets of arguments. Furthermore, one can also target the richer setting of universal constraints and the symmetric difference repairs[26]. Another interesting question is to explore whether the lower bounds for complexity also apply to the sub-classes of FDs and IDs, namely *keys* and *foreign keys*, as well as the particular case of acyclic dependencies. Moreover, we would like to explore whether consistent query answering (CQA) under inconsistency-tolerant semantics can also be tackled via the argumentation approach. Finally, one can consider incorporating information about priorities among tuples into the resulting AFs, that is, extending the translations presented in this work to the setting of prioritized repairing and consistent query answering[23,24,45]. Here, ref.[13] has already considered FDs and DCs, thus the question remains open only for the case of IDs and LTGDs.

Another promising direction to consider next is the exploration of an explainability dimension, similar to that in the setting of ontological KBs[8,10,46]. Given an instance $\mathcal{D}$ including a database $\mathcal{T}$ and a collection B of dependencies, then the proposed AF $\mathcal{F}_{\mathcal{D}}$ lets one determine the *causes* why some tuples are not in some repair (or all repairs). We note that, for IDs, the auxiliary arguments modeling each dependency in B can serve this purpose. For FDs, we believe that annotating arguments (or the attack relation between a pair of arguments) by the FDs involved in the conflict can achieve the goal. As a result, one can look at the AF $\mathcal{F}_{\mathcal{D}}$ and read from it the FDs or IDs that a tuple s failing $\exists$-REP or $\forall$-REP participates in. Then, subsets of the atoms and/or possibly tuples in a database can be considered as *explanations*. Such explanations seem interesting in modeling scenarios where the data (database) has higher confidence than the dependencies; for example, if dependencies are mined over some part of the existing data. An explanation then informs that the data (and hence tuples therein) should be kept, whereas dependencies need to be screened and further analyzed.

Author contributions

All authors contributed to the discussions. The first three authors contributed significantly to the technical writing phase of the paper.

Data availability

Data sharing not applicable as the paper does not include an experimental evaluation.

Acknowledgments

We thank all anonymous reviewers for their valuable feedback. The manuscript has undergone substantial revision in response to their detailed comments, which have significantly improved the quality of the work. Research was funded by the German Research Foundation (DFG), grants TRR 318/3 2026–438445824 and VI 1045-1/1–432788559, the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the project WHALE (LFN 1-04) funded under the Lamarr Fellow Network programme and project SAIL, grant NW21-059D, and by the German Federal Ministry of Research, Technology and Space (BMFTR) within the project KI-Akademie OWL under grant 16IS24057B.

Conflict of interest

The authors declare that they have no conflict of interest.

Dates

Received 12 December 2025; Revised 12 December 2025; Accepted 10 March 2026; Published online 29 June 2026

References

[1] Abiteboul S, Hull R, Vianu V. 1995. *Foundations of databases*. Boston, USA: Addison-Wesley. http://webdam.inria.fr/Alice

[2] ten Cate B, Fontaine G, Kolaitis PG. 2012. On the data complexity of consistent query answering. *ICDT'12: Proceedings of the 15th International Conference on Database Theory, Berlin, Germany, 26–29 March 2012*. New York, NY, USA: Association for Computing Machinery. pp. 22–33 doi: 10.1145/2274576.2274580

[3] Barceló P, Fontaine G. 2017. On the data complexity of consistent query answering over graph databases. *Journal of Computer and System Sciences* 88:164−194

[4] Wijsen J. 2003. Condensed representation of database repairs for consistent query answering. *Proceedings of the 9th International Conference on Database Theory*, ICDT'03. Berlin, Heidelberg: Springer-Verlag. pp. 378–393 doi: 10.5555/645505.656435

[5] Lopatenko A, Bertossi LE. 2007. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. *ICDT 2007: Proceedings of the 11th International Conference on Database Theory, Barcelona, Spain, 10–12 January 2007*, *Lecture Notes in Computer Science*, eds. Schwentick T, Suciu D. Berlin, Vol. 4353. Heidelberg: Springer. pp. 179–193 doi: 10.1007/11965893_13

[6] Dung PM. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321−357

[7] Nielsen SH, Parsons S. 2006. A generalization of Dung's abstract framework for argumentation: arguing with sets of attacking arguments. *Argu-*

mentation in Multi-Agent Systems, Third International Workshop, ArgMAS 2006, Hakodate, Japan, 8 May 2006, Revised Selected and Invited Papers, Lecture Notes in Computer Science, eds. Maudet N, Parsons S, Rahwan I. Vol. 4766. Berlin, Heidelberg: Springer. pp. 54–73 doi: 10.1007/978-3-540-75526-5_4

[8] Arioua A, Croitoru M. 2016. Dialectical characterization of consistent query explanation with existential rules. *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, Key Largo, Florida, USA, 16–18 May 2016*, eds. Markov Z, Russell I. Washington D.C., USA: AAAI Press. pp. 621–625. www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12800

[9] Arioua A, Croitoru M, Vesic S. 2017. Logic-based argumentation with existential rules. *International Journal of Approximate Reasoning* 90:76−106

[10] Arioua A, Tamani N, Croitoru M. 2015. Query answering explanation in inconsistent datalog +/- knowledge bases. *Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, 1–4 September 2015, Proceedings, Part I. Lecture Notes in Computer Science*, eds. Chen Q, Hameurlain A, Toumani F, Wagner RR, Decker H. Vol. 9261. Cham: Springer. pp. 203–219 doi: 10.1007/978-3-319-22849-5_15

[11] Young AP, Modgil S, Rodrigues O. 2016. Prioritised default logic as argumentation with partial order default priorities. arXiv:1609.05224

[12] Yun B, Vesic S, Croitoru M. 2020. Sets of attacking arguments for inconsistent datalog knowledge bases. *Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, 4–11 September 2020. Frontiers in Artificial Intelligence and Applications*, eds. Prakken H, Bistarelli S, Santini F, Taticchi C. Vol. 326. Amsterdam, NL: IOS Press. pp. 419–430 doi: 10.3233/FAIA200526

[13] Bienvenu M, Bourgaux C. 2020. Querying and repairing inconsistent prioritized knowledge bases: complexity analysis and links with abstract argumentation. *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, 12–18 September 2020*, eds. Calvanese D, Erdem E, Thielscher M. California: IJCAI. pp. 141–151 doi: 10.24963/KR.2020/15

[14] Arieli O, Borg A, Heyninck J. 2019. A review of the relations between logical argumentation and reasoning with maximal consistency. *Annals of Mathematics and Artificial Intelligence* 87(3):187−226

[15] Afrati FN, Kolaitis PG. 2009. Repair checking in inconsistent databases: algorithms and complexity. *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, 23–25 March 2009, Proceedings. ACM International Conference Proceeding Series*, ed. Fagin R. Vol. 361. New York, NY, USA: ACM. pp. 31–41 doi: 10.1145/1514894.1514899

[16] Arenas M, Bertossi LE, Chomicki J. 2001. Scalar aggregation in FD-inconsistent databases. *Database Theory - ICDT 2001, 8th International Conference, London, UK, 4–6 January 2001, Proceedings, Vol. 1973 of Lecture Notes in Computer Science*, eds. Van den Bussche J, Vianu V. Berlin, Heidelberg: Springer. pp. 39–53 doi: 10.1007/3-540-44503-X_3

[17] Chomicki J, Marcinkowski J. 2005. Minimal-change integrity maintenance using tuple deletions. *Information and Computation* 197(1):90–121

[18] Hannula M, Wijsen J. 2022. A dichotomy in consistent query answering for primary keys and unary foreign keys. *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, 12–17 June 2022*, eds. Libkin L, Barceló New York, NY, USA: ACM. pp. 437–449 doi: 10.1145/3517804.3524157

[19] Livshits E, Kimelfeld B, Roy S. 2020. Computing optimal repairs for functional dependencies. *ACM Transactions on Database Systems* 45(1):1–46

[20] Staworko S, Chomicki J. 2010. Consistent query answers in the presence of universal constraints. *Information Systems* 35(1):1−22

[21] Bertossi L. 2006. Consistent query answering in databases. *SIGMOD Record* 35:68−76

[22] Bertossi LE. 2019. Database repairs and consistent query answering: origins and further developments. *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, 30 June–5 July 2019*, eds. Suciu D, Skritek S, Koch C. New York, NY, USA: ACM. pp. 48–58 doi: 10.1145/3294052.3322190

[23] Kimelfeld B, Livshits E, Peterfreund L. 2017. Detecting ambiguity in prioritized database repairing. *20th International Conference on Database Theory (ICDT 2017), Leibniz International Proceedings in Informatics (LIPIcs)*, eds. Benedikt M, Orsi G. Vol. 68. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. pp. 17:1–17:20 doi: 10.4230/LIPIcs.ICDT.2017.17

[24] Kimelfeld B, Livshits E, Peterfreund L. 2020. Counting and enumerating preferred database repairs. *Theoretical Computer Science* 837:115−157

[25] 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

[26] Bienvenu M, Bourgaux C. 2023. Inconsistency handling in prioritized databases with universal constraints: complexity analysis and links with active integrity constraints. *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, Rhodes, Greece. 2–8 September 2023*. California: IJCAI Organization. pp. 97–106 doi: 10.24963/kr.2023/10

[27] Väänänen J. 2007. *Dependence logic*. Cambridge, England: Cambridge University Press. https://home.hvl.no/prosjekter/easllc2012/docs/vaananen/dependence_logic_short_version.pdf

[28] Hannula M, Hella L. 2022. Complexity thresholds in inclusion logic. *Information and Computation* 287:104759

[29] Mahmood Y. 2022. *Parameterized aspects of team-based formalisms and logical inference*. Thesis. Institutional Repository of Leibniz Universität, Hannover, Germany.

[30] Arioua A, Tamani N, Croitoru M, Buche P. 2014. Query failure explanation in inconsistent knowledge bases using argumentation. *Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, 9–12 September 2014. Vol. 266 of Frontiers in Artificial Intelligence and Applications*, eds. Parsons S, Oren N, Reed C, Cerutti F. Amsterdam, The Netherlands: IOS Press. pp. 101−108 doi: 10.3233/978-1-61499-436-7-101

[31] Croitoru M, Thomopoulos R, Vesic S. 2015. Introducing preference-based argumentation to inconsistent ontological knowledge bases. *PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, 26–30 October 2015, Proceedings. Vol. 9387 of Lecture Notes in Computer Science*, eds. Chen Q, Torroni P, Villata S, Yung-jen Hsu J, Omicini A. Cham: Springer. pp. 594–602 doi: 10.1007/978-3-319-25524-8_42

[32] Croitoru M, Vesic S. 2013. What can argumentation do for inconsistent ontology query answering? *Scalable Uncertainty Management - 7th International Conference, SUM 2013, Proceedings. Washington, DC, USA, 16–18 September 2013. Vol. 8078 of Lecture Notes in Computer Science*, eds. Liu W, Subrahmanian VS, Wijsen J. Berlin, Heidelberg: Springer. pp. 15–29 doi: 10.1007/978-3-642-40381-1_2

[33] Ho L, Arch-int S, Acar E, Schlobach S, Arch-int N. 2022. An argumentative approach for handling inconsistency in prioritized datalog± ontologies. *AI Communications* 35:243−267

[34] König M, Rapberger A, Ulbricht M. 2022. Just a matter of perspective. In *Computational models of argument*, eds. Toni F, Polberg S, Booth R, Caminada M, Kido H. Amsterdam, The Netherlands: IOS Press. pp. 212–223 doi: 10.3233/FAIA220154

[35] Bondarenko A, Dung PM, Kowalski RA, Toni F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *Artificial intelligence* 93(1−2):63−101

[36] Dvořák W, Woltran S. 2020. Complexity of abstract argumentation under a claim-centric view. *Artificial Intelligence* 285:103290

[37] Mahmood Y, Hecher M, Ngomo A-CN. 2025. Dung's argumentation framework: unveiling the expressive power with inconsistent databases. *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025*, eds. Walsh T, Shah J, Kolter Z. Washington, DC, USA: AAAI Press. pp. 15058–15066 doi: 10.1609/AAAI.V39I14.33651

[38] Mahmood Y, Virtema J, Barlag T, Ngomo A-CN. 2024. Computing

repairs under functional and inclusion dependencies via argumentation. *International Symposium on Foundations of Information and Knowledge Systems*, eds. Meier A, Ortiz M. Cham: Springer. pp. 23–42 doi: 10.1007/978-3-031-56940-1_2

[39] Arming S, Pichler R, Sallinger E. 2016. Complexity of repair checking and consistent query answering. *19th International Conference on Database Theory (ICDT 2016). Leibniz International Proceedings in Informatics (LIPIcs)*. Vol. 48. Schloss Dagstuhl: Leibniz-Zentrum für Informatik. pp. 21:1–21:18 doi: 10.4230/LIPIcs.ICDT.2016.21

[40] Dvorák W, Dunne PE. 2017. Computational problems in formal argumentation and their complexity. *FLAP* 4(8):21:1−21:18

[41] Coste-Marquis S, Devred C, Marquis P. 2005. Symmetric argumentation frameworks. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU 2005, Barcelona, Spain, 6–8 July 2005, Proceedings. Lecture Notes in Computer Science*, ed. Godo L. Vol. 3571. Berlin, Heidelberg: Springer. pp. 317–328 doi: 10.1007/11518655_28

[42] Amgoud L, Cayrol C, Lagasquie-Schiex MC, Livet P. 2008. On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems* 23(10):1062−1093

[43] Nouioua F, Risch V. 2011. Argumentation frameworks with necessities. *Scalable Uncertainty Management - 5th International Conference, Proceedings, SUM 2011, Dayton, OH, USA, 10–13 October 2011. Lecture Notes in Computer Science*, eds. Benferhat S, Grant J. Vol. 6929. Berlin, Heidelberg: Springer. pp. 163–176 doi: 10.1007/978-3-642-23963-2_14

[44] Dimopoulos Y, Dvorák W, König M, Rapberger A, Ulbricht M, et al. 2023. Sets attacking sets in abstract argumentation. *21st International Workshop on Nonmonotonic Reasoning, 2–4 September 2023, Rhodes, Greece*. Bonn, Germany: CEUR-WS.org. pp. 22–31 doi: 10.34726/5386

[45] Fagin R, Kimelfeld B, Kolaitis PG. 2015. Dichotomies in the complexity of preferred repairs. *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, 31 May–4 June 2015*. New York, NY, USA: ACM. pp. 3–15 doi: 10.1145/2745754.2745762

[46] Bienvenu M, Bourgaux C, Goasdoué F. 2019. Computing and explaining query answers over inconsistent dl-lite knowledge bases. *Journal of Artificial Intelligence Research* 64:563−644

Copyright: © 2026 by the author(s). Published by Maximum Academic Press, Fayetteville, GA. This article is an open access article distributed under Creative Commons Attribution License (CC BY 4.0), visit https://creativecommons.org/licenses/by/4.0/.