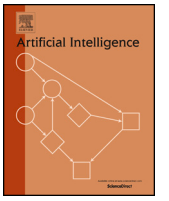
Artificial Intelligence 335 (2024) 104197

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



On measuring inconsistency in graph databases with regular path constraints

John Grant [a], Francesco Parisi [b,*]

[a] Department of Computer Science and UMIACS, University of Maryland, College Park, MD, USA
[b] Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria, Italy

ARTICLE INFO

Keywords:
Inconsistency measures
Graph databases
Computational complexity

ABSTRACT

Real-world data are often inconsistent. Although a substantial amount of research has been done on measuring inconsistency, this research concentrated on knowledge bases formalized in propositional logic. Recently, inconsistency measures have been introduced for relational databases. However, nowadays, real-world information is always more frequently represented by graph-based structures which offer a more intuitive conceptualization than relational ones. In this paper, we explore inconsistency measures for graph databases with regular path constraints, a class of integrity constraints based on a well-known navigational language for graph data. In this context, we define several inconsistency measures dealing with specific elements contributing to inconsistency in graph databases. We also define some rationality postulates that are desirable properties for an inconsistency measure for graph databases. We analyze the compliance of each measure with each postulate and find various degrees of satisfaction; in fact, one of the measures satisfies all the postulates. Finally, we investigate the data and combined complexity of the calculation of all the measures as well as the complexity of deciding whether a measure is lower than, equal to, or greater than a given threshold. It turns out that for a majority of the measures these problems are tractable, while for the other different levels of intractability are exhibited.

1. Introduction

Graph databases, and more generally data and knowledge graphs, have gained increasing attention in recent years as a promising formalism to integrate and exploit data and knowledge from diverse sources on a large scale [1–3]. Data models such as graph databases have become popular as they allow for the effective representation and access to data mainly consisting of entities from the domain of interest (represented by nodes), and relationships between them (represented by edges). The importance of graph data models stems from the fact that there are a variety of domains (such as social networks, transport networks, biological pathways, citation networks, and kinship networks) for which graph-based representations offer a more intuitive conceptualization than relational ones. Therefore, graph databases are preferred over relational databases in several applications where the topology of the data is as important as the data itself, as for instance in the above-mentioned domains [4,1,2].

However, data are mostly obtained from large-scale data extraction pipelines, which are notoriously brittle and can introduce errors and inconsistencies in these graphs [5–7], analogously to what happens for traditional relational data [8]. This has lead to an extensive body of work on handling inconsistent data. For instance, approaches for dealing with inconsistent relational databases include

* Corresponding author.
E-mail addresses: grant@cs.umd.edu (J. Grant), fparisi@dimes.unical.it (F. Parisi).

https://doi.org/10.1016/j.artint.2024.104197
Received 5 March 2024; Received in revised form 25 July 2024; Accepted 29 July 2024

Available online 2 August 2024

consistent query answering frameworks [9,10], inconsistency management policies [11], interactive data repairing and cleaning systems [12–14], as well as interactive data exploration tools [15,16]. In general, handling conflicting information in an important challenge in AI and has a long research history in the field of relational data. Moreover, given the increasing widespread of graph data, witnessed by an unprecedented growth of interconnected data [17], several issues concerning data quality have become of fundamental importance for graph data as well [18–20].

An important drawback of relying on data of poor quality is that it can significantly limit the implementation of effective AI solutions [8,21], such as in typical statistical relational learning tasks (e.g., link prediction, community search, community and anomaly detection) where data are in the form of a graph consisting of nodes (entities) and labeled edges (relationships between entities) [22]. So, having information on the quality of data used in machine learning and data-driven approaches is crucial, as poor quality data can have serious adverse consequences on the quality of decisions made using AI [23]. *Measuring inconsistency* [24,25] is a well-understood approach that can be used towards assessing data quality, as it provides ways to quantify the severity of the inconsistency and thereby help in understanding the primary sources of conflicts and devising ways to deal with them. In this regard, inconsistency measurement has been extensively investigated for propositional knowledge bases (e.g., [26–31], to mention a few) ever since the idea of measuring inconsistency was introduced more than 40 years ago in [24]. Moreover, more recently, it has been explored in several other settings such as software specifications [32], relational databases [33–35], and ontologies [36,37], among others.

However, despite data quality being an important issue in graph data, to the best of our knowledge, the problem of *measuring inconsistency in graph databases* has not been addressed so far. Here it is worthwhile to mention that there is a paper that uses bigraphs to study inconsistency measures [38]. However, that paper studied inconsistency measures on propositional knowledge bases; the bigraphs were used to represent the minimal inconsistent subsets. In this paper, we explore inconsistency measures for graph databases (GDBs) consisting of edge-labeled directed graphs [39–41,4], a simple yet powerful graph data model that is widely adopted in practice where, for example, it forms the basis of the Resource Description Framework (RDF) standard used for encoding machine-readable content on the Web [4,2]. Many online social networks can be viewed as edge-labeled directed graphs. For instance, users of a social network may correspond to vertices and an edge from user u to user v may exist with a friend label if u friended v (and v accepted), a $\mathsf{pfriend}$ label if u friended v but was not accepted, and a host of labels $\mathsf{endorsed\text{-}t}$ if u endorsed v w.r.t. topic t and $\mathsf{not\_endorsed\text{-}t}$ if u was given the option of endorsing v for topic t but did not do so. Likewise, other kinds of relationships between persons may be considered, such as kinship ones, e.g. $\mathsf{child\_of}$, $\mathsf{sister\_of}$, and so on. Moreover, a social network may also consider companies and employers to be vertices, universities and schools to be vertices, and place edges between persons and such vertices labeled with relationships like $\mathsf{employs}$ or $\mathsf{alumnus\_of}$ with the obvious meaning [42]. It is worth mentioning that other graph data models, such as property graphs [43], can be translated to/from edge-labeled directed graphs without loss of information [44,2]. In fact, although edge-labeled directed graphs have a simple structure, they can encode complex information.

As in the relational case, the notion of consistency in GDBs is related to the fact that it is expected that data comply with a set of integrity constraints expressing some semantic properties of the represented world. These properties can be formally expressed by relying on a query language. In particular, GDBs are commonly queried through navigational languages, such as *regular path queries* (RPQs) that can capture pairs of nodes connected by paths whose labels satisfy some regular expression [40,45,46]. For instance, the RPQ $\mathsf{endorsed\text{-}GDB}^+$ specifies all paths consisting of a sequence of one-or-more directed edges whose label is $\mathsf{endorsed\text{-}GDB}$ (each representing the fact that a user endorsed another one w.r.t. the topic GDB), expressing the transitive $\mathsf{endorsed\text{-}GDB}$ relationship. The result of evaluating such a query over a GDB is the set of pairs of nodes, corresponding to pairs of users, such that the former transitively endorse the latter w.r.t. the considered topic. The need for RPQs in graph databases has been long discussed [47] and they have been implemented in various systems. For instance, RPQs form the conceptual core of *property paths* in the SPARQL standard [48], which have been implemented in several SPARQL engines. Likewise, in the Cypher query language [49], one can find RPQ-like features [4].

A well-known class of integrity constraints that are based on RPQs is that of *regular path constraints* (RPCs) [50,51]. Intuitively, an RPC imposes the constraint that the evaluation of an RPQ is contained in the evaluation of another RPQ. As an example, the RPC $\mathsf{child\_of} \subseteq \mathsf{son\_of} \mid \mathsf{daughter\_of}$, intuitively states that every child is a son or a daughter (we formally introduce the semantics of RPQs and RPCs in Section 2).

1.1. Contributions

In this paper, we investigate inconsistency measures for GDBs with RPCs and, in particular, analyze rationality postulates compliance as well as the computational complexity of several inconsistency measures that we conceive for the considered setting. Our main contributions can be summarized as follows.

- We introduce inconsistency measures (IMs for short) for GDBs with RPCs. These measures are loosely inspired by IMs for propositional logic and relational databases. However, given the different nature of the data model, we cannot apply well-known methods for propositional logic or for relational databases; rather, we use the principles behind those methods as inspiration to define new IMs that apply to graph databases. In particular, in Section 3 we formally introduce eleven IMs, namely those whose name is reported in the second row of Table 1 (or, equivalently, in the first column of Table 2; both tables are discussed below). In general, the inconsistency measures measure the inconsistency by blaming elements in the GDB (i.e., nodes, labels, and edges) from which conflicts originate due to not complying with the integrity constraints. However, we also define a limiting measure (also used for propositional logic) that simply distinguishes between consistent and inconsistent databases and another measure that measures the number of constraints that are violated. This is different from measuring the inconsistency of a set of formulas,

Table 1
Status of postulates for graph database inconsistency measures. ✓ means satisfied, while ✗ means not satisfied.

	Inconsistency measure										
Postulate	I_B	I_C	I_M	I_P	I_S	I_E	I_L	I_V	I_{E-}	I_{E+}	I_{V-}
Monotony	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Edge Independence	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Vertex Independence	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Edge Penalty	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗
Vertex Penalty	✗	✗	✓	✗	✓	✓	✗	✓	✗	✗	✗
Edge Super-Additivity	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗	✗
Vertex Super-Additivity	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓
MIMS-Separability	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗
MIMS-Normalization	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓

Table 2
Complexity of Lower Value, Upper Value, Exact Value, and Inconsistency Measurement problems under data and combined complexity. For a complexity class C, C-c means C-complete, while C means membership in C.

	Lower Value		Upper Value		Exact Value		Inconsistency Measurement	
	FLV	**LV**	**FUV**	**UV**	**FEV**	**EV**	**FIM**	**IM**
I_B	NL	P	NL	P	NL	P	FNL	FP
I_C	NL	P	NL	P	NL	P	FNL	FP
I_P	NL	P	NL	P	NL	P	FNL	FP
I_E	NL	P	NL	P	NL	P	FNL	FP
I_L	NL	P	NL	P	NL	P	FNL	FP
I_V	NL	P	NL	P	NL	P	FNL	FP
I_{E-}	$coNP$-c	$coNP$-c	NP-c	NP-c	D^p-c	D^p-c	$FP^{NP[\log n]}$-c	$FP^{NP[\log n]}$-c
I_{V-}	$coNP$-c	$coNP$-c	NP-c	NP-c	D^p-c	D^p-c	$FP^{NP[\log n]}$-c	$FP^{NP[\log n]}$-c
I_{E+}	$coNP$-c	$coNP$-c	NP-c	NP-c	D^p-c	D^p-c	$FP^{NP[\log n]}$-c	$FP^{NP[\log n]}$-c
I_S	CNL	PP	CNL	PP	CNL	PP	$\#P$-c	$\#P$-c
I_M	CNL	PP	CNL	PP	CNL	PP	$\#P$-c	$\#P$-c

all of which have the same status, as is the case of propositional knowledge bases. This is also different from the case of relational databases, where the structure of data is fixed by the database schema over which integrity constraints are defined.

- Rationality postulates are properties defined for propositional inconsistency measures in order to describe general desirable behaviors of the measures, though it is not universally agreed upon which postulates should be satisfied by an inconsistency measure to be considered as a "good" one [52]. Still, analyzing the compliance of inconsistency measures w.r.t. postulates is an important way to evaluate a measure, and for this reason, it is often investigated in the evaluation of inconsistency measures [26, 53]. Given the importance of postulate satisfaction, we introduce the GDB counterpart of some rationality postulates defined for propositional knowledge bases as well as for relational databases and check for compliance by the GDB IMs. Table 1 summarizes the results obtained in Section 4, and allows us to compare the IMs in terms of the satisfied postulates. For instance, while the first three postulates are satisfied by all the considered IMs, the other highlight that IMs behave differently in the different settings considered by the postulates. However, it is important to note that we do not claim that all postulates should be satisfied by all measures. We use the postulates to understand the behavior of the considered measures from different perspectives which are encoded by the individual postulates, though one may think that, in general, an IM behaves more in line with our intuition if it satisfies more postulates.
- Finally, in Section 5, we investigate the computational complexity of the problems of deciding whether a given value is lower than (**LV**), greater than (**UV**), or equal to (**EV**) the inconsistency measured using a given inconsistency measure for GDBs. We investigate both the combined and the data complexity [54,55] of these problems. While under combined complexity the set of constraints is part of the input, it is considered fixed (i.e., not impacting on the size of the input) under data complexity analysis. We use **FLV**, **FUV**, **FEV** to refer to the variant of the above-mentioned problems where the set of constraints is fixed. A summary of the results obtained for the above-mentioned decision problems, as well as for the problem of computing the actual value of an inconsistency measure (**IM** and **FIM** problems), is reported in Table 2. Six of the eleven measures we consider are tractable and among the other five there are differences exhibiting various levels of intractability expressed by different complexity classes [56–59].

1.2. Organization

After recalling some basic notions for graph databases and regular path constraints in Section 2.1, we briefly review the complexity classes used in the paper in Section 2.2. In Section 3, we first present the general concept of an inconsistency measure for graph

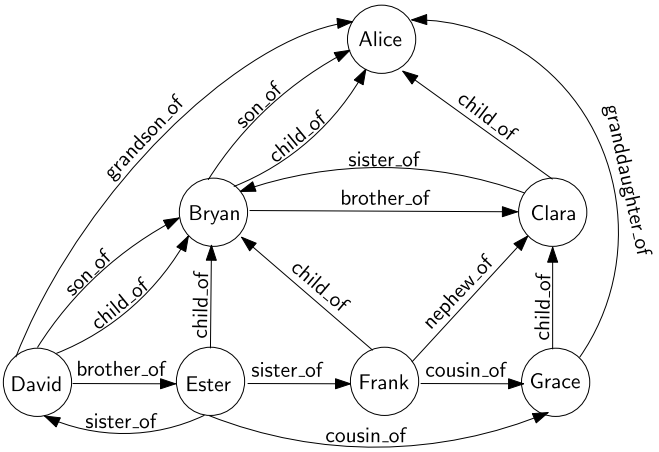


Fig. 1. Graph database G_{ex}.

databases, and then introduce our measures by considering several criteria for counting elements, e.g. vertices and edges, from which inconsistency arises; we illustrate the inconsistency measures introduced by a detailed example in Section 3.2. Then, we introduce our rationality postulates, discuss some relationship between them, and analyze the satisfaction of rationality postulates in Section 4. We start Section 5 by formally defining the decision problems **(F)LV**, **(F)UV**, and **(F)EV**, as well as the function problem **(F)IM**, and then analyze the complexity of these problems. After introducing some preliminary lemmas, we first present the results for tractable measures in Section 5.1, and then those for intractable ones in Section 5.2; the results are then discussed in Section 5.3. Related work is discussed in Section 6. Finally, in Section 7, the paper is concluded and future work is outlined.

2. Preliminaries

We first introduce our notation for graph databases and regular path constraints, and then review some basic notions about complexity.

2.1. Graph databases and regular path constraints

A graph database (GDB) is finite edge-labeled directed graph [39–41,4]. Let Σ be a finite alphabet (i.e., a set of symbols, also called *labels*). A graph database $G = (V, E)$ over Σ consists of *i*) a finite set V of vertices (or nodes), and *ii*) a set of labeled edges $E \subseteq V \times \Sigma \times V$. A tuple $(u, \ell, v) \in E$, where $u, v \in V$ and $\ell \in \Sigma$, denotes an edge from node u to node v whose label is ℓ. The labels indicate the different types of relationships between vertices; multiple labeled-edges between the same pairs of vertices are possible.

An example of graph database G_{ex} consisting of 7 vertices is shown in Fig. 1. Here, the set of edge labels is $\Sigma_{ex} = \{\mathsf{child\_of}, \mathsf{son\_of}, \mathsf{daughter\_of}, \mathsf{brother\_of}, \mathsf{sister\_of}, \mathsf{nephew\_of}, \mathsf{niece\_of}, \mathsf{cousin\_of}, \mathsf{grandson\_of}, \mathsf{granddaughter\_of}\}$.

The *size* of G is the number of edges plus the number of isolated vertices (i.e., those having neither incoming nor outgoing edges).[1] For two graph databases $G = (V, E)$ and $G' = (V', E')$ over alphabet Σ, we say that G is a subgraph of G' (denoted as $G \subseteq G'$) iff $V \subseteq V'$ and $E \subseteq E'$. Moreover, we write $G \subset G'$ if $G \subseteq G'$ and $G \neq G'$.

2.1.1. Regular path constraints

We focus on a class of integrity constraints for GDBs, called *regular path constraints* [50,51], that are based on a navigational language for GDBs, that is the language of *regular path queries*.

A navigational language for GDBs expresses properties of paths. Given GDB $G = (V, E)$, a path π in G from vertex v_0 to vertex v_n is a sequence of edges of the form: $\pi = (v_0, \ell_1, v_1)\,(v_1, \ell_2, v_2) \ldots (v_{n-1}, \ell_n, v_n)$, where $n \geq 0$ and for all $i \in [0..n-1]$, $(v_i, \ell_{i+1}, v_{i+1}) \in E$. The label of π, denoted $\lambda(\pi)$, is the string $\ell_1 \ell_2 \ldots \ell_n$ in Σ^*, where Σ^* is the set of all strings of finite length consisting of labels in Σ, including the empty string. Observe that $\lambda(\pi)$ is the empty string ϵ if $n = 0$, that is, if $\pi = (v_0, \epsilon, v_0)$ is the empty path.

A regular path query (RPQ) Q is a regular expression [60] defined over the set Σ of edge labels [40]. We recall that a regular expression Q defines a formal language as sets of strings over Σ. In particular, ϵ is a regular expression that denotes the set containing the empty string as its only element. If $\ell \in \Sigma$, then ℓ is a regular expression that denotes the set whose only element is label ℓ. If e and f are regular expressions denoting the sets of strings S_e and S_f, then *i*) $e|f$ is a regular expression denoting the set $S_e \cup S_f$; *ii*) $e.f$ is a regular expression denoting the set of all concatenations of s_1 and s_2 with $s_1 \in S_e$ and $s_2 \in S_f$; *iii*) e^* is a regular expression denoting closure of S_e, that is, the set of zero or more concatenations of strings from S_e. In our context, a regular expression specifies

[1] Clearly, nothing changes if we choose $|E| + |V|$ for the size of $G = (V, E)$.

all paths whose edge labels, when concatenated, form a string in the language of the regular expression. Intuitively speaking, regular expressions allow for concatenating paths, for applying a union/disjunction of paths, and for applying a path zero or many times.

The evaluation $Q(G)$ of an RPQ Q over a graph database $G = (V, E)$ is the set of pairs (u, v) of vertices in V for which there is a path π in G from u to v such that the label $\lambda(\pi)$ satisfies the regular expression Q, that is, $\lambda(\pi)$ is in the language defined by Q. If Q does not mention the Kleene-star (i.e., if Q defines a finite language) then Q is said to be non-recursive.

Example 1. *For the graph database G_{ex} shown in Fig. 1, RPQ $Q_1 = \mathsf{son\_of}.\mathsf{son\_of}^*$ specifies all paths formed from a sequence of one-or-more edges with the label* son_of*. Thus Q_1 expresses a transitive relationship in the graph of Fig. 1. The query could be equivalently rewritten as $Q_1 = \mathsf{son\_of}^+$, where ℓ^+ is usually used as shorthand for $\ell.\ell^*$. The evaluation of Q_1 over G_{ex} is the set of pairs of vertices of G_{ex} that are connected by such a path, that is, $Q_1(G_{ex}) = \{(\mathsf{David}, \mathsf{Bryan}), (\mathsf{Bryan}, \mathsf{Alice})\ (\mathsf{David}, \mathsf{Alice})\}$. As another example, for the RPQ $Q_2 = \mathsf{child\_of}.(\mathsf{brother\_of}|\mathsf{sister\_of})$, we have that $Q_2(G_{ex}) = \{(\mathsf{David}, \mathsf{Clara}), (\mathsf{Ester}, \mathsf{Clara}), (\mathsf{Frank}, \mathsf{Clara}), (\mathsf{Grace}, \mathsf{Bryan})\}$.*

Graph database constraints based on the class of RPQs are known as regular path constraints (RPCs) [50,51]. An RPC over Σ is an expression of the form $Q_1 \subseteq Q_2$, where Q_1 and Q_2 are RPQs over Σ. A *word constraint* is an RPC in which both Q_1 and Q_2 are words, i.e. simply sequences of labels, instead of arbitrary regular expressions.

An RPC $Q_1 \subseteq Q_2$ imposes the constraint that the evaluation of RPQ Q_1 is contained in the evaluation of RPQ Q_2. That is, a graph database G satisfies the RPC $Q_1 \subseteq Q_2$, denoted as $G \vDash Q_1 \subseteq Q_2$, if and only if $Q_1(G) \subseteq Q_2(G)$ (otherwise, we say that G does not satisfy or violates $Q_1 \subseteq Q_2$ and write $G \nvDash Q_1 \subseteq Q_2$).

Example 2. *For the GDB G_{ex} shown in Fig. 1, let Γ be the set of the following RPCs (inspired by [41]):*

C_1: $\mathsf{child\_of} \subseteq \mathsf{son\_of}|\mathsf{daughter\_of}$*, meaning that a child is a son or a daughter;*
C_2: $\mathsf{child\_of}.(\mathsf{brother\_of}|\mathsf{sister\_of}) \subseteq \mathsf{nephew\_of}|\mathsf{niece\_of}$*, meaning that a child of a brother or a sister is a niece or a nephew;*
C_3: $\mathsf{child\_of}.\mathsf{child\_of} \subseteq \mathsf{grandson\_of}|\mathsf{granddaughter\_of}$*, meaning that a child of a child is a grandson or a granddaughter;*
C_4: $\mathsf{son\_of}.\mathsf{child\_of} \subseteq \mathsf{grandson\_of}$*, meaning that a son of a child of a person is a grandson of that person.*

Considering the first constraint, C_1, there are four cases where an edge with label child_of *does not also have the label* son_of *or* daughter_of*, that is, the pairs:* (Frank, Bryan)*,* (Ester, Bryan)*,* (Grace, Clara)*, and* (Clara, Alice)*. Therefore, G_{ex} violates C_1, and we write $G_{ex} \nvDash C_1$. On the other hand, it can be easily checked that the fourth constraint is satisfied, that is $G_{ex} \vDash C_4$.*

As for the second constraint, the RPQ on the left-hand side of C_2 specifies paths consisting of two edges: the first one labeled with child_of*, and the second one labeled with* brother_of *or* sister_of*. Hence, there are three paths in G_{ex} that violate C_2:*

- *the path $\pi_1 =$* (David, child_of, Bryan) (Bryan, brother_of, Clara)*;*
- *the path $\pi_2 =$* (Ester, child_of, Bryan) (Bryan, brother_of, Clara)*;*
- *and the path $\pi_3 =$* (Grace, child_of, Clara) (Clara, sister_of, Bryan)*.*

In fact, all need an edge with label nephew_of *or* niece_of *from the first to the third vertex. That is, G_{ex} violates C_2 since the pairs* (David, Clara)*,* (Ester, Clara)*, and* (Grace, Bryan) *belong to the answer of the RPQ on the left-hand side of C_2 but not to the answer of that of its right-hand side.*

As for the third constraint, there are two paths that violate C_3:

- *the path $\pi_4 =$* (Ester, child_of, Bryan)(Bryan, child_of, Alice)*;*
- *and the path $\pi_5 =$* (Frank, child_of, Bryan)(Bryan, child_of, Alice)*.*

Both need an edge with label grandson_of *or* granddaughter_of*. Therefore, G_{ex} violates C_3 because of the pairs* (Ester, Alice) *and* (Frank, Alice) *that belong to the answer of the RPQ on the left-hand side but not to the answer of the right-hand side of C_3.*

Given a (finite) set Γ of RPCs, we use $G \vDash \Gamma$ to denote the fact that for each RPC $Q_1 \subseteq Q_2$ in Γ, $G \vDash Q_1 \subseteq Q_2$. A GDB G is said to be *consistent* w.r.t. Γ iff $G \vDash C$; otherwise, G is said to be *inconsistent* (w.r.t. Γ). In our running example, G_{ex} is inconsistent w.r.t. $\Gamma = \{C_1, C_2, C_3, C_4\}$; in such a case, we write $G_{ex} \nvDash \Gamma$.

Except for the examples where Γ is given explicitly, we assume that Γ is a given set of RPCs for Σ. We use $||\Gamma||$ to denote the size of Γ, that is the sum of the sizes of the RPQs occurring in the constraints in Γ, $||\Gamma|| = \sum_{Q_1 \subseteq Q_2 \in \Gamma} ||Q_1|| + ||Q_2||$, where the size $||Q||$ of an RPQ Q is the length of the regular expression (number of symbols).

2.2. Complexity classes

We briefly review the complexity classes used to characterize the complexity of decision and function problems considered in this paper; we refer the reader to [56] for further details on complexity classes.

P (resp., NP) is the class of the decision problems that can be solved by a deterministic (resp., non-deterministic) Turing machine in polynomial time w.r.t. the size of the input of the problem. $coNP$ is the class of the decision problems whose complement is in NP. The classes Σ_k^p, Π_k^p and Δ_k^p, with $k \geq 0$, are defined as follows:

- $\Sigma_0^p = \Pi_0^p = \Delta_0^p = P$;
- $\Sigma_1^p = NP$ and $\Pi_1^p = coNP$;
- $\Delta_k^p = P^{\Sigma_{k-1}^p}$, $\Sigma_k^p = NP^{\Sigma_{k-1}^p}$, and $\Pi_k^p = co\Sigma_k^p$, $\forall k > 0$.

Thus, P^C (resp., NP^C) denotes the class of problems that can be solved in polynomial time using an oracle in the class C by a deterministic (resp., non-deterministic) Turing machine. Under the standard complexity-theoretic assumptions, we have that $\Sigma_k^p \subseteq \Delta_{k+1}^p \subseteq \Sigma_{k+1}^p \subseteq PSPACE$ and $\Pi_k^p \subseteq \Delta_{k+1}^p \subseteq \Pi_{k+1}^p \subseteq PSPACE$.

A decision problem is in D_k^p iff it is the conjunction of a decision problem in Σ_k^p and a decision problem in Π_k^p. Hence, D_1^p (or simply D^p) denotes the class of the problems that are a conjunction of a problem in NP and one in $coNP$. It holds that $D^p \subseteq \Delta_2^p = P^{NP}$.

NL is the class of the decision problems that can be solved by a non-deterministic Turing machine using a logarithmic amount of memory space, that is, NL can be defined as $NSPACE(\log n)$. It is well-known that $NL \subseteq P$, and that $NL = coNL$, where $coNL$ is the class of problems whose complements are in NL. The class NL is closed under the operations complementation, union, and therefore intersection, concatenation, and Kleene star.

We will also use the class CP [61] from the counting polynomial hierarchy defined in [57]. These classes rely on a counting quantifier C defined as follows. Given a predicate $H(x, y)$ with free variables x and y, $C_y^k H(x, y)$ holds iff $|\{y : H(x, y) \text{ is true}\}| \geq k$, i.e., the counting quantifier is true for predicate H and bound k iff the number of values of y such that $H(x, y)$ holds is at least k. We will use the polynomially bounded version of the counting quantifier which is defined as follows. Given a class C of decision problems, we say that a problem A is in CC iff there is a problem $B \in C$, a polynomial-time computable function f, and a polynomial p such that x is a positive instance of A iff $C_{y,|y|\leq p(x)}^{f(x)}(x, y) \in B$. That is, instance $x \in A$ iff there are at least $f(x)$ many y's whose size is polynomially bounded by that of x such that a predicate for (x, y) holds, with checking the predicate being in B.

A canonical problem for CP is deciding whether a SAT formula has at least k many satisfying assignments. The class CP coincides with the class PP of the decision problems that can be solved in polynomial time by a probabilistic Turing machine [61,57]. In particular, PP can be also defined by using ordinary non-deterministic Turing machines by altering the way a machine accepts its input. The class PP consists of the languages for which there is a non-deterministic Turing machine such that an input is in the language if and only if more than half of the computations on that input end up accepting (in polynomial time). It is worth mentioning that the polynomial hierarchy $PH = \bigcup_{i\geq 0} \Sigma_i^p$ is contained in the class P^{PP} [62]. The relationships between the classes NP, CP and $coNP$ are as follows: $NP \subseteq CP$ and $coNP \subseteq CP$. Differently from NP that is closed under union and intersection and is not known to be closed under complement, the class CP is closed under complement. Like NP, CP is closed under union and intersection, though this question remained open for several years [63].

We now recall some complexity classes for function problems. FP (resp., FNP) is the class of the function problems that can be solved by a deterministic (resp., non-deterministic) Turing machine in polynomial time. Similarly, FNL is the class of the function problems that can be solved by a non-deterministic Turing machine in logarithmic space. $\#P$ is the complexity class of the functions f such that f counts the number of accepting paths of a non-deterministic polynomial-time Turing machine [59]. For a complexity class C, FP^C is the class of functions computable by a deterministic polynomial-time Turing machine using a C-oracle. Thus, FP^{NP} is the class of problems that can be solved by a polynomial-time Turing machine that can ask a *polynomial* number of queries to an NP oracle. If a logarithmic number of queries is asked by the machine, then we have the class $FP^{NP[\log n]}$. $FP^{\#P}$ is the class of functions computable by a polynomial-time Turing machine with a $\#P$ oracle. For each complexity class $\#C \in \#PH$ it holds that $FP^{\#P} = FP^{\#C}$, since $\#PH \subseteq FP^{\#P[1]}$ [64] and $FP^{FP^{\#P[1]}} \subseteq FP^{\#P}$.

3. Inconsistency measures

In this section, we first define the concept of an inconsistency measure for graph databases and then present the ones we will use later.

3.1. Inconsistency measures for graph databases

For propositional logic knowledge bases and relational databases some classifications have been defined for inconsistency measures. One classification distinguishes between *absolute* and *relative* measures. An absolute inconsistency measure assigns a value that indicates the totality of the inconsistency. An example of such a measure for propositional logic is the number of minimal inconsistent sets of formulas. Another example is the number of distinct formulas that occur in some minimal inconsistent subset. The corresponding examples for relational databases replace 'formulas' by 'elements' ('tuples' in the case of definite relational databases). This is in contrast to a relative inconsistency measure whose value indicates in some way the proportion of the knowledge base or database that is inconsistent. In this paper, we will deal with absolute inconsistency measures for graph databases as well as the special case of the drastic inconsistency measure that simply distinguishes the consistent case from the inconsistent case.

We start with some definitions. Let $\mathcal{G}$ be the set of all finite graph databases. We fix an alphabet Σ and a set Γ of RPCs over Σ and write $\mathcal{G}_\Sigma^\Gamma$ for the set of all such finite graph databases over Σ. Recall that $G \in \mathcal{G}_\Sigma^\Gamma$ is consistent iff $G \models \Gamma$.

Definition 1 *(Inconsistency Measure). A function $I : \mathcal{G}_\Sigma^\Gamma \to \mathbb{R}_\infty^{\geq 0}$ is an* inconsistency measure *iff the following condition (called* Consistency*) holds for all $G \in \mathcal{G}_\Sigma^\Gamma$: $I(G) = 0$ iff G is consistent.*

Thus, an inconsistency measure (IM) assigns to every graph database either a nonnegative number or infinity. This number must be 0 if and only if the graph database is consistent. Although in general infinity is admitted as an inconsistency value, none of the inconsistency measures that we will define takes the value ∞. We explain the reason for this after we define the concept of a problematic path.

In the following, we introduce some basic notions that are needed to define IMs and their properties in the context of graph databases.

Let Q be an RPQ over Σ. We call every pair (u, v) obtained in the evaluation of Q over G an *answer pair*. So $Q(G)$ is the set of answer pairs for Q in G. Recall that an answer pair is obtained from a path π that satisfies Q where u is the starting vertex and v is the ending vertex. We call such a path an *answer path* and write $\mathsf{Paths}(Q, G)$ for the set of all answer paths for Q. Note that there may be several answer paths that correspond to the same answer pair. For every answer path π we indicate the answer pair associated with π as $(u, v)_\pi$.

The inconsistency of a graph database is caused by the RPCs. Recall that every RPC in Γ has the form $Q_1 \subseteq Q_2$ for some RPQs over Σ. Let $(u, v) \in Q_1(G) \setminus Q_2(G)$ for some $Q_1 \subseteq Q_2 \in \Gamma$. We call such a pair *problematic* and write $\mathsf{ProblematicPairs}(G)$ for the set of problematic vertex pairs. All other vertex pairs are said to be *free*. If $\pi \in \mathsf{Path}(Q_1, G)$ such that $(u, v)_\pi \in \mathsf{ProblematicPairs}(G)$ then $\pi \in \mathsf{ProblematicPaths}(G)$. Again, a path that is not problematic is *free*. As explained earlier, each path π has the form $(v_0, \ell_1, v_1) \dots (v_{n-1}, \ell_n, v_n)$ with the corresponding answer pair (v_0, v_n).

Although the set V of vertices is finite, in the presence of recursion arbitrarily long paths may be problematic. In such a case the number of problematic paths is infinite. Infinity is allowed as the value of an inconsistency measure. The disadvantage of using a measure that counts the number of problematic paths is that it cannot distinguish between the case where infinity comes from a single recursion or multiple recursions or finitely many additional problematic paths to one or more recursions. Instead, we will use an inconsistency measure that counts the minimal problematic paths. This is a finite number even if there are infinitely many problematic paths. Let π_1 and π_2 be distinct paths. We write $\pi_1 \subset \pi_2$ iff π_1 is a proper subsequence of π_2. We write $\pi \in \mathsf{MinimalProblematicPaths}(G)$ iff $\pi \in \mathsf{ProblematicPaths}(G)$ and there is no $\pi' \in \mathsf{ProblematicPaths}(G)$ such that $\pi' \subset \pi$.

For a path $\pi = (v_0, \ell_1, v_1) \dots (v_{n-1}, \ell_n, v_n)$, we write $\mathsf{Vertices}(\pi) = \{v_0, \dots, v_n\}$, that is, the set of vertices along the path. We also write $|\pi| = n + 1$ indicating the number of vertices along the path. Furthermore, using e for an edge, we slightly abuse notation by writing $e \in \pi$ if e is one of the edges of π; moreover, we write $\ell \in \pi$ if ℓ is a label of an edge in π, and $v \in \pi$ if v is a vertex for an edge in π. This way we define additional problematic sets as follows:

- $\mathsf{ProblematicEdges}(G) = \{(v, \ell, v') \mid \exists \pi \in \mathsf{ProblematicPaths}(G)$ and $(v, \ell, v') \in \pi\}$,
- $\mathsf{ProblematicLabels}(G) = \{\ell \mid \exists \pi \in \mathsf{ProblematicPaths}(G)$ and $\ell \in \pi\}$,
- $\mathsf{ProblematicVertices}(G) = \{v \mid \exists \pi \in \mathsf{ProblematicPaths}(G)$ and $v \in \pi\}$.

Thus, the problematic edges (resp., labels, vertices) are the edges (resp., labels, vertices) that appear on some problematic path. The edges, labels, and vertices that are not problematic are called *free*. As the problematic edges, labels, and vertices are all defined using problematic paths, we obtain the following. A vertex v is free if and only if all edges that have v as an endpoint are free. Similarly, a label ℓ is free if and only if all edges whose label is ℓ are free. We also tighten the definition of a problematic edge and vertex as follows. An edge e (resp., vertex v) is called *minimal problematic* iff $\exists \pi \in \mathsf{MinimalProblematicPaths}(G)$ and $e \in \pi$ (resp., $v \in \pi$).

As the RPCs involve subsets, the situation for inconsistency measures is nonmonotonic. That is, the addition or deletion of information may decrease or increase an inconsistency measure.

Example 3. *Consider a GDB G over $\Sigma = \{\mathsf{parent\_of}, \mathsf{grandparent\_of}\}$ such that there is a single RPC in Γ:* $\mathsf{parent\_of}.\mathsf{parent\_of} \subseteq \mathsf{grandparent\_of}$*, meaning that a parent of a parent is a grandparent. Let $V = \{\mathsf{Ann}, \mathsf{Bob}, \mathsf{Carol}\}$, and $E = \{(\mathsf{Ann}, \mathsf{parent\_of}, \mathsf{Bob}), (\mathsf{Bob}, \mathsf{parent\_of}, \mathsf{Carol}), (\mathsf{Ann}, \mathsf{grandparent\_of}, \mathsf{Carol})\}$. The GDB $G = (V, E)$ is consistent (w.r.t. Γ) and so has inconsistency measure 0 for every inconsistency measure. But if the edge* $(\mathsf{Ann}, \mathsf{grandparent\_of}, \mathsf{Carol})$ *is deleted, the database becomes inconsistent and must have a positive measure for all inconsistency measures.*

Propositional logic and other logics for which inconsistency measures have been investigated are mostly monotonic, that is, the addition of information cannot decrease inconsistency, and the deletion of information cannot increase inconsistency for absolute inconsistency measures. There is one paper, [31], that studies measuring inconsistency in nonmonotonic logics in great detail. Another recent paper, [65], deals with measuring inconsistency in an environment that also includes nonmonotonicity. We use some of the ideas from [65] but the details for the GDBs with RPCs are substantially different.

We introduce monotonicity using the concept of a *monotonic subgraph*. The idea is that a monotonic subgraph does not have an inconsistency that is not also in the graph. Let G' be a subgraph of G. We say that G' is a *monotonic subgraph* of G iff $\mathsf{ProblematicPaths}(G') \subseteq \mathsf{ProblematicPaths}(G)$. We write $G' \subseteq_m G$ iff G' is a monotonic subgraph of G. So in Example 3, the subgraph G' with the edge $(\mathsf{Ann}, \mathsf{grandparent\_of}, \mathsf{Carol})$ deleted, is not a monotonic subgraph. It is clear from the definitions that $\subseteq_m$ is a transitive relation on subgraphs. We say that G' is a *minimal inconsistent monotonic subgraph* (MIMS) of G iff $G' \subseteq_m G$, G' is inconsistent, and there is no inconsistent G'' such that $G'' \subseteq_m G'$ and $G'' \neq G'$. It may be useful to think of the minimal inconsistent subgraphs as the counterpart of the minimal inconsistent subsets of a propositional knowledge base or a relational database, though the two concepts are different. We write $\mathsf{MIMS}(G)$ for the set of all MIMSs of G. Just as in the propositional case, the empty set is consistent (it satisfies

all RPCs for graph databases) and is not a MIMS. So now we have monotonic subgraphs instead of subsets. Also, we define the concept of graph union as follows: for $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

Now we are ready to define inconsistency measures. An inconsistency measure for graph databases requires two inputs: the graph database G and the set of integrity constraints Γ; that is, it is a function $I_x(G, \Gamma)$. Usually Γ is known, so we just write it as $I_x(G)$. In case where we consider different sets of constraints we indicate which set is being used.

Definition 2 *(GDB Inconsistency Measures). For any GDB G (and set Γ of RPCs), the inconsistency measures I_B, I_C, I_M, I_P, I_S, I_E, I_L, and I_V are as follows:*

- $I_B(G) = 1$ *iff G is not consistent (0 otherwise).*
- $I_C(G) = |\{C \mid C \in \Gamma \text{ and } G \not\models C\}|$.
- $I_M(G) = |\mathsf{MIMS}(G)|$.
- $I_P(G) = |\mathsf{ProblematicPairs(G)}|$.
- $I_S(G) = |\mathsf{MinimalProblematicPaths(G)}|$.
- $I_E(G) = |\mathsf{ProblematicEdges(G)}|$.
- $I_L(G) = |\mathsf{ProblematicLabels(G)}|$.
- $I_V(G) = |\mathsf{ProblematicVertices(G)}|$.

In Definition 2, we start by defining the drastic measure I_B, which simply differentiates between consistent and inconsistent graph databases. Then, we define one inconsistency measure, I_C, that considers only the violated constraints and counts them. The next inconsistency measure, I_M, counts the number of minimal inconsistent monotonic subgraphs thinking of each such subgraph as representing one inconsistency. Finally, we define inconsistency measures that deal with problematic elements: I_P counts the number of problematic pairs; I_S counts the number of minimal problematic paths; I_E counts the number of problematic edges; I_L counts the number of problematic labels; and I_V counts the number of problematic vertices.

The following result provides a relationship between I_M and I_S. It will be useful to define a function s, the subgraph constructor function, that maps a path in a graph to a subgraph as follows. Let $\pi = (v_1, \ell_1, v_2) \dots (v_{n-1} \ell_{n-1}, v_n)$ be a path in G. Define $s(\pi) = G_\pi = (V_\pi, E_\pi)$ by $V_\pi = \{v_1, \dots, v_n\}$ and $E_\pi = \{(v_1, \ell_1, v_2), \dots, (v_{n-1}, \ell_{n-1}, v_n)\}$. That is, G_π is the subgraph of G that contains exactly the vertices and edges of the path.

Proposition 1. *For every graph database G, $I_M(G) \leq I_S(G)$.*

Proof. If G is consistent then $I_M(G) = I_S(G) = 0$. Assume that G is inconsistent.

Let $G' \in \mathsf{MIMS}(G)$. We now show that there is a minimal problematic path $\pi \in G$ such that $s(\pi) = G'$. By the definition of MIMS there must be at least one problematic path and hence at least one minimal problematic path in G', say π'. By the monotonicity of G', π' is also a problematic path in G and by the minimality of G', π' is also a minimal problematic path in G. Let $G'' = s(\pi')$. Then $G'' \subseteq_m G' \subseteq_m G$. By the minimality of G', $G'' = G'$. So for each MIMS of G there is at least one minimal problematic path in G that is not in a different MIMS. This shows that $I_M(G) \leq I_S(G)$. □

The following example shows that it is possible to have $I_M(G) < I_S(G)$, that is, there may be several different elements of $\mathsf{MinimalProblematicPaths}(G)$ that are mapped by function s to the same element of $\mathsf{MIMS}(G)$.

Example 4. *Let $\Sigma = \{\mathsf{twin\_of}, \mathsf{same\_as}\}$, $\Gamma = \{\mathsf{twin\_of}.\mathsf{twin\_of} \subseteq \mathsf{same\_as}\}$, $V = \{\mathsf{ann}, \mathsf{sue}\}$, and $E = \{(\mathsf{ann}, \mathsf{twin\_of}, \mathsf{sue}), (\mathsf{sue}, \mathsf{twin\_of}, \mathsf{ann})\}$. In this case $\mathsf{MIMS}(G) = \{G\}$; hence $I_M(G) = 1$. But there are two minimal problematic paths, namely* $(\mathsf{ann}, \mathsf{twin\_of}, \mathsf{sue})(\mathsf{sue}, \mathsf{twin\_of}, \mathsf{ann})$ *and* $(\mathsf{sue}, \mathsf{twin\_of}, \mathsf{ann})(\mathsf{ann}, \mathsf{twin\_of}, \mathsf{sue})$. *Hence $I_S(G) = 2$. Note that adding a new node and/or a new edge does not help to resolve the inconsistency.*

For the case of propositional logic knowledge bases there is an inconsistency measure usually called the "repair measure" that counts the minimal number of formulas that need to be deleted to make the knowledge base consistent [66]. The situation is more complicated for graph databases because both vertices and edges may be deleted and because of nonmonotonicity the addition of edges may restore consistency as well. So there are three measures analogous to the repair measure as the addition of isolated vertices cannot restore consistency. We need a notation to indicate the subgraph of a graph obtained by deleting some edges or vertices as well as the enlarged graph obtained by adding edges. For a database $G = (V, E)$, let $E' \subseteq E$ (resp., $V' \subseteq V$). We use the notation $G - E'$ (resp., $G - V'$) for the subgraph G' of G such that $G' = (V, E \setminus E')$ (resp., $G' = (V \setminus V', E')$ where $E' \subseteq E$ is the subset of edges not adjacent to V'). Finally, let E' be a set of edges whose vertices are in V. We write $G + E'$ for the graph $(V, E \cup E')$.

Our last three inconsistency measures are as follows.

Definition 3 *(GDB Inconsistency Measures (continued)). For any GDB G, the inconsistency measures I_{E-}, I_{E+}, and I_{V-} are as follows:*

- $I_{E-}(G) = \min\{|E'| \mid G - E' \text{ is consistent}\}$.
- $I_{E+}(G) = \min\{|E'| \mid G + E' \text{ is consistent}\}$.

- $I_{V-}(G) = \min\{|V'| \mid G - V' \text{ is consistent}\}$.

Hence, I_{E-} counts the minimal number of edges that need to be removed to restore consistency; I_{E+} counts the minimal number of edges that need to be added to restore consistency; and I_{V-} counts the minimal number of vertices that need to be removed to restore consistency.

3.2. Computing the inconsistency measures on the example

To illustrate the inconsistency measures just introduced in Definitions 2 and 3, we do the computation for the GDB G_{ex} of our running example (see Fig. 1) with the constraints of Example 2. Here, Γ consists of four RPCs. It will be useful to break the problem into several parts as follows. The graph remains the same; however, we add the RPCs one at a time. So here we will use implicitly the two parameter version of the measures: including both the graph (which remains the same) and the set of constraints. We will use the notation $\Gamma_i = \{C_1, \ldots, C_i\}$, but just write '$\Gamma_i$:' followed by the various $I(G)$s instead of $I(G, \Gamma_i)$. Γ_1 contains only C_1; Γ_2 adds C_2 and so on. For each case we explain the computations. Clearly, any constraint that is violated remains violated as new constraints are added.

Γ_1: To recap, $C_1 = \text{child\_of} \subseteq \text{son\_of}|\text{daughter\_of}$. As observed in Example 2, there are four cases where an edge with label child_of does not also have the label son_of or daughter_of (see $I_P(G)$ below for the corresponding four problematic pairs). We obtain that:
- $I_B(G) = 1$ because G is inconsistent.
- $I_C(G) = 1$ because the single constraint in Γ_1 is violated.
- $I_M(G) = 4$ because $|\text{MIMS}(G)| = 4$, one MIS per problematic pair, each containing a single edge:
  $G_1 = (\{\text{Frank}, \text{Bryan}\}, \{(\text{Frank}, \text{child\_of}, \text{Bryan})\})$,
  $G_2 = (\{\text{Ester}, \text{Bryan}\}, \{(\text{Ester}, \text{child\_of}, \text{Bryan})\})$,
  $G_3 = (\{\text{Grace}, \text{Clara}\}, \{(\text{Grace}, \text{child\_of}, \text{Clara})\})$,
  $G_4 = (\{\text{Clara}, \text{Alice}\}, \{(\text{Clara}, \text{child\_of}, \text{Alice})\})$.
- $I_P(G) = 4$ because there are four problematic pairs: (Frank, Bryan), (Ester, Bryan), (Grace, Clara), and (Clara, Alice).
- $I_S(G) = 4$ because there is one minimal problematic path containing a single edge for each problematic pair.
- $I_E(G) = 4$ because there are four problematic edges, one for each problematic pair.
- $I_L(G) = 1$ because there is one problematic label, child_of.
- $I_V(G) = 6$ because there are six problematic vertices: Frank, Ester, Grace, Bryan, Clara, and Alice.
- $I_{E-}(G) = 4$ because the four problematic edges must all be deleted to restore consistency.
- $I_{E+}(G) = 4$ because for each problematic edge we must add an edge with the same vertices and label son_of or daughter_of to restore consistency.
- $I_{V-}(G) = 2$ because it suffices to remove the two vertices Bryan and Clara to restore consistency.

Γ_2: As shown in Example 2, there are three paths that violate $C_2 = \text{child\_of}.(\text{brother\_of}|\text{sister\_of}) \subseteq \text{nephew\_of}|\text{niece\_of}$, namely $\pi_1 = (\text{David}, \text{child\_of}, \text{Bryan})\ (\text{Bryan}, \text{brother\_of}, \text{Clara})$, $\pi_2 = (\text{Ester}, \text{child\_of}, \text{Bryan})\ (\text{Bryan}, \text{brother\_of}, \text{Clara})$, and $\pi_3 = (\text{Grace}, \text{child\_of}, \text{Clara})\ (\text{Clara}, \text{sister\_of}, \text{Bryan})$. These paths need an edge with the label nephew_of or niece_of from the first to the third vertex. Recall that $\Gamma_2 = \{C_1, C_2\}$ so the inconsistencies remain from Γ_1. We obtain that:
- $I_B(G) = 1$ because G is inconsistent.
- $I_C(G) = 2$ because both constraints in Γ_2 are violated.
- $I_M(G) = 5$ because $|\text{MIMS}(G)| = 5$, namely $G_1, \ldots, G_4$ as above and $G_5 = (\{\text{David}, \text{Bryan}, \text{Clara}\}, \{(\text{David}, \text{child\_of}, \text{Bryan}), (\text{Bryan}, \text{brother\_of}, \text{Clara})\})$. Note that $s(\pi_2)$ contains G_2 and $s(\pi_3)$ contains G_3; hence these do not increase the value of I_M.
- $I_P(G) = 7$ because there are three new problematic pairs: (David, Clara), (Ester, Clara), and (Grace, Bryan).
- $I_S(G) = 5$ because there is one new minimal problematic path, namely, (David, child_of, Bryan)(Bryan, brother_of, Clara).
- $I_E(G) = 7$ because there are three new problematic edges: (David, child_of, Bryan), (Bryan, brother_of, Clara), and (Clara, sister_of, Bryan).
- $I_L(G) = 3$ because there are two new problematic labels: brother_of and sister_of.
- $I_V(G) = 7$ because there is one new problematic vertex: David.
- $I_{E-}(G) = 5$ because one more edge, namely (Bryan, brother_of, Clara) must be removed to restore consistency.
- $I_{E+}(G) = 7$ because three new edges must be added for the vertex pairs: (David, Clara), (Ester, Clara), and (Grace, Bryan), each with either the label nephew_of or niece_of to restore consistency.
- $I_{V-}(G) = 2$ as no additional vertices must be removed to restore consistency.

Γ_3: There are two paths that violate $C_3 = \text{child\_of}.\text{child\_of} \subseteq \text{grandson\_of}|\text{granddaughter\_of}$, namely the path $\pi_4 = (\text{Ester}, \text{child\_of}, \text{Bryan})\ (\text{Bryan}, \text{child\_of}, \text{Alice})$ and the path $\pi_5 = (\text{Frank}, \text{child\_of}, \text{Bryan})(\text{Bryan}, \text{child\_of}, \text{Alice})$ both need an edge with the label grandson_of or granddaughter_of. However, both of these paths already contain a previously given minimal problematic path. We obtain that:
- $I_B(G) = 1$ because G is inconsistent.
- $I_C(G) = 3$ because all the constraints in Γ_3 are violated.
- $I_M(G) = 5$ because $|\text{MIMS}(G)| = 5$ as explained above.
- $I_P(G) = 9$ because there are two new problematic pairs: (Ester, Alice) and (Frank, Alice).
- $I_S(G) = 5$ because there is no new minimal problematic path, as explained above.

- $I_E(G) = 8$ because there is one new problematic edge, (Bryan, child_of, Alice).
- $I_L(G) = 3$ because there are no new problematic labels.
- $I_V(G) = 7$ because there are no new problematic vertices.
- $I_{E-}(G) = 5$ because no additional edge needs to be removed to restore consistency.
- $I_{E+}(G) = 9$ because two new edges must be added: either grandson_of or granddaughter_of to each of the vertex pairs (Ester, Alice) and (Frank, Alice) to restore consistency.
- $I_{V-}(G) = 3$ as it suffices to remove one more vertex, Alice, to restore consistency.

Γ_4 There is no path that violates $C_4 = \text{son\_of}.\text{child\_of} \subseteq \text{grandson\_of}$. Hence the inconsistency measures are the same as for Γ_3.

4. Rationality postulates for graph database inconsistency measures

Definition 1 allows for a broad range of functions, many of which have no good intuitive rationale for being an inconsistency measure. Consider, for example, the function that assigns to a consistent graph database the value 0 and to an inconsistent one its size. But the size of the inconsistent graph database is not a good inconsistency measure: a large one may have few inconsistencies while a small one may have many. The situation was the same for propositional knowledge bases. So researchers looked for properties that they thought a good inconsistency measure should satisfy. These properties of inconsistency measures are called rationality postulates. Next we define the postulates we will use for GDBs.

Our postulates are based on common postulates for propositional knowledge bases and relational databases. Here we describe briefly these postulates for those cases. Monotony states that the inconsistency measure of a subset of the formulas (resp., relational database elements) cannot exceed the measure of the original set. (Recall that for a definite relational database an element is simply a tuple.) Free-Formula Independence (resp., Free-Element Independence) states that the removal of a free formula (resp., element) does not change the inconsistency measure. Penalty states that the removal of a problematic formula (resp., element) decreases the inconsistency measure. Super-Additivity states that the inconsistency measure of the union of two disjoint sets of formulas (resp., elements) is at least as large as their sum. MI-Separability also deals with the union but it states that if the sets of minimal inconsistent subsets (MISs) of the two sets of formulas (resp., elements) form a partition of the set of inconsistent subsets of the union then the inconsistency measure of the union is exactly the sum of the inconsistent measures of the two sets. Finally, MI-Normalization states that the inconsistency measure of a minimal inconsistent set of formulas (resp., elements) is 1.

Our postulates follow these concepts with some modifications. As we deal with several different kinds of objects and as our emphasis is on vertices and edges, we define some postulates separately for vertices and edges. There is also an issue with the analogs of the Penalty postulate. The problematic edges and vertices are based on all the problematic paths, but our inconsistency measure that calculates the number of problematic paths does this only for the minimal problematic paths, as explained earlier.

Definition 4 *(Postulates for GDB Inconsistency Measures). For a given Σ and Γ, let G and G' be GDBs, e and v an edge and a vertex of G respectively, and I a GDB inconsistency measure. The postulates for the GDB inconsistency measures are as follows:*

Monotony *If $G' \subseteq_m G$, then $I(G') \leq I(G)$.*
Edge Independence *If $G - \{e\} \subseteq_m G$ and e is free in G, then $I(G - \{e\}) = I(G)$.*
Vertex Independence *If $G - \{v\} \subseteq_m G$ and v is free in G, then $I(G - \{v\}) = I(G)$.*
Edge Penalty *If $G - \{e\} \subseteq_m G$ and e is minimal problematic in G, then $I(G - \{e\}) < I(G)$.*
Vertex Penalty *If $G - \{v\} \subseteq_m G$ and v is minimal problematic in G, then $I(G - \{v\}) < I(G)$.*
Edge Super-Additivity *If $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, $G_1 \subseteq_m G_1 \cup G_2$, $G_2 \subseteq_m G_1 \cup G_2$, and $E_1 \cap E_2 = \emptyset$, then $I(G_1 \cup G_2) \geq I(G_1) + I(G_2)$.*
Vertex Super-Additivity *If $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, and $V_1 \cap V_2 = \emptyset$, then $I(G_1 \cup G_2) \geq I(G_1) + I(G_2)$.*
MIMS-Separability *If $\text{MIMS}(G_1) \cap \text{MIMS}(G_2) = \emptyset$ and $\text{MIMS}(G_1) \cup \text{MIMS}(G_2) = \text{MIMS}(G_1 \cup G_2)$, then $I(G_1 \cup G_2) = I(G_1) + I(G_2)$.*
MIMS-Normalization *If $\text{MIMS}(G) = \{G\}$, then $I(G) = 1$.*

Hence, in the context of GDBs, Monotony means that the inconsistency measure of a monotonic subgraph is not greater than that of the graph database. The two versions of Independence mean that if the subgraph obtained by removing a free edge (resp., vertex) is monotonic, then the subgraph has the same inconsistency measure as the graph database. The two versions of Penalty mean that if the subgraph obtained by removing a minimal problematic edge (resp., vertex) is monotonic, then the subgraph has a smaller inconsistency measure than the graph database. The two versions of Super-Additivity mean that if two GDBs have no common edge (resp., vertex), then the inconsistency measure of the union of the GDBs is at least the sum of the inconsistency measures of the two GDBs. We also require that each graph be a monotonic subgraph of the union. This need not be stated for Vertex Super-Additivity because the vertices in the two GDBs are distinct. MIMS-Separability means that if no MIMS of one GDB is also a MIMS of the other GDB and the MIMSs of the union of the two GDBs is exactly the union of the MIMSs of the two GDBs separately, then the inconsistency measure of the union is exactly the sum of the inconsistency measures of the two GDBs. MIMS-Normalization means that the inconsistency measure of a GDB that is its own MIMS is 1. Here, minimal inconsistent monotonic subgraphs are used instead of minimal inconsistent subsets.

As stated next, for Penalty and Super-Additivity there is a relationship between the edge and vertex versions.

Proposition 2. *For any GDB inconsistency measure I, it holds that:*

(a) *If I satisfies Monotony and Edge Penalty then I satisfies Vertex Penalty.*
(b) *If I satisfies Edge Super-Additivity then I satisfies Vertex Super-Additivity.*

Proof. We first consider item *(a)*. Suppose that I satisfies Monotony and Edge Penalty. Let v be a minimal problematic vertex in G and $G - \{v\} \subseteq_m G$. For such a vertex v, there must be at least one edge e adjacent to it such that e is minimal problematic in G. The removal of v from G removes e as well as any other edges adjacent to v. By the hypotheses, if $M \in \mathsf{MIMS}(G - \{v\})$ then $M \in \mathsf{MIMS}(G)$, that is, $G - \{v\} \subseteq_m G$, since the removal of a minimal problematic vertex cannot generate a new problematic path in $G - \{v\}$ which is not already in G. This implies that also $M \in \mathsf{MIMS}(G - \{e\})$ since $G - \{v\} \subseteq G - \{e\}$ and v and e are minimal problematic. By Edge Penalty, $I(G - \{e\}) < I(G)$ and then by Monotony $I(G - \{v\}) \leq I(G - \{e\})$. This means that $I(G - \{v\}) < I(G)$, that is, I satisfies Vertex Penalty.

Consider now item *(b)*. Suppose that I satisfies Edge Super-Additivity and G_1 and G_2 are such that $V_1 \cap V_2 = \emptyset$. Then the conditions of Edge Super-Additivity are satisfied and so $I(G_1 \cup G_2) \geq I(G_1) + I(G_2)$. □

With the same spirit of postulates for inconsistency measures developed for other settings, our postulates are intended to offer indications for analyzing and understanding the behavior of inconsistency measures for GDBs; moreover, they enable comparing inconsistency measures through some formal properties. To this end, we consider the satisfaction of these postulates by the inconsistency measures we defined earlier.

Theorem 1. *The satisfaction of the postulates for the inconsistency measures introduced in Definitions 2 and 3 is as given in Table 1.*

Proof. We do the proof individually for each inconsistency measure. In order to save space, in the counterexamples we do not show that all the conditions, mainly the monotonicity of subsets, are satisfied. Instead we concentrate on showing how the relationship of the inconsistency measures is violated. We also indicate the reuse of a counterexample for a different postulate or inconsistency measure. Moreover, in the following, we use the notation given in Definition 4, e.g., G and G' denote GDBs, and e and v denote an edge and a vertex of G, respectively.

I_B We recall that $I_B(G) = 1$ iff G is not consistent (0 otherwise).

Monotony It suffices to consider the case where G' is inconsistent. Hence $I_B(G') = 1$ and by the definition of $\subseteq_m$, G must also be inconsistent. Thus $I_B(G) = 1$.
Edge Independence By Monotony it suffices to consider the case where G is inconsistent. Hence $I_B(G) = 1$ and as e is free, $G - \{e\}$ must also be inconsistent. So $I_B(G - \{e\}) = 1$.
Vertex Independence The proof is the same as for Edge Independence with v substituted for e.
Edge Penalty Let $\Sigma = \{\mathsf{mother\_of}, \mathsf{parent\_of}\}$, $\Gamma = \{\mathsf{mother\_of} \subseteq \mathsf{parent\_of}\}$, and $G = (\{\mathsf{ann}, \mathsf{sue}, \mathsf{pat}, \mathsf{jane}\}, \{(\mathsf{ann}, \mathsf{mother\_of}, \mathsf{sue}), (\mathsf{pat}, \mathsf{mother\_of}, \mathsf{jane})\})$. G is inconsistent; hence $I_B(G) = 1$. The edge $e = (\mathsf{ann}, \mathsf{mother\_of}, \mathsf{sue})$ is minimal problematic, but $G - \{e\}$ is also inconsistent. So $I_B(G - \{e\}) = 1$.
Vertex Penalty The same counterexample can be used as for Edge Penalty with $v = \mathsf{ann}$ substituted for e.
Edge Super-Additivity Let $\Sigma = \{\mathsf{mother\_of}, \mathsf{parent\_of}\}$, $\Gamma = \{\mathsf{mother\_of} \subseteq \mathsf{parent\_of}\}$, $G_1 = (\{\mathsf{ann}, \mathsf{sue}\}, \{(\mathsf{ann}, \mathsf{mother\_of}, \mathsf{sue})\})$, and $G_2 = (\{\mathsf{pat}, \mathsf{jane}\}, \{(\mathsf{pat}, \mathsf{mother\_of}, \mathsf{jane})\})$. Then, $E_1 \cap E_2 = \emptyset$, but $I_B(G_1 \cup G_2) = 1 \not\geq I_B(G_1) + I_B(G_2) = 2$.
Vertex Super-Additivity The same counterexample can be used as for Edge Super-Additivity as $V_1 \cap V_2 = \emptyset$.
MIMS-Separability The same counterexample can be used as for Edge Super-Additivity because the MIMS-Separability conditions hold for G_1 and G_2.
MIMS Normalization Such a G must be inconsistent and hence $I_B(G) = 1$.

I_C Recall that $I_C(G) = |\{C \mid C \in \Gamma \text{ and } G \not\models C\}|$.

Monotony By the definition of $\subseteq_m$, if $G' \not\models C$ then $G \not\models C$, where $C \in \Gamma$.
Edge Independence We prove the contrapositive. By Monotony, $I_C(G - \{e\}) \leq I_C(G)$. If $I_C(G - \{e\}) < I_C(G)$ then there is $C \in \Gamma$ such that $G - \{e\}) \models C$ but $G \not\models C$. This means that for some path $\pi \in \mathsf{ProblematicPaths}(G)$, $e \in \pi$. Hence e is not free in G.
Vertex Independence The proof is the same as for Edge Independence with v substituted for e.
Edge Penalty The same counterexample can be used as for I_B.
Vertex Penalty The same counterexample can be used as for I_B.
Edge Super-Additivity The same counterexample can be used as for I_B.
Vertex Super-Additivity The same counterexample can be used as for I_B.
MIMS-Separability The same counterexample can be used as for I_B.
MIMS Normalization Let $\Sigma = \{\mathsf{mother\_of}, \mathsf{parent\_of}, \mathsf{gives\_birth\_to}\}$, $\Gamma = \{\mathsf{mother\_of} \subseteq \mathsf{parent\_of}, \mathsf{mother\_of} \subseteq \mathsf{gives\_birth\_to}\}$, and $G = (\{\mathsf{ann}, \mathsf{sue}\}, \{(\mathsf{ann}, \mathsf{mother\_of}, \mathsf{sue})\})$. Then $\mathsf{MIMS}(G) = \{G\}$ and $I_C(G) = 2$.

I_M Recall that $I_M(G) = |\mathsf{MIMS}(G)|$.

Monotony Suppose that $M \in \mathsf{MIMS}(G')$ and $G' \subseteq_m G$. Then $M \subseteq_m G'$ and by the transitivity of $\subseteq_m$, $M \subseteq_m G$. That is, M is an inconsistent monotonic subset of G. M must also be minimal inconsistent in G. Hence $M \in \mathsf{MIMS}(G)$.

Edge Independence As e is free, there is no $M \in \mathsf{MIMS}(G)$ for which e is an edge in M. Also, as $G - \{e\} \subseteq_m G$, $\mathsf{MIMS}(G - \{e\}) = \mathsf{MIMS}(G)$, from which the result follows.

Vertex Independence The proof is the same as for Edge Independence with v (resp., vertex) substituted for e (resp., edge).

Edge Penalty As e is minimal problematic, there is a $G' \in \mathsf{MIMS}(G)$ such that e is an edge in G'. $\mathsf{MIMS}(G - \{e\}) = \mathsf{MIMS}(G) \setminus \{G' \in \mathsf{MIMS}(G) | e \in G'\}$. Hence $I_M(G - \{e\}) < I_M(G)$.

Vertex Penalty This follows from Edge Penalty and Proposition 2(a).

Edge Super-Additivity By the conditions of this postulate, $\mathsf{MIMS}(G_1) \cup \mathsf{MIMS}(G_2) \subseteq \mathsf{MIMS}(G_1 \cup G_2)$, from which the result follows.

Vertex Super-Additivity By $V_1 \cap V_2 = \emptyset$, $\mathsf{MIMS}(G_1 \cup G_2) = \mathsf{MIMS}(G_1) \cup \mathsf{MIMS}(G_2)$ from which the result follows.

MIMS-Separability Immediate from the conditions of this postulate and the definition of I_M.

MIMS-Normalization Immediate from the definition of I_M.

I_P Recall that $I_P(G) = |\mathsf{ProblematicPairs(G)}|$.

Monotony It follows from the definition of $\subseteq_m$, that if $(u, v) \in \mathsf{ProblematicPairs}(G')$ then $(u, v) \in \mathsf{ProblematicPairs}(G)$.

Edge Independence As e is free there is no $\pi \in \mathsf{ProblematicPaths}(G)$ such that $e \in \pi$. Hence $\mathsf{ProblematicPairs}(G - \{e\}) = \mathsf{ProblematicPairs}(G)$, from which the result follows.

Vertex Independence The proof is the same as for Edge Independence with v substituted for e.

Edge Penalty Let $\Sigma = \{\mathsf{friend\_of}, \mathsf{likes}, \mathsf{neighbor\_of}, \mathsf{acquaintance\_of}\}$, $\Gamma = \{\mathsf{friend\_of} \subseteq \mathsf{likes}, \mathsf{neighbor\_of} \subseteq \mathsf{acquaintance\_of}\}$, and $G = (\{\mathsf{ann}, \mathsf{pat}\}, \{(\mathsf{ann}, \mathsf{friend\_of}, \mathsf{pat}), (\mathsf{ann}, \mathsf{neighbor\_of}, \mathsf{pat})\})$. There is one problematic pair, namely (ann, pat); hence $I_P(G) = 1$. The edge $e = (\mathsf{ann}, \mathsf{friend\_of}, \mathsf{pat})$ is minimal problematic, but $G - \{e\}$ has the same problematic pair. So $I_P(G - \{e\}) = 1$ as well, violating the postulate.

Vertex Penalty Let $\Sigma = \{\mathsf{parent\_of}, \mathsf{grandparent\_of}, \mathsf{ancestor\_of}\}$, $\Gamma = \{\mathsf{parent\_of}.\mathsf{parent\_of} \subseteq \mathsf{ancestor\_of}, \mathsf{grandparent\_of} \subseteq \mathsf{ancestor\_of}\}$, and $G = (\{\mathsf{joe}, \mathsf{ann}, \mathsf{pat}\}, \{(\mathsf{joe}, \mathsf{parent\_of}, \mathsf{ann}), (\mathsf{ann}, \mathsf{parent\_of}, \mathsf{pat}), (\mathsf{joe}, \mathsf{grandparent\_of}, \mathsf{pat})\})$. There is one problematic pair, namely (joe, pat); hence $I_P(G) = 1$. The vertex ann is minimal problematic, but $G - \{\mathsf{ann}\}$ has the same problematic pair. So $I_P(G - \{\mathsf{ann}\}) = 1$, violating the postulate.

Edge Super-Additivity As in the counterexample to Edge Penalty, again $\Sigma = \{\mathsf{friend\_of}, \mathsf{likes}, \mathsf{neighbor\_of}, \mathsf{acquaintance\_of}\}$ and $\Gamma = \{\mathsf{friend\_of} \subseteq \mathsf{likes}, \mathsf{neighbor\_of} \subseteq \mathsf{acquaintance\_of}\}$. Let $G_1 = (\{\mathsf{ann}, \mathsf{pat}\}, \{(\mathsf{ann}, \mathsf{friend\_of}, \mathsf{pat})\})$ and $G_2 = (\{\mathsf{ann}, \mathsf{pat}\}, \{(\mathsf{ann}, \mathsf{neighbor\_of}, \mathsf{pat})\})$. Then, the Edge Super-Additivity conditions are satisfied but $I_P(G_1 \cup G_2) = I_P(G_1) = I_P(G_2) = 1$ as each graph has the same problematic pair, (ann, pat). This violates the postulate.

Vertex Super-Additivity By $V_1 \cap V_2 = \emptyset$, $\mathsf{ProblematicPairs}(G_1 \cup G_2) = \mathsf{ProblematicPairs}(G_1) \cup \mathsf{ProblematicPairs}(G_2)$, while $\mathsf{ProblematicPairs}(G_1) \cap \mathsf{ProblematicPairs}(G_2) = \emptyset$. So $I_P(G_1 \cup G_2) = I_P(G_1) + I_P(G_2)$.

MIMS-Separability The same counterexample can be used as for Edge Super-Additivity.

MIMS-Normalization A single MIMS can have only a single problematic pair.

I_S Recall that $I_S(G) = |\mathsf{MinimalProblematicPaths(G)}|$.

Monotony By the definition of $\subseteq_m$, if $\pi \in \mathsf{MinimalProblematicPaths}(G')$ then $\pi \in \mathsf{ProblematicPaths}(G)$. Also, there cannot be a $\pi' \subset \pi$ such that $\pi' \in \mathsf{Problematic\ Paths}(G)$; hence $\pi \in \mathsf{MinimalProblematicPaths}(G)$. Thus $I_S(G') \leq I_S(G)$.

Edge Independence As e is free there is no $\pi \in \mathsf{ProblematicPaths}(G)$ such that $e \in \pi$. Hence there is no $\pi \in \mathsf{MinimalProblematicPaths}(G)$ such that $e \in \pi$. From this the result follows.

Vertex Independence The proof is the same as for Edge Independence with v substituted for e.

Edge Penalty Recall that before Proposition 1, we defined the subgraph constructor function s that maps a minimal problematic path to its corresponding MIMS. We proceed as in the proof of Edge Penalty for I_M but now we get $\mathsf{MinimalProblematicPaths}(G - \{e\}) = \mathsf{MinimalProblematicPaths}(G) \setminus \{\pi \in \mathsf{MinimalProblematicPaths}(G) \,|\, e \in \pi\}$. Hence $I_S(G - \{e\}) < I_S(G)$.

Vertex Penalty This follows from Edge Penalty and Proposition 2(a).

Edge Super-Additivity By $E_1 \cap E_2 = \emptyset$, $\mathsf{MinimalProblematicPaths}(G_1 \cup G_2) \supseteq \mathsf{MinimalProblematicPaths}(G_1) \cup \mathsf{MinimalProblematicPaths}(G_2)$. Hence, $I_S(G_1 \cup G_2) \geq I_S(G_1) + I_S(G_2)$.

Vertex Super-Additivity This follows from Edge Super-Additivity and Proposition 2 (b).

MIMS-Separability In the proof of Proposition 1 we showed that every minimal problematic path defines a single MIMS; however, there may be several minimal problematic paths that define the same MIMS (see Example 4). By the conditions of MIMS-Separability, this means that $\mathsf{MinimalProblematicPaths}(G_1 \cup G_2) = \mathsf{MinimalProblematicPaths}(G_1) \cup \mathsf{MinimalProblematicPaths}(G_2)$ and $\mathsf{MinimalProblematicPaths}(G_1) \cap \mathsf{MinimalProblematicPaths}(G_2) = \emptyset$. Hence $I_S(G_1 \cup G_2) = I_S(G_1) + I_S(G_2)$.

MIMS-Normalization The GDB given in Example 4 is a counterexample as explained there.

I_E Recall that $I_E(G) = |\mathsf{ProblematicEdges(G)}|$.

Monotony By the definition of $\subseteq_m$, if $e \in \text{ProblematicEdges}(G')$ then $e \in \text{ProblematicEdges}(G)$.
Edge Independence As e is free there is no $\pi \in \text{ProblematicPaths}$ such that $e \in \pi$. From this the result follows.
Vertex Independence The proof is the same as for Edge Independence with v substituted for e.
Edge Penalty The result follows from the definition of I_E and the fact that every minimal problematic edge is a problematic edge.
Vertex Penalty This follows from Edge Penalty and Proposition 2(a).
Edge Super-Additivity By $E_1 \cap E_2 = \emptyset$, $\text{ProblematicEdges}(G_1 \cup G_2) \supseteq \text{ProblematicEdges}(G_1) \cup \text{ProblematicEdges}(G_2)$ and $\text{ProblematicEdges}(G_1) \cap \text{ProblematicEdges}(G_2) = \emptyset$. Therefore, $I_E(G_1 \cup G_2) \geq I_E(G_1) + I_E(G_2)$.
Vertex Super-Additivity This follows from Edge Super-Additivity and Proposition 2(b).
MIMS-Separability Let $\Sigma = \{\text{son\_of}, \text{brother\_of}, \text{nephew\_of}, \text{grandson\_of}\}$, $\Gamma = \{\text{son\_of}.\text{brother\_of} \subseteq \text{nephew\_of}, \text{son\_of}.\text{son\_of} \subseteq \text{grandson\_of}\}$, $G_1 = (\{\text{jim}, \text{bob}, \text{ann}\}, \{(\text{jim}, \text{son\_of}, \text{bob}), (\text{bob}, \text{brother\_of}, \text{ann})\})$, and $G_2 = (\{\text{jim}, \text{bob}, \text{steve}\}, \{(\text{jim}, \text{son\_of}, \text{bob}), (\text{bob}, \text{son\_of}, \text{steve})\})$. We have that $I_E(G_1 \cup G_2) = 3 \neq I_E(G_1) + I_E(G_2) = 4$ because of the common edge.
MIMS-Normalization Let $\Sigma = \{\text{parent\_of}, \text{grandparent\_of}\}$, $\Gamma = \{\text{parent\_of}.\text{parent\_of} \subseteq \text{grandparent\_of}\}$, and $G = (\{\text{sue}, \text{pat}, \text{ann}\}, \{(\text{sue}, \text{parent\_of}, \text{pat}), (\text{pat}, \text{parent\_of}, \text{ann})\})$. Then, G is a MIMS but $I_E(G) = 2$.

I_L Recall that $I_L(G) = |\text{ProblematicLabels(G)}|$.

Monotony By the definition of $\subseteq_m$, if $\ell \in \text{ProblematicLabels}(G')$ then $\ell \in \text{ProblematicLabels}(G)$.
Edge Independence The label ℓ of e may be free or problematic. If it is free then $I_L(G - \{e\}) = I_L(G)$. If ℓ is problematic, as e is free, there must be a problematic edge e' for which the label is ℓ. But e' is not removed, hence again $I_L(G - \{e\}) = I_L(G)$.
Vertex Independence The removal of a vertex results in the removal of all edges that have that vertex. All of these edges must also be free. As in the proof of Edge Independence, no problematic label is removed.
Edge Penalty The same counterexample can be used as for I_B.
Vertex Penalty The same counterexample can be used as for I_B.
Edge Super-Additivity The same counterexample can be used as for I_B.
Vertex Super-Additivity The same counterexample can be used as for I_B.
MIMS-Separability The same counterexample can be used as for I_B.
MIMS-Normalization Let $\Sigma = \{\text{parent\_of}, \text{mother\_of}, \text{grandparent\_of}\}$, $\Gamma = \{\text{parent\_of}.\text{mother\_of} \subseteq \text{grandparent\_of}\}$, and $G = (\{\text{sue}, \text{pat}, \text{ann}\}, \{(\text{sue}, \text{parent\_of}, \text{pat}), (\text{pat}, \text{mother\_of}, \text{ann})\})$. Then, G is a MIMS but $I_L(G) = 2$.

I_V Recall that $I_V(G) = |\text{ProblematicVertices(G)}|$.

Monotony By the definition of $\subseteq_m$, if $v \in \text{ProblematicVertices}(G')$ then $v \in \text{ProblematicVertices}(G)$.
Edge Independence As e is free there is no $\pi \in \text{ProblematicPaths}$ such that $e \in \pi$. Therefore, $\text{ProblematicVertices}(G - \{e\}) = \text{ProblematicVertices}(G)$.
Vertex Independence The proof is the same as for Edge Independence with v substituted for e.
Edge Penalty Let $\Sigma = \{\text{friend\_of}, \text{likes}\}$, $\Gamma = \{\text{friend\_of} \subseteq \text{likes}\}$, and $G = \{(\text{ann}, \text{pat}, \text{joe}\}, \{(\text{ann}, \text{friend\_of}, \text{pat}), (\text{ann}, \text{friend\_of}, \text{joe}), (\text{pat}, \text{friend\_of}, \text{joe})\})$. Here, $I_V(G) = 3$. The edge $e = (\text{ann}, \text{friend\_of}, \text{pat})$ is minimal problematic, but $G - \{e\}$ has the same problematic vertices. So $I_V(G - \{e\}) = 3$ violating the postulate.
Vertex Penalty When a minimal problematic vertex v is removed, $I_V(G - \{v\}) < I_V(G)$.
Edge Super-Additivity The same counterexample can be used as for I_P. But in this case, $I_V(G_1 \cup G_2) = I_V(G_1) = I_V(G_2) = 2$.
Vertex Super-Additivity By $V_1 \cap V_2 = \emptyset$, we have that $\text{ProblematicVertices}(G_1 \cup G_2) = \text{ProblematicVertices}(G_1) \cup \text{ProblematicVertices}(G_2)$ and also that $\text{ProblematicVertices}(G_1) \cap \text{ProblematicVertices}(G_2) = \emptyset$. So $I_V(G_1 \cup G_2) = I_V(G_1) + I_V(G_2)$.
MIMS-Separability Let $\Sigma = \{\text{friend\_of}, \text{likes}\}$, $\Gamma = \{\text{friend\_of} \subseteq \text{likes}\}$, $G_1 = (\{\text{ann}, \text{pat}\}, \{(\text{ann}, \text{friend\_of}, \text{pat})\})$, and $G_2 = (\{\text{ann}, \text{joe}\}, \{(\text{ann}, \text{friend\_of}, \text{joe})\})$. Both G_1 and G_2 are MIMSs, but $I_V(G_1 \cup G_2) = 3 \neq I_V(G_1) + I_V(G_2) = 4$.
MIMS-Normalization Let $\Sigma = \{\text{friend\_of}, \text{likes}\}$, $\Gamma = \{\text{friend\_of} \subseteq \text{likes}\}$, and $G = (\{\text{ann}, \text{pat}\}, \{(\text{ann}, \text{friend\_of}, \text{pat})\}$. G is a MIMS, but $I_V(G) = 2$.

I_{E-} Recall that $I_{E-}(G) = \min\{|E'| \text{ such that } G - E' \text{ is consistent}\}$.

Monotony By the definition of $\subseteq_m$, any set of edges needed to be deleted from G' to restore consistency must still be deleted from G.
Edge Independence As e is free, the same sets of edges must be deleted from G' and G to restore consistency.
Vertex Independence The proof is the same as for Edge Independence with v substituted for e.
Edge Penalty Let $\Sigma = \{\text{mother\_of}, \text{parent\_of}, \text{grandmother\_of}\}$, $\Gamma = \{\text{mother\_of}.\text{mother\_of} \subseteq \text{grandmother\_of}, \text{mother\_of}.\text{parent\_of} \subseteq \text{grandmother\_of}\}$ and $G = (\{\text{ann}, \text{pat}, \text{sue}\}, \{(\text{ann}, \text{mother\_of}, \text{pat}), (\text{pat}, \text{mother\_of}, \text{sue}), (\text{pat}, \text{parent\_of}, \text{sue})\})$. Then $I_{E-}(G) = 1$ because it suffices to delete the edge $(\text{ann}, \text{mother\_of}, \text{pat})$ to restore consis-

tency. In G every edge is minimal problematic. Let $G' = G - \{(\text{pat}, \text{mother\_of}, \text{sue})\}$. Then $G' \subsetneq_m G$ but $I_{E-}(G') = 1$ also.

Vertex Penalty Let $\Sigma = \{\text{parent\_of}, \text{grandparent\_of}\}$, $\Gamma = \{\text{parent\_of}.\text{parent\_of} \subseteq \text{grandparent\_of}\}$, and $G = (\{\text{ann}, \text{joe}, \text{sue}, \text{sam}\}, \{(\text{ann}, \text{parent\_of}, \text{sue}), (\text{joe}, \text{parent\_of}, \text{sue}), (\text{sue}, \text{parent\_of}, \text{sam})\})$. Every vertex is minimal problematic. Then $I_{E-}(G) = 1$ because it suffices to delete $(\text{sue}, \text{parent\_of}, \text{sam})$ to restore consistency. Let $G' = G - \{\text{ann}\}$. Again, $I_{E-}(G') = 1$.

Edge Super-Additivity By $E_1 \cap E_2 = \emptyset$ any set of edges that must be removed from $G_1 \cup G_2$ to restore consistency must include the union of a set that restores consistency to G_1 and a set that restores consistency to G_2. Hence $I_{E-}(G_1 \cup G_2) \geq I_{E-}(G_1) + I_{E-}(G_2)$

Vertex Super-Additivity This follows from Edge Super-Additivity and Proposition 2(b).

MIMS-Separability Let $\Sigma = \{\text{parent\_of}, \text{grandparent\_of}\}$, $\Gamma = \{\text{parent\_of}.\ \text{parent\_of} \subseteq \text{grandparent\_of}\}$, $G_1 = (\{\text{ann}, \text{sue}, \text{joe}\}, \{(\text{ann}, \text{parent\_of}, \text{sue}), (\text{sue}, \text{parent\_of}, \text{joe})\})$, and $G_2 = (\{\text{ann}, \text{sue}, \text{pat}\}, \{(\text{ann}, \text{parent\_of}, \text{sue}), (\text{sue}, \text{parent\_of}, \text{pat})\})$. Both G_1 and G_2 form a MIMS and hence $I_{E-}(G_1) = I_{E-}(G_2) = 1$. Also, the conditions for MIMS-Separability are satisfied. But $I_{E-}(G_1 \cup G_2) = 1$ because it suffices to remove the common edge $(\text{ann}, \text{parent\_of}, \text{sue})$ to restore consistency.

MIMS-Normalization It suffices to remove one edge of a MIMS to restore consistency.

I_{E+} Recall that $I_{E+}(G) = \min\{|E'| \text{ such that } G + E' \text{ is consistent}\}$.

Monotony By the definition of $\subseteq_m$, any set of edges that need to be inserted into G' to restore consistency must still be inserted into G.

Edge Independence As e is free, the same sets of edges must be inserted into G' and G to restore consistency.

Vertex Independence The proof is the same as for Edge Independence with v substituted for e.

Edge Penalty The same example can be used as for I_{E-}. In this case $I_{E+}(G) = 1$ because it suffices to insert the single edge $(\text{ann}, \text{grandmother\_of}, \text{sue})$. Using the same G', also, $I_{E+}(G') = 1$.

Vertex Penalty Let $\Sigma = \{\text{friend\_of}\}$, $\Gamma = \{\text{friend\_of}.\text{friend\_of} \subseteq \text{friend\_of}\}$, and $G = (\{\text{ann}, \text{joe}, \text{pat}, \text{sue}\}, \{(\text{ann}, \text{friend\_of}, \text{joe}), (\text{ann}, \text{friend\_of}, \text{pat}), (\text{joe}, \text{friend\_of}, \text{sue}), (\text{pat}, \text{friend\_of}, \text{sue})\})$. Then $I_{E+}(G) = 1$ as it suffices to insert the single edge $(\text{ann}, \text{friend\_of}, \text{sue})$ to restore consistency. In this example every vertex is minimal problematic. Let $G' = G - \{\text{joe}\}$. Then $I_{E+}(G') = 1$ because the same single insertion is needed to restore consistency.

Edge Super-Additivity Let $\Sigma = \{\text{mother\_of}, \text{gave\_birth\_to}, \text{parent\_of}\}$, $\Gamma = \{\text{mother\_of} \subseteq \text{parent\_of}, \text{gave\_birth\_to} \subseteq \text{parent\_of}\}$, $G_1 = (\{\text{ann}, \text{sue}\}, \{(\text{ann}, \text{mother\_of}, \text{sue})\})$, and $G_2 = (\{\text{ann}, \text{sue}\}, \{(\text{ann}, \text{gave\_birth\_to}, \text{sue})\})$. Then $I_{E+}(G_1 \cup G_2) = 1 < I_{E+}(G_1) + I_{E+}(G_2) = 2$ because in all three cases it suffices to add $(\text{ann}, \text{parent\_of}, \text{sue})$ to restore consistency.

Vertex Super-Additivity By $V_1 \cap V_2 = \emptyset$ any set of edges that must be inserted into $G_1 \cup G_2$ to restore consistency must be the union of a set that restores consistency to G_1 and a set that restores consistency to G_2. Hence $I_{E+}(G_1 \cup G_2) = I_{E+}(G_1) + I_{E+}(G_2)$.

MIMS-Separability The same example can be used as for I_{E-}. Now $I_{E+}(G_1) = I_{E+}(G_2) = 1$ also because it suffices to add the single edge $(\text{ann}, \text{grandparent\_of}, \text{joe})$ to both G_1 and G_2 to restore consistency. The same is true for $G_1 \cup G_2$, hence $I_{E+}(G_1 \cup G_2) = 1$.

MIMS-Normalization Let $\Sigma = \{\text{mother\_of}, \text{grandmother\_of}, \text{grandparent\_of}\}$, $\Gamma = \{\text{mother\_of}.\text{mother\_of} \subseteq \text{grandmother\_of}, \text{mother\_of}.\text{mother\_of} \subseteq \text{grandparent\_of}, \text{and } G = (\{\text{ann}, \text{pat}, \text{joe}\}, \{(\text{ann}, \text{mother\_of}, \text{pat}), (\text{pat}, \text{mother\_of}, \text{joe})\})$, Here G is a MIMS but $I_{E+}(G) = 2$ because the two edges, $(\text{ann}, \text{grandmother\_of}, \text{joe})$ and $(\text{ann}, \text{grandparent\_of}, \text{joe})$ must be added to restore consistency.

I_{V-} Recall that $I_{V-}(G) = \min\{|V'| \text{ such that } G - V' \text{ is consistent}\}$.

Monotony By the definition of $\subseteq_m$, any set of vertices needed to be deleted from G' to restore consistency must still be deleted from G.

Edge Independence As e is free, the same sets of vertices must be deleted from G' and G to restore consistency.

Vertex Independence The proof is the same as for Edge Independence with v substituted for e.

Edge Penalty The same example can be used as for I_{E-} to obtain $I_{V-}(G) = I_{V-}(G - \{(\text{pat}, \text{mother}, \text{sue})\}) = 1$.

Vertex Penalty The same example can be used as for I_{E-} to obtain $I_{V-}(G) = I_{V-}(G - \{\text{ann}\}) = 1$.

Edge Super-Additivity The same example can be used as for I_{E+} with the change that it suffices remove the vertex ann to obtain $I_{V-}(G_1 \cup G_2) = 1 < I_{V-}(G_1) + I_{V-}(G_2) = 2$.

Vertex Super-Additivity By $V_1 \cap V_2 = \emptyset$ any set of vertices that must be removed from $G_1 \cup G_2$ to restore consistency must be the union of a set that restores consistency to G_1 and a set that restores consistency to G_2. Hence $I_{V-}(G_1 \cup G_2) = I_{V-}(G_1) + I_{V-}(G_2)$.

MIMS-Separability The same example can be used as for I_{E-} with the change that it suffices remove the common vertex ann instead of the edge $(\text{ann}, \text{parent\_of}, \text{sue})$ to restore consistency; hence $I_{V-}(G_1 \cup G_2) = 1 < I_{V-}(G_1) + I_{V-}(G_2) = 2$.

MIMS-Normalization It suffices to remove a single vertex from a MIMS to restore consistency. □

All the measures we are considering satisfy Monotony, Edge Independence and Vertex Independence. In fact, I_M satisfies all nine postulates and I_S satisfies eight. We think this indicates that these are particularly good inconsistency measures for graph databases. However, as we will see in the next section, this comes at a cost: these two measures are the most involved from a computational standpoint. The tractable measures are I_B, I_C, I_P, I_E, I_L, and I_V. I_B and I_C are quite different from the others. I_B is a limiting case and I_C does not deal with the elements of the graph. Their use is for the special purpose of checking for inconsistency (I_B) and counting the constraints that cause inconsistencies (I_C). Among the other four tractable measures, I_E violates only the two postulates involving MIMSs. I_V is similar; however, it has the disadvantage of also violating Edge Penalty and Edge Super-Additivity. This suggests that among the tractable measures, I_E is the best from the two standpoints we considered: compliance to rationality postulates and computational cost. The other three measures, the repair-based ones I_{E-}, I_{V-}, and I_{E+} turn out to be somewhere in the middle, with I_{E-} being better than the other two from the point of view of postulates satisfaction (all three measures are equally intractable, but less than I_M and I_S under standard complexity assumptions).

5. The complexity of graph database inconsistency measures

We investigate the combined and the data-complexity [54,55] of the following three decision problems, which intuitively ask if a given value v is, respectively, lower than, greater than, or equal to the value returned by a given IM when applied to a given GDB. Since every considered IM returns an integer, it suffices to focus on an integer threshold v.

Definition 5 *(Lower (**LV**), Upper (**UV**), and Exact Value (**EV**)). Let I be an IM. Given a GDB G and a set Γ of RPCs, and a positive value $v \in \mathbb{N}^{>0}$,*

- $\boldsymbol{LV}_I(G, \Gamma, v)$ *is the problem of deciding whether* $I(G) \geq v$.

Given G and a non-negative value $v' \in \mathbb{N}^{\geq 0}$,

- $\boldsymbol{UV}_I(G, \Gamma, v')$ *is the problem of deciding whether* $I(G) \leq v'$, *and*
- $\boldsymbol{EV}_I(G, \Gamma, v')$ *is the problem of deciding whether* $I(G) = v'$.

When the input consists of a GDB and a value only (i.e., when the set Γ of RPCs is fixed) we refer to these problems as Fixed Lower (**FLV**), Fixed Upper (**FUV**), and Fixed Exact Value (**FEV**) problems. Hence these problems will be denoted as $\mathbf{FLV}_I(G, v)$, $\mathbf{FUV}_I(G, v')$, and $\mathbf{FEV}_I(G, v')$, respectively, omitting the (fixed) set of constraints.

We also consider the problem of determining the value of an inconsistency measure.

Definition 6 *(Inconsistency Measurement (**IM**) problem). Let I be an inconsistency measure. Given a GDB G and a set Γ of RPCs, $\boldsymbol{IM}_I(G, \Gamma)$ is the problem of computing the value of $I(G)$.*

Also for the inconsistency measurement problem, if the input consists only of the GDB G, we refer to this problem as the Fixed Inconsistency Measurement problem, denoted as $\mathbf{FIM}_I(G)$.

Given an inconsistency measure, some of the complexity results concerning a given problem (e.g. **LV**) can be established by showing results for another problem (e.g. **UV**) if certain conditions hold. The following lemmas provide some relationships between the complexity of the above-mentioned problems that will be useful in our complexity analysis.

The first lemma states that, for the considered inconsistency measures, characterizing the complexity of either **LV** or **UV** (resp., **FLV** or **FUV**) w.r.t. certain classes of the polynomial hierarchy is sufficient to establish the complexity of both problems. Moreover, showing the membership of **LV** or **UV** (resp., **FLV** or **FUV**) in such classes is sufficient to establish a membership result for **EV**(resp., **FEV**).

Lemma 1. *Let G be a GDB, and C a complexity class in $\{\Sigma_k^p, \Pi_k^p\}$ with $k > 0$. For all the inconsistency measures I introduced in Definitions 2 and 3:*

- $\boldsymbol{LV}_I(G, \Gamma, v)$ *is C-complete iff* $\boldsymbol{UV}_I(G, \Gamma, v)$ *is co$C$-complete;*
- *if* $\boldsymbol{LV}_I(G, \Gamma, v)$ *or* $\boldsymbol{UV}_I(G, \Gamma, v)$ *is in C, then* $\boldsymbol{EV}_I(G, \Gamma, v)$ *is in D_k^p.*

*The results hold also for **FLV**, **FUV**, and **FEV** (in place of **LV**, **UV**, and **EV**, respectively).*

Proof. First observe that for all the inconsistency measures, I, that we defined, $I(G)$ is an integer value. Thus, $I(G) \geq v$ iff $I(G) \not\leq v - 1$. That is, $\mathbf{LV}_I(G, \Gamma, v)$ is true iff $\mathbf{UV}_I(G, \Gamma, v-1)$ is false. Thus, if $\mathbf{LV}_I(G, \Gamma, v)$ is Σ_k^p-complete (resp., Π_k^p-complete), then $\mathbf{UV}_I(G, \Gamma, v)$ is Π_k^p-complete (resp., Σ_k^p-complete), and vice versa. The second part of the statement follows from the first one and the definition of D_k^p. Moreover, reasoning similarly, we obtain that the results hold also for **FLV**, **FUV**, and **FEV**. □

The second lemma provides a relationship between upper-bound results on the complexity of **UV** (**FUV**) and upper-bound results on the complexity of **IM** (**FIM**). Before stating the lemma, we introduce some notation. Let $\mathbf{G}_n$ be the set of all GDBs of size n, and I an IM. We use $I[\mathbf{G}_n]$ to denote the image of $\mathbf{G}_n$ under the function I, that is, the set of all the values I can take as its argument varies over GDBs of size n; this is called the expressivity of the measure [26].

Lemma 2. *Let G be a GDB, and C a complexity class in $\{\Sigma_k^p, \Pi_k^p\}$ with $k > 0$. For all the inconsistency measures, I, introduced in Definitions 2 and 3, if $\mathbf{UV}_I(G, \Gamma, v)$ (resp., $\mathbf{FUV}_I(G, v)$) is in C, then $\mathbf{IM}_I(G, \Gamma)$ (resp., $\mathbf{FIM}_I(G)$) is in $FP^{\Sigma_k^p[\log f(n)]}$ with $f : \mathbb{N} \to \mathbb{N}$ such that $|I[\mathbf{G}_n]|$ is $O(f(n))$.*

Proof. Observe that $I[\mathbf{G}_n]$ can be upper bounded in advance. The value of $I(G)$ can be computed by performing a binary search in the interval $[0, |I[\mathbf{G}_n]|]$, that is by asking $O(\log f(n))$ queries to a Σ_k^p-oracle solving $\mathbf{UV}_I(G, \Gamma, v)$ (resp., $\mathbf{FUV}_I(G, v)$), where v is chosen as usual in a binary search. □

The last lemma connects the results for **FLV**, **FUV**, and **FEV** and those for **LV**, **UV**, and **EV**, respectively.

Lemma 3. *Let I be an inconsistency measure and C a complexity class in $\{\Sigma_k^p, \Pi_k^p\}$.*

- *If $\mathbf{FLV}_I(G, v)$ (resp., $\mathbf{FUV}_I(G, v)$, $\mathbf{FEV}_I(G, v)$) is C-hard, then $\mathbf{LV}_I(G, \Gamma, v)$ (resp., $\mathbf{UV}_I(G, \Gamma, v)$, $\mathbf{EV}_I(G, \Gamma, v)$) is C-hard.*
- *If $\mathbf{FIM}_I(G)$ is FP^C-hard (resp., $FP^{C[\log n]}$-hard), then $\mathbf{IM}_I(G, \Gamma)$ is FP^C-hard (resp., $FP^{C[\log n]}$-hard).*
- *If $\mathbf{LV}_I(G, \Gamma, v)$ (resp., $\mathbf{UV}_I(G, \Gamma, v)$, $\mathbf{EV}_I(G, \Gamma, v)$) is in C, then $\mathbf{FLV}_I(G, v)$ (resp., $\mathbf{FUV}_I(G, v)$, $\mathbf{FEV}_I(G, v)$) is in C.*
- *If $\mathbf{IM}_I(G, \Gamma)$ is in FP^C (resp., $FP^{C[\log n]}$), then $\mathbf{FIM}_I(G)$ is in FP^C (resp., $FP^{C[\log n]}$).*

Proof. This follows from the fact that hardness results under data complexity carry over to the case of combined complexity (where the set of constraints is part of the input, not considered fixed). On the other hand, membership results under combined complexity carry over to the special case of data complexity (where the set of constraints is not part of the input, and thus considered fixed). □

5.1. Tractable measures

We start our complexity analysis by showing that six of the eleven measures we defined are tractable. This is an encouraging result showing that the complexity of calculating the inconsistency measures for graph databases, as for definite relational databases [35], is far better than the case for propositional knowledge bases [29].

5.1.1. The measures I_B, I_C, and I_P

We start with the measures I_B, I_C, and I_P for which the computation is similar. The following proposition states that the problem of computing the value of the drastic measure is polynomial under combined complexity and in NL under data complexity. Moreover, the same results also hold for the measure I_C, which counts the number of violated constraints, and for I_P which counts the number of problematic pairs.

Proposition 3. *Let $G = (V, E)$ be a GDB. For $I \in \{I_B, I_C, I_P\}$, it holds that $\mathbf{IM}_I(G, \Gamma)$ is in FP and $\mathbf{FIM}_I(G)$ is in FNL.*

Proof. We start with IM I_C, which is defined as $I_C(G) = |\{C \mid C \in \Gamma \text{ and } G \not\models C\}|$. Let $C = Q_1 \subseteq Q_2$ be a constraint in Γ. $G \not\models C$ if there is a problematic pair (u, v), meaning that $(u, v) \in Q_1(G)$ and $(u, v) \notin Q_2(G)$. Deciding whether a given pair of vertices of G belongs to the answer of an RPQ Q can be done in $O(|E| \cdot ||Q||)$ [46]. Hence, for each $C = Q_1 \subseteq Q_2 \in \Gamma$, deciding whether $G \not\models C$ can be done in $O(|V|^2 \cdot |E| \cdot (||Q_1|| + ||Q_2||)$. Since $||\Gamma|| = \sum_{Q_1 \subseteq Q_2 \in \Gamma} ||Q_1|| + ||Q_2||$, it follows that $\mathbf{IM}_{I_C}(G, \Gamma)$ can be computed in time $O(|V|^2 \cdot |E| \cdot ||\Gamma||)$, and thus it is in FP. Since this also suffices to check for all the problematic pairs of vertices, the result for $\mathbf{IM}_{I_P}(G, \Gamma)$ follows.

Consider now $\mathbf{FIM}_{I_C}(G)$. The result follows from the fact that the data complexity of deciding the answer of an RPQ is in NL [46], and NL is closed under complement. Hence, for a constraint $C = Q_1 \subseteq Q_2$, checking whether $G \not\models C$, that is checking whether there is a pair (u, v) such that $(u, v) \in Q_1(G)$ and $(u, v) \notin Q_2(G)$, is in NL as well. Finally, the union of such checks (one for each constraint in Γ), is NL since NL is closed under the union operation and the size of Γ is fixed (bounded by a constant). Thus, $\mathbf{FIM}_{I_C}(G)$ can be solved in FNL. As for $\mathbf{FIM}_{I_P}(G)$, it is the union of the sets of problematic pairs, that as said above can be computed in FNL.

Finally, for $\mathbf{IM}_{I_B}(G, \Gamma)$ (resp., $\mathbf{FIM}_{I_B}(G)$), we have that $\mathbf{IM}_{I_B}(G, \Gamma) = 1$ (resp., $\mathbf{FIM}_{I_B}(G) = 1$) iff $\mathbf{IM}_{I_C}(G, \Gamma) \geq 1$ (resp., $\mathbf{FIM}_{I_C}(G, \Gamma) \geq 1$), from which the results for I_B follow. □

As a consequence of Proposition 3, we have that $\mathbf{LV}_I$, $\mathbf{UV}_I$, and $\mathbf{EV}_I$, for $I \in \{I_B, I_C, I_P\}$, can be solved in polynomial time. In fact, once the value of $I(G)$ is computed, deciding whether $I(G)$ is lower than, greater than, or equal to a given value v is trivial. For similar reasons, we have that $\mathbf{FLV}_I$, $\mathbf{FUV}_I$, and $\mathbf{FEV}_I$ are in NL for $I \in \{I_B, I_C, I_P\}$.

5.1.2. Measures counting the problematic edges, vertices, and labels

We now consider the measures I_E, I_L, and I_V counting the number of problematic edges, labels, and vertices, respectively. The proofs of the complexity of these measures are similar and can be handled as a group, as they rely on the following lemma.

Lemma 4. *Given a GDB $G = (V, E)$ and an edge $(v, \ell, v') \in E$, deciding whether there is a path $\pi \in \mathsf{ProblematicPaths}(G)$ such that $(v, \ell, v') \in \pi$ can be accomplished in P under combined complexity and in NL under data-complexity.*

Proof. The algorithm for checking whether an edge is problematic can be obtained by exploiting the fact that checking whether a labeled graph contains a directed path π' from a node x to node y such that the string of labels of π' satisfies a given regular expression is tractable [67]. In particular, the algorithm in [67] relies on constructing the so-called intersection graph $I(G, A_Q)$ of the input graph G and a non-deterministic finite automaton A_Q accepting the language of the given regular expression Q. Before introducing our algorithm for checking whether there is a path $\pi \in \mathsf{ProblematicPaths}(G)$ such that $(v, \ell, v') \in \pi$, it is useful to recall how $I(G, A_Q)$ is obtained.

The nodes of the intersection graph $I(G, A_Q)$ are the pairs (n_1, n_2) where n_1 is a node of G and n_2 is a node of A_Q. For instance, consider the case where $\Sigma = \{a, b, c, d\}$ and $G = (V, E)$ where $V = \{v_1, v_2\}$ and $E = \{(v_1, a, v_2), (v_1, c, v_2), (v_2, b, v_1)\}$. Assume that there is one constraint: $a.b.c \subseteq d$. Considering the regular expression $Q = a.b.c$, the automaton A_Q consists of the nodes q_0, q_1, q_2, q_3, where q_0 represents the initial state, and q_3 is the final state, and the edges $(q_0, a, q_1), (q_1, b, q_2), (q_2, c, q_3)$, meaning that we start with the initial state q_0, then move to q_1 by reading a, then move to q_2 by reading b, then move to q_3 by reading c, and then stop in the final node q_3. Thus, for our example, the intersection graph $I(G, A_Q)$ consists of the following 8 nodes, where we use a pair to name a node obtained by combining a node of G with one of A_Q: $(v_1, q_0), (v_1, q_1), (v_1, q_2), (v_1, q_3), (v_2, q_0), (v_2, q_1)$, $(v_2, q_2), (v_2, q_3)$.

The edges of the intersection graph $I(G, A_Q)$ are such that there is an edge $e = ((n_1, n_2), \ell, (n_1', n_2'))$ if and only if there is an edge (n_1, ℓ, n_1') in G and there is an edge (n_2, ℓ, n_2') in A_Q. Continuing with our example, we have that the edges of the intersection graph are $((v_1, q_0), a, (v_2, q_1))$, $((v_2, q_1), b, (v_1, q_2))$, and $((v_1, q_2), c, (v_2, q_3))$.

In [67], it is shown that there is a path from a node x to a node y of G satisfying a regular expression Q if and only if there is a path in $I(G, A_Q)$ from the node (x, q_0), where q_0 is the initial state of A_Q, to a node in $\{(y, q_{f_1}), \ldots, (y, q_{f_k})\}$, where $q_{f_1}, \ldots, q_{f_k}$ are the final states of A_Q. For instance, in our example there is a path from v_1 to v_2 of G satisfying $Q = a.b.c$ if and only if there is a path in $I(G, A_Q)$ from the initial state (v_1, q_0) to the final state (v_2, q_3). The path in G is $(v_1, a, v_2)(v_2, b, v_1)(v_1, c, v_2)$. Observe that this procedure can be used to decide whether a given pair is an answer pair in $Q(G)$ for an RPQ Q over G (in our example $(v_1, v_2) \in a.b.c(G)$). Moreover, all this can be decided in P under combined complexity and in NL under data-complexity [46].

To check whether edge (v, ℓ, v') is contained in a problematic path, we proceed as follows. For each node u of G, for each constraint $Q_1 \subseteq Q_2$ in Γ, we use the above-described procedure to compute the set S of all nodes u' such that $(u, u') \in Q_1(G)$ and (v, ℓ, v') belongs to a path from u to u' in G. To this end, we traverse the intersection graph $I(G, A_{Q_1})$ by first finding the paths from u to v, then advancing to the appropriate states of the intersection graph via ℓ and v', and then continuing from v' (and the appropriate states) to other reachable nodes by ensuring that u' is reached if possible. Then, for each pair $(u, u') \in S$, we check whether $(u, u') \notin Q_2(G)$. If so, then there is a path $\pi \in \mathsf{ProblematicPaths}(G)$ such that $(v, \ell, v') \in \pi$. Again, all this can be done in P under combined complexity and in NL under data-complexity. □

The result of Lemma 4 entails that checking whether a given label or vertex is contained in a problematic path is tractable as well. Hence, we obtain the following result.

Theorem 2. *For any GDB G and inconsistency measure $I \in \{I_E, I_L, I_V\}$, it holds that $\mathbf{FIM}_I(G)$ is in FNL and $\mathbf{IM}_I(G, \Gamma)$ is in FP.*

Proof. For each edge (label, vertex) of G, we check whether there is a problematic path containing it by using the result of Lemma 4, from which the statement follows. □

Thus, we also have that, for $I \in \{I_E, I_L, I_V\}$, $\mathbf{LV}_I$, $\mathbf{UV}_I$, and $\mathbf{EV}_I$, are in P, and that $\mathbf{FLV}_I$, $\mathbf{FUV}_I$, and $\mathbf{FEV}_I$ are in NL.

5.2. Intractable measures

In this section, we present the results for the inconsistency measures that turn out to be intractable. We present our results by considering the inconsistency measures in ascending level of complexity. In particular, we start with the measures whose complexity turns out to be on the first level of the polynomial hierarchy, that is I_{E-}, I_{V-} (Section 5.2.1), and I_{E+} (Section 5.2.2), and then continue with the complexity results for I_S (Section 5.2.3) and I_M (Section 5.2.4) using classes from the counting polynomial hierarchy.

5.2.1. Measures counting the number of edge and vertex deletions

We start the complexity analysis of the intractable measures by the two measures that count the minimal number of edges and vertices that need to be removed to restore consistency, namely I_{E-} and I_{V-}.

Theorem 3. *Let $G = (V, E)$ be a GDB. For $I \in \{I_{E-}, I_{V-}\}$, it holds that $\mathbf{UV}_I(G, \Gamma, v)$ is in NP and $\mathbf{FUV}_I(G, v)$ is NP-hard.*

Proof. (Membership results). We first consider the measure I_{E-}. A guess-and-check strategy for deciding whether $I_{E-}(G) \leq \nu$ is as follows. First, guess a set of edges E_ν such that $E_\nu \subseteq E$ and $|E_\nu| \leq \nu$, and then check in polynomial time that $(G - E_\nu) \models \Gamma$ (this is equivalent to deciding whether $\mathbf{UV}_{I_B}(G - E_\nu, \Gamma, 0)$ is true, which is in P). This strategy is complete since I_{E-} is monotonic and if there is such an E_ν then there is an E_m such that $E_m \subseteq E_\nu$, $(G - E_m) \models \Gamma$, and $|E_m| = min\{|E'| \mid (G - E') \models \Gamma\}$, meaning that $I_{E-}(G) = |E_m| \leq \nu$.

The membership result for $\mathbf{UV}_{I_{V-}}$ can be proved by reasoning as above, except that we need to guess a set of nodes V_ν such that $V_\nu \subseteq V$ and $|V_\nu| \leq \nu$, and then check in polynomial time that $(G - V_\nu) \models \Gamma$.

(Hardness result for $\mathbf{FUV}_{I_{E-}}$). We show a reduction from 3-COLORABILITY to $\mathbf{FUV}_{I_{E-}}(G, \nu)$. An instance of 3-COLORABILITY consists of a graph $H = \langle N, F \rangle$, where N is a set of nodes and F is a set of undirected edges. A 3-COLORABILITY instance is *true* iff there is a total function $f : N \to \{r, g, b\}$ (*red, green, blue*) such that $f(n_i) \neq f(n_j)$ whenever $(n_i, n_j) \in F$. Function f is said to be a 3-coloring function for H.

Let $H = \langle N, F \rangle$ be a 3-COLORABILITY instance. We construct an instance of $\mathbf{FUV}_{I_{E-}}(G, \nu)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{e, r, g, b, z\}$ such that $V = N$ and $E = \{\{(u, e, v), (v, e, u)\} \mid \{u, v\} \in F\} \cup \{\{(u, r, u), (u, g, u), (u, b, u)\} \mid u \in N\}$. Thus G contains a pair of directed edges labeled with e for each undirected edge in F plus three self-loops labeled with r, g and b, respectively, for each node in N.
- Γ is a set of RPCs consisting of the following six constraints:
  1) $r.b \subseteq z$, $r.g \subseteq z$, $b.g \subseteq z$, that is, each node cannot have more than one self-loop labeled with a color. Indeed, the evaluation of RPQ z is $z(G) = \emptyset$, while the evaluations of the left-hand side RPQs in these constraints are the sets of pairs (u, u) such that there is a path $(u, c_1, u)(u, c_2, u)$ for the specified pairs of colors $c_1, c_2 \in \{r, g, b\}$. Thus these constraints are satisfied if and only if each node has at most one self-loop labeled with $c \in \{r, g, b\}$.
  2) $r.e.r \subseteq z$, $b.e.b \subseteq z$, $g.e.g \subseteq z$, that is, there cannot be two nodes having self-loops labeled with the same color $c \in \{r, g, b\}$ and connected by an edge. Since the evaluation of the RPQ z is $z(G) = \emptyset$, the constraints entail that the sets of pairs (u, v) for which there is a path $(u, c, u)(u, e, v)(v, c, v)$ with $c \in \{r, g, b\}$ must be empty.
- Finally, $\nu = 2|V|$.

($\Rightarrow$) Let f be a 3-coloring function for H. We show that $\mathbf{FUV}_{I_{E-}}(G, \nu)$ is true. Let R be the set of self-loops corresponding to colors *not* assigned to nodes by f, that is, $R = \{(u, c, u) \mid f(u) \neq c\}$. Since f is a 3-coloring function, we have that $G - R \models \Gamma$. Moreover, $|R|$ is the minimal number of edges to be removed to restore consistency, since each node has 3 self-loops in G and two of them need to be removed to satisfy the first group of constraints in Γ. Thus $I_{E-}(G) = 2|V|$, meaning that $\mathbf{FUV}_{I_{E-}}(G, \nu)$ is true.

($\Leftarrow$) We now show that if $\mathbf{FUV}_{I_{E-}}(G, \nu)$ is true, and thus $I_{E-}(G) \leq 2|V|$, then there is a 3-coloring function for H. Let E' be a set of edges such that $I_{E-}(G) = min\{|E'| \mid G - E' \text{ is consistent}\}$. Since each node has 3 self-loops in G and two of them need to be removed to satisfy the first group of constraints in Γ, it follows that $|E'| \geq 2|V|$. Therefore $I_{E-}(G) = 2|V|$, meaning that no other edge needs to be removed to get consistency. Hence, a 3-coloring function for H is $f(u) = c$ for each $(u, c, u) \in (E \setminus E')$. This is a 3-coloring function for H since it associates each node of H with a color (f is total), and since $G - E'$ is consistent w.r.t. Γ, and in particular the second group of constraints of Γ is satisfied, it follows that there cannot be two nodes associated with the same color $c \in \{r, g, b\}$ and connected by an edge in G. This in turns means that no nodes connected by an edge in H are associated with the same color by the function f.

(Hardness result for $\mathbf{FUV}_{I_{V-}}$). We show that 3-COLORABILITY can be also reduced to $\mathbf{FUV}_{I_{V-}}(G, \nu)$. Let $H = \langle N, F \rangle$ be a 3-COLORABILITY instance. We construct an instance of $\mathbf{FUV}_{I_{V-}}(G, \nu)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{e, d, z\}$ such that V contains a vertex vc for each $v \in N$ and $c \in \{r, g, b\}$, and $E = \{\{(vr, d, vg), (vg, d, vb), (vb, d, vr)\} \mid v \in N\} \cup \{\{(ur, e, vr), (ug, e, vg), (ub, e, vb)\} \mid \{u, v\} \in F\}$. Thus, G contains three vertices, namely vr, vg, vb, for each vertex $v \in N$. These vertices form a cycle of edges labeled as d. Moreover, for each edge (u, v) in F, there is an edge labeled with e between uc and vc for each color $c \in \{r, g, b\}$.
- Γ is a set of RPCs consisting of the following two constraints: $d \subseteq z$ and $e \subseteq z$, stating that there cannot be pairs of nodes connected by an edge labeled as d or e. That is, for a given set $V' \subseteq V$, we have that $G' = G - V'$ is consistent w.r.t. Γ iff no edge is in G'.
- Finally, $\nu = 2|V|$.

($\Rightarrow$) Let f be a 3-coloring function for H. We show that $\mathbf{FUV}_{I_{V-}}(G, \nu)$ is true. Let R be the set of vertices of G corresponding to the pairs of the form (vertex of F, color *not* assigned to such vertex) by f, that is, $R = \{vc \mid f(v) \neq c\}$. Since f is a 3-coloring function, and thus $2|V|$ vertices and their adjacent edges are removed, we have that $G - R$ consists of isolated vertices only and thus $(G - R) \models \Gamma$. Moreover, $|R|$ is the minimal number of vertices that need to be removed to restore consistency. Thus $I_{V-}(G) = 2|V|$, meaning that $\mathbf{FUV}_{I_{V-}}(G, \nu)$ is true.

($\Leftarrow$) We now show that if $\mathbf{FUV}_{I_{V-}}(G, \nu)$ is true, and thus $I_{V-}(G) \leq 2|V|$, then there is a 3-coloring function for H. Let V' be a set of vertices such that $I_{V-}(G) = min\{|V'| \mid G - V' \text{ is consistent}\}$. Since $(G - V') \models \Gamma$, we have that no edge is in $(G - V')$. This can be achieved by removing at least $2|V|$ vertices, that is, $|V'| \geq 2|V|$. Therefore $I_{V-}(G) = 2|V|$, meaning that no other vertex needs to be removed to get consistency. Hence, a 3-coloring function for H is $f(v) = c$ for each $vc \in (V \setminus V')$. This is a 3-coloring function for H since it associates each node of H with a color (f is total), and since $G - V'$ is consistent w.r.t. Γ, it follows that there cannot be

two nodes associated with the same color $c \in \{r, g, b\}$ and connected by an edge in G. This in turns means that no nodes connected by an edge in H are associated with the same color by f. □

The following corollary follows from Theorem 3 and previous results stated in Lemmas 1, 2, and 3.

Corollary 1. *For any GDB G and IM $I \in \{I_{E-}, I_{V-}\}$, $\mathbf{FUV}_I(G, v)$ and $\mathbf{UV}_I(G, \Gamma, v)$ are NP-complete; $\mathbf{FLV}_I(G, v)$ and $\mathbf{LV}_I(G, \Gamma, v)$ are coNP-complete; $\mathbf{FEV}_I(G, v)$ and $\mathbf{EV}_I(G, \Gamma, v)$ are in D^p; and $\mathbf{FIM}_I(G)$ and $\mathbf{IM}_I(G, \Gamma)$ are in $FP^{NP[\log n]}$.*

Proof. From the results of Theorem 3 and Lemma 3, it follows that $\mathbf{FUV}_I$ and $\mathbf{UV}_I$ are NP-complete for $I \in \{I_{E-}, I_{V-}\}$. Then, using Lemma 1, it follows that $\mathbf{FLV}_I$ and $\mathbf{LV}_I$ are $coNP$-complete for $I \in \{I_{E-}, I_{V-}\}$. Hence, from these results and Lemma 1, we have that $\mathbf{EV}_I$ (and $\mathbf{FEV}_I$) is in D^p for $I \in \{I_{E-}, I_{V-}\}$. Moreover, since $I_{E-}(G) \in \{0, 1, \ldots, |E|\}$ and $I_{V-}(G) \in \{0, 1, |V|\}$, and thus $|I[\mathbf{G}_n]|$ is $O(n)$ for $I \in \{I_{E-}, I_{V-}\}$, using Lemma 2, we have that computing $I_{E-}(G)$ and $I_{V-}(G)$ is in $FP^{NP[\log n]}$. □

The characterization of the complexity of $\mathbf{FEV}_I$ (and thus of $\mathbf{EV}_I$, cf. Lemma 3) for $I \in \{I_{E-}, I_{V-}\}$, is strengthened by the following theorem where it is shown that $\mathbf{FEV}_I$ is D^p-hard. Thus, it follows that $\mathbf{FEV}_I$ and $\mathbf{EV}_I$ with $I \in \{I_{E-}, I_{V-}\}$ are complete for D^p.

Theorem 4. *For any GDB G and IM $I \in \{I_{E-}, I_{V-}\}$, $\mathbf{FEV}_I(G, v)$ is D^p-hard.*

Proof. We show reductions to our problems from the D^p-hard problem Exact Vertex Cover [68], which is defined as follows. Given a graph $H = \langle N, F \rangle$, where N is a set of nodes and F is a set of (undirected) edges, and a number K, decide whether a minimal vertex cover for H has size exactly K. A vertex cover of size K for H is a subset $N' \subseteq N$ with $|N'| = K$ such that for each edge $(u, w) \in F$ at least one of u or w belongs to N'. A vertex cover of size K for H is minimal if there is no vertex cover of size $K' < K$ for H.

(Hardness result for $\mathbf{FEV}_{I_{E-}}$). Given $H = \langle N, F \rangle$ and K, we construct an instance of $\mathbf{FEV}_{I_{E-}}(G, v)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{e, f, g, h, z\}$ such that $V = N$ and $E = \{\{(u, e, v), (u, f, v), (u, g, v)\} \mid \{u, v\} \in F\} \cup \{\{(u, h, u)\} \mid u \in N\}$. Thus, G contains three edges labeled with e, f, and g, respectively, for each edge in F, plus a self-loop labeled with h for each node in N.
- $\Gamma = \{h.e.h \subseteq z, h.f.h \subseteq z, h.g.h \subseteq z\}$, that is, there cannot be two nodes having self-loops labeled with h and connected by an edge labeled with either e, f or g.
- Finally, $v = K$.

We show that K is the size of a minimal vertex cover for H iff $\mathbf{FEV}_{I_{E-}}(G, v)$ is true.

(⇒) Let N' be a minimal vertex cover for H, with $|N'| = K$. Let $R = \{(u, h, u) \mid u \in N'\}$. Thus R consists of a self-loop labeled with h for each vertex belonging to the minimal vertex cover. That is, each vertex in $G - R$ has a self-loop labeled with h only if it does not belong to N'. Moreover, since for each edge $(u, \ell, w) \in G$ with $\ell \in \{e, f, g\}$ at least one of u or w belongs to N', it follows that in $G - R$ there are no vertices having self-loops labeled with h and connected by an edge labeled with ℓ; otherwise the edge in H corresponding to such edge in G would not be covered by the vertex cover. Therefore, $G - R \vDash \Gamma$. Finally, we show that $|R|$ is the minimal number of edges to be removed to restore consistency. In fact, for each pair of vertices u, w in G, removing one of the self-loops of u or w is minimal w.r.t. removing all three edges $(u, \ell, w) \in G$ with $\ell \in \{e, f, g\}$ in order to locally restore consistency w.r.t. Γ. Moreover, removing the self-loops in R corresponding to the minimal vertex cover is necessary to avoid the presence in $G - R$ of a pair of connected vertices both having a self-loop.

(⇐) If $\mathbf{FEV}_{I_{E-}}(G, v)$ is true, then the minimal number of edges to be removed to restore consistency is K. As mentioned earlier, the cardinality-minimal set of edges to be removed consists of self-loops only, as removing the other kinds of edges requires three removals for each pair of vertices, instead of only one. Let R be a set of self-loops such that $I_{E-}(G) = |R| = K$. Let $N' = \{v \mid (v, h, v) \in R\}$. Then N' is a vertex cover for H since for each $(u, \ell, w) \in G$ with $\ell \in \{e, f, g\}$ (and thus for each edge of H) at least one of u or w belongs to N'. Moreover, the size of N' is minimal as so is that of R, from which it follows that K is the size of a minimal vertex cover for H.

(Hardness result for $\mathbf{FEV}_{I_{V-}}$). Given $H = \langle N, F \rangle$ and K, we construct an instance of $\mathbf{FEV}_{I_{V-}}(G, v)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{e, z\}$ such that $V = N$ and $E = \{\{(u, e, v)\} \mid \{u, v\} \in F\}$.
- $\Gamma = \{e \subseteq z\}$. Since the evaluation of the RPQ z is $z(G) = \emptyset$, the constraint entails that the sets of pairs (u, v) such that there is an edge (u, e, v) between them must be empty. That is, there cannot be two nodes connected by an edge in G.
- Finally, $v = K$.

We show that K is the size of a minimal vertex cover for H iff $\mathbf{FEV}_{I_{V-}}(G, v)$ is true.

(⇒) Let N' be a minimal vertex cover for H, with $|N'| = K$. Since for each edge $(u, w) \in F$ at least one of u or w belongs to N', it follows that $G - N'$ is consistent, as removing the vertices in N' is sufficient to remove all edges of G. Moreover, $|N'|$ is the minimal

number of vertices to be removed to restore consistency, as otherwise N' would not be minimal. Therefore $I_{V-}(G) = K$, meaning that $\mathbf{FEV}_{I_{V-}}(G, v)$ is true.

($\Leftarrow$) If $\mathbf{FEV}_{I_{V-}}(G, v)$ is true, then the minimal number of vertices to be removed to restore consistency is K. As there is a problematic pair for each edge, this means that K is the minimal number of vertices that are incident to all edges, that is, the size of a minimal vertex cover for H. □

Corollary 2. *For any GDB G and IM $I \in \{I_{E-}, I_{V-}\}$, i) $\mathbf{FEV}_I(G, v)$ and $\mathbf{EV}_I(G, \Gamma, v)$ are D^p-complete; and ii) $\mathbf{FIM}_I(G)$ and $\mathbf{IM}_I(G, \Gamma)$ are $FP^{NP[\log n]}$-complete.*

Proof. Item *i*) follows from Theorem 4 and Corollary 1. As for Item *ii*), we can show a reduction from the $FP^{NP[log\ n]}$-complete problem MIN-VERTEX COVER [56], which is defined as follows. Given a graph H, compute the size of a minimal vertex cover for H. Hence, the statement follows by using the reductions in the proof of Theorem 4, where it is shown that K is the size of a minimal vertex cover for H iff $I(G) = K$ for $I \in \{I_{E-}, I_{V-}\}$, and Lemma 3. □

We conclude our analysis of the complexity of the two measures that count the minimal number of edges and vertices that need to be removed to restore consistency by observing that they are intractable even in the presence of word constraints only.

Proposition 4. *For IM $I \in \{I_{E-}, I_{V-}\}$, all the hardness results in Table 2 still hold if the set of integrity constraints consists of word constraints only.*

Proof. The result follows from the fact that only RPCs of the form $Q_1 \subseteq Q_2$ in which both Q_1 and Q_2 are words, that is sequences of labels, are used in the hardness proofs of Theorems 3 and 4 and in all the other hardness results concerning I_{E-} and I_{V-} (as only the constructions given in the hardness proofs of Theorems 3 and 4 are exploited). □

5.2.2. Measure counting the number of edge additions

Next we consider the single IM that counts the minimal number of edges of the graph that need to be added to restore consistency, I_{E+}. It turns out that the complexity of this measure is the same as the ones counting the minimal number of deletions.

Theorem 5. *For any GDB $G = (V, E)$ over Σ, it holds that $\mathbf{UV}_{I_{E+}}(G, \Gamma, v)$ is in NP and $\mathbf{FUV}_{I_{E+}}(G, v)$ is NP-hard.*

Proof. (Membership). A guess-and-check strategy for deciding whether $I_{E+}(G) \leq v$ is as follows. First, guess a set of edges E_v whose vertices are in V and whose labels are in Σ and such that $|E_v| \leq v$, and then check in polynomial time that $(G + E_v) \models \Gamma$ (the latter is equivalent to deciding whether $\mathbf{UV}_{I_B}(G + E_v, \Gamma, 0)$ is true, which is in P). Observe that the number of possible edges in E_v is bounded by $|\Sigma||V|^2$, which is polynomial w.r.t. the size of the input. Moreover, this strategy is complete since I_{E+} is monotonic and if there is such an E_v then there is an E_m such that $E_m \subseteq E_v$, $(G + E_m) \models \Gamma$, and $|E_m| = min\{|E'| \mid (G + E') \models \Gamma\}$, meaning that $I_{E+}(G) = |E_m| \leq v$.

(Hardness). We show a reduction from 3-COLORABILITY to $\mathbf{FUV}_{I_{E+}}(G, v)$. Recall that an instance of 3-COLORABILITY consists of a graph $H = \langle N, F \rangle$, where N is a set of nodes and F is a set of undirected edges. A 3-COLORABILITY instance is *true* iff there is a total function $f : N \to \{r, g, b\}$ such that $f(n_i) \neq f(n_j)$ whenever $(n_i, n_j) \in F$. (The function f is said to be a 3-coloring function for H.)

Let $H = \langle N, F \rangle$ be a 3-COLORABILITY instance. We construct an instance of $\mathbf{FUV}_{I_{E+}}(G, v)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{e, r, g, b, r', g', b', n, z\}$ such that $V = N \cup \{c\}$ and $E = \{\{(u, e, v), (v, e, u)\} \mid \{u, v\} \in F\} \cup \{\{(c, n, v)\} \mid v \in N\}$. Thus, G contains a pair of directed edges labeled with e for each undirected edge in F plus an edge labeled with n from the new vertex c to each node in N.
- Γ is a set of RPCs consisting of the following eight constraints:
  1) $n \subseteq r|g|b$, $r|g|b \subseteq n$ that is, there must be at least an edge labeled with a color between the vertex c and each vertex in V. Indeed, the evaluation of the RPQ n is $n(G) = \{(c, v) \mid v \in V\}$, and the constraints together imply that $r(G) \cup g(G) \cup b(G) = n(G)$. Thus, these constraints are satisfied if each vertex can be reached from c by means of an edge labeled with a color in $\{r, g, b\}$.
  2) $r \subseteq r.r'.r$, $g \subseteq g.g'.g$, $b \subseteq b.b'.b$, that is, for each color $l \in \{r, g, b\}$, if there is an edge labeled with l, then there must be an inverse edge labeled with l'. Indeed, for each color $l \in \{r, g, b\}$, the constraints of item 1) entail that the evaluation of the RPQ l must be a set of pairs such that $l(G) \subseteq \{(c, v) \mid v \in V\}$. Thus, the new constraints imply that for each pair $(c, v) \in l(G)$, there must be a path $(c, l, v)(v, l', c)(c, l, u)$; observe that, although u and v are not necessarily the same node, if an edge (c, l, u) exists, then (c, u) belongs to $l(G)$ as well. Therefore, these constraints are satisfied if for each vertex v that can be reached from c by means of an edge labeled with a color $l \in \{r, g, b\}$, there is an edge labeled with l' in the opposite direction, that is, from v to c.
  3) $r.e.r' \subseteq z$, $g.e.g' \subseteq z$, $b.e.b' \subseteq z$, that is, there cannot be two nodes having an incoming and outgoing edge labeled with l and l', respectively, where l is a color in $\{r, g, b\}$ and connected by an edge. Since the evaluation of the RPQ z is $z(G) = \emptyset$, the constraints entail that the sets of pairs (u, v) such that there is a path $(c, l, u)(u, e, v)(v, l', c)$ with $l \in \{r, g, b\}$ must be empty.
- Finally, $v = 2|V|$.

($\Rightarrow$) Let f be a 3-coloring function for H. We show that $\mathbf{FUV}_{I_{E+}}(G, v)$ is true. Let $A = \{(c, l, u) \mid f(u) = l\} \cup \{(u, l', c) \mid f(u) = l\}$. Since f is a 3-coloring function, we have that $G + A \models \Gamma$. Moreover, $|A|$ is the minimal number of edges to be added to restore consistency, since at least an edge labeled with a color $l \in \{r, g, b\}$ from the vertex c to each vertex in V needs to be added to satisfy the first two constraints in Γ, and for each added edge, an additional edge in the opposite direction labeled with $l' \in \{r, g, b\}$ must be added to satisfy the constraints in item 2). Thus $I_{E+}(G) = 2|V|$, meaning that $\mathbf{FUV}_{I_{E+}}(G, v)$ is true.

($\Leftarrow$) We now show that if $\mathbf{FUV}_{I_{E+}}(G, v)$ is true, and thus $I_{E+}(G) \leq 2|V|$, then there is a 3-coloring function for H. Let E' be the set of edges such that $I_{E+}(G) = min\{|E'| \mid G + E' \text{ is consistent}\}$. Since at least one edge (c, l, u) with $l \in \{r, g, b\}$ for each $u \in V$ needs to be added to satisfy the first two constraints in Γ, and for each of them a corresponding inverse edge (u, l', c) must be added to satisfy the constraints in item 2), it follows that $|E'| \geq 2|V|$. Therefore $I_{E+}(G) = 2|V|$, meaning that no other edge needs to be added to get consistency. Hence, a 3-coloring function for H is $f(u) = l$ for each $(c, l, u) \in E'$. This is a 3-coloring function for H since it associates each node of H with a color (f is total), and since $G + E'$ is consistent w.r.t. Γ, and in particular the third group of constraints of Γ is satisfied, it follows that there cannot be two nodes associated with the same color in $\{r, g, b\}$ and connected by an edge in G. This in turns means that no nodes connected by an edge in H are associated with the same color by the function f. □

The following is a consequence of Theorem 5 and previous lemmas.

Corollary 3. *For any GDB G, $\mathbf{FUV}_{I_{E+}}(G, v)$ and $\mathbf{UV}_{I_{E+}}(G, \Gamma, v)$ are NP-complete; moreover, $\mathbf{FLV}_{I_{E+}}(G, v)$ and $\mathbf{LV}_{I_{E+}}(G, \Gamma, v)$ are coNP-complete; $\mathbf{FEV}_{I_{E+}}(G, v)$ and $\mathbf{EV}_{I_{E+}}(G, \Gamma, v)$ are in D^p; and $\mathbf{FIM}_{I_{E+}}(G)$ and $\mathbf{IM}_{I_{E+}}(G, \Gamma)$ are in $FP^{NP[\log n]}$.*

Proof. From the results of Theorem 5 and Lemma 3, it follows that $\mathbf{FUV}_{I_{E+}}$ and $\mathbf{UV}_{I_{E+}}$ are NP-complete. Moreover, using Lemma 1, it follows that $\mathbf{FLV}_{I_{E+}}$ and $\mathbf{LV}_{I_{E+}}$ are $coNP$-complete. Hence, from these results and Lemma 1, we have that $\mathbf{EV}_{I_{E+}}$ and $\mathbf{FEV}_{I_{E+}}$ are in D^p. Finally, since $I_{E+}(G) \in \{0, 1, \ldots, |\Sigma||V|^2\}$, and thus $|I_{E+}[\mathbf{G}_n]|$ is $O(f(n))$, with $f(n) = |\Sigma||V|^2$ which is polynomial in the input size, using Lemma 2, we have that computing $I_{E+}(G)$ is in $FP^{NP[\log n]}$. □

The next theorem shows that $\mathbf{FEV}_{I_{E+}}$ is D^p-hard, which, together with the previous results, entails that $\mathbf{FEV}_{I_{E+}}$ and $\mathbf{EV}_{I_{E+}}$ are D^p-complete.

Theorem 6. *For any GDB G, $\mathbf{FEV}_{I_{E+}}(G, v)$ is D^p-hard.*

Proof. We show a reduction to our problem from the D^p-hard problem Exact Vertex Cover [68], which is recalled in what follows. Given a graph $H = \langle N, F \rangle$, where N is a set of nodes and F is a set of (undirected) edges, and a number K, decide whether a minimal vertex cover for H has size exactly K. A vertex cover of size K for H is a subset $N' \subseteq N$ with $|N'| = K$ such that for each edge $(u, w) \in F$ at least one of u or w belongs to N'. A vertex cover of size K for H is minimal if there is no vertex cover of size $K' < K$ for H.

Given $H = \langle N, F \rangle$ and K, we construct an instance of $\mathbf{FEV}_{I_{E+}}(G, v)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{a, a', b, c, e\}$ such that $V = N \cup \{s\}$, where s is a new vertex, and $E = \{(u, e, v), (v, e, u)\} \mid \{u, v\} \in F\} \cup \{(v, b, s), (s, c, v) \mid v \in N\}$. Thus, G contains a pair of edges for each undirected edge in H, and a new vertex s with a pair of edges toward and from it labeled with b and c, respectively, to/from any vertex in H. Observe that there is no edge labeled with a or a' in G.
- $\Gamma = \{a \subseteq b, a' \subseteq c, a' \subseteq a'.a.a', e \subseteq (a.c)|(b.a')\}$. The first constraint in Γ requires that the pairs of vertices connected by an edge labeled with a is a subset of those connected by an edge labeled with b, that is $a(G) \subseteq b(G) = \{(v, s) \mid v \in N\}$. The second constraint imposes that $a'(G) \subseteq c(G) = \{(s, v) \mid v \in N\}$. The constraint $a' \subseteq a'.a.a'$ together with the previous ones entails that if there is an edge (s, a', v) then there must be a path $(s, a', v)(v, a, s)(v, a', u)$, which implies that there must be (v, a, s). Finally, the last constraint in Γ requires that the pairs of vertices connected by an edge of H is a subset of those connected by a path consisting of a first edge labeled with a and a second one labeled with c, or by a path consisting of a first edge labeled with b and a second one labeled with a'.
- Finally, $v = 2K$.

We show that K is the size of a minimal vertex cover for H iff $\mathbf{FEV}_{I_{E+}}(G, v)$ is true.

($\Rightarrow$) Let N' be a minimal vertex cover for H, with $|N'| = K$. Let $A = \{(v, a, s) \mid v \in N'\} \cup \{(s, a', v) \mid v \in N'\}$. That is, A consists of an a-labeled edge and an a'-labeled edge for each vertex belonging to the minimal vertex cover. We now show that $G + A \models \Gamma$. It is easy to see that $G + A$ is consistent w.r.t. the first two constraints in Γ, since $a(G) \subseteq b(G)$ and $a'(G) \subseteq c(G)$. Moreover, the third constraint is satisfied since there is an edge (s, a', v) iff there is (v, a, s). Finally, since N' consists of a set of vertices such that for each edge $(u, w) \in F$ at least one of u or w belongs to N', it is the case that the set of pairs (v_1, v_2) such that there is a path from v_1 to v_2 consisting of a first edge labeled with a and a second one labeled with c, or the first edge labeled with b and a second one labeled with a', covers all the edges of the graph. That is, also the last constraint in Γ is satisfied. Observe that achieving consistency by

adding fewer a-labeled edges would contradict the fact that N' is a minimal vertex cover for H. Therefore, since $|A| = 2|N'| = 2K$ is minimal, it follows that $I_{E+}(G) = v$.

($\Leftarrow$) If $\mathbf{FEV}_{I_{E+}}(G, v)$ is true, then the minimal number of edges to be added to restore consistency is K. The cardinality-minimal set of edges to be added consists of a-labeled and a'-labeled edges only, as adding any the other kind of edges does not suffice to get consistency. Let $A \subseteq \{(v, a, s), (s, a', v) \mid v \in V\}$ be a set of $2K$ edges such that $G + A \models \Gamma$. Let $N' = \{v \mid (v, a, s) \in A\}$. Then N' is a vertex cover for H since for each edge $(u, e, w) \in G$ (and thus for each edge (u, w) of H) at least one of u or w belongs to N' (this is entailed by the satisfaction of the fourth constraint in Γ). Finally, the size of N' is minimal since the size of E is minimal. Thus, K is the size of a minimal vertex cover for H. □

Corollary 4. *For any GDB G, i) $\mathbf{FEV}_{I_{E+}}(G, v)$ and $\mathbf{EV}_{I_{E+}}(G, \Gamma, v)$ are D^p-complete; and ii) $\mathbf{FIM}_{I_{E+}}(G)$ and $\mathbf{IM}_{I_{E+}}(G, \Gamma)$ are $FP^{NP[\log n]}$-complete.*

Proof. Item *i*) follows from Theorem 6 and Corollary 3. As for Item *ii*), we can show a reduction from the $FP^{NP[log\ n]}$-complete problem MIN-VERTEX COVER [56]. The statement follows by using the reductions in the proof of Theorem 6 and Lemma 3. □

Before concluding this section, we point out that all the hardness results for the measure I_{E+} hold for non-recursive RPCs, that is, even in the presence of RPCs where both the left- and right-hand side queries do not mention the Kleene-star operator.

5.2.3. Measure counting the minimal problematic paths

We show that the last two measures under consideration, counting the number of minimal problematic paths, I_S, and counting the number of MIMSs, I_M, have the same complexity. We start with I_S, and in particular with the following lemma that will be useful to characterize its complexity.

Lemma 5. *Given a path π in a GDB G, deciding whether $\pi \in$ MinimalProblematicPaths(G) is in P under combined complexity and in NL under data-complexity.*

Proof. Recall that checking whether a labeled graph contains a directed path π' from a node x to a node y such that the string of labels of π' satisfies a given regular expression is tractable [67]. As discussed in the proof of Lemma 4, the algorithm in [67] relies on constructing the intersection graph $I(G, A_Q)$ of the input graph G and non-deterministic finite automaton A_Q accepting the language of the given regular expression Q.

To check whether π is a minimal problematic path, we first look for a constraint C such that π is a minimal problematic path w.r.t. G and C considered in isolation, and then we consider any other constraint and check that there is no path π' which is a proper subsequence of π and is problematic.

Let u and v be the first and last nodes of π. For each constraint $C = Q_1 \subseteq Q_2$ in Γ, we use the above-mentioned procedure to check if there is a path from u to v in G such that $\lambda(\pi)$ satisfies Q_1, that is $(u, v) \in Q_1(G)$, and during the traversal of the intersection graph $I(G, A_{Q_1})$, we check whether there is a node c visited twice (note that c is a node of the intersection graph). If there is such a node then π cannot be minimal since removing the subsequence between the node in G corresponding to c and itself would result in a subsequence of π such that $(u, v) \in Q_1(G)$. Otherwise, we check whether $(u, v) \notin Q_2(G)$ by using the intersection graph $I(G, A_{Q_2})$. If so, (u, v) is problematic and is π is a minimal problematic path w.r.t. G and C considered in isolation.

We now consider any other constraint $C' = Q'_1 \subseteq Q'_2$ in Γ and check that there is no path π' which is a proper subsequence of π and is problematic. Thus, using the intersection graphs $I(G, A_{Q'_1})$ and $I(G, A_{Q'_2})$, for each node x in π we check the following conditions:

1) there is a subsequence π_x of π from node x to node y in G such that $\lambda(\pi_x)$ satisfies Q'_1, that is, there is a path in $I(G, A_{Q'_1})$ from the node (x, q_0), where q_0 is the initial state of $A_{Q'_1}$, to a node in $\{(y, q_{f_1}), \ldots, (y, q_{f_k})\}$, where $q_{f_1}, \ldots, q_{f_k}$ are the final states of $A_{Q'_1}$; if such path π_x exists, then $(x, y) \in Q'_1(G)$;
2) $(x, y) \notin Q'_2(G)$, by using the intersection graph $I(G, A_{Q'_2})$.

If one of these two conditions is false, then π_x is not a problematic path which is proper subsequence of π. If this holds for any constraint C' in Γ, then there is no path π' which is a proper subsequence of π and is problematic. Thus, π is a minimal problematic path since we have shown that there is a constraint C such that π is a minimal problematic path w.r.t. G and C considered in isolation, and there is no path π' which is a proper subsequence of π and is problematic. All this can be decided in P under combined complexity and in NL under data-complexity since we need to repeat the procedure for checking whether a labeled graph contains a directed path between a given pairs of nodes satisfying a given regular expression (in P under combined complexity and in NL under data-complexity [46]) a polynomial number of times w.r.t. the sizes of G and Γ, once for each pair of nodes checked. □

We are now ready to state the complexity of computing the value of I_S.

Theorem 7. *For any GDB G, $\mathbf{FIM}_{I_S}(G)$ and $\mathbf{IM}_{I_S}(G, \Gamma)$ are #P-complete.*

Proof. (Membership). We show that membership is in $\#P$ for $\mathbf{IM}_{I_S}(G,\Gamma)$, from which the result for $\mathbf{FIM}_{I_S}(G)$ follows. In fact, $\mathbf{IM}_{I_S}(G,\Gamma)$ can be computed by counting the number of accepting paths of a non-deterministic polynomial-time Turing machine M such that (*i*) M non-deterministically guesses a sequence π of edges of G such that each leaf of the resulting computation tree is a path of G; (*ii*) then, M checks in polynomial time that $\pi \in \mathsf{MinimalProblematicPaths(G)}$ (cf. Lemma 5).
(Hardness). We show a reduction from the $\#P$-complete problem S-T PATHS [69] that, given a (directed) graph $H = \langle N, F\rangle$ and two vertices s and t in N, asks for the number of paths from s to t that visit every node at most once, that is, the number of *simple* paths from s to t. Given $H = \langle N, F\rangle$ and $s, t \in N$, we construct an instance of $\mathbf{FIM}_{I_S}(G)$ as follows.

- $G = (V, E)$ is a GDB over $\Sigma = \{a, b, e, z\}$ such that $V = N \cup \{v_a, v_b\}$ and $E = \{\{(u, e, v)\} \mid \{u, v\} \in F\} \cup \{(v_a, a, s), (t, b, v_b)\}$.
- $\Gamma = \{a.e^*.b \subseteq z\}$. Since $z(G) = \emptyset$, the constraint states that there cannot be a pair $(u, v) \in V$ such that there is a path between u and v where the first edge has label a, the last edge has label b, and all intermediate edges have label e.

Therefore, the constraint in Γ entails that there cannot be any path in G connecting s and t since v_a and v_b are connected with these nodes only and (v_a, v_b) is the only candidate for the answer to the RPQ on the left side of the constraint. This means that, if there is a path from s to t in H, then (v_a, v_b) is a problematic pair of G, and every s-t path in H one-to-one corresponds to a problematic path of G.

Given this, we now show that the number of paths in H from s to t that visit every node at most once is equal to the number of minimal problematic paths of G, that is $I_S(G)$, from which it will follow that $\mathbf{FIM}_{I_S}(G)$ is $\#P$-hard.

First, observe that if a problematic path is not minimal, and thus contains one or more sequences of edges that can be removed so that the remaining ones still result in a problematic path, then each sequence that can be removed corresponds to a cycle, that is, some vertex is visited twice or more. Consider a simple s-t path π in H, then there is a corresponding simple v_a-v_b path π' in G (obtained by expanding π with the edges in $\{(v_a, a, s), (t, b, v_b)\}$). Analogously to π, every node in π' may be visited at most once. Then, it follows that there cannot be a v_a-v_b path in G which is a proper subsequence of π'; that is, π' is a minimal problematic path. On the other hand, suppose now that π'' is minimal problematic v_a-v_b path in G. Then, there is a corresponding s-t path π''' in H that is minimal, in the sense that there is no (unlabeled) s-t path in H that is a proper subsequence of π''', entailing that it is simple. Therefore, $\mathbf{FIM}_{I_S}(G)$ is $\#P$-hard.

Finally, the result for $\mathbf{IM}_{I_S}(G,\Gamma)$ follows by using a result similar to that of Lemma 3, according to which the data-complexity $\#P$-hardness carries over to the case of combined complexity, where the set of constraints is part of the input (that is, this set is not assumed to be fixed as in the previous case). □

The following theorem states that the three decision problems for I_S are in the class PP. It is worth noting that these problems are unlikely to belong to classes contained in PP (under the standard complexity assumption $P \neq NP$). Specifically, $\mathbf{FUV}_{I_S}(G, \nu)$ (resp., $\mathbf{FLV}_{I_S}(G, \nu)$, $\mathbf{FEV}_{I_S}(G, \nu)$), and thus $\mathbf{UV}_{I_S}(G, \Gamma, \nu)$ (resp., $\mathbf{LV}_{I_S}(G, \Gamma, \nu)$, $\mathbf{EV}_{I_S}(G, \Gamma, \nu)$), is unlikely to be in NP (resp., $coNP$, D^p) as checking whether a given set of paths is a set of minimal problematic paths cannot be accomplished in polynomial time due to the exponential size of $|\mathsf{MinimalProblematicPaths}(G)|$ w.r.t. G.

Theorem 8. *For any GDB G, $\boldsymbol{LV}_{I_S}(G,\Gamma,\nu)$, $\boldsymbol{UV}_{I_S}(G,\Gamma,\nu)$, and $\boldsymbol{EV}_{I_S}(G,\Gamma,\nu)$ are in PP. Moreover, $\boldsymbol{FLV}_{I_S}(G,\nu)$, $\boldsymbol{FUV}_{I_S}(G,\nu)$, and $\boldsymbol{FEV}_{I_S}(G,\nu)$ are in CNL.*

Proof. For **LV**, **UV**, and **EV**, we show the memberships in CP. The result follows from the fact that CP coincides with PP [61,57].

The membership of $\mathbf{LV}_{I_S}(G,\Gamma,\nu)$ in CP follows from the fact that it is equivalent to checking whether $|\mathsf{MinimalProblematicPaths(G)}| \geq \nu$, where ν is a constant value given in the input, and checking whether a sequence of edges of G is a minimal problematic path can be accomplished in polynomial time. More formally, given $\langle G,\Gamma,\pi\rangle$, where π a sequence of edges of G, let B be the problem of deciding whether $\pi \in \mathsf{MinimalProblematicPaths}(G)$. Since the complexity of B is in P (cf. Lemma 5), the size of the minimal paths of G is polynomially bounded w.r.t. the size of G, and $\langle G,\Gamma,\nu\rangle$ is positive instance of $\mathbf{LV}_{I_S}(G,\Gamma,\nu)$ iff $C^{\nu}_{\pi}(G,\Gamma,\pi) \in B$, it follows that $\mathbf{LV}_{I_S}(G,\Gamma,\nu)$ is in CP.

Since $I_S(G) \geq \nu$ iff $I_S(G) \not\leq \nu - 1$, that is $\mathbf{LV}_{I_S}(G,\Gamma,\nu)$ is true iff $\mathbf{UV}_{I_S}(G,\Gamma,\nu-1)$ is false, and CP is closed under complement [61, 57] it follows that $\mathbf{UV}_{I_S}(G,\Gamma,\nu)$ is in CP as well. Finally, since $\mathbf{EV}_{I_S}(G,\Gamma,\nu)$ is true iff $\mathbf{LV}_{I_S}(G,\Gamma,\nu)$ is true and $\mathbf{UV}_{I_S}(G,\Gamma,\nu)$ is true, and CP is closed for intersection [63], we have that $\mathbf{EV}_{I_S}(G,\Gamma,\nu)$ is in CP too.

Finally, the memberships of $\mathbf{FLV}_{I_S}(G,\nu)$, $\mathbf{FUV}_{I_S}(G,\nu)$, and $\mathbf{FEV}_{I_S}(G,\nu)$ in CNL follow by reasoning as above but observing that the above-mentioned problem B is in NL under data complexity (cf. Lemma 5). □

5.2.4. Measure counting the MIMSs

In Proposition 1 we showed that $I_M \leq I_S$. We start this subsection by giving a condition which guarantees the equality of I_M and I_S. Recall that s denotes the subgraph constructor function defined before the statement of Proposition 1.

Lemma 6. *Let $\pi \in \mathsf{MinimalProblematicPaths}(G)$ such that π does not contain a cycle. Therefore, if $\pi' \in \mathsf{MinimalProblematicPaths}(G)$ such that $s(\pi') = s(\pi)$ then $\pi' = \pi$.*

Proof. Let $\pi = (v_1, \ell_1, v_2) \ldots (v_{n-1}, \ell_{n-1}, v_n) \in \mathsf{MinimalProblematicPaths}(G)$ such that π does not contain a cycle. Therefore, for all $i \neq j$, $v_i \neq v_j$. If $\pi' \in \mathsf{MinimalProblematicPaths}(G)$ such that $s(\pi') = s(\pi)$, then π and π' must have the same vertices and edges.

Consider what can be the first edge in π'. If it is not (v_1, ℓ_1, v_2) then there must be an edge that ends at v_1 but that would make a cycle. Hence, the first edge must be (v_1, ℓ_1, v_2). Similarly, the second edge (if there is one) must be (v_2, ℓ_1, v_3) and so on. Therefore $\pi' = \pi$. □

Proposition 5. *If G is such that none of its minimal problematic paths contain a cycle then $I_M(G) = I_S(G)$. In particular, if G is acyclic then $I_M(G) = I_S(G)$.*

Proof. The lemma above shows that the minimal problematic paths without cycles are in a one-to-one correspondence with their corresponding MIMSs. □

The following lemma addresses the complexity of deciding whether a subgraph (of a given graph) is a monotonic subgraph.

Lemma 7. *Given a GDB G and a subgraph G' of G, deciding whether $G' \subseteq_m G$ is in P under combined complexity and in NL under data-complexity.*

Proof. Recall that $G' \subseteq_m G$ iff $\mathsf{ProblematicPaths}(G') \subseteq \mathsf{ProblematicPaths}(G)$, that is, there is no path $\pi \in \mathsf{ProblematicPaths}(G')$ which is not in $\mathsf{ProblematicPaths}(G)$. Consider now what happens for a path π having the form $(v_0, \ell_1, v_1) \dots (v_{n-1}, \ell_n, v_n)$ and such that it is problematic in G' but not in G. We have that $(v_0, v_n) \in Q_1(G') \setminus Q_2(G')$ for some constraint $Q_1 \subseteq Q_2 \in \Gamma$. Since G' is a subgraph of G, it is the case that $(v_0, v_n) \in Q_1(G)$. Moreover, since π is not problematic in G, it must be the case that $(v_0, v_n) \in Q_2(G)$ as well. Therefore, the reason for which π is problematic in G' but not in G is that there is a path $\pi' \in \mathsf{Path}(Q_2, G)$ which is missing in G'. In order to check this condition we can check whether for each edge e belonging to G but not to G', e belongs to a path $\pi' \in \mathsf{Path}(Q_2, G)$ starting in v_0 and ending in v_n, where $(v_0, v_n) \in Q_1(G')$. That is, $\mathsf{ProblematicPaths}(G') \subseteq \mathsf{ProblematicPaths}(G)$ iff for each constraint $Q_1 \subseteq Q_2 \in \Gamma$, for each pair $(v_0, v_n) \in Q_1(G') \setminus Q_2(G')$, there is no edge e in G but not in G' belonging to a path $\pi' \in \mathsf{Path}(Q_2, G)$ starting in v_0 and ending in v_n. This can be checked in P under combined complexity and in NL under data-complexity by using a strategy similar to that used in the proof of Lemma 4 to check whether an edge belongs to a problematic path. That is, for each $e \in G \setminus G'$, with $e = (v, l, v')$, we traverse the intersection graph $I(G, A_{Q_2})$ by first finding the paths from v_0 to v, then advancing to the appropriate states of the intersection graph via ℓ and v', and then continuing from v' (and the appropriate states) to other reachable nodes by ensuring that v_n is reached if possible. Finally, since we need to repeat this procedure only for the pairs in $Q_1(G') \setminus Q_2(G')$, which are polynomial in number, the statement follows. □

As stated below, deciding whether a subgraph is a MIMS (of a given graph) is tractable.

Lemma 8. *Given a subgraph G' of a GDB G, deciding whether $G' \in \mathsf{MIMS}(G)$ is in P under combined complexity and in NL under data-complexity.*

Proof. Recall that G' is a MIMS of G iff $G' \subseteq_m G$, G' is inconsistent, and there is no inconsistent G'' such that $G'' \subseteq_m G'$ and $G'' \neq G'$. Checking whether $G' \subseteq_m G$ is in P under combined complexity and in NL under data-complexity (cf. Lemma 7), and this is also the case for checking whether G' is inconsistent as it corresponds to deciding $\mathbf{LV}_{I_B}(G', \Gamma, 1)$ (or $\mathbf{FLV}_{I_B}(G', 1)$ under data complexity). For minimality, we check that there is no inconsistent subgraph G'' such that $G'' \subseteq_m G'$ and $G'' \neq G'$, as follows. For each edge e of G', check whether the subgraph $G' - \{e\}$ obtained by removing e from G' is consistent and if not, check whether $G' - \{e\} \subseteq_m G'$ (which processes are in P under combined complexity, and in NL under data-complexity). If all such subgraphs are either consistent or the ones that are inconsistent are not monotonic subgraphs of G', then $G' \in \mathsf{MIMS}(G)$. But if there is an inconsistent monotonic subgraph, $G' - \{e\}$, then G' is not minimal inconsistent and hence $G' \notin \mathsf{MIMS}(G)$. □

We are now ready to state the complexity of the measure I_M.

Theorem 9. *For any GDB G, $\mathbf{FIM}_{I_M}(G)$ and $\mathbf{IM}_{I_M}(G, \Gamma)$ are #P-complete.*

Proof. (Membership). We show that the membership is in $\#P$ for $\mathbf{IM}_{I_M}(G, \Gamma)$, from which the result for $\mathbf{FIM}_{I_M}(G)$ follows. $\mathbf{IM}_{I_M}(G, \Gamma)$ can be computed by counting the number of accepting paths of a non-deterministic polynomial-time Turing machine M such that (*i*) M non-deterministically guesses a sequence G' of edges of G such that each leaf of the resulting computation tree represents a subgraph of G; (*ii*) then, M checks in polynomial time that $G' \in \mathsf{MIMS}(G)$ (cf. Lemma 8).
(Hardness). In the hardness proof of Theorem 7, it is shown that the number of s-t paths in a (directed) graph H that visit every node at most once is equal to the number of minimal problematic paths of a graph G, that is $I_S(G)$ (where a single integrity constraint is used). The hardness results follow from the fact that none of the minimal problematic paths of G contain a cycle, and thus $I_M(G) = I_S(G)$, cf. Proposition 5. □

Finally, we consider the three decision problems for I_M which turn out to be in PP, similarly to what we have shown for I_S.

Theorem 10. *For any GDB G, $\mathbf{LV}_{I_M}(G,\Gamma,v)$, $\mathbf{UV}_{I_M}(G,\Gamma,v)$, and $\mathbf{EV}_{I_M}(G,\Gamma,v)$ are in PP. Moreover, $\mathbf{FLV}_{I_M}(G,v)$, $\mathbf{FUV}_{I_M}(G,v)$, and $\mathbf{FEV}_{I_M}(G,v)$ are in CNL.*

Proof. The membership of $\mathbf{LV}_{I_M}(G,\Gamma,v)$ in CP (which coincides with PP [61,57]) follows from the fact that it is equivalent to checking whether $|\mathsf{MIMS(G)}| \geq v$, where v is a constant value given in the input, and checking whether a subgraph of G is a MIMS can be accomplished in polynomial time. More formally, given $\langle G,\Gamma,G'\rangle$, where G' is a subgraph of G, let B be the problem of deciding whether $G' \in \mathsf{MIMS}(G)$. Since the complexity of B is in P (cf. Lemma 8), the size of G' is polynomially bounded w.r.t. the size of G, and $\langle G,\Gamma,v\rangle$ is positive instance of $\mathbf{LV}_{I_M}(G,\Gamma,v)$ iff $C^v_{G'}(G,\Gamma,G') \in B$, it follows that $\mathbf{LV}_{I_M}(G,\Gamma,v)$ is in CP. Moreover, since $\mathbf{LV}_{I_M}(G,\Gamma,v)$ is true iff $\mathbf{UV}_{I_M}(G,\Gamma,v-1)$ is false, and CP is closed under complement [61,57], it follows that $\mathbf{UV}_{I_M}(G,\Gamma,v)$ is in CP as well. Finally, since $\mathbf{EV}_{I_M}(G,\Gamma,v)$ is true iff $\mathbf{LV}_{I_M}(G,\Gamma,v)$ is true and $\mathbf{UV}_{I_M}(G,\Gamma,v)$ is true, and CP is closed for intersection [63], we have that $\mathbf{EV}_{I_M}(G,\Gamma,v)$ is in CP too.

Finally, the memberships of $\mathbf{FLV}_{I_M}(G,v)$, $\mathbf{FUV}_{I_M}(G,v)$, and $\mathbf{FEV}_{I_M}(G,v)$ in CNL follow by reasoning as above but observing that the above-mentioned problem B is in NL under data complexity (cf. Lemma 8). □

5.3. Discussion of the complexity results

A summary of the complexity results obtained for the problems studied in Section 5 is reported in Table 2. We provided a complete characterization of the complexity of the considered inconsistency measures by examining both the data and combined complexity for all of them. Table 2 shows that we found three distinct groups of measures from the computational standpoint. The first group, the largest one, consists of the measures I_B, I_C, I_P, I_E, I_L, and I_V, that are tractable; in particular, they are in NL under data complexity, which is the typical complexity evaluation carried out for databases since the integrity constraints are usually fixed. Most of the measures in this group count elementary elements of the graph database, such as edges and (pairs of) vertices, that are problematic as they contribute to some conflict. The second group of measures consists of the repair-based ones, namely I_{E-}, I_{V-}, and I_{E+}, for which we provide tight complexity bounds for the first level of the polynomial hierarchy [56], even under data complexity (and for restricted classes of RPCs, that is word constraints and non-recursive ones). Also, these measures count elementary elements of the graph database, but they focus on sets of elements needed for addition and deletion that can restore consistency, which turned out to be computationally more involved. Finally, the third group consists of the two measures that count complex objects such as minimal problematic paths and minimal monotonic inconsistent subgraphs, and they turned out to be the most intractable ones (under standard complexity assumptions) [59,57]. Still, they are less costly than several inconsistency measures for indefinite relational databases [35] and propositional knowledge bases [29].

6. Related work

Although the idea of measuring inconsistency was introduced more than 40 years ago, in [24], at that time it did not seem to be an important issue. The problem became more noticeable in the 1990s when it became possible to store large amounts of information. Then, in the early 2000s, several AI researchers started to investigate this issue systematically [70]. The bulk of this work since then has been for propositional knowledge bases, that is, where the information is presented as a set of formulas in propositional logic. By now a substantial amount of work has been done along these lines, which includes for instance the approaches developed in [71–78,26,79]. In the last ten years the work has been extended to other frameworks. For instance, the problem of measuring inconsistency has been addressed also in the context of probabilistic knowledge bases [80–83], with [82] focusing on computational strategies via linear programming. [25] surveys most of the research on inconsistency measures and gives some extensions.

However, information in many cases is not restricted to propositional logic formulas. An approach that lifts the idea of inconsistency measure from propositional knowledge bases to a range of different frameworks used for storing real world data, e.g. graph databases, has been proposed in [65]. That approach is based on first translating a general information space, which encompasses e.g. relational and graph databases, into an inconsistency equivalent propositional knowledge base, and then applying propositional inconsistency measures to find the inconsistency of the general information space. This way the results about inconsistency measures for propositional knowledge bases can be applied to a range of frameworks. The weak point of such an approach is that it is not specifically tailored to the way information is represented and, for instance, it makes no distinction between database elements (e.g. nodes and edges) and integrity constraints. This implies that inconsistency measures blame simultaneously data elements and constraints without considering that inconsistency in databases typically refers to the data, rather than the constraints.

There are several interesting works exploring the problem of quantifying inconsistency in relational databases. [84] first developed single-dependency axioms for dirtiness functions quantifying inconsistency w.r.t. one or more functional dependencies, where dirtiness functions for multiple dependencies are built on top of dirtiness functions for single dependencies. The approach in [85–87] deals with relational databases from the point of view of first-order logic, as in logic programming. Its purpose is to show how database inconsistency measures can be applied to integrity checking [88,89], relaxing repairs, and repair checking. However, the degrees of inconsistency defined in [87] form a partially ordered set; hence it is not always possible to compare the inconsistency of different databases. An inconsistency measure based on an abstract repair semantics is proposed in [90], where the degree of inconsistency depends on the distance between the database instance and the set of possible repairs under a given repair semantics; an instantiation for cardinality-repairs that can be computed via answer-set programs is proposed in [33]. Provenance-informed annotations of the base tuples are used in [91,92] to characterize the level of inconsistency of data and query results. In particular, building upon the computed annotations, different measures of inconsistency which consider single and multiple violations of denial constraints are

introduced. Moreover, inconsistency measures have been considered as the basis of progress indicators for data-cleaning systems in [34], where properties that account for operational aspects of repair systems are introduced as well as a measure satisfying such properties. Finally, the Shapley value [93] of database tuples relative to inconsistency measures is investigated in [94] to calculate the contribution of a tuple to inconsistency for inconsistent databases with respect to functional dependencies.

Most of the literature on inconsistency measures focuses on absolute inconsistency measures. Some works deal with relative inconsistency measures for propositional knowledge bases [30,95]. Rationality postulates compliance and the data complexity of absolute and relative inconsistency measures for relational databases have been investigated in [35] and [96], respectively. Moreover, (absolute) dimensional inconsistency measures for spatio-temporal databases have been investigated in [97], where the issue is measuring inconsistency along some dimensions such as space, time, and (moving) objects.

To the best of our knowledge, this is the first piece of work addressing the problem of measuring inconsistency in graph databases, a widely used formalism to represent real-world information. As in other approaches focusing on different formalisms, e.g. propositional logic and relational databases, we have initiated a systematic study of measuring inconsistency in graph databases by proposing some measures, analyzing the proposed measures by rationality postulates, and investigating the (combined and data) complexity of the proposed inconsistency measures for graph databases.

7. Conclusions and future work

Given the vast amount of data stored in graph databases and the ubiquitous problem of data inconsistency, we think that measuring inconsistency in graph databases is an important issue. In this paper, we have made a first significant effort in dealing with various aspects of this issue. Based on what has been done in the past by the AI community, and in particular on central concepts from the definitions of inconsistency measures for propositional and relational databases, we defined inconsistency measures for graph databases with regular path constraints. Given the particular structure of the considered data model, we devised new inconsistency measures for graph databases by relying on graph elements, such as edges, vertices, and paths, which can help in understanding the source of the inconsistencies for this case. We then checked compliance with various postulates tailored to the graph database context. We also proved the complexity of calculating the value of the inconsistency measures as well as checking for lower and upper bounds and equality with a specific value. Some, but not all of the measures, are tractable. These results give insight into measuring inconsistency for graph databases.

There are several issues concerning the measurement of inconsistency in graph databases that we have not considered. First, as we mentioned at the beginning of Section 3.1, one way to classify inconsistency measures is to distinguish between absolute and relative measures. An absolute measure is one that measures, in some way, the amount of the inconsistency. That is, in our case it answers the question,"How much inconsistency is in the graph database?". On the other hand, a relative measure is one that measures, in some way, the proportion of the inconsistency. That is, in our case, it answers the question,"How inconsistent is the graph database?".

In typical cases there is a relative measure that corresponds to an absolute measure. For example, consider the measure I_E which measures the number of problematic edges. This measures a totality, hence it is an absolute measure. The corresponding relative measure, say I_E^r, measures the ratio of the problematic edges to the total number of edges. The measures we have considered, except for the special case of I_B are absolute measures. For relative measures the postulates we have considered may not be appropriate. For example, Monotony states that as the graph database is enlarged, the inconsistency measure does not decrease. But if we add elements that are consistent with the rest of the graph database, while the totality of the inconsistency does not change, we expect the ratio of the inconsistency, a relative measure, to decrease. This issue was investigated thoroughly for propositional knowledge bases in [30] and subsequently in [96] for relational databases. We will define relative measures that correspond to our absolute measures and consider the appropriate postulates for them. We will also determine the complexity of these measures. In this regard, it is worth mentioning that the results in Table 2 can be used to establish the complexity of some relative measures. For instance, since the measure I_E is tractable, it can be easily shown that its relative version I_E^r defined earlier is tractable as well. In general, if a relative measure is defined as the ratio of an absolute to a polynomial-time computable function, then the complexity of the absolute measure can give an upper bound on the complexity of the relative one. Analogously, complexity lover-bounds of absolute measures can help in providing lower bounds on the complexity of their relative variants. Hence, concerning what was just mentioned, our work provides a fundamental theoretical basis for investigating relative inconsistency measures in graph databases.

The second issue is the extension of RPQs with the ability to traverse edges in both directions, yielding the class of two-way RPQs, called 2RPQs [40]. 2RPQs are defined as follows. Let $\Sigma^{\pm}$ be the language that extends Σ with the inverse ℓ^- of each label ℓ in Σ. Then, a 2RPQ Q over Σ is an RPQ over $\Sigma^{\pm}$. Consequently, the evaluation of a 2RPQ over a graph database G is defined by considering the graph database $G^{\pm}$ that is obtained from G by adding all (inverse) edges of the form (u, ℓ^-, v) for each edge (v, ℓ, u) in G. That is, the evaluation $Q(G)$ of a 2RPQ Q (defined over Σ) over G is the evaluation of the RPQ Q (defined over $\Sigma^{\pm}$) over $G^{\pm}$. An example of 2RPQ for the GDB G_{ex} of Fig. 1 is $\mathsf{nephew\_of} \cdot \mathsf{child\_of}^-$. The answer of this query is the set consisting of the pair $(\mathsf{Frank}, \mathsf{Grace})$ which is obtained by the path formed by a (direct) $\mathsf{nephew\_of}$ edge followed by an inverse $\mathsf{child\_of}$ edge.

If our graph database uses the 2RPQ interpretation, we say that the edges in the graph are *explicit* edges, while the inverse edges, not specifically shown in the graph, are *implicit* edges. As far as our inconsistency measures are concerned, everything works fine: we just assume that we are dealing with the graph consisting of both the explicit and implicit edges. Intuitively, in this situation, given a GDB G with 2RPCs, we can evaluate $I(G)$ by means of $I(G^{\pm})$. Hence, the complexity results stated in Table 2 still hold. Moreover, the postulate satisfaction results of Table 1 also hold if we interpret the objects the postulates deal with (e.g. free edges) as defined over $I(G^{\pm})$ and w.r.t. 2RPCs (e.g. the union of implicit and explicit free edges).

The issue concerning the 2RPQ interpretation is that we may need to treat explicit and implicit edges differently. This happens if all our graphs automatically have the 2RPQ interpretation. Consider a measure such as I_{E-}, asking for the minimal number of edges that need to be deleted to restore consistency. In this case we can delete only explicit edges, and when an explicit edge is deleted the corresponding implicit edge is deleted as well. This leads to a problem, for example, if the deletion of an explicit edge also leads to the deletion of an implicit edge which is needed to satisfy a constraint. Even in the case of I_E, measuring the number of problematic edges, we may need to deal with explicit and implicit edges differently. There is a difference between the case where the explicit edge is free but the corresponding implicit edge is problematic versus the case where they are both problematic. We plan to consider the issue of 2RPQ interpretations in detail and consider the relevant complexity issues. Still, even in this case, our work provides a fundamental basis for investigating inconsistency measures in graph databases with the 2RPQ interpretation.

Finally, a third issue is computing the contribution of the inconsistency of individual elements of the graph database to the total amount of inconsistency. This has been done for propositional and relational databases by computing the Shapley value [93] of each formula (resp., tuple) [74,94,98]. We believe that it is worthwhile to investigate this issue for graph databases now that we defined the important inconsistency measures for this topic. We may find that some edges or vertices contribute much more to the inconsistency than others. This is useful to know if we wish to remove as much inconsistency as possible with a minimal number of modifications. In fact, Shapley values can be used to initiate the cleaning of the graph database from inconsistency. In this regard we will also consider different ways of making the graph database consistent or perhaps nearly consistent [18–20]. This will lead to various approaches to consistent query answering in graph databases [41], that we plan to explore in future work.

CRediT authorship contribution statement

John Grant: Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Francesco Parisi:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

We wish to thank the reviewers for many helpful comments and suggestions.

The second author acknowledges the support of the PNRR projects *FAIR - Future AI Research* (PE00000013) and *Tech4You - Technologies for climate change adaptation and quality of life improvement* (ECS0000009), under the NRRP MUR program funded by the Next Generation EU.

References

[1] M. Arenas, C. Gutierrez, J.F. Sequeda, Querying in the age of graph databases and knowledge graphs, in: Proc. of International Conference on Management of Data (SIGMOD), ACM, 2021.
[2] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, A.N. Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J.F. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, ACM Comput. Surv. 54 (4) (2022) 71:1–71:37.
[3] A. Hogan, Knowledge graphs: a guided tour (invited paper), in: Proc. of International Research School in Artificial Intelligence in Bergen, (AIB), vol. 99, 2022, pp. 1:1–1:21.
[4] R. Angles, M. Arenas, P. Barceló, A. Hogan, J.L. Reutter, D. Vrgoc, Foundations of modern query languages for graph databases, ACM Comput. Surv. 50 (5) (2017) 68:1–68:40.
[5] X.L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, W. Zhang, From data fusion to knowledge fusion, Proc. VLDB Endow. 7 (10) (2014) 881–892.
[6] K. Rabbani, M. Lissandrini, K. Hose, Extraction of validating shapes from very large knowledge graphs, Proc. VLDB Endow. 16 (5) (2023) 1023–1032.
[7] J.Z. Pan, S. Razniewski, J. Kalo, S. Singhania, J. Chen, S. Dietze, H. Jabeen, J. Omeliyanenko, W. Zhang, M. Lissandrini, R. Biswas, G. de Melo, A. Bonifati, E. Vakaj, M. Dragoni, D. Graux, Large language models and knowledge graphs: opportunities and challenges, Trans. Graph Data Knowl. 1 (1) (2023) 2:1–2:38.
[8] A.Y. Levy, Combining artificial intelligence and databases for data integration, in: Artificial Intelligence Today: Recent Trends and Developments, Springer, 1999, pp. 249–268.
[9] M. Arenas, L.E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proc. of Symposium on Principles of Database Systems (PODS), 1999, pp. 68–79.
[10] M. Calautti, L. Libkin, A. Pieris, An operational approach to consistent query answering, in: Proc. of ACM Symposium on Principles of Database Systems (PODS), 2018, pp. 239–251.
[11] M.V. Martinez, F. Parisi, A. Pugliese, G.I. Simari, V.S. Subrahmanian, Policy-based inconsistency management in relational databases, Int. J. Approx. Reason. 55 (2) (2014) 501–528.
[12] B. Fazzinga, S. Flesca, F. Furfaro, F. Parisi, DART: a data acquisition and repairing tool, in: EDBT 2006 Workshops on Inconsistency and Incompleteness in Databases (IIDB), 2006, pp. 297–317.
[13] S. Hao, N. Tang, G. Li, J. He, N. Ta, J. Feng, A novel cost-based model for data repairing, IEEE Trans. Knowl. Data Eng. 29 (4) (2017) 727–742.

[14] J. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, N. Tang, Interactive and deterministic data cleaning, in: Proc. of International Conference on Management of Data (SIGMOD), 2016, pp. 893–907.
[15] T. Bleifuß, L. Bornemann, D.V. Kalashnikov, F. Naumann, D. Srivastava, Dbchex: interactive exploration of data and schema change, in: Proc. of Biennial Conference on Innovative Data Systems Research (CIDR), 2019.
[16] A. Giuzio, G. Mecca, E. Quintarelli, M. Roveri, D. Santoro, L. Tanca, INDIANA: an interactive system for assisting database exploration, Inf. Syst. 83 (2019) 40–56.
[17] S. Sakr, A. Bonifati, H. Voigt, A. Iosup, K. Ammar, R. Angles, W.G. Aref, M. Arenas, M. Besta, P.A. Boncz, K. Daudjee, E.D. Valle, S. Dumbrava, O. Hartig, B. Haslhofer, T. Hegeman, J. Hidders, K. Hose, A. Iamnitchi, V. Kalavri, H. Kapp, W. Martens, M.T. Özsu, E. Peukert, S. Plantikow, M. Ragab, M. Ripeanu, S. Salihoglu, C. Schulz, P. Selmer, J.F. Sequeda, J. Shinavier, G. Szárnyas, R. Tommasini, A. Tumeo, A. Uta, A.L. Varbanescu, H. Wu, N. Yakovets, D. Yan, E. Yoneki, The future is big graphs: a community view on graph processing systems, Commun. ACM 64 (9) (2021) 62–71.
[18] S. Abriola, M.V. Martínez, N. Pardal, S. Cifuentes, E.P. Baque, On the complexity of finding set repairs for data-graphs, J. Artif. Intell. Res. 76 (2023) 721–759.
[19] J. Chen, H. Dong, J. Hastings, E. Jiménez-Ruiz, V. López, P. Monnin, C. Pesquita, P. Skoda, V.A.M. Tamma, Knowledge graphs for the life sciences: recent developments, challenges and opportunities, Trans. Graph Data Knowl. 1 (1) (2023) 5:1–5:33.
[20] B. Xue, L. Zou, Knowledge graph quality management: a comprehensive survey, IEEE Trans. Knowl. Data Eng. 35 (5) (2023) 4969–4988.
[21] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S.C. Guttula, S. Mujumdar, S. Afzal, R.S. Mittal, V. Munigala, Overview and importance of data quality for machine learning tasks, in: KDD, ACM, 2020, pp. 3561–3562.
[22] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proc. IEEE 104 (1) (2016) 11–33.
[23] I.H. Sarker, Data science and analytics: an overview from data-driven smart computing, decision-making and applications perspective, SN Comput. Sci. 2 (5) (2021) 377.
[24] J. Grant, Classifications for inconsistent theories, Notre Dame J. Form. Log. XIX 3 (1978) 435–444.
[25] J. Grant, M.V. Martinez, Measuring Inconsistency in Information, College Publications, 2018.
[26] M. Thimm, On the expressivity of inconsistency measures, Artif. Intell. 234 (2016) 120–151.
[27] K. Mu, Measuring inconsistency with constraints for propositional knowledge bases, Artif. Intell. 259 (2018) 52–90.
[28] G.D. Bona, J. Grant, A. Hunter, S. Konieczny, Towards a unified framework for syntactic inconsistency measures, in: Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI), 2018, pp. 1803–1810.
[29] M. Thimm, J.P. Wallner, On the complexity of inconsistency measurement, Artif. Intell. 275 (2019) 411–456.
[30] P. Besnard, J. Grant, Relative inconsistency measures, Artif. Intell. 280 (2020) 103231.
[31] M. Ulbricht, M. Thimm, G. Brewka, Handling and measuring inconsistency in non-monotonic logics, Artif. Intell. 286 (2020) 103344.
[32] K. Mu, Z. Jin, R. Lu, W. Liu, Measuring inconsistency in requirements specifications, in: Proc. of European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), 2005, pp. 440–451.
[33] L.E. Bertossi, Repair-based degrees of database inconsistency, in: Proc. of Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR), 2019, pp. 195–209.
[34] E. Livshits, R. Kochirgan, S. Tsur, I.F. Ilyas, B. Kimelfeld, S. Roy, Properties of inconsistency measures for databases, in: Proc. of International Conference on Management of Data (SIGMOD), 2021, pp. 1182–1194.
[35] F. Parisi, J. Grant, On measuring inconsistency in definite and indefinite databases with denial constraints, Artif. Intell. 318 (2023) 103884.
[36] L. Zhou, H. Huang, G. Qi, Y. Ma, Z. Huang, Y. Qu, Measuring inconsistency in DL-Lite ontologies, in: Proc. of Int. Conf. on Web Intelligence (WI), 2009, pp. 349–356.
[37] X. Zhang, K. Wang, Z. Wang, Y. Ma, G. Qi, Z. Feng, A distance-based framework for inconsistency-tolerant reasoning and inconsistency measurement in DL-Lite, Int. J. Approx. Reason. 89 (2017) 58–79.
[38] G.D. Bona, J. Grant, A. Hunter, S. Konieczny, Classifying inconsistency measures using graphs, J. Artif. Intell. Res. 66 (2019) 937–987.
[39] P. Buneman, S.B. Davidson, M.F. Fernandez, D. Suciu, Adding structure to unstructured data, in: Proc. of 6th International Conference on Database Theory (ICDT), vol. 1186, 1997, pp. 336–350.
[40] D. Calvanese, G.D. Giacomo, M. Lenzerini, M.Y. Vardi, Rewriting of regular expressions and regular path queries, J. Comput. Syst. Sci. 64 (3) (2002) 443–465.
[41] P. Barceló, G. Fontaine, On the data complexity of consistent query answering over graph databases, J. Comput. Syst. Sci. 88 (2017) 164–194.
[42] F. Parisi, N. Park, A. Pugliese, V.S. Subrahmanian, Top-k user-defined vertex scoring queries in edge-labeled graph databases, ACM Trans. Web 12 (4) (2018) 21:1–21:35.
[43] R. Angles, The property graph database model, in: Proc. of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, vol. 2100, 2018.
[44] M.A. Rodriguez, P. Neubauer, Constructions from dots and lines, Bull. Am. Soc. Inf. Sci. Technol. 36 (6) (2010) 35–41.
[45] P.T. Wood, Query languages for graph databases, SIGMOD Rec. 41 (1) (2012) 50–60.
[46] P.B. Baeza, Querying graph databases, in: Proceedings of the 32nd ACM Symposium on Principles of Database Systems (PODS), ACM, 2013, pp. 175–188.
[47] P. Buneman, S.B. Davidson, G.G. Hillebrand, D. Suciu, A query language and optimization techniques for unstructured data, in: Proceedings of International Conference on Management of Data (SIGMOD), 1996, pp. 505–516.
[48] W3C, SPARQL 1.1 query language, https://www.w3.org/TR/sparql11-query, 2013.
[49] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, A. Taylor, Cypher: an evolving query language for property graphs, in: Proceedings of International Conference on Management of Data (SIGMOD), 2018, pp. 1433–1445.
[50] S. Abiteboul, V. Vianu, Regular path queries with constraints, J. Comput. Syst. Sci. 58 (3) (1999) 428–452.
[51] G. Grahne, A. Thomo, Query containment and rewriting using views for regular path queries under constraints, in: Proceedings of the Twenty-Second Symposium on Principles of Database Systems (PODS), 2003, pp. 111–122.
[52] P. Besnard, Revisiting postulates for inconsistency measures, in: Proc. of European Conference Logics in Artificial Intelligence (JELIA), vol. 8761, 2014, pp. 383–396.
[53] M. Thimm, On the evaluation of inconsistency measures, in: J. Grant, M.V. Martinez (Eds.), Measuring Inconsistency in Information, College Publications, 2018, pp. 19–60.
[54] A.K. Chandra, D. Harel, Computable queries for relational data bases, J. Comput. Syst. Sci. 21 (2) (1980) 156–178.
[55] M.Y. Vardi, The complexity of relational query languages (extended abstract), in: Proc. of Symposium on Theory of Computing (STOC), 1982, pp. 137–146.
[56] C.M. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, Massachusetts, 1994.
[57] K.W. Wagner, The complexity of combinatorial problems with succinct input representation, Acta Inform. 23 (3) (1986) 325–356.
[58] L.A. Hemaspaandra, H. Vollmer, The satanic notations: counting classes beyond #P and other definitional adventures, SIGACT News 26 (1) (1995) 2–13.
[59] L.G. Valiant, The complexity of computing the permanent, Theor. Comput. Sci. 8 (1979) 189–201.
[60] J.E. Hopcroft, R. Motwani, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, 3rd edition, Addison-Wesley Longman Publishing Co., Inc., 2006.
[61] J. Simon, On the difference between one and many (preliminary version), in: Proceedings of Fourth Colloquium on Automata, Languages and Programming (ICALP), 1977, pp. 480–491.
[62] S. Toda, On the computational power of PP and (+)P, in: 30th Annual Symposium on Foundations of Computer Science, 1989, pp. 514–519.

[63] R. Beigel, N. Reingold, D.A. Spielman, PP is closed under intersection, J. Comput. Syst. Sci. 50 (2) (1995) 191–202.
[64] S. Toda, O. Watanabe, Polynomial time 1-Turing reductions from #PH to #P, Theor. Comput. Sci. 100 (1) (1992) 205–221.
[65] J. Grant, F. Parisi, General information spaces: measuring inconsistency, rationality postulates, and complexity, Ann. Math. Artif. Intell. 90 (2022) 235–269.
[66] J. Grant, A. Hunter, Distance-based measures of inconsistency, in: Proc. of ECSQARU, 2013, pp. 230–241.
[67] A.O. Mendelzon, P.T. Wood, Finding regular simple paths in graph databases, in: Proceedings of the Fifteenth International Conference on Very Large Data Bases (VLDB), 1989, pp. 185–193.
[68] C.H. Papadimitriou, M. Yannakakis, The complexity of facets (and some facets of complexity), J. Comput. Syst. Sci. 28 (2) (1984) 244–259.
[69] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[70] K. Knight, Measuring inconsistency, J. Philos. Log. 31 (1) (2002) 77–98.
[71] A. Hunter, Measuring inconsistency in knowledge via quasi-classical models, in: Proc. of National Conference on Artificial Intelligence and Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI), 2002, pp. 68–73.
[72] A. Hunter, S. Konieczny, Measuring inconsistency through minimal inconsistent sets, in: Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR), 2008, pp. 358–366.
[73] Y. Ma, G. Qi, G. Xiao, P. Hitzler, Z. Lin, An anytime algorithm for computing inconsistency measurement, in: Proc. of Int. Conf. on Knowledge Science, Engineering and Management (KSEM), 2009, pp. 29–40.
[74] A. Hunter, S. Konieczny, On the measure of conflicts: Shapley inconsistency values, Artif. Intell. 174 (14) (2010) 1007–1026.
[75] K. Mu, W. Liu, Z. Jin, D.A. Bell, A syntax-based approach to measuring the degree of inconsistency for belief bases, Int. J. Approx. Reason. 52 (7) (2011) 978–999.
[76] J. Grant, A. Hunter, Semantic inconsistency measures using 3-valued logics, Int. J. Approx. Reason. 156 (2023) 38–60.
[77] K. McAreavey, W. Liu, P.C. Miller, Computational approaches to finding and measuring inconsistency in arbitrary knowledge bases, Int. J. Approx. Reason. 55 (8) (2014) 1659–1693.
[78] M. Thimm, Stream-based inconsistency measurement, Int. J. Approx. Reason. 68 (2016) 68–87.
[79] S. Jabbour, Y. Ma, B. Raddaoui, L. Sais, Quantifying conflicts in propositional logic through prime implicates, Int. J. Approx. Reason. 89 (2017) 27–40.
[80] D. Picado-Muiño, Measuring and repairing inconsistency in probabilistic knowledge bases, Int. J. Approx. Reason. 52 (6) (2011) 828–840.
[81] M. Thimm, Inconsistency measures for probabilistic logics, Artif. Intell. 197 (2013) 1–24.
[82] N. Potyka, Linear programs for measuring inconsistency in probabilistic logics, in: Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR), 2014.
[83] G.D. Bona, M. Finger, Measuring inconsistency in probabilistic logic: rationality postulates and Dutch book interpretation, Artif. Intell. 227 (2015) 140–164.
[84] M.V. Martinez, A. Pugliese, G.I. Simari, V.S. Subrahmanian, H. Prade, How dirty is your relational database? An axiomatic approach, in: Proc. of European Conference Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), 2007, pp. 103–114.
[85] H. Decker, Inconsistency-tolerant database repairs and simplified repair checking by measure-based integrity checking, T. Large-Scale Data- and Knowledge-Centered Systems, vol. 34, 2017, pp. 153–183.
[86] H. Decker, S. Misra, Database inconsistency measures and their applications, in: Proc. of International Conference on Information and Software Technologies (ICIST), 2017, pp. 254–265.
[87] H. Decker, Measuring database inconsistency, in: J. Grant, M.V. Martinez (Eds.), Measuring Inconsistency in Information, College Publications, 2018, pp. 271–311.
[88] H. Decker, D. Martinenghi, Classifying integrity checking methods with regard to inconsistency tolerance, in: Proc. of International Conference on Principles and Practice of Declarative Programming (PPDP), 2008, pp. 195–204.
[89] H. Decker, D. Martinenghi, Inconsistency-tolerant integrity checking, IEEE Trans. Knowl. Data Eng. 23 (2) (2011) 218–234.
[90] L.E. Bertossi, Measuring and computing database inconsistency via repairs, in: Proc. of International Conference on Scalable Uncertainty Management (SUM), 2018, pp. 368–372.
[91] O. Issa, A. Bonifati, F. Toumani, Evaluating top-k queries with inconsistency degrees, Proc. VLDB Endow. 13 (11) (2020) 2146–2158.
[92] O. Issa, A. Bonifati, F. Toumani, INCA: inconsistency-aware data profiling and querying, in: Proc. of International Conference on Management of Data (SIGMOD), 2021, pp. 2745–2749.
[93] K. Hausken, M. Mohr, The value of a player in n-person games, Soc. Choice Welf. 18 (3) (2001) 465–483.
[94] E. Livshits, B. Kimelfeld, The Shapley value of inconsistency measures for functional dependencies, in: Proc. of International Conference on Database Theory (ICDT), vol. 186, 2021, pp. 15:1–15:19.
[95] G. Xiao, Y. Ma, Inconsistency measurement based on variables in minimal unsatisfiable subsets, in: Proc. of 20th European Conference on Artificial Intelligence (ECAI), 2012, pp. 864–869.
[96] F. Parisi, J. Grant, Relative inconsistency measures for indefinite databases with denial constraints, in: Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2023, pp. 3321–3329.
[97] J. Grant, M.V. Martinez, C. Molinaro, F. Parisi, Dimensional inconsistency measures and postulates in spatio-temporal databases, J. Artif. Intell. Res. 71 (2021) 733–780.
[98] K. Mu, Responsibility for inconsistency, Int. J. Approx. Reason. 61 (2015) 43–60.