# On Measuring Inconsistency in Graph Databases

John Grant[1], Francesco Parisi[2]

[1]Department of Computer Science and UMIACS, University of Maryland, College Park, USA
[2]Department of Informatics, Modeling, Electronics and System Engineering (DIMES), University of Calabria, Italy

**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 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 [1], a class of integrity constraints based on a well-known navigational language for graph data. We discuss several inconsistency measures dealing with specific elements contributing to inconsistency in graph databases. Then, 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 others different levels of intractability are exhibited.

**Keywords**

Inconsistency measures, Graph databases, Computational complexity.

## 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 [2, 3, 4]. 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 [5, 2, 3].

However, data are mostly obtained from large-scale data extraction pipelines, which are notoriously brittle and can introduce errors and inconsistencies in these graphs [6, 7, 8], analogously to what happens for traditional relational data [9]. This has lead to an extensive body of work on handling inconsistent data. For instance, approaches for dealing with inconsistent relational databases include consistent query answering frameworks [10, 11], inconsistency

*SEBD 2025: 33rd Symposium on Advanced Database System, June 16–19, 2025, Ischia (Italy)*
grant@cs.umd.edu (J. Grant); fparisi@dimes.unical.it (F. Parisi)
0000-0001-9977-1355 (F. Parisi)

management policies [12], interactive data repairing and cleaning systems [13, 14, 15], as well as interactive data exploration tools [16, 17]. In fact, handling conflicting information has a long research history in the field of relational data. Given the increasing widespread use of graph data, witnessed by an unprecedented growth of interconnected data [18], several issues concerning data quality have become of fundamental importance for graph data as well [19, 20, 21].

An important drawback of relying on data of poor quality is that it can significantly limit the implementation of effective AI solutions [9, 22], 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 labelled edges (relationships between entities) [23]. 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 [24]. *Measuring inconsistency* [25, 26] 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., [27, 28, 29, 30, 31, 32], to mention a few) ever since the idea of measuring inconsistency was introduced more than 45 years ago in [25]. It has been explored in other settings such as software specifications [33], relational databases [34, 35, 36, 37, 38, 39, 40, 41, 42], and ontologies [43, 44], among others [45, 46].

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. In this paper, we explore inconsistency measures for graph databases (GDBs) consisting of edge-labelled directed graphs [47, 48, 49, 5], 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 [5, 3]. 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 pfriend label if $u$ friended $v$ but was not accepted, and a host of labels endorsed-t if $u$ endorsed $v$ w.r.t. topic $t$ and not_endorsed-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. child_of, 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 employs or alumnus_of with the obvious meanings [50]. It is worth mentioning that other graph data models, such as property graphs [51], can be translated to/from edge-labelled directed graphs without loss of information [52, 3]. 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 expectation 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 [48, 53, 54]. For instance, the RPQ endorsed-GDB$^+$ specifies all paths consisting of a sequence of one-or-more directed edges whose label is endorsed-GDB (each

representing the fact that a user endorsed another one w.r.t. the topic GDB), expressing the transitive endorsed-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 [55] and they have been implemented in various systems. For instance, RPQs form the conceptual core of *property paths* in the SPARQL standard [56], which have been implemented in several SPARQL engines. Likewise, in Cypher [57], one can find RPQ-like features [5].

A well-known class of integrity constraints that are based on RPQs is that of *regular path constraints* (RPCs) [58, 59]. 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 child_of $\subseteq$ son_of | daughter_of, intuitively states that every child is a son or a daughter.

In this paper, we investigate inconsistency measures for GDBs with RPCs. 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. We investigate the combined and the data complexity [60, 61] of four natural problems concerning the inconsistency measures we conceive for the considered setting. The interested reader can find more details, proofs of the results stated, as well as additional results, e.g., compliance of inconsistency measures w.r.t. rationality postulates, in the full version of this paper [1].

## 2. Preliminaries

A graph database (GDB) is a finite edge-labeled directed graph [47, 48, 49, 5]. Let $\Sigma$ be a finite alphabet (i.e., a set of symbols, also called *labels*). A GDB $G = (V, E)$ over $\Sigma$ 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 $\ell$. 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 Figure 1.

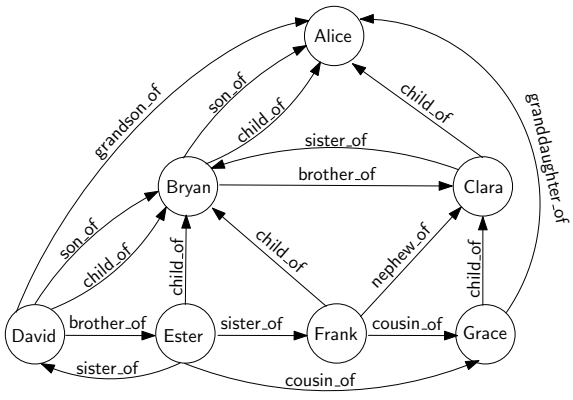


Figure 1: Graph database $G_{ex}$.

For $G = (V, E)$ and $G' = (V', E')$ over alphabet $\Sigma$, we say that $G$ is a subgraph of $G'$ (denoted as $G \subseteq G'$) iff $V \subseteq V'$ and $E \subseteq E'$. We write $G \subset G'$ if $G \subseteq G'$ and $G \neq G'$.

The *size* of $G$ is $|E| + |V|$.

Given GDB $G = (V, E)$, a path $\pi$ in $G$ from vertex $v_0$ to vertex $v_n$ is a sequence of edges of the form: $\pi = (v_0, \ell_1, v_1)(v_1, \ell_2, v_2) \dots (v_{n-1}, \ell_n, v_n)$, 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 $\pi$, denoted $\lambda(\pi)$, is the string $\ell_1 \ell_2 \dots \ell_n$ in $\Sigma^*$, where $\Sigma^*$ is the set of all strings of finite length consisting of labels in $\Sigma$, including the empty string. Observe that $\lambda(\pi)$ is the empty string $\epsilon$ 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 [62] defined over the set $\Sigma$ of edge labels [48]. In our context, a regular expression specifies all paths whose edge labels, when concatenated, form a string in the language of the regular expression.

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 $\pi$ 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 Figure 1, RPQ $Q_1 = \mathsf{son\_of.son\_of}^*$ 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 Figure 1. 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 RPQ $Q_2 = \mathsf{child\_of.(brother\_of|sister\_of)}$, 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) [58, 59]. An RPC over $\Sigma$ is an expression of the form $Q_1 \subseteq Q_2$, where $Q_1$ and $Q_2$ are RPQs over $\Sigma$. A *word constraint* is an RPC in which both $Q_1$ and $Q_2$ are words, i.e. simply sequences of labels. 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 GDB $G$ satisfies the RPC $Q_1 \subseteq Q_2$, denoted as $G \models Q_1 \subseteq Q_2$, iff $Q_1(G) \subseteq Q_2(G)$ (otherwise, we say that $G$ does not satisfy or violates $Q_1 \subseteq Q_2$ and write $G \not\models Q_1 \subseteq Q_2$).

**Example 2.** *For the GDB $G_{ex}$ shown in Figure 1, let $\Gamma$ be the set of the following RPCs: $C_1$:* child_of $\subseteq$ son_of|daughter_of*, meaning that a child is a son or a daughter; $C_2$:* child_of.(brother_of|sister_of) $\subseteq$ nephew_of|niece_of*, meaning that a child of a brother or a sister is a niece or a nephew; $C_3$:* child_of.child_of $\subseteq$ grandson_of|granddaughter_of*, meaning that a child of a child is a grandson or a granddaughter; $C_4$:* son_of.child_of $\subseteq$ grandson_of*, i.e., 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:* $(\mathsf{Frank}, \mathsf{Bryan})$*,* $(\mathsf{Ester}, \mathsf{Bryan})$*,* $(\mathsf{Grace}, \mathsf{Clara})$*, and* $(\mathsf{Clara}, \mathsf{Alice})$*. Therefore, $G_{ex}$ violates $C_1$, and we write $G_{ex} \not\models C_1$. On the other hand, it can be easily checked that the fourth constraint is satisfied, that is $G_{ex} \models C_4$.*

*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 $C_2$: $\pi_1 =$* $(\mathsf{David}, \mathsf{child\_of}, \mathsf{Bryan})$ $(\mathsf{Bryan}, \mathsf{brother\_of}, \mathsf{Clara})$*; $\pi_2 =$* $(\mathsf{Ester}, \mathsf{child\_of}, \mathsf{Bryan})$ $(\mathsf{Bryan}, \mathsf{brother\_of}, \mathsf{Clara})$*; $\pi_3 =$* $(\mathsf{Grace}, \mathsf{child\_of}, \mathsf{Clara})$ $(\mathsf{Clara}, \mathsf{sister\_of}, \mathsf{Bryan})$*. 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* $(\mathsf{David}, \mathsf{Clara})$*,* $(\mathsf{Ester}, \mathsf{Clara})$*, and* $(\mathsf{Grace}, \mathsf{Bryan})$ *belong to the answer of the RPQ on the left-hand side of $C_2$ but not to the answer of its right-hand side.*

*As for the third constraint, there are two paths that violate $C_3$: $\pi_4 =$* $(\mathsf{Ester}, \mathsf{child\_of}, \mathsf{Bryan})(\mathsf{Bryan}, \mathsf{child\_of}, \mathsf{Alice})$ *and $\pi_5 =$* $(\mathsf{Frank}, \mathsf{child\_of}, \mathsf{Bryan})$ $(\mathsf{Bryan}, \mathsf{child\_of}, \mathsf{Alice})$*. Both need an edge with label* grandson_of *or* granddaughter_of*. Therefore, $G_{ex}$ violates $C_3$ because the pairs* $(\mathsf{Ester}, \mathsf{Alice})$ *and* $(\mathsf{Frank}, \mathsf{Alice})$ *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 $\Gamma$ of RPCs, we use $G \models \Gamma$ to denote the fact that for each RPC $Q_1 \subseteq Q_2$ in $\Gamma$, $G \models Q_1 \subseteq Q_2$. A GDB $G$ is said to be *consistent* w.r.t. $\Gamma$ iff $G \models \mathcal{C}$; otherwise, $G$ is said to be *inconsistent* (w.r.t. $\Gamma$). In Example 2, $G_{ex}$ is inconsistent w.r.t. $\Gamma = \{C_1, C_2, C_3, C_4\}$, that is, $G_{ex} \not\models \Gamma$. Except for the examples where $\Gamma$ is given explicitly, we assume that $\Gamma$ is a given set of RPCs for $\Sigma$. The size of $\Gamma$ is the sum of the sizes of the RPQs occurring in the constraints in $\Gamma$.

## 3. Inconsistency Measures for Graph Databases

Let $\mathcal{G}$ be the set of all finite GDBs. We fix an alphabet $\Sigma$ and a set $\Gamma$ of RPCs over $\Sigma$ and write $\mathcal{G}_\Sigma^\Gamma$ for the set of all such finite GDBs over $\Sigma$. A function $I : \mathcal{G}_\Sigma^\Gamma \to \mathbb{R}_\infty^{\geq 0}$ is an *inconsistency measure* (IM) iff the following condition (called *Consistency*) holds for all $G \in \mathcal{G}_\Sigma^\Gamma$: $I(G) = 0$ iff $G \models \Gamma$. Thus, an IM assigns to every GDB either a nonnegative number or infinity. This number must be 0 iff the GDB is consistent.

Let $Q$ be an RPQ over $\Sigma$. 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 $G$. An answer pair is obtained from a path $\pi$ 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 $Q$. There may be several answer paths that correspond to the same answer pair. For every answer path $\pi$, we indicate the answer pair associated with $\pi$ as $(u, v)_\pi$.

The inconsistency of a GDB is caused by the RPCs. Recall that every RPC in $\Gamma$ has the form $Q_1 \subseteq Q_2$ for some RPQs over $\Sigma$. 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 $\pi \in \mathsf{ProblematicPaths}(G)$. A path that is not problematic is *free*.

**Example 3.** *Continuing with Example 2, the problematic pairs are* (Frank, Bryan)*,* (Ester, Bryan)*,* (Grace, Clara)*, and* (Clara, Alice) *(due to violations of $C_1$), and* (David, Clara)*,* (Ester, Clara)*, and* (Grace, Bryan) *(due to $C_2$), and* (Ester, Alice) *and* (Frank, Alice) *(due to $C_3$).*

Using $e$ for an edge, we slightly abuse notation by writing $e \in \pi$ if $e$ is one of the edges of $\pi$; moreover, we write $\ell \in \pi$ if $\ell$ is a label of an edge in $\pi$, and $v \in \pi$ if $v$ is a vertex for an edge in $\pi$. This way we define additional problematic sets as follows:

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

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 *free*.

**Example 4.** $\mathsf{ProblematicEdges}(G_{ex})$ = {(Frank, child_of, Bryan), (Ester, child_of, Bryan), (Grace, child_of, Clara), (Clara, child_of, Alice), (David, child_of, Bryan), (Bryan, brother_of, Clara), (Clara, sister_of, Bryan), (Bryan, child_of, Alice)}, $\mathsf{ProblematicLabels}(G_{ex})$ = {child_of, brother_of, sister_of}, *and* $\mathsf{ProblematicVertices}(G_{ex})$ = {Frank, Ester, Grace, Bryan, Clara, Alice, David}.

Finally, let $\pi_1$ and $\pi_2$ be distinct paths. We write $\pi_1 \subset \pi_2$ iff $\pi_1$ is a proper subsequence of $\pi_2$. We write $\pi \in \mathsf{MinimalProblematicPaths}(G)$ iff $\pi \in \mathsf{ProblematicPaths}(G)$ and there is no $\pi' \in \mathsf{ProblematicPaths}(G)$ such that $\pi' \subset \pi$.

**Example 5.** *With a little effort, it can be checked that* $\mathsf{MinimalProblematicPaths}(G_{ex}) = \{(\mathsf{Frank}, \mathsf{child\_of}, \mathsf{Bryan}), (\mathsf{Ester}, \mathsf{child\_of}, \mathsf{Bryan}), (\mathsf{Grace}, \mathsf{child\_of}, \mathsf{Clara}), (\mathsf{Clara}, \mathsf{child\_of}, \mathsf{Alice}),$ *and* $(\mathsf{David}, \mathsf{child\_of}, \mathsf{Bryan})(\mathsf{Bryan}, \mathsf{brother\_of}, \mathsf{Clara})\}$.

**Definition 1** (GDB Inconsistency Measures)**.** *For any GDB $G$ (and set $\Gamma$ of RPCs), the inconsistency measures $I_B$, $I_C$, $I_P$, $I_E$, $I_L$, $I_V$, and $I_S$, are as follows: $I_B(G) = 1$ iff $G$ is not consistent (0 otherwise); $I_C(G) = |\{C \mid C \in \Gamma$ and $G \not\models C\}|$; $I_P(G) = |\mathsf{ProblematicPairs(G)}|$; $I_E(G) = |\mathsf{ProblematicEdges(G)}|$; $I_L(G) = |\mathsf{ProblematicLabels(G)}|$; $I_V(G) = |\mathsf{ProblematicVertices(G)}|$; $I_S(G) = |\mathsf{MinimalProblematicPaths(G)}|$.*

In Definition 1, we start by defining the drastic measure $I_B$, which simply differentiates between consistent and inconsistent GDBs. Next, we define one IM, $I_C$, that considers only the violated constraints and counts them. Then, we define IMs that deal with problematic elements: $I_P$ counts the number of problematic pairs; $I_E$, $I_L$, and $I_V$ count the number of problematic edges, labels, and vertices, respectively; $I_S$ counts the number of minimal problematic paths. Observe that, although the set $V$ of 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 IM. 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 use $I_S$ that counts the minimal problematic paths. This is a finite number even if there are infinitely many problematic paths.

**Example 6.** *For the GDB $G_{ex}$ and $\Gamma$ of Example 2, we have that $I_B(G_{ex}) = 1$ because $G_{ex}$ is inconsistent; $I_C(G_{ex}) = 3$ because 3 constraints in $\Gamma$ are violated; $I_P(G_{ex}) = 9$ because there are 9 problematic pairs, as shown in Example 3; $I_E(G_{ex}) = 8$ because there are 8 problematic edges, as shown in Example 4; $I_L(G_{ex}) = 3$ because there are 3 problematic labels (cf. Example 4); $I_V(G_{ex}) = 7$ because there are 7 problematic vertices (cf. Example 4); and $I_S(G_{ex}) = 5$ because there are 5 minimal problematic paths, as stated in Example 5.*

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 7.** *Consider a GDB $G$ over $\Sigma = \{\mathsf{parent\_of}, \mathsf{grandparent\_of}\}$ such that there is a single RPC in $\Gamma$:* $\mathsf{parent\_of.parent\_of} \subseteq \mathsf{grandparent\_of}$. *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. $\Gamma$) and so has inconsistency measure 0 for every IM. 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 IMs.*

We now introduce 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 7, 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 $G'' \neq G'$. We write $\mathsf{MIMS}(G)$ for the set of all MIMSs of $G$.

**Example 8.** $\mathsf{MIMS}(G_{ex})$ *consists of the following subgraphs:* $G_1 = (\{\mathsf{Frank}, \mathsf{Bryan}\}, \{(\mathsf{Frank}, \mathsf{child\_of}, \mathsf{Bryan})\})$, $G_2 = (\{\mathsf{Ester}, \mathsf{Bryan}\}, \{(\mathsf{Ester}, \mathsf{child\_of}, \mathsf{Bryan})\})$, $G_3 = (\{\mathsf{Grace}, \mathsf{Clara}\}, \{(\mathsf{Grace}, \mathsf{child\_of}, \mathsf{Clara})\})$, $G_4 = (\{\mathsf{Clara}, \mathsf{Alice}\}, \{(\mathsf{Clara}, \mathsf{child\_of}, \mathsf{Alice})\})$, $G_5 = (\{\mathsf{David}, \mathsf{Bryan}, \mathsf{Clara}\}, \{(\mathsf{David}, \mathsf{child\_of}, \mathsf{Bryan}), (\mathsf{Bryan}, \mathsf{brother\_of}, \mathsf{Clara})\})$.

The next inconsistency measure, $I_M$, counts the number of minimal inconsistent monotonic subgraphs thinking of each such subgraph as representing one inconsistency.

**Definition 2** (GDB IMs (cont.))**.** *For any GDB $G$ (and set $\Gamma$ of RPCs), $I_M(G) = |\mathsf{MIMS}(G)|$.*

In our example, $I_M(G_{ex}) = I_S(G_{ex}) = 5$. For every GDB $G$, $I_M(G) \leq I_S(G)$, and it is possible to have $I_M(G) < I_S(G)$ [1].

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 [63]. For GDBs, both vertices and edges may be deleted and, because of nonmonotonicity, the addition of edges may restore consistency. So there are three measures analogous to the repair measure, as the addition of isolated vertices cannot restore consistency. 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 $(V, E \cup E')$.

**Definition 3** (GDB Inconsistency Measures (continued))**.** *For any GDB $G$ (and set $\Gamma$ of RPCs), $I_{E-}(G) = \min\{|E'| \mid G - E' \text{ is consistent}\}$, $I_{E+}(G) = \min\{|E'| \mid G + E' \text{ is consistent}\}$, and $I_{V-}(G) = \min\{|V'| \mid G - V' \text{ is consistent}\}$.*

**Example 9.** $I_{E-}(G_{ex}) = 5$ *because the following problematic edges must all be deleted to restore consistency:* $(\mathsf{Frank}, \mathsf{child\_of}, \mathsf{Bryan})$, $(\mathsf{Ester}, \mathsf{child\_of}, \mathsf{Bryan})$, $(\mathsf{Grace}, \mathsf{child\_of}, \mathsf{Clara})$, $(\mathsf{Clara}, \mathsf{child\_of}, \mathsf{Alice})$, $(\mathsf{Bryan}, \mathsf{brother\_of}, \mathsf{Clara})$. *Also,* $I_{E+}(G_{ex}) = 9$ *and* $I_{V-}(G_{ex}) = 3$.

## 4. Complexity

We investigate the combined and the data-complexity [60, 61] of the following problems.

**Definition 4** (Lower (**LV**), Upper (**UV**), and Exact Value (**EV**) )**.** *Let $I$ be an IM. Given a GDB $G$ and a set $\Gamma$ of RPCs, and a positive value $\nu \in \mathbb{N}^{>0}$, $\boldsymbol{LV}_I(G, \Gamma, \nu)$ is the problem of deciding whether $I(G) \geq \nu$. Given $G$ and a non-negative value $\nu' \in \mathbb{N}^{\geq 0}$, $\boldsymbol{UV}_I(G, \Gamma, \nu')$ is the problem of deciding whether $I(G) \leq \nu'$, and $\boldsymbol{EV}_I(G, \Gamma, \nu')$ is the problem of deciding whether $I(G) = \nu'$.*

**Table 1**
Complexity of Lower Value, Upper Value, Exact Value, and Inconsistency Measurement problems. For a complexity class $\mathcal{C}$, $\mathcal{C}$-c means $\mathcal{C}$-complete, while $\mathcal{C}$ means membership in $\mathcal{C}$. ($CNL$ is a class from the counting polynomial hierarchy [64]; see [65] for details on complexity classes).

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

When the input consists of a GDB and a value only (i.e., the set $\Gamma$ 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 $\mathbf{FLV}_I(G, \nu)$, $\mathbf{FUV}_I(G, \nu')$, and $\mathbf{FEV}_I(G, \nu')$.

**Definition 5** (Inconsistency Measurement (**IM**) problem)**.** *Let $I$ be an inconsistency measure. Given a GDB $G$ and a set $\Gamma$ of RPCs, $\mathbf{IM}_I(G, \Gamma)$ 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 $\mathbf{FIM}_I(G)$.

**Theorem 1** ([1])**.** *For any GDB $G$ and a set $\Gamma$ of RPCs, the complexity of the problems defined in Definitions 4 and 5, and of their "Fixed" versions, is as given in Table 1.*

Table 1 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 [65], even under data complexity (it can be also shown that the considered results still hold for restricted classes of RPCs, that is word constraints and non-recursive ones [1]). 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 turn 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 turn out to be the most intractable ones [66, 64]. Still, they are less costly than several inconsistency measures for indefinite relational databases [41] and propositional knowledge bases [30].

### Acknowledgments

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.

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

## References

[1] J. Grant, F. Parisi, On measuring inconsistency in graph databases with regular path constraints, Artif. Intell. 335 (2024) 104197.

[2] 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.

[3] 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 (2022) 71:1–71:37.

[4] A. Hogan, Knowledge graphs: A guided tour (invited paper), in: Proc. of International Research School in Artificial Intelligence in Bergen, (AIB), volume 99, 2022, pp. 1:1–1:21.

[5] 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 (2017) 68:1–68:40.

[6] 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 (2014) 881–892.

[7] K. Rabbani, M. Lissandrini, K. Hose, Extraction of validating shapes from very large knowledge graphs, Proc. VLDB Endow. 16 (2023) 1023–1032.

[8] 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, TGDK 1 (2023) 2:1–2:38.

[9] A. Y. Levy, Combining artificial intelligence and databases for data integration, in: Artificial Intelligence Today: Recent Trends and Developments, Springer, 1999, pp. 249–268.

[10] 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.

[11] 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.

[12] M. V. Martinez, F. Parisi, A. Pugliese, G. I. Simari, V. S. Subrahmanian, Policy-based inconsistency management in relational databases, Int. J. Approx. Reasoning 55 (2014) 501–528.

[13] 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.
[14] 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 (2017) 727–742.
[15] 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.
[16] 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.
[17] 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.
[18] 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 (2021) 62–71.
[19] 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.
[20] 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, TGDK 1 (2023) 5:1–5:33.
[21] B. Xue, L. Zou, Knowledge graph quality management: A comprehensive survey, IEEE Trans. Knowl. Data Eng. 35 (2023) 4969–4988.
[22] 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.
[23] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proc. IEEE 104 (2016) 11–33.
[24] I. H. Sarker, Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective, SN Comput. Sci. 2 (2021) 377.
[25] J. Grant, Classifications for inconsistent theories, Notre Dame Journal of Formal Logic XIX (1978) 435–444.
[26] J. Grant, M. V. Martinez, Measuring Inconsistency in Information, College Publications, 2018.
[27] M. Thimm, On the expressivity of inconsistency measures, Artif. Intell. 234 (2016) 120–151.
[28] K. Mu, Measuring inconsistency with constraints for propositional knowledge bases, Artif. Intell. 259 (2018) 52–90.
[29] G. D. Bona, J. Grant, A. Hunter, S. Konieczny, Classifying inconsistency measures using graphs, J. Artif. Intell. Res. 66 (2019) 937–987.
[30] M. Thimm, J. P. Wallner, On the complexity of inconsistency measurement, Artif. Intell. 275 (2019) 411–456.

[31] P. Besnard, J. Grant, Relative inconsistency measures, Artif. Intell. 280 (2020) 103231.
[32] M. Ulbricht, M. Thimm, G. Brewka, Handling and measuring inconsistency in non-monotonic logics, Artif. Intell. 286 (2020) 103344.
[33] 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.
[34] 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.
[35] H. Decker, D. Martinenghi, Inconsistency-tolerant integrity checking, IEEE Trans. Knowl. Data Eng. 23 (2011) 218–234.
[36] 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.
[37] 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.
[38] O. Issa, A. Bonifati, F. Toumani, Evaluating top-k queries with inconsistency degrees, Proc. VLDB Endow. 13 (2020) 2146–2158.
[39] 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.
[40] 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.
[41] F. Parisi, J. Grant, On measuring inconsistency in definite and indefinite databases with denial constraints, Artif. Intell. 318 (2023) 103884.
[42] 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.
[43] 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.
[44] 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. Reasoning 89 (2017) 58–79.
[45] 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.
[46] J. Grant, F. Parisi, General information spaces: measuring inconsistency, rationality postulates, and complexity, Annals of Math. and Artif. Intell. 90 (2022) 235–269.
[47] 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), volume 1186, 1997, pp. 336–350.
[48] D. Calvanese, G. D. Giacomo, M. Lenzerini, M. Y. Vardi, Rewriting of regular expressions and regular path queries, J. Comput. Syst. Sci. 64 (2002) 443–465.
[49] P. Barceló, G. Fontaine, On the data complexity of consistent query answering over graph databases, J. Comput. Syst. Sci. 88 (2017) 164–194.

[50] 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 (2018) 21:1–21:35.
[51] R. Angles, The property graph database model, in: Proc. of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, volume 2100, 2018.
[52] M. A. Rodriguez, P. Neubauer, Constructions from dots and lines, Bulletin of the American Society for Information Science and Technology 36 (2010) 35–41.
[53] P. T. Wood, Query languages for graph databases, SIGMOD Rec. 41 (2012) 50–60.
[54] P. B. Baeza, Querying graph databases, in: Proceedings of the 32nd ACM Symposium on Principles of Database Systems (PODS), ACM, 2013, pp. 175–188.
[55] 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.
[56] W3C, SPARQL 1.1 Query Language, 2013. Https://www.w3.org/TR/sparql11-query.
[57] 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.
[58] S. Abiteboul, V. Vianu, Regular path queries with constraints, J. Comput. Syst. Sci. 58 (1999) 428–452.
[59] 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.
[60] A. K. Chandra, D. Harel, Computable queries for relational data bases, J. Comput. Syst. Sci. 21 (1980) 156–178.
[61] M. Y. Vardi, The complexity of relational query languages (extended abstract), in: Proc. of Symposium on Theory of Computing (STOC), 1982, pp. 137–146.
[62] J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation (3rd Edition), Addison-Wesley Longman Publishing Co., Inc., 2006.
[63] J. Grant, A. Hunter, Distance-based measures of inconsistency, in: Proc. of ECSQARU, 2013, pp. 230–241.
[64] K. W. Wagner, The complexity of combinatorial problems with succinct input representation, Acta Inf. 23 (1986) 325–356.
[65] C. M. Papadimitriou, Computational complexity, Addison-Wesley, Reading, Massachusetts, 1994.
[66] L. G. Valiant, The complexity of computing the permanent, Theor. Comput. Sci. 8 (1979) 189–201.