Computing Repairs Under Functional and Inclusion Dependencies via Argumentation

Yasir Mahmood[1][0000−0002−5651−5391], Jonni Virtema[2][0000−0002−1582−3718], Timon Barlag[3][0000−0001−6139−5219], and Axel-Cyrille Ngonga Ngomo[1][0000−0001−7112−3516]

[1] DICE group, Department of Computer Science, Paderborn University, Germany
yasir.mahmood@uni-paderborn.de
axel.ngonga@upb.de
[2] Department of Computer Science, University of Sheffield, United Kingdom
j.t.virtema@sheffield.ac.uk
[3] Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany
barlag@thi.uni-hannover.de

Abstract. We discover a connection between finding subset-maximal repairs for sets of functional and inclusion dependencies, and computing extensions within argumentation frameworks (AFs). We study the complexity of existence of a repair, and deciding whether a given tuple belongs to some (or every) repair, by simulating the instances of these problems via AFs. We prove that subset-maximal repairs under functional dependencies correspond to the naive extensions, which also coincide with the preferred and stable extensions in the resulting AFs. For inclusion dependencies one needs a pre-processing step on the resulting AFs in order for the extensions to coincide. Allowing both types of dependencies breaks this relationship between extensions and only preferred semantics captures the repairs. Finally, we establish that the complexities of the above decision problems are **NP**-complete and $\mathbf{\Pi_2^P}$-complete, when both functional and inclusion dependencies are allowed.

Keywords: complexity theory · database repairs · integrity constraints · abstract argumentation

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 does not conform to the imposed integrity constraints. A rich theory has been developed to deal with inconsistent databases. 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 [14,8], attribute-based repairs [33], and cardinality-based repairs [28].

Dung's abstract argumentation framework [19] has been specifically designed to model conflicts and support relationships among arguments. An abstract argumentation framework (AF) represents arguments and their conflicts through directed graphs and allows for a convenient exploration of the conflicts at an abstract level. AFs have been explored extensively for representation and reasoning with inconsistent knowledge-bases (KBs) covering datalog, existential rules, and description logics (see e.g., [4,5,6,34,35,11] and [3] 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 integrity constraint, 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 functional dependencies (FDs) (or in general, denial constraints) have been investigated in the context of AFs, as discussed in [11]. We expand this area of research by establishing further connections between repairs and abstract argumentation frameworks when further integrity constraints are allowed.

In this paper, we focus on subset repairs of relational databases when the integrity constraints are functional and inclusion dependencies (IDs). We are interested in the computational problems of deciding the existence of a repair, and determining whether a given tuple belongs to some (or every) repair. We show how subset-maximal repairs for a set of functional dependencies and inclusion dependencies can be obtained by computing the naive, preferred, or stable extensions (see Section 2 for definitions) in the related AFs. Repairs under functional dependencies correspond to the naive extensions, which also coincide with the preferred and stable extensions in the resulting AFs. For inclusion dependencies one needs a pre-processing step on the resulting AFs for the extensions to coincide. Allowing both types of dependencies breaks this relationship between extensions and only preferred semantics captures the maximal repairs. Finally, we consider the complexity of deciding whether a tuple belongs to at least one or to all repairs, respectively. See Table 1 for the complexity results.

By employing Dung's argumentation framework 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 [1,2,15,23,27,31] (see [9,10] for an overview). The notion of conflict graphs and hypergraphs has been introduced before in the case of functional dependencies [25,26,30].

Atoms	AF-semantics for REP	Complexity Results REP	∃-REP	∀-REP
FDs	$\sigma \in \{\mathsf{naive}, \mathsf{pref}, \mathsf{stab}\}$ (Thm. 3)	(trivial)	(trivial)	$\in \mathbf{P}$
IDs	pref (Thm. 8)	$\in \mathbf{P}^{[1]}$	$\in \mathbf{P}^{[1]}$	$\in \mathbf{P}^{[1]}$
FDs+IDs	pref (Thm. 14)	**NP** (Thm. 15)	**NP** (Thm. 15)	$\mathbf{\Pi}_2^{\mathbf{P}}$ (Thm. 17)

Table 1. Overview of our main contributions. The complexity results depict completeness, unless specified otherwise. The second column indicates the AF-semantics corresponding to subset-repairs (REP) for dependencies in the first column, and the later three columns present the complexity of each problem. The **P**-results are already known in the literature, whereas the remaining results are new.

In particular, a correspondence between repairs and subset maximal independent sets of the conflict graph for FDs [2] and denial constraints [15] has been established. Moreover, a recent work defines conflict (hyper)graphs for a richer setting of universal constraints considering symmetric difference repairs [12]. Notice that the same definition also yields a correspondence between repairs and the naive extensions when the conflict graph is seen as an argumentation framework. Nevertheless, up to our knowledge, no work has considered a similar graph representation when inclusion dependencies are taken into account. Hannula and Wijsen [23] 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. Our work differs from the previous work, since it combines functional dependencies (a subclass of equality-generating dependencies) and inclusion dependencies (a subclass of tuple-generating dependencies). Moreover, one of our main contributions lies in connecting repairs under FDs and IDs to the extensions of argumentation frameworks in Dung's setting [19]. The connection between AFs and preferred repairs has been explored in the context of prioritized description logic [11] and datalog knowledge bases [7,17,18,24]. Employing abstract argumentation, we utilize tuples in a database as arguments, aligning with the approach presented in [11]. In contrast, Croitoru et al.'s work in the datalog setting [17,18] takes an orthogonal approach, employing structured argumentation to construct arguments from a given knowledge base.

2 Preliminaries

We assume that the reader is familiar with basics of complexity theory. We will encounter, in particular, the complexity classes $\mathbf{P}$, $\mathbf{NP}$, and $\mathbf{\Pi}_2^{\mathbf{P}}$. In the following, we shortly recall the necessary definitions from databases and argumentation.

We begin by restricting our attention to unirelational databases as these suffice for establishing our desired connections to argumentation frameworks as well as to our hardness results (see Table 1). Towards the end (Sec. 4), we highlight the changes required to expand this approach to the multirelational setting.

The unirelational case is also connected to the literature in team-semantics [32], which 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 and Hella [22] 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 $\mathsf{dep}(\boldsymbol{x};\boldsymbol{y})$ and $\boldsymbol{x} \subseteq \boldsymbol{y}$. [4]

Databases and Repairs For our setting, an instance of a database is a single table denoted as T. We call each entry in the table a *tuple* which is associated with an identifier. Formally, a table corresponds to a relational schema denoted as $T(x_1, \ldots, x_n)$, where T is the relation name and $x_1, \ldots, x_n$ are distinct attributes. For an attribute x 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 $(s(x_1), \ldots, s(x_k))$. For a database T, $\mathsf{dom}(T)$ denotes the *active domain* of T, defined as the collection of all the values that occur in the tuples of T.

Let $T(x_1, \ldots, x_n)$ be a schema and T be a database. A *functional dependency* (FD) over T is an expression of the form $\mathsf{dep}(\boldsymbol{x};\boldsymbol{y})$ (also denoted as $\boldsymbol{x} \to \boldsymbol{y}$) for sequences $\boldsymbol{x}, \boldsymbol{y}$ of attributes in T. A database T satisfies $\mathsf{dep}(\boldsymbol{x};\boldsymbol{y})$, denoted as $T \models \mathsf{dep}(\boldsymbol{x};\boldsymbol{y})$ if for all $s, t \in T$: if $s(\boldsymbol{x}) = t(\boldsymbol{x})$ then $s(\boldsymbol{y}) = t(\boldsymbol{y})$. That is, every two tuples from T that agree on $\boldsymbol{x}$ also agree on $\boldsymbol{y}$. Moreover, an *inclusion dependency* (ID) is an expression of the form $\boldsymbol{x} \subseteq \boldsymbol{y}$ for two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ of attributes with same length. The table T satisfies $\boldsymbol{x} \subseteq \boldsymbol{y}$ ($T \models \boldsymbol{x} \subseteq \boldsymbol{y}$) if for each $s \in T$, there is some $t \in T$ such that $s(\boldsymbol{x}) = t(\boldsymbol{y})$. We call each such t the *satisfying tuple for* s and the ID $\boldsymbol{x} \subseteq \boldsymbol{y}$. By a dependency atom, we mean either a functional or an inclusion dependency.

Let T be a database and B be a collection of dependency atoms. Then T is *consistent* with respect to B, denoted as $T \models B$, if $T \models b$ for each $b \in B$. Moreover, T is *inconsistent* with respect to B if there is some $b \in B$ such that $T \not\models b$. A *subset-repair* of T with respect to B is a subset $P \subseteq T$ which is consistent with respect to B, and maximal in the sense that no set P' exists such that it is consistent with respect to B and $P \subset P' \subseteq T$. In the following, we simply speak of a repair when we intend to mean a subset-repair. Furthermore, we often consider a database T without explicitly highlighting its schema. Let $\mathcal{B} = \langle T, B \rangle$ where T is a database and B is a set of dependency atoms, then $\mathsf{repairs}(\mathcal{B})$ denotes the set of all repairs for $\mathcal{B}$. Since an empty database satisfies each dependency trivially, we restrict the notion of a repair to non-empty databases. The problem we are interested in (REP) asks to decide whether there exists a repair for an instance $\mathcal{B}$.

Problem: REP	
Input:	an instance $\mathcal{B} = \langle T, B \rangle$.
Question:	is it true that $\mathsf{repairs}(\mathcal{B}) \neq \emptyset$?

[4] We borrow this notation and write $\mathsf{dep}(\boldsymbol{x};\boldsymbol{y})$ and $\boldsymbol{x} \subseteq \boldsymbol{y}$ for FDs and IDs, respectively.

Two further problems of interest are *brave* and *cautious* reasoning for a tuple $s \in T$. Given an instance $\mathcal{B} = \langle T, B \rangle$ and tuple $s \in T$, then brave (cautious) reasoning for s denoted as $\exists$-REP$(s, \mathcal{B})$ ($\forall$-REP$(s, \mathcal{B})$) asks whether s belongs to some (every) repair for $\mathcal{B}$.

Abstract Argumentation We use Dung's argumentation framework [19] 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 $\sigma \in \{\mathsf{conf}, \mathsf{naive}, \mathsf{adm}, \mathsf{pref}, \mathsf{stab}\}$, we write $\sigma(\mathcal{F})$ for the set of *all extensions* of semantics σ in $\mathcal{F}$ [5] Now, we are ready to define the corresponding decision problem asking for extension existence with respect to a semantics σ.

Problem:	Ext_σ
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 $a \in A$, the problems Cred_σ and Skep_σ ask whether a is in some σ-extension of $\mathcal{F}$ ("*credulously* accepted") or every σ-extension of $\mathcal{F}$ ("*skeptically* accepted"), respectively. The complexity of reasoning in argumentation is well understood, see [20, Table 1] for an overview. In particular, $\mathrm{Cred}_{\mathsf{naive}}$ and $\mathrm{Skep}_{\mathsf{naive}}$ are in $\mathbf{P}$, whereas, $\mathrm{Cred}_{\mathsf{pref}}$ and $\mathrm{Skep}_{\mathsf{pref}}$ are $\mathbf{NP}$-complete and $\mathbf{\Pi}_2^{\mathbf{P}}$-complete, respectively. Moreover, the problem to decide whether there is a non-empty extension is in $\mathbf{P}$ for naive and $\mathbf{NP}$-complete for pref-semantics. This makes naive-semantics somewhat easier and pref the hardest among the considered semantics in this work.

[5] We disallow the empty set ($\emptyset$) in extensions for the sake of compatibility with repairs. Nevertheless, one can allow $\emptyset$ as an extension in AFs and the empty database as repairs, without affecting our complexity results

3 Inconsistent Databases via Argumentation Frameworks

In the first two subsections, we consider instances containing only one type of dependency atoms to an AF. Then, we combine both (dependence and inclusion) atoms in the third subsection. Given an instance $\mathcal{B} = \langle T, B\rangle$ comprising a database T and a set B of dependencies, the goal is to capture all the subset-repairs for $\mathcal{B}$ by σ-extensions of the resulting AF ($\mathcal{F}_\mathcal{B}$) for some semantics σ.

In Section 3.1, we encode an instance $\mathcal{D} = \langle T, D\rangle$ with database T and a collection D of FDs into an AF $\mathcal{F}_\mathcal{D}$. This is achieved by letting each tuple $s \in T$ be an argument. Then the attack relation for $\mathcal{F}_\mathcal{D}$ simulates the violation between a pair $s, t \in T$ failing some $d \in D$. Although the construction for FDs is similar to the approach adopted by Bienvenu and Bourgaux [11], we do not consider priorities among tuples in the database and therefore establish that a weaker AF-semantics is already enough to capture repairs in our setting.

In Section 3.2, we simulate an instance $\mathcal{I} = \langle T, I\rangle$ including a collection I of inclusion dependencies (IDs) via AFs. The first observation is that the semantics of IDs requires the notion of *support* or *defense* rather than *conflict* between tuples. Then, we depict each tuple as an argument as well as use auxiliary arguments to simulate inclusion dependencies (i.e., to model the semantics for IDs). This is achieved by letting s_i be an argument for each $s \in T$ and $i \in I$ such that s_i attacks s. Then, the arguments defending s against s_i correspond precisely to the satisfying tuples $t \in T$ for s and i. 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 $\mathcal{I}$ and the extensions for AFs under preferred semantics. Finally, we establish that after a pre-processing on the resulting AF, the stable and naive extensions also yield repairs for $\mathcal{I}$.

Having established that both FDs and IDs can be modeled in AFs via attacks, Section 3.3 generalizes this approach by allowing both types of dependencies.

3.1 Simulating Functional Dependencies via AFs

We transform an instance $\mathcal{D} = \langle T, D\rangle$ with database T and a collection D of FDs to an AF $\mathcal{F}_\mathcal{D}$ defined as follows.

Definition 1. *Let $\mathcal{D} = \langle T, D\rangle$ be an instance of* REP *including a database T and a collection D of FDs. Then, $\mathcal{F}_\mathcal{D}$ denotes the following AF.*

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

We call $\mathcal{F}_\mathcal{D}$ the argumentation framework generated by the instance $\mathcal{D}$. Moreover, we also call R the conflict graph *for $\mathcal{D}$.*

Note that, for a given instance $\mathcal{D}$, the framework $\mathcal{F}_\mathcal{D}$ can be generated in polynomial time. The attack relation R is constructed for each $d \in D$ by taking each pair $s, t \in T$ in turn and checking whether $\{s,t\} \models d$ or not.

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

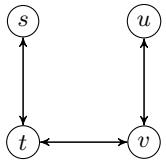

Fig. 1. Argumentation framework for modelling FDs in Example 2.

Example 2. Consider $\mathcal{D} = \langle T, D\rangle$ with database $T = \{s, t, u, v\}$ as depicted inside table in Figure 1 and FDs $\{\mathsf{dep}(\text{Emp\_ID}; \text{Dept}), \mathsf{dep}(\text{Sup\_ID}; \text{Building})\}$. Informally, each employee is associated with a unique department and employees supervised by the same supervisor work in the same building. Observe that, $\{s, t\} \not\models \mathsf{dep}(\text{Emp\_ID}; \text{Dept})$, $\{u, v\} \not\models \mathsf{dep}(\text{Emp\_ID}; \text{Dept})$, and $\{t, v\} \not\models \mathsf{dep}(\text{Sup\_ID}; \text{Building})$. The resulting AF $\mathcal{F}_\mathcal{D}$ is depicted on the right side of Figure 1. 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 a subset $P \subseteq T$ satisfying each $d \in D$ contains precisely those tuples $s \in T$, which are not in conflict with each other. Clearly, 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 [16, Prop. 4 & 5].

Theorem 3. *Let $\mathcal{D} = \langle T, D\rangle$ be an instance of* REP *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 $P \subseteq T$, $P \in \mathsf{repairs}(\mathcal{D})$ iff $P \in \sigma(\mathcal{F}_\mathcal{D})$ for $\sigma \in \{\mathsf{naive}, \mathsf{stab}, \mathsf{pref}\}$.*

Proof. Let $\mathcal{D} = \langle T, D\rangle$ be an instance of REP and $P \subseteq T$ such that $P \models d$ for each $d \in D$. Then, P is clearly conflict-free in $\mathcal{F}_\mathcal{D}$. Moreover, since P is a repair (and hence a maximal subset) of T, there is no $t \in T \setminus P$ such that $P \cup \{t\}$ is also conflict-free. As a result, P is a naive extension in $\mathcal{F}_\mathcal{D}$. Finally, the same holds for preferred and stable extensions since the attack relation in $\mathcal{F}_\mathcal{D}$ is symmetric.

Conversely, let $P \subseteq A$ be naive in $\mathcal{F}_\mathcal{D}$. Then, $\{s, t\} \models d$ for each $s, t \in P$ and $d \in D$ since P is conflict-free. Moreover, P is also subset maximal and therefore a repair for $\mathcal{D}$. □

An interesting corollary of Theorem 3 reproves that a subset-repair for $\mathcal{D}$ can be computed in polynomial time [20]. Moreover we can also decide if a given tuple $s \in T$ is in some (or all) repairs, in polynomial time. In fact, the basic properties of functional dependencies allow us to make the following observation regarding the acceptability of tuples with respect to $\mathcal{D}$.

Remark 4. Let $\mathcal{D} = \langle T, D\rangle$ be an instance of REP where D is a set of FDs. Then, $\exists$-REP$(s, \mathcal{D})$ is true for every $s \in T$, and $\forall$-REP$(s, \mathcal{D})$ is true for a tuple $s \in T$ iff $\{s, t\} \models d$ for each $t \in T$ and $d \in D$.

We conclude this section by observing that adding a size restriction for a repair renders the complexity of REP **NP**-hard. Moreover, this already holds

for propositional databases, that is, when $\mathsf{dom}(T) = \{0, 1\}$. The following result was proven in the context of team-semantics and maximal satisfying subteams for propositional dependence logic.

Theorem 5. *[29, Theorem 3.32] There is an instance $\mathcal{D}$ 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.*

3.2 Simulating Inclusion Dependencies via AFs

Let $\mathcal{I} = \langle T, I \rangle$ be an instance of REP with a database T and collection I of IDs. For $i \in I$ (say $i = \boldsymbol{x} \subseteq \boldsymbol{y}$) and $s \in T$, let $t_1, \ldots, t_m \in T$ be such that $s(\boldsymbol{x}) = t_j(\boldsymbol{y})$ for $j \leq m$. Then we say that, each such t_j *supports* s for the dependency $i \in I$ denoted as $S_i(s) := \{t_1, \ldots, t_m\}$. Clearly, $T \models i$ if and only if $S_i(s) \neq \emptyset$ for each $s \in T$. Moreover, if $S_i(s) = \emptyset$ for some $s \in T$ and $i \in I$, then s cannot belong to a repair for I. In the following, we formalize this notion and simulate the semantics for IDs via AFs.

Definition 6. *Let $\mathcal{I} = \langle T, I \rangle$ be an instance of* REP *including a database T and a collection I of IDs. Then $\mathcal{F}_\mathcal{I}$ is the following AF.*

- $A := T \cup \{s_i \mid s \in T, i \in I\}$.
- $R := \{(s_i, s), (s_i, s_i) \mid s \in T, i \in I\} \cup \{(t, s_i) \mid s \in T, i \in I, t \in S_i(s)\}$.

Intuitively, for each $i \in I$ and tuple $s \in T$, the presence of attacks (t, s_i) for each $t \in S_i(s)$ simulates the support relationship between s and tuples in $S_i(s)$. In other words, each $t \in S_i(s)$ attacks s_i and consequently, defends s against s_i. The whole idea captured in this translation is that a tuple $s \in T$ is in a repair for $\mathcal{I}$ if and only if for each $i := \boldsymbol{x} \subseteq \boldsymbol{y} \in I$, there is some $t \in T$ such that $s(\boldsymbol{x}) = t(\boldsymbol{y})$ if and only if the argument $s \in A$ is defended against each s_i in $\mathcal{F}_\mathcal{I}$.

Example 7. Consider $\mathcal{I} = \langle T, I \rangle$ with database $T = \{s, t, u, v\}$ and IDs $I := \{\text{Sup\_ID} \subseteq \text{Emp\_ID}, \text{Covers\_For} \subseteq \text{Dept}\}$. For brevity, we denote IDs by $I = \{1, 2\}$. The database and the supporting tuples $S_i(w)$ for each $i \in I, w \in T$ are depicted in the table inside Figure 2. 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, $s(\text{Sup\_ID}) = t(\text{Emp\_ID})$, $s(\text{Covers\_For}) = t(\text{Dept}) = u(\text{Dept})$, and therefore $S_1(s) = \{t\}$, $S_2(s) = \{t, u\}$. Then we have the AF $\mathcal{F}_\mathcal{I}$ as depicted in Figure 2. The AF $\mathcal{F}_\mathcal{I}$ has a unique preferred extension, given by $\{s, t\}$. Clearly, this is also the only repair for $\mathcal{I}$.

It is worth mentioning that, $\{s, t, u, v\}$ constitutes a naive extension for $\mathcal{F}_\mathcal{I}$ in Example 7, although this is not a repair for $\mathcal{I}$. Clearly, the semantics for IDs in $\mathcal{F}_\mathcal{I}$ requires admissibility (defending against attacking arguments). We now prove that the repairs for $\mathcal{I}$ are precisely the preferred extension in $\mathcal{F}_\mathcal{I}$.

Theorem 8. *Let $\mathcal{I} = \langle T, I \rangle$ be an instance of* REP *where I is a set of IDs and let $\mathcal{F}_\mathcal{I}$ denote the argumentation framework generated by $\mathcal{I}$. Then for every subset $P \subseteq T$, $P \in \mathsf{repairs}(\mathcal{I})$ iff $P \in \mathsf{pref}(\mathcal{F}_\mathcal{I})$.*

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

S_1	S_2
t	t,u
t	s
s,v	v
t	-

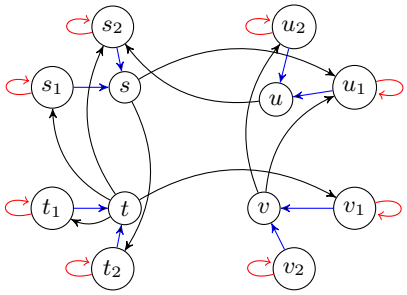

Fig. 2. The AF $\mathcal{F}_\mathcal{I}$ modelling $\mathcal{I}$ in Example 7: the red self-loops together with blue arcs depict the attacks for each tuple $s \in T$ due to IDs $i \in I$ and the black arcs model the attacks due to the support set $S_i(s)$.

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

Conversely, let $P \subseteq T$ denote a repair for $\mathcal{I} = \langle T, I \rangle$. Then P is conflict-free in $\mathcal{F}_\mathcal{I}$ since the attacks in R contain at least one argument among the auxiliary arguments (s_i) which are not in P (as $P \subseteq T$). Moreover, for each $s \in P$ and $i := \boldsymbol{x} \subseteq \boldsymbol{y} \in I$, there is some $t \in P$ such that: $s(\boldsymbol{x}) = t(\boldsymbol{y})$. This implies that each $s \in P$ is defended against the attack $s_i \in A$. Consequently, P is admissible. To prove that P is also preferred in $\mathcal{F}_\mathcal{I}$, assume to the contrary that there is an admissible $P' \supset P$ in $\mathcal{F}_\mathcal{I}$. Since P' is also conflict-free, using the same argument as for P we notice that $P' \subseteq T$. Now, P' being preferred (together with the claim in reverse direction) implies that P' is a repair for $\mathcal{I}$ contradicting the fact that P is a subset-maximal repair for $\mathcal{I}$. As a consequence, P is preferred in $\mathcal{F}_\mathcal{I}$.

This proves the correctness of our theorem. □

Notice that a framework $\mathcal{F}_\mathcal{I}$ may not have stable extensions for certain instances $\mathcal{I}$ including databases T and IDs I. This holds because some arguments can neither be accepted in an extension (e.g., when $S_i(s) = \emptyset$ for some $s \in T$ and $i \in I$), nor attacked by arguments in an extension (since arguments in A only attack auxiliary arguments). The argument corresponding to the tuple v in Example 7 depicts such an argument. As a result, the stable and preferred extensions do not coincide in general. Nevertheless, we prove that after a preprocessing, naive, stable and preferred extensions still coincide.

A pre-processing algorithm for $\mathcal{F}_\mathcal{I}$. Observe that an undefended argument in an AF $\mathcal{F}$ cannot belong to any preferred extension of $\mathcal{F}$. The intuition behind pre-processing is to remove such arguments, which are not defended against some of their attacks in $\mathcal{F}_\mathcal{I}$. This corresponds to (recursively) removing those tuples in $s \in T$, for which $S_i(s) = \emptyset$ for some $i \in I$. The pre-processing (denoted $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$) applies the following procedure as long as possible.

* For each $s_i \in A$ such that s_i is not attacked by any $t \neq s_i$: remove s and s_j for each $j \in I$, 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-loops results in a unique naive extension which is also stable and preferred. In the following, we also denote by $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$ the AF obtained after applying the pre-processing on $\mathcal{F}_\mathcal{I}$. Notice that PRE is basically an adaptation to the AFs of the well-known algorithm for finding a maximal satisfying subteam for inclusion logic formulas [22, Lem. 12].

Lemma 9. *Let $\mathcal{I}$ be an instance of* REP *and $\mathcal{F}_\mathcal{I}$ denote the argumentation framework generated by $\mathcal{I}$. Then $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$ can be computed from $\mathcal{F}_\mathcal{I}$ in polynomial time. Moreover, $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$ has a unique naive extension which is also stable and preferred.*

Proof. The procedure $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$ removes recursively all the arguments corresponding to assignments s such that $S_i(s) = \emptyset$ for some $i \in I$. Notice that $S_i(s)$ can be computed for each $i \in I$ and $s \in T$ in polynomial time. Then, PRE stores in a data structure (such as a queue) all the arguments s for which $S_i(s) = \emptyset$. 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 $S_i(t)$ 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 $\mathcal{F}_\mathcal{I}$.

Let $\mathsf{PRE}(\mathcal{F}_\mathcal{I}) = (A', R')$ denote the AF generated by the pre-processing. To prove the equivalence between extensions, notice that the set of arguments (S) without self-attacks in A' form an admissible extension since $S_i(s) \neq \emptyset$ for every $s \in S$. Further, 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 and conflict-free in $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$ and arguments in $A' \setminus S$ contain self-attacks.

This establishes the correctness of the lemma together with Theorem 8. □

One consequence of Lemma 9 is that a database in the presence of IDs admits a unique repair (Thm. 8). Moreover, the following observation follows from the proof of Lemma 9. Intuitively, we can also determine $\exists$-REP$(s, \mathcal{I})$ and $\forall$-REP$(s, \mathcal{I})$ for each $s \in T$, once the pre-processing has terminated resulting in $\mathsf{PRE}(\mathcal{F}_\mathcal{I})$.

Remark 10. Let $\mathcal{I}$ be an instance of REP and $\mathcal{F}_\mathcal{I}$ denote the argumentation framework generated by $\mathcal{I}$. Then, $\exists$-REP$(s, \mathcal{I})$ and $\forall$-REP$(s, \mathcal{I})$ is true for every $s \in T$ such that $s \in \mathsf{PRE}(\mathcal{F}_\mathcal{I})$.

Example 11 (Continue). Reconsider the instance $\mathcal{I} = \langle T, I\rangle$ from Example 7. Observe that the argument v is not defended against v_2 and therefore cannot be in a repair. Then 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 process for u, we notice that no other argument needs to be removed. Hence, the set $\{s, t\}$ is a repair for $\mathcal{I}$ as well as a σ-extension in the reduced AF for $\sigma \in \{\mathsf{naive}, \mathsf{stab}, \mathsf{pref}\}$.

3.3 Simulating Functional and Inclusion Dependencies via AFs

Consider an instance $\mathcal{B} = \langle T, B\rangle$ with a database T such that $B = D \cup I$ includes functional (D) and inclusion (I) dependencies. We apply the pre-processing as a first step, thereby, removing those tuples from T failing some $i \in I$. In other words, we remove (recursively) all the tuples s from T, such that, there is some $i \in I$ with $S_i(s) = \emptyset$. Then, the framework generated by $\mathcal{B}$ is $\mathcal{F}_\mathcal{B} := (A, R_\mathrm{D} \cup R_\mathrm{I})$, specified as below.

- $A := T \cup \{s_i \mid s \in T, i \in I\}$.
- $R_\mathrm{D} := \{(s,t), (t,s) \mid$ there is some $d \in D$, s.t. $\{s,t\} \not\models d\}$.
- $R_\mathrm{I} := \{(s_i, s), (s_i, s_i) \mid s \in T, i \in I\} \cup \{(t, s_i) \mid s \in T, i \in I, t \in S_i(s)\}$.

In the presence of both types of dependencies, pre-processing allows to reduce the number of arguments in the resulting AF without affecting the connection between extensions and repairs. The removed tuples are limited to those that cannot belong to any repair. However, to avoid the impression that pre-processing contributes towards the computation of repairs, one may choose not to apply it and consider the original database. Interestingly, even if we apply pre-processing, some tuples may not be accepted in combination with each other, as depicted in the following example.

Example 12. Consider $\mathcal{B} = \langle T, B\rangle$ with database $T = \{s, t, u\}$ and atoms $B = D \cup I$ where $D = \{\mathsf{dep}(\text{Sup\_ID}; \text{Building})\}$ and $I = \{\text{Covers\_For} \subseteq \text{Dept}\}$. Moreover, the database T and the support $S_{\text{Covers\_For} \subseteq \text{Dept}}(w)$ for each $w \in T$ is depicted in the table inside Figure 3. Then, $\{s, t\} \not\models \mathsf{dep}(\text{Sup\_ID}; \text{Building})$, and $\{t, u\} \not\models \mathsf{dep}(\text{Sup\_ID}; \text{Building})$. The resulting AF $\mathcal{F}_\mathcal{B}$ is shown in Figure 3, where the edges due to the IDs are depicted in red and blue. Then, the only preferred extensions for $\mathcal{F}_\mathcal{B}$ is $\{t\}$. Also, the only repair for $\mathcal{B}$ 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 $\mathcal{F}_\mathcal{I}$ where $\mathcal{I} = \langle T, I\rangle$ (ignoring black arcs), none of them is preferred for $\mathcal{F}_\mathcal{B}$.

One consequence of allowing both (functional and inclusion) dependencies is that the preferred and naive extensions do not coincide in general. Moreover, both $\exists$-REP and $\forall$-REP are non-trivial and distinct (cf. Rem. 4 and 10).

T	Emp_ID	Sup_ID	Dept.	Building	Covers_for	S_i
s	JonX1	AxeK4	Production	B4	Sales	t
t	TimX3	AxeK4	Sales	B2	Sales	t
u	AxeK4	AxeK4	Marketing	B4	Production	s

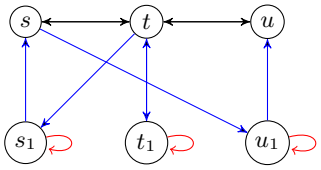

Fig. 3. Argumentation framework for modelling dependencies in Example 12. Black arcs depict conflicts due to functional, and blue ones due to inclusion dependency.

Example 13 (Cont.). Reconsider the instance $\mathcal{B}$ from Example 12. Then, $\{s, u\}$ is a naive extension for $\mathcal{F}_\mathcal{B}$ but not preferred. Moreover, t is the only tuple for which $\exists$-REP$(t, \mathcal{B})$ and $\forall$-REP$(t, \mathcal{B})$ is true.

As the preceding examples demonstrate, in the presence of both types of dependencies, the repairs correspond to somewhat costly (that is, preferred) semantics for AFs.

Theorem 14. *Let $\mathcal{B} = \langle T, B\rangle$ be an instance of* REP *where B includes FDs and IDs. Further, let $\mathcal{F}_\mathcal{B}$ denote the argumentation framework generated by $\mathcal{B}$. Then for every subset $P \subseteq T$, $P \in \mathsf{repairs}(\mathcal{B})$ iff $P \in \mathsf{pref}(\mathcal{F}_\mathcal{B})$.*

Proof. The correctness follows from the proof of Theorem 3 and 8. The conflict-freeness and admissibility of P implies that each FD and ID in B, respectively, is true in P. The converse follows the same line of argument. Finally, the maximality of repairs in $\mathcal{B}$ corresponds to the maximality of extensions in $\mathcal{F}_\mathcal{B}$. □

The existence of a non-empty extension and the credulous reasoning for preferred semantics are both **NP**-complete, and skeptical reasoning is even harder and $\mathbf{\Pi}_2^\mathbf{P}$-complete. Next we establish that when both types of dependencies are allowed, REP, $\exists$-REP and $\forall$-REP actually have the same complexity as, respectively, the existence, credulous and skeptical reasoning for preferred semantics.

Theorem 15. *Let $\mathcal{B} = \langle T, B\rangle$ be an instance of* REP *where B is a set of FDs and IDs. Further, let $s \in T$ be a tuple. Then, the problems* REP *and* $\exists$-REP$(s, \mathcal{B})$ *are both* **NP***-complete.*

Proof. The membership is easy in both cases. Given $P \subseteq T$ such that $s \in P$ (resp., P is non-empty), one can decide in polynomial time whether $P \models B$. Notice that we do not need to check the maximality, since if there is a repair (non-empty) for $\mathcal{B}$ containing s then there is also a maximal (non-empty) repair containing s.

For hardness, we first reduce from SAT to $\exists$-REP$(s, \mathcal{B})$. Towards the end, we highlight the required changes to reduce SAT to REP. Let $\varphi := \{C_i \mid i \leq m\}$ be

a formula over propositions $\{p_1, \ldots, p_n\}$. Then, we construct a database T and a collection B of FDs and IDs over a set $V := \{t_i, u_i \mid 0 \leq i \leq m\}$ of attributes. Our encoding works as follows.

- B contains a single FD $\mathsf{dep}(t_0; u_0)$ to encode that each proposition takes at most one value in $\{0, 1\}$. Moreover, B contains m inclusion dependencies $t_i \subseteq u_i$ for each $1 \leq i \leq m$ to assure that each clause C_i is satisfied.
- The database $T := \{s_\varphi\} \cup \{s_j, \bar{s}_j \mid 1 \leq j \leq n\}$ is constructed in such a way that: (C1) the pair $s_j, \bar{s}_j$ fails the FD $\mathsf{dep}(t_0; u_0)$ for each $1 \leq j \leq n$, thereby ensuring that any repair contains at most one tuple from $\{s_j, \bar{s}_j\}$, and (C2) for each ID $t_i \subseteq u_i$ the value of $s_\varphi(t_i)$ is shared only by the tuple $s \in T \setminus \{s_\varphi\}$ such that their corresponding literal satisfies the clause C_i. Formally, we let $\mathsf{dom}(T) := \{c_i \mid i \leq m\} \cup \{p_1, \ldots, p_n\} \cup \{0, 1\}$. Then, (C1) is achieved by setting $s_j(t_0) = \bar{s}_j(t_0) = p_j$, $s_j(u_0) = 1$, and $\bar{s}_j(u_0) = 0$. Moreover, we also let $s_\varphi(t_0) = s_\varphi(u_0) = 0$. To achieve (C2) we let $s_\varphi(t_i) = c_i$ for each $1 \leq i \leq m$ and $s_\varphi(u_i) = 0$. Then, we let $s_j(t_i) = s_j(u_i) = c_i$ if $p_j \in C_i$, $\bar{s}_j(t_i) = \bar{s}_j(u_i) = c_i$ if $\neg p_j \in C_i$, and we let $s_j(t_i) = s_j(u_i) = 0 = \bar{s}_j(t_i) = \bar{s}_j(u_i)$ in the remaining cases.

Clearly, $T \not\models B$ due to the presence of the pair $s_j, \bar{s}_j$ and the FD $\mathsf{dep}(t_0; u_0)$. Then, a repair $P \subseteq T$ for $\mathsf{dep}(t_0; u_0)$ contains exactly one tuple from each such pair. Finally, $P \models t_i \subseteq u_i$ for $1 \leq i \leq m$ iff P contains at least one tuple $s \in \{s_j, \bar{s}_j\}$ corresponding to $x \in \{p_j, \neg p_j\}$ such that $x \in C_i$ for each $C_i \in \varphi$. This completes the proof since φ is satisfiable if and only if $\exists$-REP$(s_\varphi, \mathcal{B})$ is true.

To reduce SAT into REP, we use an additional ID, $t_{m+1} \subseteq u_{m+1}$ and modify T in such a way that every repair P for $\mathcal{B}$ necessarily contains s_φ, thereby proving the equivalence as before. This is achieved by adding a new element $c_{m+1} \in \mathsf{dom}(T)$ and setting $s_\varphi(t_{m+1}) = s_\varphi(u_{m+1}) = c_{m+1}$, as well as $s_j(t_{m+1}) = \bar{s}_j(t_{m+1}) = c_{m+1}$ and $s_j(u_{m+1}) = \bar{s}_j(u_{m+1}) = 0$ for each $1 \leq j \leq n$. This has the effect that one cannot construct a subset-repair for $\mathcal{B}$ by excluding s_φ and therefore, there is a non-empty repair for $\mathcal{B}$ iff φ is satisfiable. This completes the proof for both cases. □

We provide an example for better understanding of the reductions from the proof of Theorem 15.

Example 16. Let $\varphi := \{x \vee y, \neg x \vee \neg y, \neg x \vee y\}$ be a propositional formula. Then, our reduction for $\exists$-REP yields a database $T := \{s_\varphi, s_x, s_y, \bar{s}_x, \bar{s}_y\}$ and a collection $B := \{\mathsf{dep}(t_0; u_0)\} \cup \{t_i \subseteq u_i \mid 1 \leq i \leq 3\}$ of dependencies as depicted in the left side of Table 2. Notice that the only satisfying assignment for φ is given by $\{x \mapsto 0, y \mapsto 1\}$, corresponding to the repair $\{s_\varphi, \bar{s}_x, s_y\}$ for $\mathcal{B}$ containing s_φ, and consequently $\exists$-REP$(s_\varphi, \mathcal{B})$ is true.

Moreover, although each of $\{s_x, s_y\}, \{s_x, \bar{s}_y\}, \{\bar{s}_x, \bar{s}_y\}$ is also a repair for $\mathcal{B}$, none of them contains s_φ. Then, the second part of our reduction (for REP) adds the inclusion dependency $t_4 \subseteq u_4$ to B and expands the database T by two columns in the right side of Table 2. This results in $\{s_\varphi, \bar{s}_x, s_y\}$ being the only REP for $\mathcal{B}$ corresponding to the satisfying assignment for φ.

	t_0	u_0	t_1	u_1	t_2	u_2	t_3	u_3	t_4	u_4
s_φ	0	0	c_1	0	c_2	0	c_3	0	c_4	c_4
s_x	x	1	c_1	c_1	0	0	0	0	c_4	0
$\bar{s}_x$	x	0	0	0	c_2	c_2	c_3	c_3	c_4	0
s_y	y	1	c_1	c_1	0	0	c_3	c_3	c_4	0
$\bar{s}_y$	y	0	0	0	c_2	c_2	0	0	c_4	0

Table 2. The database corresponding to the formula φ from Example 16.

Next, we prove that $\forall$-REP$(s, \mathcal{B})$ is even harder and $\mathbf{\Pi_2^P}$-complete.

Theorem 17. *Let $\mathcal{B} = \langle T, B\rangle$ be an instance of* REP *including a set B of FDs and IDs. Further, let $s \in T$ be a tuple. Then, the problem $\forall$-REP$(s, \mathcal{B})$ is $\mathbf{\Pi_2^P}$-complete.*

Proof. For membership, one can guess a subset $P \subseteq T$ as a counter example for s, that is, $s \notin P$ and P is a subset-repair for $\mathcal{B}$, which can be decided in polynomial time. However, to determine whether P is maximal, one has to use oracle calls for guessing subsets $P' \subseteq T$, with $s \notin P'$ to determine whether $P' \models B$ and $P' \supset P$. This gives an upper bound of $\mathbf{coNP^{NP}}$ (equivalently $\mathbf{\Pi_2^P}$).

For hardness, we use a similar idea as in the proof of Theorem 15 and reduce from an instance Φ of the $\mathbf{\Pi_2^P}$-complete problem 2QBF, where $\Phi = \forall Y \exists Z \varphi(Y, Z)$ and $\varphi := \{C_i \mid i \leq m\}$ is a CNF. We let $X = Y \cup Z$ and construct an instance $\mathcal{B} := \langle T, B\rangle$ over a set $V := \{t_i, u_i \mid 0 \leq i \leq m+1\}$ of attributes. As in the proof of Theorem 15, B contains a FD $\mathsf{dep}(t_0; u_0)$ and a collection of IDs $I := \{t_i \subseteq u_i \mid 1 \leq i \leq m+1\}$ to encode whether each clause $C_i \in \varphi$ is satisfied. Moreover, the additional ID $t_{m+1} \subseteq u_{m+1}$ encodes the existentially quantified variables Z. The database is also constructed as in the proof of Theorem 15, except for the attributes $\{t_{m+1}, u_{m+1}\}$. These attributes encode the effect that s_φ *supports* variables $\{z, \bar{z}\}$ for each $z \in Z$ via the inclusion dependency $t_{m+1} \subseteq u_{m+1}$. This is achieved by letting $s_z(t_{m+1}) = \bar{s}_z(t_{m+1}) = c_{m+1}$ for each such $z \in Z$, as well as $s_z(u_{m+1}) = \bar{s}_z(u_{m+1}) = 0$ and $s_\varphi(t_{m+1}) = s_\varphi(u_{m+1}) = c_{m+1}$.

For correctness, notice that every interpretation I_Y over Y (seen as a subset of Y) corresponds to a subset $P_Y = \{s_y \mid y \in I_Y\} \cup \{\bar{s}_y \mid y \notin I_Y\}$. Then, $P_Y \models B$: $\mathsf{dep}(t_0; u_0)$ is true since P_Y includes only one of $s_y, \bar{s}_y$ and each $t_i \subseteq u_i$ is true since $s(t_i) = s(u_i)$ for each $s \in T \setminus \{s_\varphi\}$ and $i \leq m+1$. Moreover, we have that $s_\varphi \notin P_Y$. Now, in order to extend P_Y by adding s_z or $\bar{s}_z$ for any $z \in Z$, s_φ must be added as well due to the ID $t_{m+1} \subseteq u_{m+1}$. However, in order to add s_φ, we have to find I_Y and I_Z that together satisfy φ due to the IDs $t_i \subseteq u_i$ for $i \leq m$. As a result, for any interpretation I_Y over Y: $I_Y \cup I_Z \models \varphi$ if and only if P_Y is not a repair for $\mathcal{B}$ (since, $P_Y \cup P_Z \cup \{s_\varphi\}$ is a repair in such a case). Equivalently, there is a repair for $\mathcal{B}$ not containing s_φ if and only if the formula Φ is false. As a consequence, Φ is true if and only if every repair for $\mathcal{B}$ contains s_φ if and only if $\forall$-REP$(s_\varphi, \mathcal{B})$ is true. □

We provide an example for better understanding of the reduction from the proof of Theorem 17.

	t_0	u_0	t_1	u_1	t_2	u_2	t_3	u_3	t_4	u_4
s_φ	0	0	c_1	0	c_2	0	c_3	0	c_4	c_4
s_1	y_1	1	c_1	c_1	0	0	0	0	0	0
$\bar{s}_1$	y_1	0	0	0	0	0	0	0	0	0
s_2	y_2	1	c_1	c_1	c_2	c_2	c_3	c_3	0	0
$\bar{s}_2$	y_2	0	0	0	0	0	0	0	0	0
s_3	z_3	1	c_1	c_1	0	0	c_3	c_3	c_4	0
$\bar{s}_3$	z_3	0	0	0	c_2	c_2	0	0	c_4	0
s_4	z_4	1	0	0	0	0	c_3	c_3	c_4	0
$\bar{s}_4$	z_4	0	0	0	c_2	c_2	0	0	c_4	0

Table 3. The database corresponding to the instance $\mathcal{B}$ from Example 18.

Example 18. Let $\Phi = \forall y_1 y_2 \exists z_3 z_4((y_1 \vee y_2 \vee z_3) \wedge (y_2 \vee \neg z_3 \vee \neg z_4) \wedge (y_2 \vee z_3 \vee z_4))$ be a 2QBF. Then, our reduction yields a database $T := \{s_\varphi, s_1, s_2, s_3, s_4, \bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4\}$ and a collection $B := \{\mathsf{dep}(t_0; u_0)\} \cup \{t_i \subseteq u_i \mid i \leq 4\}$ of dependencies as depicted in Table 3. The reader can verify that the formula Φ is true and that $\forall$-REP$(s_\varphi, \mathcal{B})$ is true as well.

We conclude this section by noting that the least requirements for subset-repairs in the presence of both atoms is conflict-freeness and admissibility. Furthermore, although admissible extensions for $\mathcal{F}_\mathcal{B}$ yield subset-repairs for $\mathcal{B}$, those are not guaranteed to be maximal.

4 From Unirelational to Multirelational Databases

A database instance in multirelational setting consists of a collection $\mathcal{T} = (T_1, \ldots T_m)$, where each T_j is a database corresponding to a relational schema $T_j(x_{j,1}, \ldots, x_{j,n_j})$ of arity n_j. As before, T_j denotes the relation name and $x_{j,1}, \ldots, x_{j,n_j}$ are distinct attributes. In the following, we let $T = \bigcup_{j \leq m} T_j$ denote the set of all the tuples in $\mathcal{T}$. A functional dependency over $\mathcal{T}$, and the satisfaction for FDs is defined as before, i.e., an expression of the form $\mathsf{dep}(\boldsymbol{x}; \boldsymbol{y})$ for sequences $\boldsymbol{x}, \boldsymbol{y}$ of attributes in T_j for some $j \leq m$. However, an inclusion dependency may address attributes from two different tables. That is, $i = \boldsymbol{x} \subseteq \boldsymbol{y}$ is an ID between T_j and T_k if $\boldsymbol{x}$ and $\boldsymbol{y}$ are sequences of attributes over T_j and T_k, respectively. We call T_j the *source* and T_k the *target* of i, denoted as $\mathsf{source}(i)$ and $\mathsf{target}(i)$. Then, $\mathcal{T} \models \boldsymbol{x} \subseteq \boldsymbol{y}$ if for each $s \in \mathsf{source}(\boldsymbol{x} \subseteq \boldsymbol{y})$, there is some $t \in \mathsf{target}(\boldsymbol{x} \subseteq \boldsymbol{y})$ such that $s(\boldsymbol{x}) = t(\boldsymbol{y})$. By slightly abusing the notation, if $d := \mathsf{dep}(\boldsymbol{x}; \boldsymbol{y})$ is an FD over attributes in T_j, then we write $\mathsf{source}(d) = T_j$. Finally, we define the notion of (subset-maximal) repairs similar to the case of unirelational databases, i.e., $\mathcal{P} = (P_1, \ldots, P_m)$ where $P_j \subseteq T_j$ for $j \leq m$ such that $\mathcal{P}$ satisfies each dependency in B.

The construction from Section 3 for unirelational setting can be expanded to allow databases with more than one relations. The encoding for FDs remains the same as before, whereas for IDs (between T_j and T_k), we create auxiliary arguments only for tuples in $\mathsf{source}(i)$, which can be attacked by arguments

corresponding to tuples in the $\mathsf{target}(i)$. As before, we denote by $S_i(s) = \{t \mid t \in \mathsf{target}(i), s(\boldsymbol{x}) = t(\boldsymbol{y})\}$ the tuples supporting s for an ID $i = \boldsymbol{x} \subseteq \boldsymbol{y}$. Given an instance $\mathcal{B} = \langle \mathcal{T}, B \rangle$ of a multirelational database $\mathcal{T}$ and a collection $B = D \cup I$ of FDs D and IDs I, we construct the AF $\mathcal{F}_\mathcal{B} = (A, R_\mathrm{D} \cup R_\mathrm{I})$ as follows.

- $A := \{s \mid s \in T\} \cup \{s_i \mid s \in \mathsf{source}(i) \text{ for } i \in I\}$,
- $R_\mathrm{D} := \{(s,t),(t,s) \mid s,t \in \mathsf{source}(d) \text{ for } d \in D \text{ and } \{s,t\} \not\models d\}$,
- $R_\mathrm{I} := \{(s_i,s),(s_i,s_i) \mid s \in \mathsf{source}(i) \text{ for } i \in I\} \cup \{(t,s_i) \mid t \in S_i(s) \text{ for } i \in I\}$.

Then, a similar argument as in the proof of Theorem 14 allows us to establish that repairs for an instance $\mathcal{B} = \langle \mathcal{T}, B \rangle$ including a multirelational database $\mathcal{T}$ and a collection B of FDs and IDs are precisely the preferred extensions in $\mathcal{F}_\mathcal{B}$.

5 Concluding Remarks

Overview. We simulated the problem of finding repairs of an inconsistent database under functional and inclusion dependencies by Dung's 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 FDs and IDs are allowed. Further, 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.

Discussion and Future Work. We would like to point out that although the conflict relation in the presence of functional dependencies and a connection between preferred extensions and subset maximal repairs is known for FDs [11], the main contributions of our work establishes the relation between extensions of AFs when inclusion dependencies 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, and defense/support via inclusion dependencies. A precise formulation of this transformation will allow us to simulate AFs via inconsistent databases by considering FDs and IDs. However, this intuition needs further exploration and is therefore left for future work.

Further notable future work may consider whether the idea presented here can be generalized to other well-known types of tuple or equality generating dependencies. Then, one can target the richer setting of universal constraints and the symmetric difference repairs [12]. An 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. The question we pose is the following: Given an inconsistent database $\mathcal{B}$ and a (Boolean) query q, can we construct an AF $\mathcal{F}_\mathcal{B}$

and an argument a_q such that the query q is true in some (resp., every) repair if, and only if, the argument a_q is credulously (skeptically) accepted in $\mathcal{F}_{\mathcal{B}}$. Finally, one can consider incorporating the 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 [21,25,26].

Another promising direction to consider next is the exploration of an explainability dimension, similar to that in the setting of ontological KBs [4,6,13]. Given an instance $\mathcal{B}$ including a database T and a collection B of dependencies, then the proposed AF $\mathcal{F}_{\mathcal{B}}$ lets one determine the *causes* why some tuples are not in some repair (or all repairs). We note that, for IDs, the auxiliary arguments modelling 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 $\mathcal{F}_{\mathcal{B}}$ and read from it the FDs or IDs which a tuple s failing $\exists$-REP$(s, \mathcal{B})$ or $\forall$-REP$(s, \mathcal{B})$ 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.

Acknowledgment

The work has received funding from the European Union's Horizon Europe research and innovation programme within project ENEXA (101070305) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/1 2021 – 438445824 and VI 1045-1/1 – 432788559. The first author expresses gratitude to Arne Meier (Leibniz University Hannover) for the invitation to discuss the topic in Hannover, as well as for motivating and guiding the discussion on this subject.

References

1. Afrati, F.N., Kolaitis, P.G.: Repair checking in inconsistent databases: algorithms and complexity. In: Fagin, R. (ed.) Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings. ACM International Conference Proceeding Series, vol. 361, pp. 31–41. ACM (2009). https://doi.org/10.1145/1514894.1514899, https://doi.org/10.1145/1514894.1514899
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Scalar aggregation in fd-inconsistent databases. In: den Bussche, J.V., Vianu, V. (eds.) Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings. Lecture Notes in Computer Science, vol. 1973, pp. 39–53. Springer (2001). https://doi.org/10.1007/3-540-44503-X_3, https://doi.org/10.1007/3-540-44503-X_3
3. Arieli, O., Borg, A., Heyninck, J.: A review of the relations between logical argumentation and reasoning with maximal consistency. Ann. Math. Artif. Intell. **87**(3), 187–226 (2019). https://doi.org/10.1007/S10472-019-09629-7, https://doi.org/10.1007/s10472-019-09629-7
4. Arioua, A., Croitoru, M.: Dialectical characterization of consistent query explanation with existential rules. In: Markov, Z., Russell, I. (eds.) Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, Key Largo, Florida, USA, May 16-18, 2016. pp. 621–625. AAAI Press (2016), http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12800
5. Arioua, A., Croitoru, M., Vesic, S.: Logic-based argumentation with existential rules. Int. J. Approx. Reason. **90**, 76–106 (2017). https://doi.org/10.1016/J.IJAR.2017.07.004, https://doi.org/10.1016/j.ijar.2017.07.004
6. Arioua, A., Tamani, N., Croitoru, M.: Query answering explanation in inconsistent datalog +/- knowledge bases. In: Chen, Q., Hameurlain, A., Toumani, F., Wagner, R.R., Decker, H. (eds.) Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9261, pp. 203–219. Springer (2015). https://doi.org/10.1007/978-3-319-22849-5_15, https://doi.org/10.1007/978-3-319-22849-5_15
7. Arioua, A., Tamani, N., Croitoru, M., Buche, P.: Query failure explanation in inconsistent knowledge bases using argumentation. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (eds.) Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014. Frontiers in Artificial Intelligence and Applications, vol. 266, pp.

101–108. IOS Press (2014). https://doi.org/10.3233/978-1-61499-436-7-101, https://doi.org/10.3233/978-1-61499-436-7-101
8. Barceló, P., Fontaine, G.: On the data complexity of consistent query answering over graph databases. Journal of Computer and System Sciences **88**, 164–194 (2017)
9. Bertossi, L.E.: Consistent query answering in databases. SIGMOD Rec. **35**(2), 68–76 (2006). https://doi.org/10.1145/1147376.1147391, https://doi.org/10.1145/1147376.1147391
10. Bertossi, L.E.: Database repairs and consistent query answering: Origins and further developments. In: Suciu, D., Skritek, S., Koch, C. (eds.) Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019. pp. 48–58. ACM (2019). https://doi.org/10.1145/3294052.3322190, https://doi.org/10.1145/3294052.3322190
11. Bienvenu, M., Bourgaux, C.: Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation. In: Calvanese, D., Erdem, E., Thielscher, M. (eds.) Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020. pp. 141–151 (2020). https://doi.org/10.24963/KR.2020/15, https://doi.org/10.24963/kr.2020/15
12. Bienvenu, M., Bourgaux, C.: Inconsistency Handling in Prioritized Databases with Universal Constraints: Complexity Analysis and Links with Active Integrity Constraints. In: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning. pp. 97–106 (8 2023). https://doi.org/10.24963/kr.2023/10, https://doi.org/10.24963/kr.2023/10
13. Bienvenu, M., Bourgaux, C., Goasdoué, F.: Computing and explaining query answers over inconsistent dl-lite knowledge bases. Journal of Artificial Intelligence Research **64**, 563–644 (2019)
14. ten Cate, B., Fontaine, G., Kolaitis, P.G.: On the data complexity of consistent query answering. In: Proceedings of the 15th International Conference on Database Theory. pp. 22–33. ICDT '12 (2012)
15. Chomicki, J., Marcinkowski, J.: Minimal-change integrity maintenance using tuple deletions. Information and Computation **197**(1), 90–121 (2005). https://doi.org/https://doi.org/10.1016/j.ic.2004.04.007, https://www.sciencedirect.com/science/article/pii/S0890540105000179
16. Coste-Marquis, S., Devred, C., Marquis, P.: Symmetric argumentation frameworks. In: Godo, L. (ed.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3571, pp. 317–328. Springer (2005). https://doi.org/10.1007/11518655_28, https://doi.org/10.1007/11518655_28
17. Croitoru, M., Thomopoulos, R., Vesic, S.: Introducing preference-based argumentation to inconsistent ontological knowledge bases. In: Chen, Q., Torroni, P., Villata, S., Hsu, J.Y., Omicini, A. (eds.) PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9387, pp. 594–602. Springer (2015). https://doi.org/10.1007/978-3-319-25524-8_42, https://doi.org/10.1007/978-3-319-25524-8_42
18. Croitoru, M., Vesic, S.: What can argumentation do for inconsistent ontology query answering? In: Liu, W., Subrahmanian, V.S., Wijsen, J. (eds.) Scalable Uncertainty

Management - 7th International Conference, SUM 2013, Washington, DC, USA, September 16-18, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8078, pp. 15–29. Springer (2013). https://doi.org/10.1007/978-3-642-40381-1_2, https://doi.org/10.1007/978-3-642-40381-1_2
19. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. ai **77**(2), 321–357 (1995)
20. Dvorák, W., Dunne, P.E.: Computational problems in formal argumentation and their complexity. FLAP **4**(8) (2017)
21. Fagin, R., Kimelfeld, B., Kolaitis, P.G.: Dichotomies in the complexity of preferred repairs. In: Milo, T., Calvanese, D. (eds.) Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015. pp. 3–15. ACM (2015). https://doi.org/10.1145/2745754.2745762, https://doi.org/10.1145/2745754.2745762
22. Hannula, M., Hella, L.: Complexity thresholds in inclusion logic. Inf. Comput. **287**, 104759 (2022). https://doi.org/10.1016/J.IC.2021.104759, https://doi.org/10.1016/j.ic.2021.104759
23. Hannula, M., Wijsen, J.: A dichotomy in consistent query answering for primary keys and unary foreign keys. In: Libkin, L., Barceló, P. (eds.) PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022. pp. 437–449. ACM (2022). https://doi.org/10.1145/3517804.3524157, https://doi.org/10.1145/3517804.3524157
24. Ho, L., Arch-int, S., Acar, E., Schlobach, S., Arch-int, N.: An argumentative approach for handling inconsistency in prioritized datalog± ontologies. AI Commun. **35**(3), 243–267 (2022). https://doi.org/10.3233/AIC-220087, https://doi.org/10.3233/AIC-220087
25. Kimelfeld, B., Livshits, E., Peterfreund, L.: Detecting Ambiguity in Prioritized Database Repairing. In: Benedikt, M., Orsi, G. (eds.) 20th International Conference on Database Theory (ICDT 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 68, pp. 17:1–17:20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2017). https://doi.org/10.4230/LIPIcs.ICDT.2017.17, http://drops.dagstuhl.de/opus/volltexte/2017/7048
26. Kimelfeld, B., Livshits, E., Peterfreund, L.: Counting and enumerating preferred database repairs. Theor. Comput. Sci. **837**, 115–157 (2020). https://doi.org/10.1016/J.TCS.2020.05.016, https://doi.org/10.1016/j.tcs.2020.05.016
27. Livshits, E., Kimelfeld, B., Roy, S.: Computing optimal repairs for functional dependencies. ACM Trans. Database Syst. **45**(1), 4:1–4:46 (2020). https://doi.org/10.1145/3360904, https://doi.org/10.1145/3360904
28. Lopatenko, A., Bertossi, L.E.: Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In: Proceedings ICDT-2007. LNCS, vol. 4353, pp. 179–193. Springer (2007)
29. Mahmood, Y.: Parameterized aspects of team-based formalisms and logical inference (2022). https://doi.org/10.15488/13064, https://www.tib.eu/de/suchen/id/base%3Ae4c211ee856f89407f6d9a67b4c100e3fb7eafdd
30. Staworko, S., Chomicki, J., Marcinkowski, J.: Prioritized repairing and consistent query answering in relational databases. Ann. Math. Artif. Intell. **64**(2-3), 209–246 (2012). https://doi.org/10.1007/S10472-012-9288-8, https://doi.org/10.1007/s10472-012-9288-8
31. Staworko, S., Chomicki, J.: Consistent query answers in the presence of universal constraints. Inf. Syst. **35**(1), 1–22 (2010). https://doi.org/10.1016/J.IS.2009.03.004, https://doi.org/10.1016/j.is.2009.03.004

32. Väänänen, J.: Dependence Logic. Cambridge University Press (2007)
33. Wijsen, J.: Condensed representation of database repairs for consistent query answering. In: Proceedings of the 9th International Conference on Database Theory. pp. 378–393. ICDT '03, Springer-Verlag (2002)
34. Young, A.P., Modgil, S., Rodrigues, O.: Prioritised default logic as argumentation with partial order default priorities. CoRR **abs/1609.05224** (2016), http://arxiv.org/abs/1609.05224
35. Yun, B., Vesic, S., Croitoru, M.: Sets of attacking arguments for inconsistent datalog knowledge bases. In: Prakken, H., Bistarelli, S., Santini, F., Taticchi, C. (eds.) Computational Models of Argument - Proceedings of COMMA 2020, Perugia, Italy, September 4-11, 2020. Frontiers in Artificial Intelligence and Applications, vol. 326, pp. 419–430. IOS Press (2020). https://doi.org/10.3233/FAIA200526, https://doi.org/10.3233/FAIA200526