Social Choice Methods for Database Aggregation

Francesco Belardinelli
Department of Computing, Imperial College London, U.K.
Laboratoire IBISC, Université d'Evry, France
francesco.belardinelli@imperial.ac.uk

Umberto Grandi
IRIT, University of Toulouse
France
umberto.grandi@irit.fr

Knowledge can be represented compactly in multiple ways, from a set of propositional formulas, to a Kripke model, to a database. In this paper we study the aggregation of information coming from multiple sources, each source submitting a database modelled as a first-order relational structure. In the presence of integrity constraints, we identify classes of aggregators that respect them in the aggregated database, provided these are satisfied in all individual databases. We also characterise languages for first-order queries on which the answer to a query on the aggregated database coincides with the aggregation of the answers to the query obtained on each individual database. This contribution is meant to be a first step on the application of techniques from social choice theory to knowledge representation in databases.

1 Introduction

Aggregating information coming from multiple sources is a long-standing problem in both knowledge representation and multi-agent systems (see, e.g., [24]). Depending on the chosen representation for the incoming pieces of knowledge or information, a number of competing approaches has seen the light in these literatures. Belief merging [31, 29, 28] studies the problem of aggregating propositional formulas coming from different agents into a set of models, subject to integrity constraints. Judgment and binary aggregation [16, 14, 22] asks individual agents to report yes/no opinions on a set of logically-related binary issues – the agenda – in order to take a collective decision. Social welfare functions, the cornerstone problem in social choice theory (see, e.g., [3]), can also be viewed as mechanisms to merge conflicting information, namely the individual preferences of voters expressed in the form of linear orders over a set of alternatives. Other examples include graph aggregation [18], multi-agent argumentation [8, 9, 11], ontology merging [37], and clustering aggregation [20].

In this work we take a general perspective and represent individual knowledge coming from multiple sources as a profile of databases, modelled as finite relational structures [1, 34]. Our motivation lies inbetween two possibly conflicting views on the problem of information fusion. On the one hand, the study of information merging (typically knowledge or beliefs) in knowledge representation has focused on the design of rules that guarantee the consistency of the outcome, with the main driving principles inspired from the literature on belief revision[1]. On the other hand, social choice theory has focused on agent-based properties, such as fairness and representativity of an aggregation procedure, paying attention as well on possible strategic behaviour by either the agents involved in the process or an external influencing source. While there already have been several attempts at showing how specific merging or aggregation frameworks could be simulated or subsumed by one another (see, e.g., [21, 12, 23, 19]), a more general perspective allows us to find a compromise between the two views described above.

[1] Albeit we acknowledge the work of [15, 35], which aggregate individual beliefs, modelled as plausibility orders, in an "Arrovian" fashion.

L.S. Moss (Ed.): TARK 2019
EPTCS 297, 2019, pp. 50–67, doi:10.4204/EPTCS.297.4

Our Contribution. Our starting point is a set of finite relational structures on the same signature, coming from a group of agents or sources. Then, our research problem is how to obtain a collective database summarising the information received. Virtually all of the settings mentioned above (beliefs, graphs, preferences, judgments, ...) can be represented as databases, showing the generality of our approach. We propose a number of rules for database aggregation, some inspired by existing ones from the literature on computational social choice and belief merging, as well as a new one adapted from representations of incomplete information in databases [32]. We privilege computationally friendly aggregators, for which the time to determine the collective outcome is polynomial in the individual input received.

We first evaluate these rules axiomatically, using notions imported from the literature on social choice, to provide a first classification of the agent-based properties satisfied by our proposed rules. Then, when integrity constraints are present, we study how to guarantee that a given aggregator "lifts" the integrity constraint from the individual to the collective level, i.e., the aggregated databases satisfy the same constraints as the individual ones. Specifically, we investigate which rules lift classical integrity constraints from database theory, such as functional dependencies, referential integrity and value constraints. Finally, since databases are typically queried using formulas in first-order logic, a natural question to ask in a multi-agent setting is whether the aggregation of the individual answers to a query coincides with the answer to the same query on the aggregated database. We provide a partial answer to this important problem, by identifying sufficient conditions on the first-order query language.

Related Work. While we are not aware of any application of methods from social choice theory to database aggregation, possibly the closest approach to ours is the work of Baral *et al.* [4, 5] and Konieczny [27]. In [4] the authors formalize the notion of combining knowledge bases, which are represented as normal Horn-logic programs. These investigations were further pursued in [5], which considers the problem of merging information represented in the form of first-order theories, taking a syntactic rather than a semantic approach (as we do here), and focusing on finding maximally consistent sets of the union of the individual theories received. In doing so, however, the authors privilege the knowledge representation approach, and have no control on the set of agents supporting a given maximally consistent set rather than another. In [27], the author applies techniques from belief merging to the equivalent problem of aggregating knowledge bases of first-order formulas, proposing a number of rules analysed axiomatically. Both contributions stem from a long tradition on combining inconsistent theories, especially in the domain of paraconsistent logics [7, 38]. However, all these approaches focus on merging syntactic representations (e.g., logic programs, first-order theories), while here we operate on semantical instances, i.e., databases. We also mention the work of Lin and Mendelzon [33], proposing an AGM-style approach to merge first-order theories under constraints, reminescent of the distance-based rules that we will consider in Section 3.

Mildly related to the present work is the literature on database repairs. Here the focus is on principles of minimal change, which is the aspiration to keep the recovered data as faithful as possible to the original (inconsistent) database [2]. Our perspective is different, as we analyse aggregation rather then repairing. Nonetheless, we also consider distance-based procedure.

More recently, connections between social choice theory and database querying have been explored in [26], which enriches the tasks currently supported in computational social choice by means of relational databases, thus allowing for sophisticated queries about voting rules, candidates, and voters. Here our aim is symmetric, as we rather apply methods and techniques from computational social choice to database theory.

An overview of the results presented hereafter can be found in [6], which introduces the question of database aggregation and defines some aggregation procedures. Here we extend [6] by considering in detail the problems pertaining to collective rationality through lifting of integrity constraints in Section 5, as well as aggregation and query answering in Section 6.

Structure of the Paper. In Section 2 we present basic notions on databases and integrity constraints. In Sections 3 and 4 we introduce several database aggregation procedures, and we analyse them by proposing a number of axiomatic properties. Sections 5 and 6 contains our main results on the lifting of integrity constraints and aggregated query answering. Section 7 concludes the paper.

2 Preliminaries on Databases

In this section we introduce basic notions on databases that we will use in the rest of the paper. In particular, we adopt a relational perspective [1] and present databases as finite relational structures over database schemas. Hereafter we assume a countable domain $\mathscr{U}$ of elements $u, u', \ldots$, for the interpretation of relation symbols.

Definition 1 (Database Schema and Instance)**.** *We call a* (relational) database schema $\mathscr{D}$ *a finite set* $\{P_1/q_1, \ldots, P_m/q_m\}$ *of relation symbols P with arity $q \in \mathbb{N}$. Given database schema $\mathscr{D}$ and domain $\mathscr{U}$, a* $\mathscr{D}$-instance *over $\mathscr{U}$ is a mapping D associating each relation symbol $P \in \mathscr{D}$ with a* finite *q-ary relation over $\mathscr{U}$, i.e., $D(P) \underset{fin}{\subset} \mathscr{U}^q$.*

By Definition 1 a database instance is a finite (relational) model of a database schema. The *active domain* $adom(D)$ of an instance D is the set of all individuals in $\mathscr{U}$ occurring in some tuple $\vec{u}$ of some predicate interpretation $D(P)$, that is, $adom(D) = \bigcup_{P \in \mathscr{D}} \{u \in \mathscr{U} \mid u = u_i \text{ for some } \vec{u} \in D(P)\}$. Observe that, since $\mathscr{D}$ contains a finite number of relation symbols and each $D(P)$ is finite, so is $adom(D)$. We denote the set of all instances over $\mathscr{D}$ and $\mathscr{U}$ as $\mathscr{D}(\mathscr{U})$. Clearly, the formal framework for databases we adopt is quite simple, but still it is powerful enough to cover practical cases of interest [34]. Here we do not discuss the pros and cons of the relational approach to database theory and refer to the literature for further details [1].

Example 1. To illustrate the notions introduced above, consider a database schema $\mathscr{D}_F$ for a faculty F, registering data on students and staff in two ternary relations *Students*/3 and *Staff*/3, that register IDs, names, and departments of students and staff respectively. A database instance D_F of $\mathscr{D}_F$ can be given, for example, as follows:

Students		
ID	Name	Department
10	Steve	History
11	Carole	Computer Science
12	Derek	Mechanical Engineering

Staff		
ID	Name	Department
01	Rose	Mechanical Engineering
02	Audrey	Mechanical Engineering
03	Karl	History

■

To specify the properties of databases, we make use of first-order logic with equality and no function symbols. Let V be a countable set of *individual variables*, which are the only terms in the language for the time being.

Definition 2 (FO-formulas over $\mathscr{D}$)**.** *Given a database schema $\mathscr{D}$, the formulas φ of the first-order language $\mathscr{L}_{\mathscr{D}}$ are defined by the following BNF:*

$$\varphi \quad ::= \quad x = x' \mid P(x_1, \ldots, x_q) \mid \neg\varphi \mid \varphi \to \varphi \mid \forall x \varphi$$

where $P \in \mathscr{D}$, $x_1, \ldots, x_q$ is a q-tuple of variables and x, x' are variables.

We assume "=" to be a special binary predicate with fixed obvious interpretation. By Def. 2, $\mathcal{L}_{\mathcal{D}}$ is a first-order language with equality over the relational vocabulary $\mathcal{D}$ and with no function symbols. In the following we use the standard abbreviations $\exists$, $\wedge$, $\vee$, and $\neq$. Also, free and bound variables are defined as standard. For a formula $\varphi \in \mathcal{L}_{\mathcal{D}}$, we write $\varphi(x_1,\ldots,x_\ell)$, or simply $\varphi(\vec{x})$, to list in arbitrary order all free variables $x_1,\ldots,x_\ell$ of φ. A *sentence* is a formula with no free variables. Notice that the only terms in our language $\mathcal{L}_{\mathcal{D}}$ are individual variables. We can add constants for individuals with some minor technical changes to the definitions and results in the paper. However, these do not impact on the theoretical contribution and we prefer to keep notation lighter.

To interpret FO-formulas on database instances, we introduce *assignments* as functions $\sigma : V \mapsto \mathcal{U}$. Given an assignment σ, we denote by σ^x_u the assignment such that (i) $\sigma^x_u(x) = u$; and (ii) $\sigma^x_u(x') = \sigma(x')$, for every variable $x' \in V$ different from x. We can now define the semantics of $\mathcal{L}_{\mathcal{D}}$.

Definition 3 (Satisfaction)**.** *Given a $\mathcal{D}$-instance D, an assignment σ, and an FO-formula $\varphi \in \mathcal{L}_{\mathcal{D}}$, we inductively define whether D* satisfies *φ* under *σ, or $(D,\sigma) \models \varphi$, as follows:*

$(D,\sigma) \models P(x_1,\ldots,x_q)$ *iff* $\langle \sigma(x_1),\ldots,\sigma(x_q)\rangle \in D(P)$
$(D,\sigma) \models x = x'$ *iff* $\sigma(x) = \sigma(x')$
$(D,\sigma) \models \neg\varphi$ *iff* $(D,\sigma) \not\models \varphi$
$(D,\sigma) \models \varphi \to \psi$ *iff* $(D,\sigma) \not\models \varphi$ *or* $(D,\sigma) \models \psi$
$(D,\sigma) \models \forall x \varphi$ *iff for every* $u \in adom(D)$, $(D,\sigma^x_u) \models \varphi$

A formula φ is true *in D, written $D \models \varphi$, iff $(D,\sigma) \models \varphi$, for all assignments σ.*

Observe that in Def. 3 we adopt an *active-domain* semantics, that is, quantified variables range only over the active domain of D. This is standard in database theory [1], where $adom(D)$ is assumed to be the "universe of discourse".

Integrity Constraints. It is well-known that several properties and constraints on databases can be expressed as FO-sentences. Here we consider some of them for illustrative purposes.

A *functional dependency* is an expression of type $\ell_1,\ldots,\ell_k \mapsto \ell_{k+1},\ldots,\ell_q$. A database instance D satisfies a functional dependency $\ell_1,\ldots,\ell_k \mapsto \ell_{k+1},\ldots,\ell_q$ for predicate symbol P with arity q iff for every q-tuples $\vec{u}$, $\vec{u}'$ in $D(P)$, whenever $u_i = u'_i$ for all $i \leq k$, then we also have $u_i = u'_i$ for all $k < i \leq q$. If $k = 1$, we say that it is a *key dependency*. Clearly, any database instance D satisfies a functional dependency $\ell_1,\ldots,\ell_k \mapsto \ell_{k+1},\ldots,\ell_q$ iff it satifies the following FO-sentence:

$$\forall \vec{x},\vec{y}\left(P(\vec{x}) \wedge P(\vec{y}) \wedge \bigwedge_{i\leq k}(x_i = y_i) \to \bigwedge_{k<i\leq q}(x_i = y_i)\right)$$

A *value constraint* is an expression of type $n_k \in P_v$, where $D(P_v)$ contains a list of admissible values. A database instance D satisfies a value constraint $n_k \in P_v$ for predicate symbol P with arity $q \geq k$ iff for every q-tuple $\vec{u}$ in $D(P)$, $u_k \in D(P_v)$.Also for value constraints, it is easy to check that an instance D satisfies constraint $n_k \in P_v$ for symbol P iff it satisfies the following:

$$\forall x_1,\ldots,x_q(P(x_1,\ldots,x_q) \to P_v(x_k))$$

A *referential constraint* enforces the foreign key of a predicate P_1 to be the primary key of predicate P_2. A database instance satisfies a referential constraint on the last k attributes, denoted as $(P_1 \to P_2, k)$, if for every q_1-tuple $\vec{u} \in D(P_1)$, there exists a q_2-tuple $\vec{u}' \in D(P_2)$ such that for all $1 \leq j \leq k$ we have that $u_{q_1-k+j} = u'_j$. A referential constraint can also be translated as an FO-sentence:

$$\forall \vec{x}\left(P_1(\vec{x}) \to \exists \vec{y}\left(P_2(\vec{y}) \wedge \bigwedge_{j=1}^{k}(x_{q_1-k+j} = y_j)\right)\right)$$

Example 2. Clearly, in the database instance in Example 1 there is a key dependency between IDs and the other attributes in relations *Students* and *Staff*, as it is to be expected from any well-defined notion of ID. On the other hand, in relation *Staff* the department is not a key, as two different tuples have "Mechanical Engineering" as value for this attribute. ■

3 Aggregators

The main research question we investigate in this paper concerns how to define an aggregated database instance from the instances of $\mathcal{N} = \{1,\ldots,n\}$ agents. This question is typical in social choice theory, where judgements, preferences, etc., are aggregated according to some notion of rationality that will be introduced in Section 5.

For the rest of the paper we fix a database schema $\mathcal{D}$ over a common domain $\mathcal{U}$, and consider a *profile* $\vec{D} = (D_1,\ldots,D_n)$ of n instances over $\mathcal{D}$ and $\mathcal{U}$. Then, we define what is an aggregation procedure on such instances.

Definition 4 (Aggregation Procedure)**.** *Given database schema $\mathcal{D}$ and domain $\mathcal{U}$, an* aggregation procedure $F : \mathcal{D}(\mathcal{U})^n \to \mathcal{D}(\mathcal{U})$ *is a function assigning to each tuple $\vec{D}$ of instances for n agents an aggregated instance $F(\vec{D}) \in \mathcal{D}(\mathcal{U})$.*

Let $\mathcal{F}$ be the class of all aggregation procedures.

We use $N_{\vec{u}}^{\vec{D}(P)} ::= \{i \in \mathcal{N} \mid \vec{u} \in D_i(P)\}$ to denote the set of agents accepting tuple $\vec{u}$ for symbol P, under profile $\vec{D}$. Note that considering a unique domain $\mathcal{U}$ is not really a limitation of the proposed approach: instances $D_1,\ldots,D_n$, each on a possibly different domain $\mathcal{U}_i$, for $i \leq n$, can all be seen as instances on the union $\bigcup_{i\in\mathcal{N}} \mathcal{U}_i$ of all domains.

Hereafter we illustrate and discuss some examples of aggregation procedures. We begin with the class of quota rules, inspired by their homonyms in judgment aggregation [13]. This class includes the classical majority rule, as well the union and the intersection rules which are well-known in modal epistemic logic, corresponding to distributed knowledge and "everybody knows that" [25].

Union Rule (or nomination): for every $P \in \mathcal{D}$, $F(\vec{D})(P) = \bigcup_{i\leq n} D_i(P)$. Intuitively, every agent is seen as having partial but correct information about the state of the world. Union can be considered a good aggregator if databases represent the agents' knowledge bases (certain information).

Intersection Rule (or unanimity): for every $P \in \mathcal{D}$, $F(\vec{D})(P) = \bigcap_{i\leq n} D_i(P)$. Here every agent is supposed to have a partial and possibly incorrect vision of the state of the world.

Quota Rules: a *quota* rule is an aggregation rule F defined via functions $q_P : \mathcal{U}^q \to \{0,1,\ldots,n+1\}$, associating each symbol P and q-uple with a quota, by stipulating that $\vec{u} \in F(\vec{D})(P)$ iff $|\{i \mid \vec{u} \in D_i(P)\}| \geq q_P(\vec{u})$. F is called *uniform* whenever q is a constant function for all tuples and symbols. Intuitively, if a tuple $\vec{u}$ appears in at least $q_P(\vec{u})$ of the initial databases, then it is accepted for symbol P.

The (strict) majority rule is a (uniform) quota rule for $q = \lceil (n+1)/2 \rceil$; while union and intersection are quota rule for $q = 1$ and $q = n$ respectively. We call the uniform quota rules for $q = 0$ and $q = n+1$ *trivial rules*.

The literature on belief merging has proposed and studied extensively procedures based on distances [27, 29, 28], and some of these rules have also been proposed in judgment aggregation [36]. We mention below one of the archetypal rules in this class, which makes use of the symmetric distance.

Distance-Based Rule:

$$F(\vec{D}) = \operatorname*{argmin}_{D\in\mathcal{D}-instances} \sum_{i\in\mathcal{N}} \sum_{P\in\mathcal{D}} (|D_i(P) \setminus D(P)| + |D(P) \setminus D_i(P)|)$$

Intuitively, the symmetric distance minimizes the "distance" between the aggregated database $F(\vec{D})$ and each D_i, defined as the number of tuples in D_i but not in $F(\vec{D})$, plus the number of tuples in $F(\vec{D})$ but not in D_i, calculated across all $i \in \mathcal{N}$.

Computing the result of distance-based rules is typically a hard computational problem: for instance, the above version on arbitrary propositional constraints is a Θ_2^p-complete problem [30]. Tractable versions can however be obtained by restricting the minimisation to the databases obtained in the input profiles, a viable solution when the set of individual agents in the input is sufficiently large. These rules are known in the literature on judgment aggregation as *most representative voter rules* [17], and we state here the simplest one.

Average Voter Rule:

$$F(\vec{D}) \quad = \quad \underset{\{D_i \mid i \in \mathcal{N}\}}{\operatorname{argmin}} \sum_{j \in \mathcal{N}} \sum_{P \in \mathcal{D}} (|D_j(P) \setminus D_i(P)| + |D_i(P) \setminus D_j(P)|)$$

Observe the the two rules above might output a set of equally preferred extensions for a relation symbol P. i.e., they are *non-resolute* rules. We also consider a slight variant of the average voter rule.

Relation-wise Average Voter Rule:

$$F(\vec{D}) \quad = \quad \bigcup_{P \in \mathcal{D}} \underset{\{D_i(P) \mid i \in \mathcal{N}\}}{\operatorname{argmin}} \sum_{j \in \mathcal{N}} (|D_j(P) \setminus D_i(P)| + |D_i(P) \setminus D_j(P)|) \tag{1}$$

Notice that, according to (1), the average is computed for each $P \in \mathcal{D}$ independently. In particular, $F(\vec{D})$ does not correspond to any of $D_1, \ldots, D_n$ in general.

We now state a class of rules which are typically considered non-desirable in the literature on social choice theory, since they leave a somewhat large set of agents out of the aggregation.

Dictatorship of Agent $i^* \in \mathcal{N}$: we have that $F(\vec{D}) = D_{i^*}$, i.e., the dictator i^* completely determines the aggregated database.

Oligarchy of Coalition $C^* \subseteq \mathcal{N}$: for every $P \in \mathcal{D}$, $F(\vec{D})(P) = \bigcap_{i \in C^*} D_i(P)$. Oligarchy reduces to dictatorship for singletons, and to intersection for $C^* = \mathcal{N}$.

To conclude, we present a novel definition of aggregation procedure inspired by the literature on incomplete information in databases [32].

Merge with Incomplete Information: for every $P \in \mathcal{D}$, $\vec{u} \in F(\vec{D})(P)$ iff (i) for some $\vec{u}_1 \in D_1(P)$, $\ldots$, $\vec{u}_n \in D_n(P)$, for every $k \leq q$, either $u_{1,k} = \ldots = u_{n,k}$ and $u_k = u_{1,k}$, or $u_{j,k} \neq u_{j'',k}$ for some $j, j'' \leq n$, and $u_j = \bot$, where $\bot$ is a new symbol; (ii) for every $\vec{u}' \in F(\vec{D})(P)$ and $k \leq q$, $u_k = \bot$ implies $u'_k = \bot$ or for some $k \leq q$, $u_k \neq u'_k$.

That is, by (i) whenever elements $u_1, \ldots, u_k$ appear at the same positions in some tuples in the profile, then they will appear at those positions in $F(\vec{D})$. On the other hand, if different elements appear, then we insert symbol $\bot$ as a placeholder. By (ii) we discard tuples with "strictly less" information. Notice that merge with incomplete information does not conform entirely with Def. 4, as the outcome $F(\vec{D})$ is a database instance on $\mathcal{D}(\mathcal{U} \cup \{\bot\})$, rather than $\mathcal{D}(\mathcal{U})$. Nonetheless all relevant notions on databases and aggregators can be extended seamlessly in what follows.

Example 3. Suppose that the database instance D_F in Example 1 is owned by the HR department of the faculty. On the other hand, the registrar and the head office own the following instances D'_F and D''_F respectively, due to differences in updating mechanisms and possibly errors:

Students			*Staff*		
ID	Name	Department	ID	Name	Department
10	Steve	History	01	Rose	Mech. Eng.
11	Carole	CS	02	Audrey	Mech. Eng.
			04	Carl	History

Students			*Staff*		
ID	Name	Department	ID	Name	Department
10	Steve	History	01	Rose	Mech. Eng.
11	Carole	CS	02	Aubrey	Mech. Eng.
12	Derek	Mech. Eng.	03	Karl	History
13	Marc	History			

To provide a unique vision of instances D_F, D'_F, and D''_F we can in principle choose any of the aggregation procedures introduced above. For instance, the intersection, union and majority aggregated profiles can be given as follows:

Intersection:

Students			*Staff*		
ID	Name	Department	ID	Name	Department
10	Steve	History	01	Rose	Mech. Eng.
11	Carole	CS			

Union:

Students			*Staff*		
ID	Name	Department	ID	Name	Department
10	Steve	History	01	Rose	Mech. Eng.
11	Carole	CS	02	Audrey	Mech. Eng.
12	Derek	Mech. Eng.	02	Aubrey	Mech. Eng.
13	Marc	History	03	Karl	History
			04	Carl	History

Majority:

Students			*Staff*		
ID	Name	Department	ID	Name	Department
10	Steve	History	01	Rose	Mech. Eng.
11	Carole	CS			
12	Derek	Mech. Eng.			

Clearly, some aggregation procedure do not preserve all integrity constraints, e.g., unions do not preserve key dependencies. Furthermore, aggregation by the average voter rule would output D_F, and by merge with incomplete information would produce the following instance:

Merge with inc. information:

Students			*Staff*		
ID	Name	Department	ID	Name	Department
10	Steve	History	01	Rose	Mech. Eng.
11	Carole	CS	02	$\perp$	Mech. Eng.
			$\perp$	$\perp$	History

In particular, in this last instance symbol $\perp$ intuitively signals that we are uncertain about the name of staff with ID 02, as well as about the ID and name of staff in the History department. ■

Note that further aggregation procedures are possible in principle. We choose to focus on those above as they are inspired by well-studied procedures from the literature and, with the exception of the distance-based rule, they are tractable computationally.

4 The Axiomatic Method

Aggregation procedures are best characterised by means of axioms. In particular, we consider the following properties, where relation symbols $P, P' \in \mathscr{D}$, profiles $\vec{D}, \vec{D}' \in \mathscr{D}(\mathscr{U})^n$, tuples $\vec{u}$, $\vec{u}' \in \mathscr{U}^+$ are all universally quantified. We leave the treatment of the merge with incomplete information rule for the end of the section.

Unanimity (U): $F(\vec{D})(P) \supseteq \bigcap_{i \in \mathcal{N}} D_i(P)$.

By unanimity a tuple accepted by all agents also appears in the aggregated database (for the relevant relation symbol). With the exception of the distance-based rule and trivial quota rules with any of the $q_P = n+1$, the remaining aggregators from Section 3 all satisfy unanimity.

Groundedness (G): $F(\vec{D})(P) \subseteq \bigcup_{i \in \mathcal{N}} D_i(P)$.

By groundedness any tuple appearing in the aggregated database must be accepted by some agent. All aggregators in Section 3, with the exception of the distance-based rule and the trivial quota rule with any of the $q_P = 0$, satisfy this property.

Anonymity (A): for every permutation $\pi : \mathcal{N} \to \mathcal{N}$, we have $F(D_1, \ldots, D_n) = F(D_{\pi(1)}, \ldots, D_{\pi(n)})$.

By anonymity the identity of agents is irrelevant for the aggregation procedure. This is the case for all aggregators in Section 3 but dictatorship and oligarchy.

Independence (I): if $N_{\vec{u}}^{\vec{D}(P)} = N_{\vec{u}}^{\vec{D}'(P)}$ then $\vec{u} \in F(\vec{D})(P)$ iff $\vec{u} \in F(\vec{D}')(P)$.

Intuitively, if the same agents accept (resp. reject) a tuple in two different profiles, then the tuple is accepted (resp. rejected) in both aggregated instances. The axiom of independence is a widespread requirement from social choice theory, and is arguably the main cause of most impossibility theorems, such as Arrow's seminal result [3]. From a computational perspective, independent rules are typically easier to compute than non-independent ones. Quota rules satisfy independence, as well as dictatorships and oligarchies.

Positive Neutrality (N^+): if $N_{\vec{u}}^{\vec{D}(P)} = N_{\vec{u}'}^{\vec{D}(P)}$ then $\vec{u} \in F(\vec{D})(P)$ iff $\vec{u}' \in F(\vec{D})(P)$.

Negative Neutrality (N^-): if $N_{\vec{u}}^{\vec{D}(P)} = \mathcal{N} \setminus N_{\vec{u}'}^{\vec{D}(P)}$ then $\vec{u} \in F(\vec{D})(P)$ iff $\vec{u}' \notin F(\vec{D})(P)$.

Observe that both versions of neutrality differ from independence as here we consider two different tuples in the same profile, while independence deals with the same tuple in two different profiles. Again, with the exception of the distance-based rule all aggregators introduced in Section 3 satisfy positive neutrality. Most quota rules including union and intersection do not satisfy negative neutrality (see Lemma 2 below), dictatorships and oligarchies satisfy the latter axiom.

Permutation-Neutrality (N^P): Let $\rho : \mathcal{U} \to \mathcal{U}$ be a permutation over the domain $\mathcal{U}$, and $\rho(\vec{D})$ its straightforward lifting to a profile $\vec{D}$, then $F(\rho(\vec{D})) = \rho(F(\vec{D}))$.

All aggregators introduced in Section 3 satisfy permutation-neutrality. We conclude with the following axiom, that formalises the fact that an aggregator keeps on accepting a given tuple if the support for that tuple increases.

Monotonicity (M): if $\vec{u} \in F(\vec{D})(P)$ and for every $i \in \mathcal{N}$, either $D_i(P) = D'_i(P)$ or $D_i(P) \cup \{\vec{u}\} \subseteq D'_i(P)$, then $\vec{u} \in F(D')(P)$.

Combinations of the axioms above can be used to characterise some of the rules that we defined in Section 3. Some of these results, such as the following, lift to databases known results in judgement (propositional) aggregation.

Lemma 1. *An aggregation procedure satisfies A, I, and M iff it is a quota rule.*

Proof sketch. The right-to-left implication follows from the fact that quota rules satisfy independence I, anonymity A, and monotonicity M, as we remarked above. For the left-to-right implication, observe that, to accept a given tuple $\vec{u}$ in $F(\vec{D})(P)$, an independent aggregation procedure will only look at the set of agents $i \in \mathcal{N}$ such that $\vec{u} \in D_i(P)$. If the procedure is also anonymous, then acceptance is based only on the number of individuals admitting the tuple. Finally, by monotonicity, there is a minimal number of agents required to trigger collective acceptance. That number is the quota associated with the tuple and the symbol at hand. □

If we add neutrality, then we obtain the class of uniform quota rules. If we furthermore impose unanimity and groundedness, we exclude the trivial quota rules.

Lemma 2. *If the number of individuals is odd and* $|\mathscr{D}| \geq 2$*, an aggregation procedure* F *satisfies A,* N^-*,* N^+*, I and M on the full domain* $\mathscr{D}(\mathscr{U})^n$ *if and only if it is the majority rule.*

Proof. By positive neutrality the quota must be the same for all tuples and all relation symbols. By negative neutrality the two sets $N_{\vec{u}}^{\vec{D}(P)}$ and $\mathscr{N} \setminus N_{\vec{u}}^{\vec{D}(P)}$ must be treated symmetrically. Hence, the only possibility is to have a uniform quota of $(n+1)/2$. □

The corresponding versions of these results have been proved to hold in judgment and graph aggregation [13, 18]. We now show the following equivalence between majority and the distance-based rule in the absence of integrity constrains:

Lemma 3. *In absence of constraints, and for an odd number of agents, the distance-based rule coincides with the majority rule.*

Proof. With a slight abuse of notation, if $A \subseteq \mathscr{U}^m$ let $A(\vec{u})$ be its characteristic function. Recall the definition of the distance-based rule. Since the minimisation is not constrained, and all structures are finite, the definition is equivalent to the following:

$$\begin{aligned} F(\vec{D})(P) &= \underset{D \in \mathscr{D}-instances}{\operatorname{argmin}} \sum_{i \in \mathscr{N}} \sum_{P \in \mathscr{D}} \sum_{\vec{u} \in \mathscr{U}^{q_P}} (|D_i(P)(\vec{u}) - D(P)(\vec{u})|) \\ &= \underset{D \in \mathscr{D}-instances}{\operatorname{argmin}} \sum_{P \in \mathscr{D}} \sum_{\vec{u} \in \mathscr{U}^{q_P}} \sum_{i \in \mathscr{N}} (|D_i(P)(\vec{u}) - D(P)(\vec{u})|) \end{aligned}$$

Therefore, for each $P \in \mathscr{D}$ and for each $\vec{u}$, if for a majority of the individuals in $\mathscr{N}$ we have that $\vec{u} \in D_i(P)$, then $\vec{u} \in D(P)$ minimises the overall distance, and symmetrically for the case in which a majority of individuals are such that $\vec{u} \notin D_i(P)$. □

It is easy to see that the above lemma does not hold in presence of arbitrary integrity constraints (see Example 4). We conclude this section with a result on merge with incomplete information. Note that the axioms need to be slightly adapted to account for the addition of a symbol in the output of the aggregator.

Lemma 4. *Merge with incomplete information satisfies U, A, I, and* N^+*, but not* N^-*, nor M. In particular, by Lemma 1 it is not a quota rule.*

Proof. The proof that merge satisfies unanimity U is immediate, similarly for anonymity A. As regards independence I, if $N_{\vec{u}}^{\vec{D}(P)} = N_{\vec{u}}^{\vec{D}'(P)} = \mathscr{N}$ then $\vec{u} \in F(\vec{D})(P)$ and $\vec{u} \in F(\vec{D}')(P)$ by unanimity. On the other hand, if $N_{\vec{u}}^{\vec{D}(P)} = N_{\vec{u}}^{\vec{D}'(P)} \neq \mathscr{N}$ then $\vec{u} \notin F(\vec{D})(P)$ and $\vec{u} \notin F(\vec{D}')(P)$. For positive neutrality the reasoning is similar: if $N_{\vec{u}}^{\vec{D}(P)} = N_{\vec{u}'}^{\vec{D}(P)} = \mathscr{N}$ then $\vec{u} \in F(\vec{D})(P)$ and $\vec{u}' \in F(\vec{D})(P)$; whereas if $N_{\vec{u}}^{\vec{D}(P)} = N_{\vec{u}'}^{\vec{D}(P)} \neq \mathscr{N}$ then $\vec{u} \notin F(\vec{D})(P)$ and $\vec{u}' \notin F(\vec{D})(P)$. Finally, it is not difficult to find counterexamples for both negative neutrality N^- and monotonicity M. For instance, as regards N^- consider Example 3 and tuples (02,*Audrey*,*Mech. Eng.*) and (02,*Aubrey*,*Mech. Eng.*) in relation *Staff*.

□

5 Collective Rationality

In this section we analyse further the properties of the aggregation procedures introduced in Section 3. First, we present a notion of *collective rationality* that aims to capture the appropriateness of a given aggregator F w.r.t. some constraint φ on the input instances $D_1, \ldots, D_n$. Hereafter let φ be a sentence in the first-order language $\mathcal{L}_{\mathscr{D}}$ associated with $\mathscr{D}$, interpreted as an integrity constraint that is satisfied by all $D_1, \ldots, D_n$.

Definition 5 (Collective Rationality)**.** *A constraint φ is lifted by an aggregation procedure F if whenever $D_i \models \varphi$ for all $i \in \mathcal{N}$, then also $F(\vec{D}) \models \varphi$. An aggregation procedure $F : \mathscr{D}(\mathscr{U})^n \to \mathscr{D}(\mathscr{U})$ is* collectively rational *(CR) with respect to φ iff F lifts φ.*

Intuitively, an aggregator is CR w.r.t. constraint φ iff it lifts, or preserves, φ. Consider the following:

Example 4. By Example 3 unions are not collective rational w.r.t. dependency constraints. We also provide a further example of first-order collective (ir)rationality with the majority rule. Consider agents 1 and 2 with database schema $\mathscr{D} = \{P/1, Q/2\}$. Two database instances are given as $D_1 = \{P(a), Q(a,b)\}$ and $D_2 = \{P(a), Q(a,c)\}$. Clearly, both instances satisfy integrity constraint $\varphi = \forall x(P(x) \to \exists y Q(x,y))$. However, their aggregate $D = F(D_1, D_2) = \{P(a)\}$, obtained by the majority rule, does not satisfy φ. These small examples, which can be considered a *paradox* in the sense of [22], shows that not every constraint in the language $\mathcal{L}_{\mathscr{D}}$ is collective rational w.r.t. unions and majority, thus obtaining a first, simple negative result. ■

We now focusing on integrity constraints that are proper to databases, as defined in Section 2, presenting sufficient (and possibly necessary) conditions for aggregators to lift them. We begin with functional dependencies.

Proposition 5. *A quota rule lifts a functional constraint iff for all relation symbols P occurring in the functional constraint we have that $q_P > \frac{n}{2}$, where n is the number of agents.*

Proof. By assumption, every instance D_i satisfies the constraint. That is for every tuple $(u_1, \ldots, u_k)$, either there is a unique $(u_{k+1}, \ldots, u_q)$ such that $(u_1, \ldots, u_q) = \vec{u} \in D_i(P)$, or there is none. Suppose now that the constraint is falsified by the collective outcome. That is, there are $\vec{u} \neq \vec{u}'$ such that both $\vec{u} \in F(\vec{D})(P)$ and $\vec{u}' \in F(\vec{D})(P)$, and $\vec{u}$ and $\vec{u}'$ coincide on the first k coordinates. By definition of quota rules, this means that at least q_P voters are such that $\vec{u} \in D_i(P)$, and at least q_P possibly different voters had $\vec{u}' \in D_i(P)$. Since each individual can have either $\vec{u}$ or $\vec{u}'$ in $D_i(P)$, by the pigeonhole principle this is possible if and only if the quota $q_P \leq \frac{n}{2}$. □

As immediate applications of Prop. 5, the intersection rule clearly lifts any functional dependency, while the union lifts none, as previously illustrated.

Proposition 6. *An aggregation procedure F lifts a value constraint if F is grounded.*

Proof. Let $n_k \in D(P_v)$ be a value constraint, where for all $i, j \in \mathcal{N}$, we have that $D_i(P_v) = D_j(P_v)$. A grounded aggregation procedure is such that $F(\vec{D})(P) \subseteq \bigcup_{i \in \mathcal{N}} D_i(P)$. Hence, for all $\vec{u} \in F(\vec{D})(P)$, there exists an $i \in \mathcal{N}$ such that $\vec{u} \in D_i(P)$. Since all individual databases satisfy the value constraint, we have that $u_k \in D_i(P_v)$, and therefore $u_k \in F(\vec{D})(P_v) \subseteq \bigcup_{i \in \mathcal{N}} D_i(P_v)$, showing that also $F(\vec{D})(P)$ satisfies the value constraint. □

The converse of the Prop. 6 is not true in general, since a non-grounded aggregator could be easily devised while still satisfying a given value constraint. Finally, we consider referential constraints.

Proposition 7. *A quota rule lifts a referential constraint* $(P_1 \to P_2, k)$ *iff* $q_{P_2} = 1$.

Proof. Let $\vec{u} \in F(\vec{D})(P_1)$. Since all the individual databases satisfy the integrity constraint, we know that for every $i \in \mathcal{N}$ there exists a $\vec{u}_i \in D_i(P_2)$ such that its first k coordinates coincides with the last k coordinates of P_1. Since all $\vec{u}_i$ are possibly different, they may be supported by one single individual each. Therefore, the referential constraint is lifted if and only if the quota relative to P_2 is sufficiently small, i.e., $q_{P_2} = 1$. □

As an immediate application of Prop. 7, intersection and union rules are included in the results above, since they are quota rules. As for the distance-based rule or the average voter rule, we only remark that they lift all integrity constraint by their definition, provided that the minimisation is restricted to consistent databases.

For simpler integrity constraints, notably conjunctions of literals, we show a simple correspondence theorem in the spirit of [22], albeit adapted to the first-order language under consideration. First, we introduce a set $Con \subseteq \mathcal{U}$ of constants in the first-order language, interpreted as themselves in each D_i, that is, $\sigma(c) = c$ for every $c \in Con$. Then, let $lit^+ \subseteq \mathcal{L}_\mathcal{D}$ be some language containing only positive literals of form $P(c_1, \dots, c_q)$, let $lit^- \subseteq \mathcal{L}_\mathcal{D}$ be the set containing only *negative* literals $\neg P(c_1, \dots, c_q)$, and $lit = lit^+ \cup lit^-$. We can prove the following:

Theorem 8. *If an aggregator F satisfies U and G, then it is collectively rational w.r.t. lit. If* $Con = \mathcal{U}$*, then every aggregator that is collectively rational w.r.t. lit is also unanimous and grounded.*

Proof. Suppose that aggregator F satisfies U and G. Then, we show that it is collectively rational w.r.t. *lit*. In particular, if all instances $D_1, \dots, D_n$ satisfy formulas $P(c_1, \dots, c_q)$ in lit^+, then $\vec{c} \in D_i(P)$ for every $i \in \mathcal{N}$. By unanimity we have that $\bigcap_{i \in \mathcal{N}} D_i(P) \subseteq F(\vec{D})(P)$, and therefore $\vec{c} \in F(\vec{D})(P)$. Hence, F is collectively rational on lit^+. A similar reasoning holds for lit^- by using groundedness, and therefore for the whole *lit*.

As for the converse, suppose that $Con = \mathcal{U}$ and F is collectively rational wrt to *lit*. Then, choose a profile $D_1, \dots, D_n$ with $\vec{u} \in \bigcap_{i \in \mathcal{N}} D_i(P)$, that is, for every $i \in \mathcal{N}$, $D_i \models P(u_1, \dots, u_q)$. Since we assumed that $Con = \mathcal{U}$, all formulas $P(u_1, \dots, u_q)$ are in lit^+. Further, F is CR on $D_1, \dots, D_n$ and therefore $F(\vec{D}) \models P(u_1, \dots, u_q)$, that is, $\vec{u} \in F(\vec{D})(P)$, which means that F is unanimous. Similarly, and under the same assumption, any F that is collectively rational w.r.t. lit^- is grounded. □

Note that, differently from the propositional case [22, Theorem 10], here we need both axioms of unanimity and groundedness to preserve both positive and negative literals, while for propositional languages unanimity suffices.

Given the results above, a natural question is to identify the class of aggregators that can lift any integrity constraint, no matter its form. Let us first define the following class:

Definition 6 (Generalised dictatorship)**.** *An aggregation procedure* $F : \mathcal{D}(\mathcal{U})^n \to \mathcal{D}(\mathcal{U})$ *is a* generalised dictatorship *if there exists a map* $g : \mathcal{D}(\mathcal{U})^n \to \mathcal{N}$ *such that for every* $\vec{D} \in \mathcal{D}(\mathcal{U})^n$*,* $F(\vec{D}) = D_{g(\vec{D})}$*. Let GDIC be the class of generalised dictatorships.*

Generalised dictatorships include classical dictatorships, but also more interesting procedures such as the average voter rule from Section 3, or any other rule which selects the individual input that best summarises a given profile. Clearly, since each single instance satisfies the given set of constraints, a generalised dictatorship is collectively rational with respect to the full first-order language.

Theorem 9. *GDIC* $\subset CR[\mathcal{L}_\mathcal{D}]$

Observe that while for binary aggregation the theorem above is an equality [22, Theorem 16], this is not the case for database aggregation. This is due to the fact that the first-order language specifies a given database instance only up to isomorphism. The proof of this fact is rather immediate: consider a dictatorship of the first agent, modified by permuting all the elements in $\mathscr{U}$. That is, $F(\vec{D}) = \rho(D_1)$ where $\rho : \mathscr{U} \to \mathscr{U}$ is any permutation different from the identity. Clearly, $D_1 \neq \rho(D_1)$ but all constraints that were satisfied by D_1 are also satisfied by $\rho(D_1)$. Hence, this aggregator is collectively rational with respect to the full first-order language $\mathscr{L}_{\mathscr{D}}$, but is not a generalised dictatorship.

6 Aggregation and Query Answering

In this section we analyse one of the most common operations performed on databases, i.e., query answering, in the light of (rational) aggregation. Observe that any open formula $\varphi(x_1, \ldots, x_\ell)$, with free variables $x_1, \ldots, x_\ell$, can be thought of as a query [1]. Evaluating $\varphi(x_1, \ldots, x_\ell)$ on a database instance D returns the set $ans(D, \varphi)$ of tuples $\vec{u} = (u_1, \ldots, u_\ell)$ such that the assignment σ, with $\sigma(x_i) = u_i$ for $i \leq \ell$, satisfies φ, that is, $(D, \sigma) \models \varphi$. Hereafter, with an abuse of notation, we often write simply $(D, \vec{u}) \models \varphi$. Given the importance of query answering in database theory, the following question is of obvious interest.

Question 10. *What is the relation between the answer $ans(F(\vec{D}), \varphi)$ to query φ on the aggregated database $F(\vec{D})$, and answers $ans(D_1, \varphi), \ldots, ans(D_n, \varphi)$ to the same query on each instance $D_1, \ldots, D_n$?*

Clearly, given a query φ, every aggregator F on database instances induces an aggregation procedure F^* on the query answers, as illustrated by the following diagram:

$$\begin{array}{ccc} D_1, \ldots, D_n & \xrightarrow{\quad F \quad} & F(\vec{D}) \\ \varphi \big\downarrow & & \big\downarrow \varphi \\ ans(D_1, \varphi), \ldots, ans(D_n, \varphi) & \xrightarrow{\quad F^* \quad} & ans(F(\vec{D}), \varphi) \end{array}$$

Hereafter we consider some examples to illustrate this question.

Example 5. If we assume intersection as the aggregation procedure, it is easy to check that in general the answer to a query in the aggregated database is not the intersection of the answers for each single instance. To see this, let $D_1(P) = \{(a,b)\}$ and $D_2(P) = \{(a,d)\}$ and consider query $\varphi = \exists y P(x,y)$. Clearly, $ans(D_1 \cap D_2, \varphi)$ is empty, while $ans(D_1, \varphi) \cap ans(D_2, \varphi) = \{a\}$. Hence, in general $\bigcap_{i \in \mathscr{N}} ans(D_i, \varphi) \not\subseteq ans(\bigcap_{i \in \mathscr{N}} D_i, \varphi)$. The converse can also be the case. Consider instances D_1, D_2 such that $D_1(P) = \{(a,a), (a,b)\}$, $D_1(R) = \{c\}$, and $D_2(P) = \{(a,a), (a,b)\}$, $D_2(R) = \{d\}$, with query $\varphi = \forall y P(x,y)$. The intersection $ans(D_1, \varphi) \cap ans(D_2, \varphi)$ of answers is empty. However the answer w.r.t. the intersection of databases is $ans(D_1 \cap D_2, \varphi) = \{a\}$, since the active domain of the intersection only includes elements a and b. As a result, in general $ans(\bigcap_{i \in \mathscr{N}} D_i, \varphi) \not\subseteq \bigcap_{i \in \mathscr{N}} ans(D_i, \varphi)$.

A similar argument shows that the union of answers is in general different from the answer on the union of instances. ■

These examples shows that it is extremely difficult to find aggregators such that the diagram above commutes for any first-order query $\varphi \in \mathscr{L}_{\mathscr{D}}$. Hence, they naturally raise the question of syntactic restrictions on queries such that the aggregation procedure $F^* = \varphi \circ F \circ \varphi^{-1}$ on answers can be expressed explicitly in terms of F (e.g., the intersection of answers is the answer to the query on the intersection)[2]:

[2] Hereafter, with an abuse of notation, we write φ also to mean the corresponding query evaluation for formula φ.

Question 11. *Given aggregation procedures F and F^*, is there a restriction on the query language for φ such that the diagram above commutes?*

This problem is related to the following, more general question.

Question 12. *Given an aggregation procedure F and a query language $\mathscr{L}$, what is the aggregation procedure F^*? Can F^* be represented "explicitly", for instance as one of the aggregation procedure introduced in Sec. 3?*

For instance, it is immediate that if F and F^* are both dictatorships for the same agent $i^* \in \mathscr{N}$, then the whole first-order language $\mathscr{L}$ is preserved, that is, the result of querying and then aggregating by F^* is the same as aggregating by F and then querying.

The results in this section provide a first, partial answer to Question 11. Hereafter, with an abuse of notation, we consider functor F^* as an aggregator on databases. Indeed, all $ans(D_i, \varphi)$ can be seen as a finite relational structures, to which we can apply the aggregators in Sec. 3, as well as the axioms in Sec. 4.

Let us first introduce the positive existential and universal fragments $\mathscr{L}^+_\exists$ and $\mathscr{L}^+_\forall$ of first-order logic, defined respectively as follows:

$$\begin{aligned} \varphi &::= x = x' \mid P(x_1, \ldots, x_q) \mid \varphi \vee \varphi \mid \exists x \varphi \\ \varphi &::= x = x' \mid P(x_1, \ldots, x_q) \mid \varphi \wedge \varphi \mid \forall x \varphi \end{aligned}$$

Our first lemma shows a positive result related to Example 5, when union is considered instead of intersection.

Lemma 13 (Existential Fragment)**.** *The language $\mathscr{L}^+_\exists$ is preserved by unions, that is, for F and F^* equal to set-theoretical union, the diagram commutes for the query language $\mathscr{L}^+_\exists$.*

Proof. The proof is by induction on the structure of query φ. For atomic $\varphi = P(x_1, \ldots, x_q)$, we have that $\vec{u} \in ans(\bigcup_{i\in\mathscr{N}} D_i, \varphi)$ iff $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u}) \models \varphi$, iff for some $i \in \mathscr{N}$, $(D_i, \vec{u}) \models \varphi$, iff $\vec{u} \in ans(D_i, \varphi)$ for some $i \in \mathscr{N}$, iff $\vec{u} \in \bigcup_{i\in\mathscr{N}} ans(D_i, \varphi)$.

For $\varphi = \psi \vee \psi'$, $\vec{u} \in ans(\bigcup_{i\in\mathscr{N}} D_i, \varphi)$ iff $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u}) \models \varphi$, iff $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u}_1) \models \psi$ or $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u}_2) \models \psi'$, iff for some $i, j \in \mathscr{N}$, $(D_i, \vec{u}_1) \models \psi$ or $(D_j, \vec{u}_2) \models \psi'$ by induction hypothesis, where $\vec{u}_1$ and $\vec{u}_2$ are suitable subsequences of $\vec{u}$. In particular, we have both $(D_i, \vec{u}) \models \psi \vee \psi'$ and $(D_j, \vec{u}) \models \psi \vee \psi'$, that is, $\vec{u} \in \bigcup_{i\in\mathscr{N}} ans(D_i, \varphi)$. On the other hand, $\vec{u} \in \bigcup_{i\in\mathscr{N}} ans(D_i, \varphi)$ iff $\vec{u} \in ans(D_i, \varphi)$ for some $i \in \mathscr{N}$, iff $(D_i, \vec{u}_1) \models \psi$ or $(D_i, \vec{u}_2) \models \psi'$. In both cases, by induction hypothesis $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u}) \models \varphi$, that is, $\vec{u} \in ans(\bigcup_{i\in\mathscr{N}} D_i, \varphi)$.

For $\varphi = \exists x \psi$, $\vec{u} \in ans(\bigcup_{i\in\mathscr{N}} D_i, \varphi)$ iff $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u}) \models \varphi$, iff for some $v \in adom(\bigcup_{i\in\mathscr{N}} D_i)$, $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u} \cdot v) \models \psi$, and therefore for some $i, j \in \mathscr{N}$, $v \in adom(D_j)$ and $(D_i, \vec{u} \cdot v) \models \psi$. Note that if $(D_i, \vec{u} \cdot v) \models \psi$, then $v \in adom(D_i)$ as well, as φ belongs to the positive (existential) fragment of first-order logic. Hence, for some $i \in \mathscr{N}$, $v \in adom(D_i)$ and $(D_i, \vec{u} \cdot v) \models \psi$, that is, $\vec{u} \in ans(D_i, \varphi)$ for some $i \in \mathscr{N}$. On the other hand, $\vec{u} \in \bigcup_{i\in\mathscr{N}} ans(D_i, \varphi)$ iff $\vec{u} \in ans(D_i, \varphi)$ for some $i \in \mathscr{N}$, iff $v \in adom(D_i)$ and $(D_i, \vec{u} \cdot v) \models \psi$, that is, $v \in adom(\bigcup_{i\in\mathscr{N}} D_i)$ and $(\bigcup_{i\in\mathscr{N}} D_i, \vec{u} \cdot v) \models \psi$ by induction hypothesis. Hence, $\vec{u} \in ans(\bigcup_{i\in\mathscr{N}} D_i, \varphi)$. □

By Lemma 13 queries in $\mathscr{L}^+_\exists$ are preserved whenever both F and F^* are unions. The interest of such a result is that, in order to get an answer to query $\varphi \in \mathscr{L}^+_\exists$ in the aggregated databases $F(\vec{D})$, we might run the query on each instance D separately, and then aggregate the results, whichever is more efficient depending on the size of query φ and instances $D_1, \ldots, D_n$.

Further, we may wonder whether a result symmetric to Lemma 13 holds for intersections and the positive universal fragment $\mathscr{L}^+_\forall$ of first-order logic. Unfortunately, in Example 5 we provided a formula

$\varphi = \forall y P(x,y)$ in $\mathscr{L}_\forall^+$ and instances D_1, D_2 such that $ans(D_1 \cap D_2, \varphi) \not\subseteq ans(D_1, \varphi) \cap ans(D_2, \varphi)$. Hence, for F and F^* equal to set-theoretical intersection, the diagram above does not commute for the query language $\mathscr{L}_\forall^+$.

Nonetheless, we are able to prove a weaker but still significant result related to Question 12. Specifically, the next lemma shows that if in the diagram above F is unanimous and the query language is $\mathscr{L}_\forall^+$, then F^* is unanimous, in the sense that $\bigcap_{i\in\mathscr{N}} ans(D_i, \varphi) \subseteq ans(\bigcap_{i\in\mathscr{N}} D_i), \varphi)$.

Lemma 14. *Let aggregator F be unanimous and let $\mathscr{L}_\forall^+$ be the query language. Then, the induced aggregator F^* is also unanimous.*

Proof. We prove that $\bigcap_{i\in\mathscr{N}} ans(D_i, \varphi) \subseteq ans(F(\vec{D}), \varphi)$. So, if $\vec{u} \in \bigcap_{i\in\mathscr{N}} ans(D_i, \varphi)$ then for every agent $i \in \mathscr{N}$, $(D_i, \vec{u}) \models \varphi$. We now prove by induction on $\varphi \in \mathscr{L}_\forall^+$ that if for every $i \in \mathscr{N}$, $(D_i, \vec{u}) \models \varphi$, then $(F(\vec{D}), \vec{u}) \models \varphi$, that is $\vec{u} \in ans(F(\vec{D}), \varphi)$.

As to the base case for $\varphi = P(x_1, \ldots, x_q)$ atomic, $(D_i, \vec{u}) \models \varphi$ iff $\vec{u} \in D_i(P)$ for every $i \in \mathscr{N}$. In particular, $\vec{u} \in F(\vec{D})(P)$ as well by unanimity, and therefore $(F(\vec{D}), \vec{u}) \models P(x_1, \ldots, x_q)$. The case for identity is immediate.

As to the inductive case for $\varphi = \psi \wedge \psi'$, suppose that for every $i \in \mathscr{N}$, $(D_i, \vec{u}) \models \varphi$, that is, $(D_i, \vec{u}_1) \models \psi$ and $(D_i, \vec{u}_2) \models \psi$ for suitable $\vec{u}_1$ and $\vec{u}_2$ subsequences of $\vec{u}$. By induction hypothesis we obtain that $(F(\vec{D}), \vec{u}_1) \models \psi$ and $(F(\vec{D}), \vec{u}_2) \models \psi'$, i.e., $(F(\vec{D}), \vec{u}) \models \varphi$. Finally, if $(D_i, \vec{u}) \models \forall x \psi$ for every $i \in \mathscr{N}$, then for all $v \in adom(D_i)$, $(D_i, \vec{u} \cdot v) \models \psi$. In particular, for all $v \in adom(F(\vec{D}))$, $(D_i, \vec{u} \cdot v) \models \psi$ for every $i \in \mathscr{N}$, and by induction hypothesis, for all $v \in adom(F(\vec{D}))$, $(F(\vec{D}), \vec{u} \cdot v) \models \psi$, i.e., $(F(\vec{D}), \vec{u}) \models \forall x \psi$. □

Note that set-theoretical intersection is unanimous.

A result symmetric to Lemma 14 holds for language $\mathscr{L}_\exists^+$ and unions:

Lemma 15. *Let aggregator F be grounded and let $\mathscr{L}_\exists^+$ be the query language. Then, the induced aggregator F^* is also grounded.*

Proof. We prove that $ans(F(\vec{D}), \varphi) \subseteq \bigcup_{i\in\mathscr{N}} ans(D_i, \varphi)$. So, if $\vec{u} \in ans(F(\vec{D}), \varphi)$ then $(F(\vec{D}), \vec{u}) \models \varphi$. We now prove by induction on $\varphi \in \mathscr{L}_\exists^+$ that if $(F(\vec{D}), \vec{u}) \models \varphi$, then for some $i \in \mathscr{N}$, $(D_i, \vec{u}) \models \varphi$, and therefore $\vec{u} \in \bigcup_{i\in\mathscr{N}} ans(D_i, \varphi)$.

As to the base case for $\varphi = P(x_1, \ldots, x_q)$ atomic, $(F(\vec{D}), \vec{u}) \models \varphi$ iff $\vec{u} \in F(\vec{D})(P)$, iff $\vec{u} \in D_i(P)$ for some agents $i \in \mathscr{N}$ by groundedness. In particular, $(D_i, \vec{u}) \models \varphi$ as well. The case for identity is immediate.

As to the inductive case for $\varphi = \psi \vee \psi'$, suppose that $(F(\vec{D}), \vec{u}) \models \varphi$, i.e., $(F(\vec{D}), \vec{u}_1) \models \psi$ or $(F(\vec{D}), \vec{u}_2) \models \psi'$ for suitable $\vec{u}_1$ and $\vec{u}_2$ subsequences of $\vec{u}$. In the former case, by induction hypothesis we have that for some $i \in \mathscr{N}$, $(D_i, \vec{u}_1) \models \psi$, and therefore $(D_i, \vec{u}) \models \varphi$. The case for $(F(\vec{D}), \vec{u}_2) \models \psi'$ is symmetric. Finally, if $(F(\vec{D}), \vec{u}) \models \exists x \psi$, then for some $v \in adom(F(\vec{D}))$, $(F(\vec{D}), \vec{u} \cdot v) \models \psi$. In particular, by induction hypothesis, $(D_i, \vec{u} \cdot v) \models \psi$ for some $i \in \mathscr{N}$. Since ψ is a positive formula, $v \in adom(D_i)$, and therefore, $(D_i, \vec{u} \cdot v) \models \varphi$. □

Note that Lemma 14 and 15 apply to all quota rules, including union and intersection, though the query language is rather limited.

We now move towards more practical query answering and consider the language $\mathscr{L}_{CQ}$ of (unions of) conjuctive queries, which is a popular query language in the theory of databases thanks to its NP-complete query answering problem [10]. Formulas in $\mathscr{L}_{CQ}$ are defined according to the following BNF:

$$\varphi \quad ::= \quad P_1(x_1, \ldots, x_{q_1}) \wedge \ldots \wedge P_m(x_1, \ldots, x_{q_m}) \mid \varphi \vee \varphi \mid \exists x \varphi$$

We now show that the result of conjunctive queries is preserved by merge with incomplete information.

Lemma 16. *Let aggregator F be merge with incomplete information and let $\mathscr{L}_{CQ}$ be the query language. Then, the induced aggregator F^* is also the merge rule.*

Proof. We show $F^*(ans(D_1,\varphi),\ldots,ans(D_n,\varphi)) = ans(F(\vec{D}),\varphi)$, where both F and F^* are the merge rule, by induction on $\varphi \in \mathscr{L}_{CQ}$. As to the base case for $\varphi = P_1(x_1,\ldots,x_{q_1}) \wedge \ldots \wedge P_m(x_1,\ldots,x_{q_m})$, $(F(\vec{D}),\vec{u}) \models \varphi$ iff $\vec{u}_1 \in F(\vec{D})(P_1)$, $\ldots$, $\vec{u}_m \in F(\vec{D})(P_m)$, where each $\vec{u}_i$ is a suitable subsequences of $\vec{u}$. This is the case iff for every $j \leq n$, $\vec{u}'_{j,1} \in D_j(P_1)$, $\ldots$, $\vec{u}'_{j,m} \in D_j(P_m)$, where each $\vec{u}'_{j,i}$ differs from $\vec{u}_i$ as the latter might contain $\perp$ in designated positions. Again, the above is the case iff for every $j \leq n$, $(D_j,\vec{u}'_j) \models \varphi$, that is, $\vec{u} \in F^*(\{\vec{u}'_1\},\ldots,\{\vec{u}'_n\})$, where F^* is the merge rule. For $\varphi = \psi \vee \psi'$, $(F(\vec{D}),\vec{u}) \models \varphi$ iff $(F(\vec{D}),\vec{u}_1) \models \psi$ or $(F(\vec{D}),\vec{u}_2) \models \psi'$, where $\vec{u}_1$ and $\vec{u}_2$ are suitable subsequences of $\vec{u}$. By induction hypothesis, this is the case iff $\vec{u}_1 \in F^*(ans(D_1,\psi),\ldots,ans(D_n,\psi))$ or $\vec{u}_2 \in F^*(ans(D_1,\psi'),\ldots,ans(D_n,\psi'))$, that is, iff $\vec{u} \in F^*(ans(D_1,\varphi),\ldots,ans(D_n,\varphi))$. Finally, for $\varphi = \exists x\psi$, $\vec{u} \in ans(F(\vec{D}),\varphi)$ iff for some $v \in adom(F(\vec{D}))$, $(F(\vec{D}),\vec{u}\cdot v) \models \psi$, iff $\vec{u}\cdot v \in F^*(ans(D_1,\psi),\ldots,$ $ans(D_n,\psi))$ by induction hypothesis. By definition of F^*, for every $j \leq n$, $(D_j,\vec{u}'\cdot v') \models \psi$, where $\vec{u}'\cdot v'$ differs from $\vec{u}\cdot v$ as the latter might contain $\perp$ in designated positions. The above is the case iff for every $j \leq n$, $(D_j,\vec{u}') \models \varphi$, iff $\vec{u} \in F^*(ans(D_1,\varphi),\ldots,ans(D_n,\varphi))$, where F^* is the merge rule. □

By Lemma 16 we can query the individual instances and then merge the corresponding answers instead of querying the merged database.

By using the relation-wise average voter rule, we are able to prove the following preservation result. Hereafter, the average Ave of answers $ans(D_1,\varphi),\ldots,ans(D_n,\varphi)$ is computed as follows:

$$Ave(ans(D_1,\varphi),\ldots,ans(D_n,\varphi)) = \underset{ans(D_i,\varphi)|i\in\mathscr{N}\}}{\operatorname{argmin}} \sum_{j\in\mathscr{N}} (|ans(D_j,\varphi)\setminus ans(D_i,\varphi)| + |ans(D_i,\varphi)\setminus ans(D_j,\varphi)|) \quad (2)$$

Note that the relation-wise average voter rule can be defined as the union of the averages of the individual relations associated to each $P \in \mathscr{D}$.

Lemma 17. *Let aggregator F be the average rule and let first-order logic $\mathscr{L}$ be the query language. Then, $F^*(ans(D_1,\varphi),\ldots,ans(D_n,\varphi)) = ans(F(\vec{D}),\varphi)$ is a subset of $Ave(ans(D_1,\varphi),\ldots,ans(D_n,\varphi))$.*

Proof. If $\vec{u} \in F^*(ans(D_1,\varphi),\ldots,ans(D_n,\varphi)) = ans(F(\vec{D}),\varphi)$ then $(F(\vec{D}),\vec{u}) \models \varphi$. Now we prove on induction of the structure of φ that if $(F(\vec{D}),\vec{u}) \models \varphi$ then $\vec{u}$ belongs to the average of $ans(D_1,\varphi),\ldots,$ $ans(D_n,\varphi)$. For $\varphi = P(x_1,\ldots,x_q)$, if $(F(\vec{D}),\vec{u}) \models \varphi$ then $\vec{u} \in F(\vec{D})(P)$, and therefore $\vec{u}$ belongs to the average of $ans(D_1,\varphi),\ldots,ans(D_n,\varphi)$, as $F(\vec{D})$ minimises the distance for all $P \in \mathscr{D}$. For $\varphi = \psi \star \psi'$, where $\star$ is a Boolean operator, $(F(\vec{D}),\vec{u}) \models \varphi$ iff $(F(\vec{D}),\vec{u}_1) \models \psi \mathbin{\hat{\star}} (F(\vec{D}),\vec{u}_2) \models \psi'$, where $\hat{\star}$ is the interpretation of $\star$ and $\vec{u}_1$, $\vec{u}_2$ are suitable subsequences of $\vec{u}$. By induction hypothesis, then $\vec{u}_1$ and $\vec{u}_2$ minimise the distances in the answers to queries ψ and ψ' respectively, then $\vec{u}$ does so for φ, and therefore $\vec{u}$ belongs to the average of $ans(D_1,\varphi),\ldots,ans(D_n,\varphi)$. Finally, universal (resp. existential) quantification is dealt with by considering it as a finite conjunction (resp. disjunction). □

To conclude this section we discuss the results obtain so far, which can be seen as a first contribution on the relationship between database aggragation and query answering. In particular, Lemma 13 can be seen as a (partial) answer to Question 11. Similarly, Lemma 14 and 15 are related to Question 12. However, all the applicability of these results is restricted by the limited expressivity of the query languages. On the other hand, Lemma 16 and 17 show that merge with incomplete information and the average voter rule preserve (union of) conjuctive queries and the whole of first-order logic respectively. Results along these lines may find application in efficient query answering: it might be that in selected cases, rather than querying the aggregated database $F(\vec{D})$, it is more efficient to query the individual instances $D_1, \ldots, D_n$ and then aggregate the answers. Then, it is crucial to know which answers are preserved by the different aggregation procedures. The results provided in this section aimed to be a first, preliminary step in this direction.

7 Conclusions and Related Work

In this paper we have proposed a framework for the aggregation of conflicting information coming from multiple sources in the form of finite relational databases. We proposed a number of aggregators inspired by the literature on social choice theory, and adapted a number of axiomatic properties. We then focused on two natural questions which arise when dealing with the aggregation of databases. First, in Section 5 we studied what kind of integrity constraints are lifted by some of the rules we proposed, i.e., what constraints are true in the aggregated database supposing that all individual input satisfies the same constraints. Second, in Section 6 we investigated first-order query answering in the aggregated databases, characterising some languages for which the aggregation of the answers in the individual databases correspond to the answer to the query on the aggregated database.

Our initial results shed light on the possible use of choice-theoretic techniques in database merging and integration, and opens multiple interesting directions for future research. In particular, the connections to the literature on aggregation and merging can be investigated further. Firstly, Section 5 showcased results for which database aggregation behaves similarly to binary aggregation with integrity constraints (see [22]), but pointed at some crucial differences. In particular, there are natural classes of integrity constraints used in databases for which the equivalent in propositional logic, the language of choice for binary aggregation, would be tedious and lenghty. We were able to provide initial results on their preservation through aggregation. Secondly, the recent work of [18] is also strongly related to our contribution. Since graphs are a specific type of relational structures, our work directly generalise their graph aggregation framework to relations of arbitrary arity. However, the specificity of their setting allows them to obtain very powerful impossibility results, which are yet to be explored in the area of database aggregation. Thirdly, to the best of our knowledge the problem of aggregated query answering is new in the literature on aggregation, albeit a similar problem has been studied in the aggregation of argumentation graphs [11]. Also this direction deserves further investigation.

Acknowledgements. F. Belardinelli acknowledges the support of the ANR JCJC Project SVeDaS (ANR-16-CE40-0021), and U. Grandi the support of the ANR JCJC project SCONE (ANR 18-CE23-0009-01).

References

[1] S. Abiteboul, R. Hull & V. Vianu (1995): *Foundations of Databases*. Addison-Wesley.

[2] O. Arieli, M. Denecker & M. Bruynooghe (2007): *Distance semantics for database repair*. Annals of Mathematics and Artificial Intelligence 50(3), pp. 389–415, doi:10.1007/s10472-007-9074-1.

[3] K. J. Arrow (1963): *Social Choice and Individual Values*, 2nd edition. John Wiley and Sons.

[4] C. Baral, S. Kraus & J. Minker (1991): *Combining Multiple Knowledge Bases*. IEEE Transactions on Knowledge and Data Engineering 3(2), pp. 208–220, doi:10.1109/69.88001.

[5] C. Baral, S. Kraus, J. Minker & V. S. Subrahmanian (1992): *Combining knowledge bases consisting of first-order theories*. Computational Intelligence 8(1), doi:10.1016/0004-3702(80)90014-4.

[6] F. Belardinelli & U. Grandi (2019): *A Social Choice Theoretic Perspective on Database Aggregation (Extended Abstract)*. In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[7] H. A. Blair & V. S. Subrahmanian (1989): *Paraconsistent Logic Programming*. *Theoretical Computer Science* 68(2), pp. 135–154, doi:10.1016/0304-3975(89)90126-6.

[8] R. Booth, E. Awad & I. Rahwan (2014): *Interval Methods for Judgment Aggregation in Argumentation*. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

[9] M. Caminada & G. Pigozzi (2011): *On judgment aggregation in abstract argumentation*. *Autonomous Agents and Multiagent Systems* 22(1), pp. 64–102, doi:10.1007/s10458-009-9116-7.

[10] Ashok K. Chandra & Philip M. Merlin (1977): *Optimal Implementation of Conjunctive Queries in Relational Data Bases*. In: *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, ACM, New York, NY, USA, pp. 77–90, doi:10.1145/800105.803397.

[11] W. Chen & U. Endriss (2017): *Preservation of Semantic Properties during the Aggregation of Abstract Argumentation Frameworks*. In: *Proceedings of the 16th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, doi:10.1007/978-3-540-77684-0_4.

[12] F. Dietrich & C. List (2007): *Arrow's Theorem in Judgment Aggregation*. *Social Choice and Welfare* 29(1), pp. 19–33, doi:10.1007/s00355-006-0196-x.

[13] F. Dietrich & C. List (2007): *Judgment Aggregation by Quota Rules: Majority Voting Generalized*. *Journal of Theoretical Politics* 19(4), pp. 391–424, doi:10.1016/0022-0531(75)90062-9.

[14] E. Dokow & R. Holzman (2010): *Aggregation of binary evaluations*. *Journal of Economic Theory* 145(2), pp. 495–511, doi:10.1016/j.jet.2007.10.004.

[15] J. Doyle & M. P. Wellman (1991): *Impediments to Universal Preference-based Default Theories*. *Artificial Intelligence* 49(1), pp. 97–128, doi:10.1016/0004-3702(91)90007-7.

[16] U. Endriss (2016): *Judgment Aggregation*. In F. Brandt, V. Conitzer, U. Endriss, J. Lang & A. D. Procaccia, editors: *Handbook of Computational Social Choice*, Cambridge University Press, doi:10.1017/CBO9781107446984.018.

[17] U. Endriss & U. Grandi (2014): *Binary Aggregation by Selection of the Most Representative Voter*. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-2014)*.

[18] U. Endriss & U. Grandi (2017): *Graph Aggregation*. *Artificial Intelligence* 245, pp. 86–114, doi:10.1016/j.artint.2017.01.001.

[19] P. Everaere, S. Konieczny & P. Marquis (2015): *Belief Merging versus Judgment Aggregation*. In: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[20] A. Gionis, H. Mannila & P. Tsaparas (2007): *Clustering aggregation*. *ACM Transactions on Knowledge Discovery from Data* 1(1), p. 4, doi:10.1145/1217299.1217303.

[21] U. Grandi & U. Endriss (2011): *Binary Aggregation with Integrity Constraints*. In: *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*.

[22] U. Grandi & U. Endriss (2013): *Lifting Integrity Constraints in Binary Aggregation*. *Artificial Intelligence* 199–200, pp. 45–66, doi:10.1016/j.artint.2013.05.001.

[23] E. Gregoire & S. Konieczny (2006): *Logic-based approaches to information fusion*. *Information Fusion* 7(1), pp. 4 – 18, doi:10.1016/j.inffus.2005.08.001. Logic-based Approaches to Information Fusion.

[24] F. van Harmelen, V. Lifschitz & B. Porter (2007): *Handbook of Knowledge Representation*. Elsevier Science, San Diego, USA.

[25] J. Hintikka (1962): *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press.

[26] B. Kimelfeld, P. Kolaitis & J. Stoyanovich (2018): *Computational Social Choice Meets Databases*. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, doi:10.24963/ijcai.2018/44.

[27] S. Konieczny (2000): *On the Difference between Merging Knowledge Bases and Combining them*. In: *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

[28] S Konieczny, J Lang & P Marquis (2004): *DA2 merging operators*. *Artificial Intelligence* 157(1), pp. 49 – 79.

[29] S. Konieczny & R. Pino Pérez (2002): *Merging Information Under Constraints: A Logical Framework*. *Journal of Logic and Computation* 12(5), pp. 773–808, doi:10.1093/logcom/12.5.773.

[30] Sébastien Konieczny, Jérôme Lang & Pierre Marquis (2002): *Distance Based Merging: A General Framework and some Complexity Results*. In: *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR)*.

[31] P. Liberatore & M. Schaerf (1998): *Arbitration (or How to Merge Knowledge Bases)*. *IEEE Transactions on Knowledge and Data Engineering* 10(1), pp. 76–90, doi:10.1109/69.667090.

[32] L. Libkin (2015): *How to Define Certain Answers*. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*.

[33] Jinxin Lin & Alberto O. Mendelzon (1996): *Merging Databases under Constraints*. *International Journal of Cooperative Information Systems* 7, pp. 55–76, doi:10.1142/S021821579200009X.

[34] D. Maier, J. Ullman & M. Vardi (1984): *On the Foundations of the Universal Relation Model*. *ACM Transactions on Database Systems* 9(2), pp. 283–308, doi:10.1145/329.318580.

[35] P. Maynard-Zhang & D. J. Lehmann (2003): *Representing and Aggregating Conflicting Beliefs*. *Journal of Artificial Intelligence Research* 19, pp. 155–203, doi:10.1613/jair.1206.

[36] M. Miller & D. Osherson (2009): *Methods for distance-based judgment aggregation*. *Social Choice and Welfare* 32(4), pp. 575–601, doi:10.1007/s00355-008-0340-x.

[37] D. Porello & U. Endriss (2012): *Ontology merging as social choice: Judgment aggregation under the open world assumption*. *Journal of Logic and Computation* 24, pp. 1229–1249, doi:10.1093/logcom/exs056.

[38] V. S. Subrahmanian (1992): *Paraconsistent Disjunctive Deductive Databases*. *Theoretical Computer Science* 93(1), pp. 115–141, doi:10.1016/0304-3975(92)90214-Z.