Dependencies for Graphs

WENFEI FAN, University of Edinburgh, Beihang University, and Shenzhen Institute of Computing Sciences
PING LU*, BDBC, Beihang University, China

This paper proposes a class of dependencies for graphs, referred to as *graph entity dependencies* (GEDs). A GED is defined as a combination of a graph pattern and an attribute dependency. In a uniform format, GEDs can express graph functional dependencies with constant literals to catch inconsistencies, and keys carrying id literals to identify entities (vertices) in a graph. We revise the chase for GEDs and prove its Church-Rosser property. We characterize GED satisfiability and implication, and establish the complexity of these problems and the validation problem for GEDs, in the presence and absence of constant literals and id literals. We also develop a sound, complete and independent axiom system for finite implication of GEDs. In addition, we extend GEDs with built-in predicates or disjunctions, to strike a balance between the expressive power and complexity. We settle the complexity of the satisfiability, implication and validation problems for these extensions.

CCS Concepts: • **Database Management** → **Systems**;

Additional Key Words and Phrases: Graph dependencies; Conditional functional dependencies; Keys; EGDs; TGDs; Satisfiability, Implication, Validation; Axiom system; Built-in predicates; Disjunction

ACM Reference Format:
Wenfei Fan and Ping Lu. 2018. Dependencies for Graphs. *ACM Trans. Datab. Syst.* 1, 1, Article 1 (January 2018), 56 pages. https://doi.org/0000000.0000000

1 INTRODUCTION

As primitive integrity constraints for relations, functional dependencies (FDs) are found in almost every database textbook. FDs specify a fundamental part of the semantics of data, and have proven important in schema design, query optimization, and prevention of update anomalies. Moreover, FDs and their extensions such as conditional functional dependencies (CFDs) [25] and denial constraints [3] have been widely used in practice to detect semantic inconsistencies and repair relations. Among our most familiar FDs are keys. Keys provide an invariant connection between tuples and the real-world entities they represent, and are crucial to data models and transformations.

The need for FDs and keys is also evident in graphs. Unlike relational data, real-life graphs often do not come with a schema, and dependencies such as FDs and keys provide one of few means for us to specify the integrity and semantics of the data. They are useful in consistency checking, entity resolution, knowledge base expansion, spam and fraud detection, among other things.

*Corresponding author

Authors' addresses: Wenfei Fan, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, Beihang University, 37 Xue Yuan Road, Haidian District, Beijing, Shenzhen Institute of Computing Sciences, Shenzhen, China, wenfei@inf.ed.ac.uk; Ping Lu, BDBC, Beihang University, 37 Xue Yuan Road, Haidian District, Beijing, 100191, China, luping@buaa.edu.cn.

0362-5915/2018/1-ART1 $15.00
https://doi.org/0000000.0000000

Example 1.1. Consider knowledge bases and social networks, which are modeled as graphs.

(1) Consistency checking. It is common to find inconsistencies in real-life knowledge bases, *e.g.,*

- psychologist Tony Gibson is credited for creating Ghetto Blaster, while the video game was actually created by programmer Tony 'Gibbo' Gibson (Yago3);
- both Saint Petersburg and Helsinki are labeled as the capital of Finland (Yago3);
- all birds can fly, and moa are birds, although moa are "flightless" (DBPedia); and
- Philip Sclater is marked as both a child and a parent of William Lutley Sclater (DBPedia).

As shown in [30], such errors can be caught by FDs defined on graphs, referred to as GFDs.

(2) Knowledge base expansion [23]. When adding a newly extracted album to a knowledge base G, to avoid duplicates, we need keys to identify an album entity in G, defined in terms of

ψ_1: its title and the id of its primary artist, or
ψ_2: its title and the year of initial release.

As shown in [23], these can be expressed as keys for graphs. Note that the title of an album and the name of its artist cannot uniquely identify an album. For instance, an American band and a British band are both called "Bleach", and both bands had an album "Bleach".

To cope with ψ_1, we also need a key to identify artists:

ψ_3: the name of the artist, and the id of an album recorded by the artist.

As opposed to our familiar keys for relations, these keys are "recursively defined": to identify an album, we may need to identify its primary artist, and vice versa.

Moreover, to identify albums, we often need to repair data by using, *e.g.,* the following FD:

ψ_4: for any two albums, if their titles, types (*e.g.,* sound track, studio album) and release types [53] (*e.g.,* single, EP) are the same, then initial release of the two albums are the same.

Indeed, as observed in [27], we often need to combine data repairing (fixing data values) with FDs and identifying objects with keys. That is, to identify objects, we need to repair data, and vice versa.

(3) Spam detection. Fake accounts are common in social networks [17]. A rule for identifying spam is as follows, which can also be expressed as an extension of FDs on graphs (see [30]).

- If account x' is confirmed fake, both accounts x and x' like blogs $y_1, \ldots, y_k$, x posts blog z_2, x' posts z_1, and if both z_1 and z_2 have a peculiar keyword c, then x is also fake. □

Moreover, FDs and keys help us optimize queries that are costly on large graphs in the real world (*e.g.,* social graph at Facebook, which has billions of nodes and trillions of edges [36]).

Keys and FDs on graphs are a departure from their relational counterparts. (1) A relational FD $R(X \to Y)$ is defined on a relation schema R with attributes X and Y, where R specifies the "scope" of the FD, *i.e.,* $X \to Y$ is to be applied to tuples in an instance of R. In contrast, graphs are semistructured and are often schemaless. To cope with this, we need a combination of (a) a topological constraint to identify entities, to specify its "scope", *i.e.,* it identifies entities to which the FD is applied; and (b) a dependency on the attributes of the entities identified. (2) Relational FDs and keys are "value-based", while keys and FDs for graphs are often necessarily "id-based" as shown by ψ_1–ψ_3 of Example 1.1. That is, they are based on node identity. In particular, if two vertices are identified as the same entity, then they must have the same attributes and edges.

There has been work on FDs for RDF [2, 16, 19, 38, 40, 42, 56] in particular and for property graphs [30] in general, and on keys for RDF [23]. However, many questions remain open. For example, as opposed to relational FDs and keys, none of the FD proposals can express keys for graphs [23], which are based on node identity and are possibly "recursively defined" (Example 1.1).

The practical need calls for a full treatment of the topic, to answer the following questions. (1) Is there a simple class of graph dependencies for us to uniformly express FDs and keys? (2) Can we adapt the chase [51] to reason about the dependencies? (3) What is the complexity of fundamental problems associated with the dependencies? (4) Is there a finite axiom system for their implication analysis, along the same lines as Armstrong's axioms for traditional FDs [6]? (5) How can we strike a balance between their expressive power and complexity?

Contributions. This paper tackles these questions.

(1) GEDs. We propose a class of dependencies, referred to as *graph entity dependencies* and denoted by GEDs (Section 3). A GED is a combination of (a) a graph pattern Q as a topological constraint, and (b) a dependency $X \Rightarrow Y$ with sets X and Y of equality literals, which can encode FDs. The pattern Q identifies a set of entities in a graph, and $X \Rightarrow Y$ is enforced on these entities. GEDs may specify conditions carrying literals with constants, like relational CFDs [25]. They may carry id literals to identify vertices in a graph, beyond equality on attribute values.

GEDs are defined on general property graphs, and can uniformly express FDs and keys on graphs, referred to as GFDs and GKeys, respectively; GFDs and GKeys correspond to the GFDs of [30] and the keys of [23], respectively (subject to adaption of graph pattern matching with graph homomorphism instead of subgraph isomorphism). They can express traditional FDs, CFDs and equality-generating dependencies (EGDs [8]), when relations are represented as graphs. That is, GEDs are able to do the job of keys, FDs, CFDs and EGDs for graph-structured data, *e.g.,* to specify integrity, detect inconsistencies, identify entities and optimize queries, among other things. Better still, as a combination of GFDs and GKeys, GEDs are useful in (i) cross-device identity matching and fraud detection by characterizing associations among entities, and (ii) graph cleaning by interleaving data repairing with GFDs and object identification with GKeys (see Example 4.4).

(2) The chase revised. We extend the chase [51] to GEDs (Section 4). Chasing with GEDs is more involved than with traditional FDs: it may run into conflicts introduced by id literals or constant literals, and may "generate" new attributes when enforcing GEDs on a schemaless graph. Nonetheless, we show that the chase with GEDs is finite and has the Church-Rosser property. That is, all chasing sequences of a graph (pattern) by a set of GEDs are finite and yield the same result, regardless of the order of GEDs applied, retaining the nice features of the chase for relational FDs.

(3) Classical problems. We study three fundamental problems associated with GEDs (Section 5).

(a) The *satisfiability* problem is to decide, given a set Σ of GEDs, whether there exists a nonempty finite model G of Σ that satisfies Σ, denoted by $G \models \Sigma$ as usual.

(b) The *implication* problem is to decide whether a set Σ of GEDs entails another GED φ, denoted by $\Sigma \models \varphi$, *i.e.,* for any finite graph G, if $G \models \Sigma$, then $G \models \varphi$.

(c) The *validation* problem is to decide, given a finite graph G and a set Σ of GEDs, whether $G \models \Sigma$.

These problems not only are of theoretical interest, but also find practical applications. The satisfiability analysis helps us check whether a set of GEDs (possibly discovered from "dirty" graphs) makes sense before the GEDs are used as rules for data cleaning or query optimization. The implication analysis serves as an optimization strategy to get rid of redundant rules. The validation analysis can detect violations of GEDs, and catch "dirty" entities.

To understand where the complexity arises, we consider two dichotomies in the study:

- the presence of id literals vs. their absence, and
- the presence of constant values vs. their absence.

Dependencies	Satisfiability	Implication	Validation	Connection with GEDs
GEDs	coNP-c (Th. 5.4)	NP-c (Th. 5.12)	coNP-c (Th. 5.16)	$Q[\bar{x}](X \Rightarrow Y)$
GFDs	coNP-c (Th. 5.4)	NP-c (Th. 5.12)	coNP-c (Th. 5.16)	GEDs without id literals
GKeys	coNP-c (Th. 5.4)	NP-c (Th. 5.12)	coNP-c (Th. 5.16)	$Q[\bar{x}](X \Rightarrow x.\text{id} = y.\text{id})$
GED_xs	coNP-c (Th. 5.4)	NP-c (Th. 5.12)	coNP-c (Th. 5.16)	GEDs without constant literals
GFD_xs	$O(1)$ (Th. 5.4)	NP-c (Th. 5.12)	coNP-c (Th. 5.16)	GFDs without constant literals
GDCs	Σ_2^p-c (Th. 7.2)	Π_2^p-c (Th. 7.2)	coNP-c (Th. 7.2)	adding built-in predicates
$\text{GED}^\vee$s	Σ_2^p-c (Th. 7.4)	Π_2^p-c (Th. 7.4)	coNP-c (Th. 7.4)	disjunctive Y in $Q[\bar{x}](X \Rightarrow Y)$

Table 1. Complexity for reasoning about GEDs

For instance, GFDs do not allow id literals and can only enforce value equality, while GKeys support id literals and can enforce two nodes to be the same, merging their corresponding attributes. In these settings, we characterize GED satisfiability and implication, based on the chase. We also establish complexity bounds of these problems for GEDs, GFDs, keys and other sub-classes, all matching, as summarized in Table 1. As opposed to relational FDs, these problems are all intractable for GEDs. The complexity is, however, comparable to, *e.g.,* (a) relational CFDs, for which the satisfiability and implication problems are NP-complete and coNP-complete, respectively [25], and (b) EGDs, for which the implication problem is NP-complete [9, 35]. The intractability for GEDs is quite robust even for restricted special cases such as GEDs defined with tree patterns. No prior work has studied the complexity of the classical problems for graph dependencies with tree patterns.

(4) Finite axiomatizability. We study the finite axiomatizability of GEDs (Section 6). One naturally wants a finite set $\mathcal{A}$ of inference rules that is sound and complete for the implication analysis of GEDs, along the same lines as Armstrong's axioms for relational FDs (see [1]). That is, for any set Σ of GEDs and another GED φ, $\Sigma \models \varphi$ if and only if φ is provable from Σ using $\mathcal{A}$. Here we focus on finite graphs and study finite implication, rather than unrestricted implication.

We provide a set of six inference rules for GEDs, and show that it is sound and complete for GED implication analyses, based on the revised chase. We also show that the axiom system is independent (non-redundant and minimal), *i.e.,* removing any rule makes it no longer complete.

(5) Extensions. To strike a balance between the expressivity and complexity, we investigate extensions of GEDs (Section 7). We extend GEDs by supporting

- built-in predicates $=, \neq, <, >, \leq, \geq$ (denoted by GDCs); or
- limited disjunction of literals (denoted by $\text{GED}^\vee$s).

To the best of our knowledge, no previous work has studied graph dependencies defined in terms of built-in predicates or disjunction. As special cases, we can express denial constraints [3] as GDCs with built-in predicates, disjunctive EGDs [21] as $\text{GED}^\vee$s, and "domain constraints" for attributes of an entity to have a finite domain as both GDCs and $\text{GED}^\vee$s, among other things. With the increased expressive power, the extensions increase the complexity of static analyses. We show that their satisfiability and implication problems become Σ_2^p-complete and Π_2^p-complete, as opposed to coNP-complete and NP-complete for GEDs, respectively. Their validation problems remain coNP-complete, the same as for GEDs (see Table 1).

The dependency classes studied in the paper and their complexity results are summarized in Table 1, annotated with their corresponding theorems. This work is a preliminary step toward developing a dependency theory for graphs. The intractability results reveal the challenges inherent to entities with a graph structure. The revised chase, characterizations of satisfiability and implication, and axiom system provide insight into the analyses of graph dependencies.

Related work. This paper extends its conference version [29] as follows. (1) We provide stronger lower bound results for the satisfiability, implication and validation problems for GEDs and their special cases (Theorems 5.4, 5.12 and 5.16). More specifically, Lemmas 5.6, 5.7, 5.14, 5.15, 5.17 and 5.18 are new, establishing the intractability of these problems for GFDs, GKeys and GFD$_x$s that are defined with tree patterns only. (2) We provide the detailed proofs of all results, which were not given in [29]. Some of the proofs are nontrivial and are interesting in their own right. (3) The description of various concepts is expanded, improving the readability of the paper.

We categorize other related work as follows.

Relational dependencies. FDs were introduced in [18] and have been well studied for relations. Armstrong's axioms were proposed for FDs in [6], and the chase was introduced in [51]. EGDs and TGDs were defined in [8]. There was also renewed interest in extending FDs to improve data quality, *e.g.,* denial constraints [3] and CFDs [25] (see [1, 22, 24] for surveys).

For relational FDs, the satisfiability, implication and validation problems are in $O(1)$, linear time and PTIME, respectively (cf. [1]). Similar to the strong notion of satisfiability studied in this work, a consistency problem was shown undecidable for TGDs [35]; the implication problem is known to be NP-complete and undecidable for EGDs and TGDs, respectively [9]; and the validation problem was shown coNP-complete for EGDs [9] and Π_2^p-complete for TGDs [48]. The satisfiability, implication and validation problems are NP-complete, coNP-complete and in PTIME for CFDs [25], respectively. The satisfiability and implication problems are NP-complete and coNP-complete for denial constraints [7], respectively. An axiom system of four rules was developed for EGDs in [50], while TGDs are not finitely axiomatizable for finite implication. A set of four rules was shown sound and complete for the (finite) implication analysis of CFDs [25].

GEDs carry graph patterns and id literals, and are more expressive than EGDs. Their satisfiability, implication and validation problems are intractable. However, their static analyses bear complexity comparable to their counterparts for denial constraints, CFDs and EGDs. Moreover, GEDs have the finite axiomatizability and the Church-Rosser property of the chase, as for relational FDs.

One might want to encode GEDs as relational dependencies and employ relational techniques to reason about GEDs. However, (a) id literals and graph patterns with wildcard complicate the encoding; and (b) it is not clear what we can get from an encoding. To express GEDs we need both EGDs and limited TGDs. Indeed, as will be shown in Sections 3 and 4, enforcing a GED may add new attributes, beyond the capacity of EGDs. Reasoning about generic TGDs is beyond reach [9, 35]. While some special cases have been studied (*e.g.,* oblivious terminating TGDs and EGDs [43, 44]), their syntactic characterization is not yet in place, and their fundamental problems such as satisfiability and validation are still open. It is not clear whether GEDs can be expressed in any of these special cases, and even so, what results GEDs can inherit from them. Moreover, graph patterns in GEDs explicitly characterize associations among entities, which are useful in, *e.g.,* fraud detection. In addition, native graph techniques allow us to make use of, *e.g.,* the data locality of graph homomorphism to check GEDs and hence yield efficient implementation strategies, which are not offered by relational reasoning methods for EGDs and TGDs. In light of these, we give a native definition of GEDs and develop their proofs directly. (c) The chase and axiom system for GEDs are quite different from their counterparts in the relational setting. For instance, chasing with GEDs may expand a graph with new attributes and run into conflicts, in contrast to with EGDs.

FDs *for graphs*. Graph constraints are being investigated by W3C [41] and the industry (*e.g.,* [46]). The constraints currently supported are quite simple (*e.g.,* uniqueness constraints, cardinality constraints and property paths); a "standard" form of FDs is not yet in place. However, there have been several research proposals for FDs on RDF graphs. This line of work started from [42]. It defines keys, foreign keys and FDs by extending relational methods to RDF, and interpreting the "scope" of an FD with a class type that represents a relation name. Using clustered values, [56] defines FDs with conjunctive path patterns, which were extended to CFDs [38]. FDs are also defined by mapping relations to RDF [16], with tree patterns in which nodes represent relation attributes. As opposed to class names [42], tree patterns [16] and path patterns [38, 56], GEDs are specified with (possibly cyclic) graph patterns with variables and node identities.

Closer to this work are [2, 19, 39, 40] for RDF. A class of EGDs was formulated in [2] in terms of RDF triple patterns with variables, which are interpreted with homomorphism and triple embedding. Along the same lines, FDs, tuple-generating dependencies (TGDs) and forbidding dependencies were defined for RDF in [19]. FDs and EGDs were extended in [40] and [2] to support constants like CFDs [25]. Chasing algorithms were developed in [2, 39, 40] for the implication analysis of EGDs and FDs, extending relational techniques to RDF triples. The decidability of the implication and validation problems was established in [19] for the EGDs (and hence FDs), among other things. Finite axiom systems were provided for the EGDs, TGDs, and for EGDs and TGDs put together, consisting of 9, 5 and 16 rules, respectively [2, 19]. Axiom systems were also provided for various classes of FDs over relations of an arbitrary arity [39, 40], with 13 rules for the general case.

This work differs from [2, 19, 39, 40] in the following.

(1) GEDs are defined for general property graphs, not limited to RDF. (a) GEDs distinguish node identity from value equality. Their id literals enforce that nodes identified have the same attributes and edges. (b) GEDs can enforce generation of new attributes, a useful feature that is not supported by [2, 19, 39, 40]. (c) GEDs can express GFDs, GKeys and forbidding dependencies in a uniform format (Section 3). (d) GEDs support constant literals, beyond [19, 39].

(2) Our revised chase differs from the prior work in the following. (a) We study the chase of a graph (pattern) by GEDs, not limited to the implication analysis. For instance, the chase also helps us characterize the satisfiability analysis. (b) Chasing with GEDs has to deal with id literals, a major cause of invalid steps. It may also add new attributes as enforced by GEDs. (c) We establish the Church-Rosser property of the chase, which was not considered in [2, 19, 39, 40].

(3) We provide characterizations of the static analyses of GEDs, and the complexity of the satisfiability, implication and validation problems for GEDs in various settings. The satisfiability problem was not studied for EGDs or FDs of [2, 19, 39, 40]. Moreover, the complexity bounds remain to be developed for the implication and validation problems of their EGDs and FDs.

(4) The axiom system $\mathcal{A}_{\mathrm{GED}}$ differs from [2, 19, 39, 40] in the following. (a) Besides value-based reasoning, $\mathcal{A}_{\mathrm{GED}}$ deals with id-based deduction to enforce the semantics of node identities. (b) It adopts graph pattern matching in property graphs, beyond RDF and relations. (c) $\mathcal{A}_{\mathrm{GED}}$ allows attribute generation (Section 4), which is not supported by the axiom systems for EGDs and FDs [2, 19, 39, 40]. While this can be derived from TGDs and EGDs of [19] put together, the finite axiomatizability for finite implication of TGDs requires further investigation [9].

As remarked in Section 5, a class of keys was studied for RDF [23]. Over property graphs, a form of GFDs [30] was defined with a graph pattern Q that is interpreted via subgraph isomorphic mapping. These GFDs can express CFDs [25] when tuples are represented as vertices in a graph, but cannot express keys of [23]. The satisfiability, implication and validation problems are shown coNP-complete, NP-complete and coNP-complete, respectively, for GFDs of [30].

This work differs from the prior work [23, 30] as follows.

(1) GEDs can uniformly express GFDs and GKeys. Moreover, to simplify the definition of the

keys of [23] and to reason about GFDs and keys in a uniform framework, GEDs adopt the graph homomorphism semantics for graph pattern matching, as opposed to subgraph isomorphism [23, 30] (see Section 3). That is, we consider GFDs and GKeys under the homomorphism semantics.

(2) We revise the chase for GEDs, which was not studied in [30]. A form of chase was studied for keys [23], which is a simple special case of the general process studied here.

(3) We establish the complexity of the satisfiability, implication and validation problems for GEDs in various settings. These were not studied in [23], and were considered for GFDs of [30] only. As remarked earlier, we employ characterizations and proof techniques different from [30] to cope with different semantics of graph pattern matching, *e.g.,* the chase to prove upper bounds. We also give lower bounds for GKeys, GFD_xs and GED_xs, which were not studied before. In addition, we show that the lower bounds of these problems remain intact for GEDs, GFDs and GKeys defined with tree patterns only, which were not considered in [23, 30].

(4) We provide a finite axiom system for GEDs, which was not studied for GFDs and keys [23, 30].

(5) To the best of our knowledge, no previous work has studied graph dependencies defined in terms of built-in predicates or disjunction, including [23, 30].

The chase has also been studied for relational data exchange (with source-to-target TGDs and target disjunctive EGDs [14] or FDs [10, 45]), data repairing [34], query rewriting [20], ontology querying [15], and optimizing SPARQL queries [54] with the constraints of [42]. In contrast, the chase of a graph by GEDs handles attribute and label conflicts introduced by id literals, and enforces GEDs in the absence of schema; it is quite different from the relational counterparts.

FDs *for XML*. Keys [12, 28] and FDs [5] were also studied for XML, which are quite different from GEDs in formulation and semantics. Thus the results on XML do not apply to GEDs and vice versa.

Keys for XML were proposed in [12], defined in terms of paths instead of graph patterns. For a set Σ of keys for XML, it is always possible to find an XML document satisfying Σ [13], and its implication problem is in PTIME [13]. In contrast, the satisfiability and implication problems are coNP-complete and NP-complete for GKeys, respectively, even for GKeys with tree patterns.

In the presence of DTDs, the satisfiability and implication problems are undecidable for XML keys and foreign keys put together, and are NP-complete and coNP-complete [28], respectively, when keys and foreign keys are unary [28]. These problems are in linear-time when we consider keys under DTDs alone [28]. A form of XML keys was proposed by W3C [55], which differs from the keys of [12] in the path languages for specifying targets [4]. In the presence of XML Schema, the satisfiability and the implication problems are EXPTIME-hard for these XML keys [4]. In contrast, we consider schemaless graphs since real-life graphs typically do not come with a schema.

A class of FDs was studied in [5], for which the implication problem is in PTIME even in the presence of DTDs [5], in contrast to the NP-completeness for GFDs (even with tree patterns).

Novelty. Taken together, the novelty of this work consists of the following.

(1) This work proposes GEDs, a class of graph dependencies that is not a simple adaption of any known class of relational dependencies to graphs, as indicated by differences in expressive power, complexity bounds and proof techniques. GEDs are defined in terms of graph patterns as topological constraints, to characterize associations among entities in a graph; this makes it possible to apply native graph techniques to the analyses of the dependencies, *e.g.,* the locality of graph homomorphism, which is not offered by traditional techniques for relational dependencies.

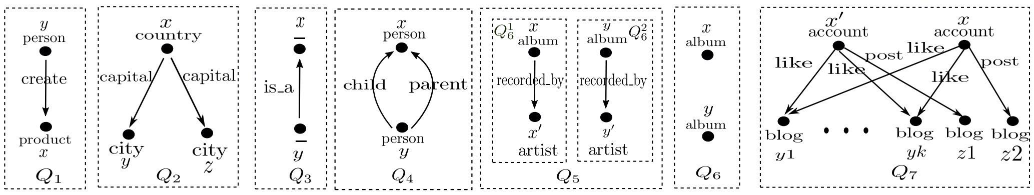

Fig. 1. Graph patterns

(2) As graph dependencies, GEDs are the first uniform formalism to express our familiar functional dependencies and keys on graphs, among other things. In addition, this work makes a first effort to provide foundations for graph dependencies that are analogous to the classical (relational) dependencies theory, including the chase with the Church-Rosser property, a finite axiom system that is sound, complete and independent, characterizations of the satisfiability and implication analyses, and nontrivial complexity bounds for fundamental problems associated with graph dependencies. Furthermore, it is the first study of graph dependencies with built-in predicates and (limited) disjunction, and helps us strike a balance between the complexity and expressive power.

2 PRELIMINARIES

Before we define GEDs, we first review some basic notations. Assume three countably infinite sets Γ, Υ and U of labels, attributes and constants, respectively.

Graphs. A *graph* G is specified as (V, E, L, F_A), where (a) V is a finite set of nodes; (b) $E \subseteq V \times \Gamma \times V$ is a finite set of edges, in which (v, ι, v') denotes an edge from node v to v', and the edge is labeled with ι, referred to as its *label*; (c) each node $v \in V$ has label $L(v)$ from Γ; and (d) each node $v \in V$ carries a tuple $F_A(v) = (A_1 = a_1, \ldots, A_n = a_n)$ of *attributes* of a finite arity, where $A_i \in \Upsilon$ and $a_i \in U$, written as $v.A_i = a_i$, and $A_i \neq A_j$ if $i \neq j$. In particular, each node $v \in V$ carries a special attribute id denoting its *node identity*.

That is, we consider finite directed graphs in which nodes and edges are labeled. Nodes carry attributes for, *e.g.,* properties, keywords and ratings, as in property graphs. Unlike relational databases, we assume no schema for graphs. As usually found in practice, for an attribute $A \in \Upsilon$ and a node $v \in V$, $v.A$ may not exist, except that v has a unique attribute $v.\mathrm{id}$ denoting its identity, *i.e.,* when two nodes have the same value for their id attribute, the two are the same node.

Graph patterns. A *graph pattern* is a directed graph $Q[\bar{x}] = (V_Q, E_Q, L_Q)$, where (1) V_Q (resp. E_Q) is a finite set of pattern nodes (resp. edges); (2) L_Q is a function that assigns a label $L_Q(u)$ (resp. $L_Q(e)$) to each node $u \in V_Q$ (resp. edge $e \in E_Q$); and (3) $\bar{x}$ is a list of distinct variables, each denoting a node in V_Q. Labels of pattern nodes and edges are drawn from Γ. We allow wildcard '_' as a special label for nodes or edges in Q.

We will use the following notion to define keys.

A pattern $Q_2[\bar{y}]$ is a *copy* of $Q_1[\bar{x}]$ via a bijection $f: \bar{x} \mapsto \bar{y}$ if $Q_2[\bar{y}]$ is $Q_1[\bar{x}]$ with variables renamed by f. More specifically, let $Q_1[\bar{x}] = (V_{Q1}, E_{Q1}, L_{Q1})$ and $Q_2[\bar{y}] = (V_{Q2}, E_{Q2}, L_{Q2})$. Then (a) $\bar{x}$ and $\bar{y}$ are disjoint, and (b) f is an isomorphism from Q_1 to Q_2; *i.e.,* for each $x \in \bar{x}$, $L_{Q1}(x) = L_{Q2}(f(x))$; and (x_1, ι, x_2) is an edge in E_{Q_1} if and only if $(f(x_1), \iota, f(x_2))$ is in E_{Q_2}, with the same label ι.

Example 2.1. Figure 1 depicts seven graph patterns, which are borrowed from [23, 30] and will be used to specify entities and associations listed in Example 1.1. (1) Pattern $Q_1[x, y]$ specifies a person entity y and a product entity x, which are connected by a create edge; among other things, Q_1 can specify a video game and its creator (see Example 1.1); similarly for $Q_2[x, y, z]$ and

$Q_4[x, y]$. (2) Pattern $Q_3[x, y]$ shows a generic is_a relationship between two entities labeled with wildcard '_' [30]. (3) Pattern $Q_5[x, x', y, y']$ consists of (a) a pattern $Q_5^1[x, x']$ with variables x and x', specifying a relationship between an album entity x and an artist entity x'; and (b) a copy $Q_5^2[y, y']$ of $Q_5^1[x, x']$ with variables renamed; we will define a key using Q_5 for knowledge base expansion; similarly for $Q_6[x, y]$ [30]. (4) Pattern $Q_7[x, x', z_1, z_2, y_1, \ldots, y_k]$ specifies two accounts x and x', $k + 2$ blogs $z_1, z_2, y_1, \ldots, y_k$ and their relationships [30]; we will use Q_7 for spam detection. □

Matches. We say that a label ι *matches* ι', denoted by $\iota \asymp \iota'$, if either (a) ι and ι' are in Γ and $\iota = \iota'$, or (b) $\iota' \in \Gamma$ and ι is '_', *i.e.,* the wildcard matches any label in Γ.

A *match* of pattern $Q[\bar{x}]$ in graph G is a homomorphism h from Q to G such that (a) for each node $u \in V_Q$, $L_Q(u) \asymp L(h(u))$; and (b) for each edge $e = (u, \iota, u')$ in Q, there exists an edge $e' = (h(u), \iota', h(u'))$ in G such that $\iota \asymp \iota'$. Note that when ι is wildcard '_', there may exist multiple edges e' with $\iota \asymp \iota'$. The match picks one of them, and denotes it by $h(\iota^u_{u'})$. Note that distinct wildcards (on different edges) may have different label valuations.

Abusing the notations, we also denote the match as a vector $h(\bar{x})$ if it is clear from the context, where $h(\bar{x})$ consists of $h(x)$ for all variables $x \in \bar{x}$. Intuitively, $\bar{x}$ is a list of entities to be identified by pattern Q, and $h(\bar{x})$ is an instantiation of $\bar{x}$ in graph G, one node for each entity.

3 GRAPH ENTITY DEPENDENCIES

We now define *graph entity dependencies* (GEDs).

GEDs. A GED φ is defined as $Q[\bar{x}](X \Rightarrow Y)$, where $Q[\bar{x}]$ is a graph pattern, and X and Y are two (possibly empty) sets of literals of $\bar{x}$. To simplify the discussion, we refer to $Q[\bar{x}]$ and $X \Rightarrow Y$ as the *pattern* and FD of φ, respectively.

A *literal* of $\bar{x}$ is one of the following: for $x, y \in \bar{x}$,

(a) *constant literal* $x.A = c$, where c is a constant in U, and A is an attribute in Υ that is not id;

(b) *variable literal* $x.A = y.B$, where A and B are attributes in Υ that are not id; or

(c) id *literal* $x.\text{id} = y.\text{id}$.

Intuitively, GED φ is a combination of (1) a *topological constraint* imposed by graph pattern Q, to identify entities in a graph, and (2) an attribute dependency $X \Rightarrow Y$ that subsumes an FD (to be clarified shortly), to be applied to the entities identified by Q. Constant literals $x.A = c$ enforce bindings of semantically related constants, along the same lines as relational CFDs [25]. An id literal $x.\text{id} = y.\text{id}$ states that x and y denote the same vertex (entity).

Example 3.1. We can use GEDs as data quality rules to detect the inconsistencies and identify entities observed in Example 1.1. These rules are borrowed from [23, 30] to show how GEDs express the dependencies studied there. The GEDs are defined with graph patterns depicted in Fig. 1.

(1) GED $\varphi_1 = Q_1[x, y](X_1 \Rightarrow Y_1)$. Here X_1 consists of a single constant literal $x.\text{type}$ = "video game", Y_1 consists of a literal $y.\text{type}$ = "programmer", and type is an attribute of person and product (not shown in Q_1). It states that a video game can only be created by programmers.

(2) GED $\varphi_2 = Q_2[x, y, z](\emptyset \Rightarrow y.\text{name} = z.\text{name})$. It states that if a country x has two capitals y and z, then y and z must have the same name. Here name is an attribute, X is empty, and Y consists of a single variable literal.

(3) GED $\varphi_3 = Q_3[x, y](x.A = x.A \Rightarrow y.A = x.A)$, where A is an attribute of x, *e.g.,* can_fly. It says that if y is_a x and if x has property A, then y inherits $x.A$, *i.e.,* y also has attribute A and $y.A = x.A$. Note that x and y are labeled '_', representing generic entities regardless of their labels.

(4) GED $\varphi_4 = Q_4[x,y](\emptyset \Rightarrow \text{false})$, where false is syntactic sugar representing the Boolean constant that can be expressed by $y.A = c$ and $y.A = d$ for distinct constants c and d. The GED states that graph pattern Q_4 is "illegal", *i.e.,* no person can be both a child and a parent of another person.

GEDs φ_1–φ_4 catch the semantic inconsistencies in knowledge bases described in Example 1.1. For instance, φ_3 can detect the inconsistency between birds and moa in DBPedia.

(5) The keys of Example 1.1 can also be expressed as GEDs:

For album: $\psi_1 = Q_5[x,x',y,y'](X_5 \Rightarrow x.\text{id} = y.\text{id})$,
$\psi_2 = Q_6[x,y](X_6 \Rightarrow x.\text{id} = y.\text{id})$.
For artist: $\psi_3 = Q_5[x,x',y,y'](X_7 \Rightarrow x'.\text{id} = y'.\text{id})$.

Here X_5 consists of a variable literal $x.\text{title} = y.\text{title}$ and an id literal $x'.\text{id} = y'.\text{id}$; X_6 includes variable literals $x.\text{title} = y.\text{title}$ and $x.\text{release} = y.\text{release}$; and X_7 consists of a variable literal $x'.\text{name} = y'.\text{name}$ and an id literal $x.\text{id} = y.\text{id}$, defined with attributes title, release, name and id.

To identify a pair of album entities x and y, we check either their title attributes and the ids of their artists (ψ_1), or their title and release attributes (ψ_2). Similarly, to identify artist entities x' and y' as required by ψ_1, we need to check the ids of a pair of albums they recorded (ψ_3) in turn.

The FD for albums is expressed as GED $\psi_4 = Q_6[x,y](X_8 \Rightarrow x.\text{release} = y.\text{release})$, where X_8 consists of variable literals $x.\text{title} = y.\text{title}$, $x.\text{type} = y.\text{type}$ and $x.\text{release\_type} = y.\text{release\_type}$. □

Semantics. To interpret GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$, we use the following notations. Consider a match $h(\bar{x})$ of Q in a graph G, and a literal l of $\bar{x}$.

We say that $h(\bar{x})$ *satisfies* l, denoted by $h(\bar{x}) \models l$, if (a) when l is $x.A = c$, then attribute $v.A$ *exists* at node $v = h(x)$, and $v.A = c$; (b) when l is $x.A = y.B$, then attributes A and B *exist* at $v = h(x)$ and $w = h(y)$, respectively, and $v.A = w.B$; and (c) when l is $x.\text{id} = y.\text{id}$, then $h(x)$ and $h(y)$ refer to the same node; hence, they have the same set of attributes and edges.

We denote by $h(\bar{x}) \models X$ if the match $h(\bar{x})$ satisfies *all* the literals in X; in particular, if X is $\emptyset$, then $h(\bar{x}) \models X$ for any match $h(\bar{x})$ of Q in G. We write $h(\bar{x}) \models X \Rightarrow Y$ if $h(\bar{x}) \models X$ implies $h(\bar{x}) \models Y$.

A graph G *satisfies* GED φ, denoted by $G \models \varphi$, if *for all* matches $h(\bar{x})$ of Q in G, $h(\bar{x}) \models X \Rightarrow Y$. Given the semantics, we also write $Q[\bar{x}](X \Rightarrow Y)$ as $Q[\bar{x}]\big(\bigwedge_{l \in X} l \Rightarrow \bigwedge_{l' \in Y} l'\big)$.

A graph G *satisfies* a set Σ of GEDs if for all $\varphi \in \Sigma$, $G \models \varphi$, *i.e.,* G satisfies each GED in Σ.

Existence of attributes. In a GED $Q[\bar{x}](X \Rightarrow Y)$, attributes are not specified in pattern Q, and we consider schemaless graphs. For a literal $x.A = c$, node $h(x)$ does not necessarily have attribute A, to accommodate the semistructured nature of graphs. Hence, (a) when $x.A = c$ is a literal in X, if $h(x)$ has *no* A-attribute, then $h(\bar{x})$ trivially satisfies $X \Rightarrow Y$ by the definition of $h(\bar{x}) \models X$. (b) In contrast, if $x.A = c$ is in Y, then for $h(\bar{x}) \models Y$, node $h(x)$ must have A-attribute; similarly for $x.A = y.B$. That is, the literals in Y can be used to enforce nodes to have certain attributes.

As a consequence, we can use, *e.g.,* $Q[x](\emptyset \Rightarrow x.A = x.A)$ to enforce that all entities x of "type" τ must have an A attribute, where Q consists of a single vertex x labeled τ. This is in the flavor of tuple generating dependencies [8], limited to attributes. Such constraints cannot be expressed as EGDs [8] or FDs for relations and RDF [2, 19, 40, 56]. However, GEDs *cannot* enforce attribute $x.A$ to have a finite domain (*e.g.,* Boolean) as opposed to database schema.

Special cases. We list some special cases of GEDs.

(1) GFDs. GFDs are syntactically defined as GEDs without id literals, *i.e.,* $Q[\bar{x}](X \Rightarrow Y)$ in which neither X nor Y contains any id literal $x.\text{id} = y.\text{id}$. We use the same syntax as GFDs of [30], but interpret graph pattern matching in terms of homomorphism as opposed to isomorphism [30].

For instance, φ_1–φ_4 and ψ_4 in Example 3.1 are GFDs, but ψ_1–ψ_3 are not.

(2) Keys. A key ψ of [23] is defined as $Q[\bar{x}, x_o]$, where $Q[\bar{x}]$ is a graph pattern and x_o is a designated node in $\bar{x}$. A graph G satisfies ψ if for any two matches $h(\bar{x})$ and $h'(\bar{x})$ of $Q[\bar{x}]$ in G such that $h(\bar{x})$ and $h'(\bar{x})$ are isomorphic, $h(x_o)$ and $h'(x_o)$ denote the same node. Pattern Q is defined as a set of RDF triples with variables and constants, and is interpreted in terms of of subgraph isomorphism.

We define a *key for graphs*, denoted by GKey, as a GED of the form $Q[\bar{z}](X \Rightarrow x_0.\text{id} = y_0.\text{id})$, where (a) $Q[\bar{z}]$ is composed of patterns $Q_1[\bar{x}]$ and $Q_2[\bar{y}]$, $Q_2[\bar{y}]$ is a copy of $Q_1[\bar{x}]$ via a bijection f: $\bar{x} \mapsto \bar{y}$ (see Example 2.1), and variables in $\bar{x}$ and $\bar{y}$ are disjoint; (b) $\bar{z}$ consists of $\bar{x}$ followed by $\bar{y}$, (c) $x_0 \in \bar{x}$ and $y_0 = f(x_0)$ are designated nodes in Q, and (d) X is a set of literals of $\bar{x}$ and $\bar{y}$, which can be constant literals, variable literals, or id literals as before.

For instance, ψ_1, ψ_2 and ψ_3 of Example 3.1 are GKeys.

GKeys can be "recursively defined" in terms of id literals. A key $\psi = Q[\bar{x}, x_o]$ of [23] can be syntactically expressed as a GKey $Q'[\bar{z}](X \Rightarrow x_0.\text{id} = y_0.\text{id})$, where X consists of literals for constant and variable bindings embedded in pattern Q', and Q' is composed of Q and a copy of Q. We interpret the key in terms of homomorphism instead of three isomorphic mappings [23].

It is to uniformly express keys and GFDs that we adopt the homomorphism semantics for graph pattern matching. To illustrate this, consider GKey ψ_3 given in Example 3.1. The GKey catches no violations if it is interpreted under subgraph isomorphism. Indeed, for any match $h[\bar{x}]$ of pattern Q_5 in a graph G, $h(x)$ and $h(y)$ have to be distinct nodes as required by isomorphism. As a result, $h[\bar{x}] \not\models X_7$ and hence, $h[\bar{x}]$ trivially satisfies ψ_3. As opposed to [23] that interprets a key with three isomorphic mappings, we interpret GEDs with a single match of pattern, and thus isomorphism is too strict to allow two variables to be mapped to the same node.

The example above tells us that if we adopt the isomorphism semantics, if an id literal $x.\text{id} = y.\text{id}$ appears in X of a GED $Q[\bar{x}](X \Rightarrow Y)$, then x and y have to be mapped to distinct nodes, and hence, any match of Q trivially satisfies the GED. When id literals appear in Y, the issue becomes more subtle when it comes to the satisfiability of a set Σ of GEDs (see Section 5.1), where a model of Σ requires that every GED in Σ finds a match of its pattern, to assure that the GEDs in Σ do not conflict with each other. Consider a GKey $\varphi = Q[x, y](\emptyset \Rightarrow x.\text{id} = y.\text{id})$, where Q consists of two isolated nodes, which are labeled "UoE", a short for "University of Edinburgh". This GKey states that all nodes labeled "UoE" are essentially the same node. One can verify that under the isomorphism semantics, GKeys like φ cannot find a model in any sensible graph. It is to deal with these complications that keys defined in [23] have to adopt three isomorphic mappings.

Note that GFDs and GKeys defined in this paper are not exactly the same as the ones given in [23, 30], since we adopt the homomorphism semantics, while [23, 30] adopt the isomorphism semantics. Nonetheless, GFDs and GKeys under the two semantics have similar properties. In this paper, we uniformly study the properties of GFDs and GKeys with the homomorphism semantics.

We allow id literals to appear in X such that we can continuously apply (recursively defined) keys without physically merging the nodes in a graph G. That is, when we identify nodes u and v in G, we do not have to immediately modify G by merging the two into one; instead, we simply let $u.\text{id} = v.\text{id}$, and apply other GEDs to G, possibly capitalizing on $u.\text{id} = v.\text{id}$. Moreover, with id literals in X we can easily keep track of propagation of object identification.

(3) GED$_\text{x}$s. We also study the class of GFDs that include no constant literals, referred to as *variable* GFDs and denoted by GFD$_\text{x}$s. For instance, φ_2, φ_3 and ψ_4 are GFD$_\text{x}$s, but φ_1 and φ_4 are not. Intuitively, (a) GFDs are an extension of relational CFDs to graphs, (b) while GFD$_\text{x}$s extend FDs from relations to graphs, carrying neither constant literals nor id literals.

symbols	notations
$G = (V, E, L, F_A)$	a graph
$Q[\bar{x}] = (V_Q, E_Q, L_Q)$	a graph pattern
$\iota \asymp \iota'$	a label ι matches ι'
$h(\bar{x})$	a match of a graph pattern in a graph
$h(\iota^u_{u'})$	the edge matched by the wildcard of the edge $(u, \_, u')$
$[x]_{\mathsf{Eq}}$, $[x.A]_{\mathsf{Eq}}$	the equivalence classes of a node x and an attribute $x.A$
G_{Eq}	the coercion of a consistent Eq on G
$\mathsf{Eq} \Rightarrow_{(\varphi,h)} \mathsf{Eq}'$	a chase step of G by Σ at Eq
$\mathsf{chase}(G, \Sigma)$	the result of any terminal chasing sequence of G by Σ
$(\mathsf{Eq}_k, G_{\mathsf{Eq}_k})$	valid chase result
$\perp$	the chase result is invalid
$G_\Sigma = (V_\Sigma, E_\Sigma, L_\Sigma, F^\Sigma_A)$	the canonical graph of a set Σ of GEDs
$G_Q = (V_Q, E_Q, L_Q, F_A)$	the canonical graph of graph pattern Q
Eq_X	the equivalence relation of X in G_Q
$\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$	the result of the chase of G_Q by Σ starting with Eq_X
$\mathcal{A}_{\mathsf{GED}}$	the axiom system $\mathcal{A}_{\mathsf{GED}}$ for GEDs in Table 3
$\Sigma_{\mathcal{A}} \vdash \varphi$	φ is provable from Σ using $\mathcal{A}$
$(y)^{(y,\iota,x)}_x$	the copy of y connected to node x, which corresponds to the edge (y, ι, x).

Table 2. Notations

Similarly, we study GEDs without constant literals, referred to as *variable* GEDs and denoted by $\mathsf{GED_x}$s. Obviously $\mathsf{GFD_x}$s are a proper sub-class of GEDs. For instance, ψ_1–ψ_3 given in Example 3.1 are $\mathsf{GED_x}$s, but they are not $\mathsf{GFD_x}$s.

(4) Forbidding GEDs. GEDs can express limited negation, in the form of $Q[\bar{x}](X \Rightarrow \text{false})$. Following [19], we refer to such GEDs as *forbidding constraints*.

(5) Relational dependencies. Following [30], one can show that FDs and CFDs can be expressed as GEDs if relation tuples are represented as nodes in a graph. In fact, equality-generating dependencies (EGDs) can be expressed as GFDs (GEDs) in the same setting. An EGD has the form $\forall \bar{z}(\phi(\bar{z}) \Rightarrow y_1 = y_2)$, where ϕ is a conjunction of relation atoms $R(w_1, \ldots, w_l)$ and equality atoms $w_i = w_j$, w_i and w_j are variables in $\bar{z}$, and so are y_1 and y_2 (cf. [1]). Each variable w corresponds to an attribute $R_w[A_w]$ in a relation atom of ϕ. The EGD can be expressed as a pair of GFDs:

(1) $\varphi_R = Q_E[\bar{x}](\emptyset \Rightarrow Y_R)$, where (a) Q_E is a pattern such that for each relation atom R in ϕ, there exists a node $x_R \in \bar{x}$ in Q_E that is labeled with R; and Q_E has no edges; and (b) Y_R consists of $x_R.A_R = x_R.A_R$ for each variable $x \in \bar{z}$, which indicates attribute $R[A_R]$; intuitively, φ_R ensures that the relations in ϕ have the attributes required; and

(2) $\varphi_E = Q_E[\bar{x}](X_E \Rightarrow Y_E)$, where (a) for each equality atom $w_i = w_j$ in ϕ, which corresponds to $R_i[A_{R_i}] = R_j[A_{R_j}]$ as remarked above, X_E includes a literal $x_{R_i}.A_{R_i} = x_{R_j}.A_{R_j}$; and (b) Y_E is $x_{R_{y1}}.A_{R_{y1}} = x_{R_{y2}}.A_{R_{y2}}$, which corresponds to $y_1 = y_2$. This enforces that ϕ entails $y_1 = y_2$.

The notations of the paper are listed in Table 2.

4 THE CHASE REVISITED FOR GEDS

We next revise the chase [51] for GEDs over graphs (Section 4.1), and show that chasing with GEDs has the Church-Rosser property (Section 4.2). As will be seen in later sections, the chase helps us characterize the static analyses of GEDs and develop a finite axiomatization for GEDs.

4.1 Chasing with GEDs

Consider a graph $G = (V, E, L, F_A)$ and a finite set Σ of GEDs. We next study the chase of G by Σ. Among other things, the chase can be used to (a) check the satisfiability and implication of GEDs (see Section 5), (b) optimize pattern queries, and (c) identify entities and catch errors in graphs.

Equivalence relations. We define the chase in terms of a sequence of equivalence relations Eq on nodes x and attributes $x.A$ in G. For each node x in V, its equivalence class, denoted by $[x]_{\mathsf{Eq}}$, is a set of nodes $y \in V$ that are identified as x. For each attribute $x.A$ of x, its equivalence class $[x.A]_{\mathsf{Eq}}$ is a set of attributes $y.B$ and constants c, if $x.A = y.B$ and $x.A = c$ are enforced by GEDs in Σ (see below), respectively. The relation is reflexive, symmetric and transitive, such that

(a) if node $y \in [x]_{\mathsf{Eq}}$, then $x \in [y]_{\mathsf{Eq}}$ and $[x]_{\mathsf{Eq}} = [y]_{\mathsf{Eq}}$; intuitively, we merge $[x]_{\mathsf{Eq}}$ and $[y]_{\mathsf{Eq}}$ into one; similarly, if attribute $y.B \in [x.A]_{\mathsf{Eq}}$, then $[y.B]_{\mathsf{Eq}} = [x.A]_{\mathsf{Eq}}$;

(b) if there exists attribute $y.B$ such that $y.B \in [x.A]_{\mathsf{Eq}}$ and $y.B \in [z.C]_{\mathsf{Eq}}$, then $[x.A]_{\mathsf{Eq}} = [z.C]_{\mathsf{Eq}}$; similarly for constant c if $c \in [x.A]_{\mathsf{Eq}}$ and $c \in [z.C]_{\mathsf{Eq}}$;

(c) if there exists node y such that $y \in [x]_{\mathsf{Eq}}$ and $y \in [z]_{\mathsf{Eq}}$, then $[x]_{\mathsf{Eq}} = [z]_{\mathsf{Eq}}$ by transitivity;

(d) if node $y \in [x]_{\mathsf{Eq}}$, then for each attribute $y.B$ of y, $[x.B]_{\mathsf{Eq}} = [y.B]_{\mathsf{Eq}}$; similarly for each attribute $x.A$ of x; intuitively, if x and y are the same node, then they have the same set of attributes and the same corresponding attribute values.

Consistency. Inconsistencies may be introduced by id literals and constant literals when enforcing GEDs. We say that an equivalence relation Eq is *inconsistent* in G if

(a) there exists node $y \in [x]_{\mathsf{Eq}}$ such that $L(x) \not\asymp L(y)$ and $L(y) \not\asymp L(x)$ (label conflict), or

(b) there is $y.B \in [x.A]_{\mathsf{Eq}}$ such that $x.A = c$ and $y.B = d$ for distinct c and d (attribute conflict).

Otherwise we say that Eq is *consistent*.

We use $\asymp$ to compare labels (recall $\asymp$ from Section 2). This is to cope with wildcards in a pattern Q when we chase Q as a graph (see Section 5 for such examples). In this case, we treat '_' in Q as a special label. Recall that $\asymp$ is asymmetric: $x \asymp y$ does not mean that $y \asymp x$.

Coercion. When Eq is consistent in graph G, we can enforce Eq on G by merging nodes and their corresponding attributes and edges, and by equalizing and extending attributes.

We define the *coercion of a consistent* Eq *on* G as graph $G_{\mathsf{Eq}} = (V', E', L', F'_A)$ as follows: for each node $x \in V$, (a) x_{Eq} is a node in V', denoting $[x]_{\mathsf{Eq}}$; (b) for each edge $(x, \iota, y) \in E$, $(x_{\mathsf{Eq}}, \iota, y_{\mathsf{Eq}})$ is in E'; (c) $L'(x_{\mathsf{Eq}})$ is '_' if all nodes in $[x]_{\mathsf{Eq}}$ are labeled '_'; otherwise $L'(x_{\mathsf{Eq}}) = L(z)$, where $z \in [x]_{\mathsf{Eq}}$ with $L(z) \neq$ '_'; and (d) $F'_A(x_{\mathsf{Eq}}) = \bigcup_{y \in [x]_{\mathsf{Eq}}} F_A(y)$, the union of the attributes of all the nodes in $[x]_{\mathsf{Eq}}$.

When Eq is consistent, G_{Eq} is well defined. In particular, when x and y are identified as the same node, $F'_A(x_{\mathsf{Eq}})$ merges the attribute sets of x and y; moreover, if A is an attribute of both x and y, then $x.A = y.A$, and hence $F'_A(x_{\mathsf{Eq}})$ is well defined. In addition, for all nodes $z_1, z_2 \in [x]_{\mathsf{Eq}}$, if $L(z_1) \neq$ '_' and $L(z_2) \neq$ '_', then $L(z_1) = L(z_2)$. In contrast, when Eq is inconsistent, G_{Eq} is undefined.

Chasing. We start with Eq_0, an initial equivalence relation consisting of (1) $[x]_{\mathsf{Eq}_0} = \{x\}$ for each node $x \in V$, and (2) $[x.A]_{\mathsf{Eq}_0} = \{x.A, c\}$ for each attribute $x.A = c$ in $F_A(x)$ (see Example 4.1 for an example). Each chase step i extends an equivalence relation Eq_{i-1} to get Eq_i by applying a GED.

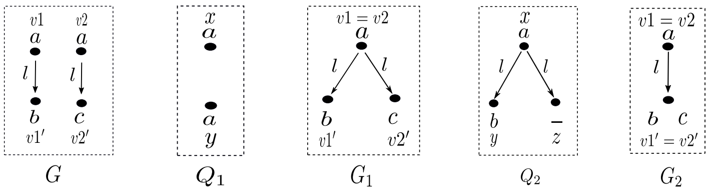

Fig. 2. Graphs and patterns used in chasing

More specifically, we define *a chase step of* G *by* Σ *at* Eq as

$$\mathsf{Eq} \Rightarrow_{(\varphi,h)} \mathsf{Eq}',$$

where $\varphi = Q[\bar{x}](X \Rightarrow Y)$ is a GED in Σ, and $h(\bar{x})$ is a match of pattern Q in the coercion G_{Eq} of Eq on graph G such that (a) $h(\bar{x}) \models X$, and (b) Eq$'$ is the equivalence relation of the extension of Eq by adding one literal $l \in Y$; more specifically, l and Eq$'$ satisfy one of the following conditions:

(1) if l is $x.A = c$ and $c \notin [h(x).A]_{\mathsf{Eq}}$, then Eq$'$ extends Eq by (a) including a new equivalence class $[h(x).A]_{\mathsf{Eq}'}$ if $h(x).A$ is not in Eq, and (b) adding c to $[h(x).A]_{\mathsf{Eq}'}$;

(2) if l is $x.A = y.B$ and $h(y).B \notin [h(x).A]_{\mathsf{Eq}}$, then Eq$'$ extends Eq by adding (a) $[h(x).A]_{\mathsf{Eq}'}$ if $h(x).A$ is not in Eq, and (b) $h(y).B$ to $[h(x).A]_{\mathsf{Eq}'}$; and

(3) if l is $x.\mathrm{id} = y.\mathrm{id}$ and $h(y) \notin [h(x)]_{\mathsf{Eq}}$, then Eq$'$ extends Eq by adding $h(y)$ to $[h(x)]_{\mathsf{Eq}'}$.

The step is *valid* if Eq$'$ is consistent in G_{Eq}.

Note that cases (1) and (2) above may expand the set of attributes of $h(x)$ when enforcing φ: attribute $h(x).A$ in Y is added if it is not already an attribute of $h(x)$, as required by $h(\bar{x}) \models Y$ (Section 3), since otherwise the chase will not lead to a graph satisfying φ (see Theorem 4.2 below).

A *chasing sequence* ρ of G by Σ is a sequence

$$(\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k),$$

where, for all $i \in [0, k-1]$, there exist a GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a match h of pattern Q in coercion graph G_{Eq_i} such that $\mathsf{Eq}_i \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{i+1}$ is a valid chase step.

The sequence is *terminal* if there exist no GED $\varphi \in \Sigma$, match h of pattern Q of φ in G_{Eq_k} and equivalence relation Eq_{k+1} such that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ is a valid chase step. More specifically, it terminates in one of the following two cases.

(a) No GEDs in Σ can be applied to expand ρ. If so, we say that ρ is *valid*, and refer to $(\mathsf{Eq}_k, G_{\mathsf{Eq}_k})$ as *its result*. It is easy to verify that in a valid ρ, for all $i \in [0, k]$, Eq_i is consistent in G_{Eq_i}.

(b) Either Eq_0 is inconsistent to start with (we will see such a case in Section 5.2), or there exist φ, h and Eq_{k+1} such that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ but Eq_{k+1} is inconsistent in G_{Eq_k}. If so, we say that the sequence ρ is *invalid*, and its result is $\perp$ (undefined).

In particular, a forbidding constraint $Q[\bar{x}](X \Rightarrow \mathsf{false})$ can be applied only when G is "inconsistent" or "dirty", and as a result, it makes the chasing sequence invalid.

Example 4.1. Consider graph G shown in Fig. 2, where v_1 and v_2 have attributes $v_1.A = v_2.A = 1$.

(1) Consider a set Σ_1 consisting of a single GED $\phi_1 = Q_1[x, y](x.A = y.A \Rightarrow x.\mathrm{id} = y.\mathrm{id})$ with Q_1 in Fig. 2. Then $\mathsf{Eq}_0 \Rightarrow_{(\phi_1,h_1)} \mathsf{Eq}_1$ is a chase step, where (a) Eq_0 consists of $[v]_{\mathsf{Eq}_0} = \{v\}$ for v ranging over v_1, v_2, v_1', v_2', and $[v_1.A]_{\mathsf{Eq}_0} = [v_2.A]_{\mathsf{Eq}_0} = \{v_1.A, v_2.A, 1\}$; (b) h_1: $x \mapsto v_1$ and $y \mapsto v_2$; and (c) Eq_1 extends Eq_0 by letting $[v_1]_{\mathsf{Eq}_1} = [v_2]_{\mathsf{Eq}_1} = \{v_1, v_2\}$. The coercion G_1 of Eq_1 on G is shown in Fig. 2, which merges v_1 and v_2. One can verify that $(\mathsf{Eq}_0, \mathsf{Eq}_1)$ is a terminal chasing sequence of G by Σ_1 since no more GEDs can be applied. Moreover, it is valid, yielding result (Eq_1, G_1).

(2) Consider $\Sigma_2 = \{\phi_1, \phi_2\}$, where $\phi_2 = Q_2[x, y, z](\emptyset \Rightarrow y.\mathrm{id} = z.\mathrm{id})$ (Q_2 in Fig. 2). Now $\mathsf{Eq}_1 \Rightarrow_{(\phi_2,h_2)}$

Eq_2 is a chase step, where h_2: $x \mapsto v_1$, $y \mapsto v_1'$, $z \mapsto v_2'$; and (b) Eq_2 extends Eq_1 by adding v_2' to $[v_1']_{\mathsf{Eq}_1}$. Then the sequence $(\mathsf{Eq}_0, \mathsf{Eq}_1)$ is still terminal, but it is invalid since there exists a chase step $\mathsf{Eq}_1 \Rightarrow_{(\phi_2, h_2)} \mathsf{Eq}_2$, where Eq_2 is inconsistent in G_1. As shown in Fig. 2, the coercion G_2 of Eq_2 on G is to merge v_1' and v_2' with distinct labels. The result of this sequence is $\bot$. □

As opposed to chase of relations or RDF with EGDs or FDs [2, 8, 19, 40], a chasing sequence with GEDs operates on a graph (pattern), and may be invalid due to label conflicts. Moreover, it supports "attribute generation" (cases (1) and (2) of chase steps above) to cope with schemaless graphs. In addition, the relational and RDF chasing rules do not deal with id literals. When $x.\mathsf{id} = y.\mathsf{id}$ is enforced, all their attributes and edges have to be merged.

4.2 The Church Rosser Property

The chase with relational FDs has the Church-Rosser property (cf. [1]). We show that chasing with GEDs retains the property. We consider finite sets Σ of GEDs, and use the following notions.

(a) Chasing with GEDs is *finite*, if for all sets Σ of GEDs and all graphs G, all chasing sequences of G by Σ are finite.

(b) Chasing with GEDs has the *Church-Rosser property*, if for all Σ and G, all terminal chasing sequences of G by Σ have the same result, regardless of in what order the GEDs are applied. That is, terminal sequences are either (a) all valid with the same $(\mathsf{Eq}, G_{\mathsf{Eq}})$, or (b) all invalid with $\bot$.

As opposed to chasing with relational FDs, chasing with GEDs may get into conflicts. Nonetheless, all terminal chasing sequences with GEDs still yield the same result.

Theorem 4.2. *Chasing with* GEDs *is finite and has the Church-Rosser property. Moreover, for any set* Σ *of* GEDs *and graph* G*, if there exists a valid terminal chasing sequences of* G *by* Σ*, then* $G_{\mathsf{Eq}} \models \Sigma$*, where* $(\mathsf{Eq}, G_{\mathsf{Eq}})$ *is the result of the terminal sequence.* □

By Theorem 4.2, we can define *the result of chasing* G *by* Σ as the result of any terminal chasing sequence of G by Σ, denoted by $\mathsf{chase}(G, \Sigma)$. We say that $\mathsf{chase}(G, \Sigma)$ is *consistent* if there exists such a valid terminal chasing sequence, with result $(\mathsf{Eq}, G_{\mathsf{Eq}})$. It is *inconsistent* otherwise, *i.e.,* when all terminal chasing sequences are invalid.

Proof: We show that for any set Σ of GEDs and any graph $G = (V, E, L, F_A)$, (1) all chasing sequences of G by Σ are finite; (2) all terminal sequences of G by Σ have the same result; and (3) if there exists a valid terminal chasing sequence of G by Σ with result $(\mathsf{Eq}, G_{\mathsf{Eq}})$, then $G_{\mathsf{Eq}} \models \Sigma$.

(1) Any chasing sequence ρ *of* G *by* Σ *is finite.* Consider a chasing sequence $\rho = (\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k)$ of G by Σ. Observe that for each $0 \leq i \leq k$, Eq_i consists of (a) at most $|V|$ equivalence classes $[x]_{\mathsf{Eq}_i}$ for all nodes in G, with total length $|V|$ when all such classes are put together; (b) at most $|V| \cdot |\Sigma| + |F_A|$ equivalence classes $[x.A]_{\mathsf{Eq}_i}$ for attributes, since GEDs in Σ check at most $|\Sigma|$ attributes for each node; and (c) at most $|\Sigma| + |F_A|$ constants, since all constants are from F_A and Σ. Hence the length $|\mathsf{Eq}_i|$ of Eq_i is at most $|G| \cdot |\Sigma| + 2 \cdot |G| + |\Sigma| \leq 4 \cdot |G| \cdot |\Sigma|$, because each attribute $x.A$ or constant c appears in at most one equivalence class. Here the size of G, denoted by $|G|$, is the sum of $|V|$, $|E|$, $|F_A|$, the sizes of the labels of all vertices and edges, and the sizes of the values of all attributes.

In the chasing step $\mathsf{Eq}_{i-1} \Rightarrow_{(\varphi, h)} \mathsf{Eq}_i$, Eq_i extends Eq_{i-1} by adding one literal l (see Section 4.1). Assume *w.l.o.g.* that l is $x.A = c$; the proofs for the cases of $x.A = y.B$ and $x.\mathsf{id} = y.\mathsf{id}$ are similar. Then $|\mathsf{Eq}_i| \leq |\mathsf{Eq}_{i-1}| + 2$. More specifically, (a) when neither c nor $x.A$ exists in Eq_{i-1}, $|\mathsf{Eq}_i| = |\mathsf{Eq}_{i-1}| + 2$; (b) when only one of c and $x.A$ exists in Eq_{i-1}, $|\mathsf{Eq}_i| = |\mathsf{Eq}_{i-1}| + 1$; (c) when both of c and $x.A$ are in Eq_{i-1}, $|\mathsf{Eq}_i| = |\mathsf{Eq}_{i-1}|$. That is, $|\mathsf{Eq}_i|$ increases the size $|\mathsf{Eq}_{i-1}|$ except in case (c). Observe that in case (c), two different equivalence classes can be merged only once. Moreover, there exist at most $4 \cdot |G| \cdot |\Sigma|$ equivalence classes to process. Given that $|\mathsf{Eq}_k| \leq 4 \cdot |G| \cdot |\Sigma|$, we have that the length of ρ is bounded by $8 \cdot |G| \cdot |\Sigma|$, *i.e.,* $k \leq 8 \cdot |G| \cdot |\Sigma|$. Hence ρ is finite.

(2) All terminal sequences of G by Σ have the same result. We show this by contradiction. Assume that there exist two terminal chasing sequences $\rho_1 = (\mathsf{Eq}_0, \mathsf{Eq}_1, \ldots, \mathsf{Eq}_k)$ and $\rho_2 = (\mathsf{Eq}_0, \mathsf{Eq}'_1, \ldots, \mathsf{Eq}'_l)$ such that they end up with different results. Assume *w.l.o.g.* that Eq_0 is consistent, since otherwise both ρ_1 and ρ_2 are invalid and end up with the same result. When Eq_0 is consistent, if Eq_k and Eq'_l are different, then we have that either $\mathsf{Eq}_k \setminus \mathsf{Eq}'_l \neq \emptyset$ or $\mathsf{Eq}'_l \setminus \mathsf{Eq}_k \neq \emptyset$. To simplify the discussion, we also use Eq to denote the set of equality literals $u = v$ for all $u \in [v]_{\mathsf{Eq}}$, *i.e.,* $x.A = c$, $x.A = y.B$ and $x.\mathrm{id} = y.\mathrm{id}$. We next show that if $\mathsf{Eq}_k \setminus \mathsf{Eq}'_l \neq \emptyset$ or $\mathsf{Eq}'_l \setminus \mathsf{Eq}_k \neq \emptyset$, then either one of ρ_1 and ρ_2 is not terminal, or both ρ_1 and ρ_2 are invalid (and hence ρ_1 and ρ_2 have the same result), a contradiction to the assumption above. To this end, we first show the following.

Lemma 4.3. *If ρ_1 is valid and $\mathsf{Eq}'_l \setminus \mathsf{Eq}_k \neq \emptyset$, then ρ_1 is not terminal.*

Proof: Assume that $S = \mathsf{Eq}'_l \setminus \mathsf{Eq}_k \neq \emptyset$. We show that there exist a GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, a match h of Q in G_{Eq_k}, and an equivalence relation Eq_{k+1} on G such that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ is a chase step of G by Σ. Hence the valid chasing sequence ρ_1 is not terminal by the definition of terminal chasing sequences (see Section 4.1).

We define φ, h, and Eq_{k+1} as follows. Suppose that Eq'_j $(1 \leq j \leq l)$ is the first equivalence relation in ρ_2 such that $S \cap \mathsf{Eq}'_j \neq \emptyset$, and $\mathsf{Eq}'_{j-1} \Rightarrow_{(\varphi_j,h_j)} \mathsf{Eq}'_j$ is the corresponding chase step in ρ_2, where $\varphi_j = Q_j[\bar{x}_j](X_j \Rightarrow Y_j)$ is a GED in Σ, h_j is a match of Q_j in $G_{\mathsf{Eq}'_{j-1}}$, and $h_j(\bar{x}_j) \models X_j$. Since both sequences start with Eq_0, we have that $j > 0$ and $\mathsf{Eq}'_{j-1} \subseteq \mathsf{Eq}_k$. We let $\varphi = \varphi_j$, and define h as follows: for each node x in Q_j, if $h_j(x) = u_{\mathsf{Eq}_{j-1}}$ (*i.e.,* $[u]_{\mathsf{Eq}_{j-1}}$, see Section 4.1 for the definition of the coercion of Eq on G), then $h(x) = u_{\mathsf{Eq}_k}$ (denoting $[u]_{\mathsf{Eq}_k}$); and for each edge (x_1, ι, y_1), because there exists an edge $(u_{\mathsf{Eq}_{j-1}}, h_j(\iota^{x_1}_{y_1}), v_{\mathsf{Eq}_{j-1}})$ in $G_{\mathsf{Eq}'_{j-1}}$ such that $h_j(x_1) = u_{\mathsf{Eq}_{j-1}}$, $h_j(y_1) = v_{\mathsf{Eq}_{j-1}}$ and $\iota \asymp h_j(\iota^{x_1}_{y_1})$, we define $h(\iota^{x_1}_{y_1}) = h_j(\iota^{x_1}_{y_1})$. We next show the following: (a) h is a match of Q_j in G_{Eq_k}, (b) $h(\bar{x}) \models X$ (*i.e.,* X_j), and (c) φ and h yield a chase step $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$. If these hold, then Eq_k can be further extended by another chase step. That is, ρ_1 is not terminal, a contradiction.

(a) By the definitions of coercion and h, it is easy to verify that h is a match of Q_j in G_{Eq_k}.

(b) We now show that $h(\bar{x}) \models X$ (*i.e.,* X_j). For each literal l in X, (i) if l is $x.A = c$ or $x.A = y.B$, we have that $h(\bar{x}) \models l$ since $F_A^{\mathsf{Eq}'_{j-1}}(x) \subseteq F_A^{\mathsf{Eq}_k}(x)$ and $h_j(\bar{x}_j) \models X_j$, where $F_A^{\mathsf{Eq}'_{j-1}}$ and $F_A^{\mathsf{Eq}_k}$ are the attribute functions for the coercions of Eq'_{j-1} and Eq_k on G, respectively. (ii) If l is $x.\mathrm{id} = y.\mathrm{id}$, since $h_j(\bar{x}_j) \models X_j$, we know that $h_j(y) \in [h_j(x)]_{\mathsf{Eq}_{j-1}}$. Since $\mathsf{Eq}'_{j-1} \subseteq \mathsf{Eq}_k$, $h_j(y) \in [h_j(x)]_{\mathsf{Eq}_k}$. By the definitions of G_{Eq_k} and h, we have that $h(x)$ and $h(y)$ are the same node in G_{Eq_k}, *i.e.,* $h(\bar{x}) \models l$. Putting these together, we can conclude that $h(\bar{x}) \models X$.

(c) Given these, we show that ρ can be extended by a chase step $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$, and thus is not terminal. As each chase step only adds one literal, assume *w.l.o.g.* that $v.A = c$ is the literal added by the chase step $\mathsf{Eq}'_{j-1} \Rightarrow_{(\varphi_j,h_j)} \mathsf{Eq}'_j$, *i.e.,* $\{v.A = c\} \in S \cap \mathsf{Eq}'_j$; the proofs for the cases of $v.A = v'.B$ and $v.\mathrm{id} = v'.\mathrm{id}$ are similar. Let Eq_{k+1} be the equivalence relation that extends Eq_k with $v.A = c$. From statements (a) and (b) above, we know that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ is a chase step. □

Using Lemma 4.3, we now continue with the proof of statement (2) of Theorem 4.2. It suffices to consider the following two cases: (a) both ρ_1 and ρ_2 are valid but have different results; and (b) one of them is valid and the other is not. We show that both cases lead to contradiction.

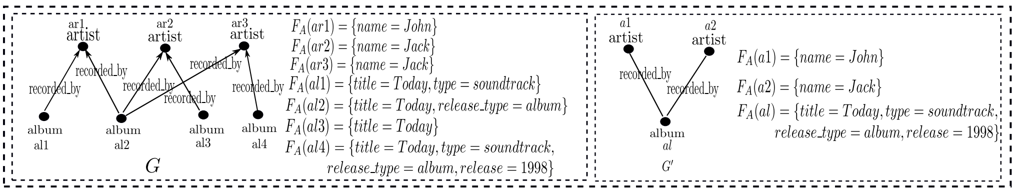
ar1
artist
ar2
artist
ar3
artist
recorded_by
recorded_by
recorded_by
recorded_by
recorded_by
recorded_by
album
al1
album
al2
album
al3
album
al4
$F_A(ar1) = \{name = John\}$
$F_A(ar2) = \{name = Jack\}$
$F_A(ar3) = \{name = Jack\}$
$F_A(al1) = \{title = Today, type = soundtrack\}$
$F_A(al2) = \{title = Today, release\_type = album\}$
$F_A(al3) = \{title = Today\}$
$F_A(al4) = \{title = Today, type = soundtrack, release\_type = album, release = 1998\}$
G
a1
artist
a2
artist
recorded_by
recorded_by
album
al
G'
$F_A(a1) = \{name = John\}$
$F_A(a2) = \{name = Jack\}$
$F_A(al) = \{title = Today, type = soundtrack, release\_type = album, release = 1998\}$

Fig. 3. Graph for repairing

In case (a), by Lemma 4.3, we must have that $\text{Eq}'_l \setminus \text{Eq}_k = \emptyset$ and $\text{Eq}_k \setminus \text{Eq}'_l = \emptyset$ since otherwise one of the two sequences is not terminal. However, from these it follows that $\text{Eq}_k = \text{Eq}'_l$, *i.e.,* ρ_1 and ρ_2 end up with the same result, a contradiction to the assumption.

In case (b), assume *w.l.o.g.* that ρ_1 is valid but ρ_2 is not. We show that ρ_1 must be invalid as well. Since ρ_2 is invalid, there exist a GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a match h of Q in $G_{\text{Eq}'_l}$ such that $\text{Eq}'_l \Rightarrow_{(\varphi,h)} \text{Eq}'_{l+1}$, $h(\bar{x}) \models X$ and Eq'_{l+1} is inconsistent in $G_{\text{Eq}'_l}$, where Eq'_{l+1} extends Eq'_l by adding a literal l from Y. By Lemma 4.3, $\text{Eq}'_l \subseteq \text{Eq}_k$. As in the proof of Lemma 4.3, one can verify that h is also a match of Q in G_{Eq_k} and $h(\bar{x}) \models X$. Thus $\text{Eq}_k \Rightarrow_{(\varphi,h)} \text{Eq}_{k+1}$ is a chase step, where Eq_{k+1} extends Eq_k by adding literal l. Since Eq'_{l+1} has conflicts and $\text{Eq}'_l \subseteq \text{Eq}_k$, we have that Eq_{k+1} also has conflicts, which contradicts the assumption that ρ_1 is valid.

(3) If there exists a valid terminal chasing sequence ρ of G by Σ, then $G_{\text{Eq}} \models \Sigma$, where $(\text{Eq}, G_{\text{Eq}})$ is the result of ρ. We prove this by contradiction. Suppose that $G_{\text{Eq}} \not\models \Sigma$. Then there exist a GED $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in Σ and a match h_1 of Q_1 in G_{Eq} such that $h_1(\bar{x}_1) \models X_1$, but $h_1(\bar{x}_1) \not\models Y_1$, *i.e.,* there exists a literal l in Y_1 such that $h_1(\bar{x}_1) \not\models l$. Then $\text{Eq} \Rightarrow_{(\varphi_1,h_1)} \text{Eq}'$ is a chase step, where Eq' is the equivalence relation of the extension of Eq by adding $h_1(l)$ (see Section 4.1), and $h_1(l)$ is the literal obtained by replacing each node x with $h_1(x)$ in l. This contradicts the assumption that ρ is a valid terminal chasing sequence. Therefore, $G_{\text{Eq}} \models \Sigma$.

This completes the proof of Theorem 4.2. □

Since the chase has the Church-Rosser property, we can use it in consistency checking, entity resolution, knowledge base expansion, graph repairing and fraud detection, among other things.

Example 4.4. Consider graph $G = (V, E, L, F_A)$ shown in Fig. 3. It consists of three artists ($ar1$, $ar2$, and $ar3$) and four albums ($al1$, $al2$, $al3$, and $al4$). Moreover, artists $ar2$ and $ar3$ have the same name, and all albums have the same title. We chase G with GEDs ψ_1–ψ_4 given in Example 3.1.

A chasing sequence is as follows: (1) merge $al1$, $al2$ and $al3$ using GKeys ψ_1; (2) add attribute $al3$.release = 1998 to album $al3$ by applying GFD ψ_4 to $al3$ and $al4$; (3) merge $al3$ and $al4$ with GKeys ψ_2; and (4) merge $ar2$ and $ar3$ using GKeys ψ_3. We finally obtain the coercion G' shown in Fig. 3. Note that before step (1), node $al3$ has neither attribute type nor attribute release_type, and we hence cannot apply GFD ψ_4 to $al3$ and $al4$. But after step (1), attributes type and release_type are added to $al3$, which are inherited from $al1$ and $al2$, respectively. Then we can apply ψ_4, and further apply GKeys ψ_3 to merge $ar2$ and $ar3$. One can verify that all chasing sequences result in G'. This process shows that to clean graph G, we need a combination of GFDs and GKeys. □

5 REASONING ABOUT GEDS

We next study three fundamental problems associated with GEDs and their sub-classes identified in Section 3. We characterize their static analyses and establish their complexity in various settings (Sections 5.1 and 5.2). We also investigate their validation problem (Section 5.3).

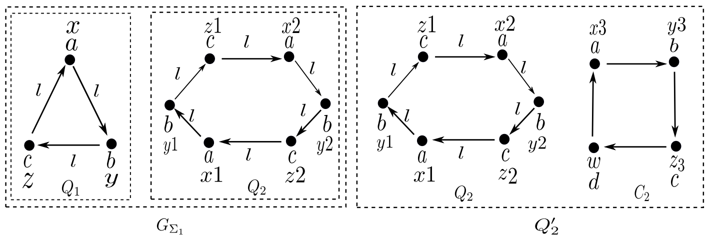

Fig. 4. The satisfiability of GEDs

5.1 The Satisfiability Problem

We study a strong notion of satisfiability. Consider a set Σ of GEDs. A *model* of Σ is a graph G such that (a) $G \models \Sigma$, and (b) for each $Q[\bar{x}](X \Rightarrow Y)$ in Σ, Q has a match in G.

Intuitively, if Σ has a model, then the GEDs in Σ are sensible and do not conflict with each other. Hence we can apply these GEDs without worrying about their conflicts.

The *satisfiability* problem for GEDs is stated as follows.

- Input: A finite set Σ of GEDs.
- Question: Does there exist a model of Σ?

We say that Σ is *satisfiable* if it has a model.

For relational FDs, the satisfiability problem is trivial: for any set Σ of FDs, there always exists a nonempty relation that satisfies Σ (cf. [24]). When it comes to GEDs defined on graphs, however, the satisfiability analysis becomes more intriguing.

Example 5.1. (1) Consider a set Σ_1 consisting of

$$\phi_1 = Q_1[x, y, z](x.A = x.B \Rightarrow y.\text{id} = z.\text{id}), \qquad \phi_2 = Q_2[x_1, y_1, z_1, x_2, y_2, z_2](\emptyset \Rightarrow x_1.A = x_1.B),$$

where patterns Q_1 and Q_2 are depicted in Fig. 4 (ignore G_{Σ_1} for the moment, which will be used later in Example 5.3). One can verify that each of ϕ_1 and ϕ_2 has a model when they are taken separately; however, Σ_1 does not have a model. To see this, consider a homomorphism f from Q_2 to Q_1, mapping x_1 and x_2 to x, y_1 and y_2 to y, and z_1 and z_2 to z. Hence for any match h of Q_1 in a graph G, the composition of h and f makes a match of Q_2 in G. When taken together, however, ϕ_1 and ϕ_2 require us to merge two nodes y and z with distinct labels (label conflict).

(2) GEDs may interact with each other even when their patterns are not homomorphic. To see this, consider Σ_2 consisting of ϕ_1 and $\phi_2' = Q_2'[\bar{x}](\emptyset \Rightarrow x_1.A = x_1.B)$, where pattern Q_2' extends Q_2 by adding a connected component C_2, as shown in Fig. 4. Obviously, Q_1 is not homomorphic to Q_2' and vice versa. However, Σ_2 is not satisfiable. To see this, suppose by contradiction that Σ_2 has a model G, in which Q_2' has a match $h_2(\bar{x})$. Then for any match h_1 of Q_1 in G, we can construct a match h_2' of Q_2' such that (a) over C_2, h_2' is the same as h_2, and (b) over Q_2, h_2' is the composition of h_1 and f given above. Then the same conflict emerges as in (1). Note that some nodes of Q_1 and Q_2' are mapped to the same vertices in a graph. Indeed, the satisfiability problem only requires that each pattern in Σ has a match in the graph, while these matches do not have to be disjoint. □

Characterization. We develop a sufficient and necessary condition to characterize the satisfiability of a set Σ of GEDs. Consider a set Σ of GEDs $\varphi_i = Q_i[\bar{x}_i](X_i \Rightarrow Y_i)$ for $i \in [1, n]$, where $Q_i = (V_i, E_i, L_i)$. Assume *w.l.o.g.* that V_i and V_j are disjoint if $i \neq j$, after naming the nodes in Q_i.

The *canonical graph* G_Σ of Σ is defined to be a graph $(V_\Sigma, E_\Sigma, L_\Sigma, F_A^\Sigma)$, where V_Σ is the union of V_i's, and similarly for E_Σ and L_Σ; but F_A^Σ is empty. Intuitively, G_Σ is the union of all graph patterns in Σ, in which patterns from different GEDs are disjoint. To simplify the presentation, we use G_{Q_i} to denote the subgraph of G_Σ corresponding to the pattern Q_i of GED $\varphi_i = Q_i[\bar{x}_i](X_i \Rightarrow Y_i)$ in Σ.

We chase the pattern graph G_Σ with Σ, and characterize the satisfiability of Σ based on the chase.

Theorem 5.2. *A set Σ of* GEDs *is satisfiable if and only if* chase(G_Σ, Σ) *is consistent.* □

Example 5.3. Recall the set Σ_1 of GEDs from Example 5.1. Its canonical graph is the union G_{Σ_1} of Q_1 and Q_2 shown in Fig. 4. One can verify that chase(G_{Σ_1}, Σ_1) is inconsistent, *i.e.,* there exists a terminal chasing sequence of G_{Σ_1} by Σ_1 with result $\perp$. This confirms the observation of Example 5.1 that Σ_1 is not satisfiable. Similarly one can see that chase(G_{Σ_2}, Σ_2) is also inconsistent, for Σ_2. □

Proof: We show that a set Σ of GEDs is satisfiable if and only if chase(G_Σ, Σ) is consistent.

($\Leftarrow$) Assume first that chase(G_Σ, Σ) is consistent. Let $(\mathsf{Eq}, G_{\mathsf{Eq}})$ be the result of chasing G_Σ by Σ. We construct a model G of Σ from G_{Eq}, by assigning attribute values. Suppose that $G_{\mathsf{Eq}} = (V, E, L, F_A^{\mathsf{Eq}})$. We define $G = (V, E, L, F_A)$, with the same sets V and E of nodes and edges and the same labeling L, except that each wildcard '_' in G_{Eq} is replaced by a new label $\#A$, while F_A is populated as follows.

(1) For each $x.A = c$ in F_A^{Eq}, add $x.A = c$ to F_A.
(2) For each $x.A = y.B$ in F_A^{Eq}, if there exists a constant c such that $c \in [x.A]_{\mathsf{Eq}}$ or $c \in [y.B]_{\mathsf{Eq}}$, then add $x.A = c$ and $y.B = c$ to F_A; otherwise add $x.A = d$ and $y.B = d$ to F_A and Eq for some fresh distinct constant d to instantiate the attributes in $[x.A]_{\mathsf{Eq}}$ and $[y.B]_{\mathsf{Eq}}$ when these equivalence classes contain no constants, *i.e.,* when their attributes are not yet instantiated.

Since chase(G_Σ, Σ) is consistent, G is well defined. Moreover, since G_{Eq} is the coercion of Eq on G_Σ, by the definition of G_Σ, for any GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G_{Eq}: $x \mapsto x_{\mathsf{Eq}}$ (*i.e.,* $[x]_{\mathsf{Eq}}$; see Section 4 for the definition of coercion); this can be verified just like in Lemma 4.3. By the definition of G, G and G_{Eq} are isomorphic, except that wildcard '_' is replaced by $\#A$. Hence h_φ is also a match of Q in G. Thus each pattern Q in Σ has a match in G. Note that only wildcard '_' in pattern Q can match '_' in graph G_{Eq} (and $\#A$ in G).

It remains to show that $G \models \Sigma$. Suppose by contradiction that there is a GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ such that $G \not\models \varphi$. That is, there exists a match h of Q in G such that $h(\bar{x}) \models X$ but $h(\bar{x}) \not\models Y$. Since G and G_{Eq} are isomorphic (except that wildcard '_' is replaced by $\#A$), one can verify that h is a match of Q in G_{Eq} such that $h(\bar{x}) \models X$ but $h(\bar{x}) \not\models Y$, *i.e.,* $G_{\mathsf{Eq}} \not\models \varphi$. This contradicts Theorem 4.2, which assures that $G_{\mathsf{Eq}} \models \Sigma$ since chase(G_Σ, Σ) is consistent. Therefore, Σ is satisfiable.

($\Rightarrow$) Now assume that Σ is satisfiable. Consider a terminal chasing sequence $\rho = (\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k)$ of G_Σ by Σ. We show that ρ is valid, and hence that chase(G_Σ, Σ) is consistent by Theorem 4.2. We first identify a property of ρ, and then use the property to show that chase(G_Σ, Σ) is consistent.

(1) We first identify a property of ρ. Since Σ is satisfiable, there exists a model $G = (V, E, L, F_A)$ of Σ such that for any GED $\varphi' = Q'[\bar{x}'](X' \Rightarrow Y')$ in Σ, there exists a match $h_{\varphi'}$ of Q' in G. By combining these matches, we obtain a mapping h from G_Σ to G. That is, for each node $x \in V_\Sigma$, if x is in pattern Q', then $h(x) = h_{\varphi'}(x)$. It is a homomorphism from G_Σ to G by the definition of G_Σ, in which different patterns of Σ are disjoint. We show the following property.

For all $i \in [0, k]$, $h(V_\Sigma) \models \mathsf{Eq}_i$, *i.e.,* $h(V_\Sigma) \models l$ for all literals l in Eq_i by taking each $v \in [u]_{\mathsf{Eq}_i}$ as a literal $u = v$, denoting $x.A = c$, $x.A = y.B$ or $x.\mathrm{id} = y.\mathrm{id}$.

To simplify the presentation, we use $h(V_\Sigma)$ to denote $h(\bar{z})$, where $\bar{z}$ consists of all variables in V_Σ, and $h(V_\Sigma) \models l$ is defined along the same lines as $h(\bar{x}) \models l$ given in Section 3. We also use $h(l)$ to denote the literal obtained by replacing each variable x with $h(x)$ in l.

We prove this property by induction on i as follows. Obviously, this holds in the base case, *i.e.,* $h(V_\Sigma) \models \mathsf{Eq}_0$, where Eq_0 includes $[x]_{\mathsf{Eq}_0} = \{x\}$ for any node x in G_Σ, and F^Σ_A is empty in G_Σ.

Assume that for any j $(0 \leq j \leq i-1)$, $h(V_\Sigma) \models \mathsf{Eq}_j$. We show that $h(V_\Sigma) \models \mathsf{Eq}_i$. Since ρ is a chasing sequence, there exist a GED $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in Σ and a match h_1 of Q_1 in $G_{\mathsf{Eq}_{i-1}}$ such that $\mathsf{Eq}_{i-1} \Rightarrow_{(\varphi_1,h_1)} \mathsf{Eq}_i$, where Eq_i extends Eq_{i-1} by adding a literal $l \in Y_1$. That is, $h_1(\bar{x}_1) \models X_1$ and $h_1(l) \notin \mathsf{Eq}_{i-1}$. To show that $h(V_\Sigma) \models \mathsf{Eq}_i$, it suffices to show that $h \circ h_1$ is a match of Q_1 in G and $h \circ h_1(\bar{x}_1) \models X_1$. Here $h \circ h_1$ is a mapping from Q_1 to G defined as follows: for each node u in Q_1, $h \circ h_1(u) = h(v)$, where $h_1(u) = v_{\mathsf{Eq}_{i-1}}$. Since $h(V_\Sigma) \models \mathsf{Eq}_{i-1}$, $h \circ h_1$ is well defined. If $h \circ h_1(\bar{x}_1) \models X_1$, then $h \circ h_1(x) \models Y_1$ since $G \models \varphi$. From this it follows that $h(V_\Sigma) \models h_1(l)$, and thus $h(V_\Sigma) \models \mathsf{Eq}_i$.

By the definitions of coercion and $h \circ h_1$, one can easily verify that $h \circ h_1$ is a match of Q_1 in G.

We next show that $h \circ h_1(\bar{x}_1) \models X_1$, *i.e.,* for all literals $l \in X_1$, $h \circ h_1(\bar{x}_1) \models l$. We verify this based on different cases of l as follows. (a) When l is $x.A = c$. Let $v = h_1(x)$. Then by $h_1(\bar{x}_1) \models X_1$, we have that $c \in [v.A]_{\mathsf{Eq}_{i-1}}$. By $h(V_\Sigma) \models \mathsf{Eq}_{i-1}$, node $v' = h(v) = h \circ h_1(x)$ has attribute A, and $v'.A = c$ in G. That is, $h \circ h_1(\bar{x}_1) \models l$. (b) When l is $x.A = y.B$. Let $v = h_1(x)$ and $v' = h_1(y)$. From $h_1(\bar{x}_1) \models X_1$ it follows that $v.A \in [v'.B]_{\mathsf{Eq}_{i-1}}$. Since $h(V_\Sigma) \models \mathsf{Eq}_{i-1}$, we have the following: $h(v) = h \circ h_1(x)$ and $h(v') = h \circ h_1(y)$ have attributes A and B, respectively, and moreover, $(h \circ h_1(x)).A = (h \circ h_1(y)).B$ in G. Hence, $h \circ h_1(\bar{x}_1) \models l$. (c) When l is $x.\mathrm{id} = y.\mathrm{id}$. Let $v = h_1(x)$ and $v' = h_1(y)$. Since $h_1(\bar{x}_1) \models X_1$, $v' \in [v]_{\mathsf{Eq}_{i-1}}$. From $h(V_\Sigma) \models \mathsf{Eq}_{i-1}$ it follows that $h(v)$ and $h(v')$ are the same node in G. That is, $h \circ h_1(x)$ and $h \circ h_1(y)$ are the same node in G. Hence $h \circ h_1(\bar{x}_1) \models l$.

Putting these together, we can conclude that $h \circ h_1(\bar{x}_1) \models X_1$. Thus $h(V_\Sigma) \models \mathsf{Eq}_i$ for all $i \in [0,k]$.

(2) Based on the mapping h, we next show that $\mathrm{chase}(G_\Sigma, \Sigma)$ is consistent. By the definition of G_Σ, it is easy to verify that Eq_0 is consistent. Thus it suffices to show the following: (†) if there exist $\varphi = Q_2[\bar{x}_2](X_2 \Rightarrow Y_2)$ in Σ and a match h_2 of Q_2 in G_{Eq_k} such that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h_2)} \mathsf{Eq}_{k+1}$ and Eq_{k+1} is inconsistent in G_{Eq_k}, then $G \not\models \varphi$, where G is the model of Σ mentioned in (1) above. For if statement (†) above holds, then it contradicts the assumption that G is a model of Σ.

Along the same lines as the proof for (1) above, one can verify that $h \circ h_2$ is a match of Q_2 in G. Then it suffices to show that $h \circ h_2(\bar{x}_2) \models X_2$, but $h \circ h_2(\bar{x}_2) \not\models Y_2$. For if these hold, then $G \not\models \varphi$. Since $h(V_\Sigma) \models \mathsf{Eq}_k$, an argument similar to the one for (1) can verify that $h \circ h_2(\bar{x}_2) \models X_2$. To show that $h \circ h_2(\bar{x}_2) \not\models Y_2$, note that Eq_{k+1} is inconsistent in G_{Eq_k}, and Eq_{k+1} extends Eq_k by adding a literal $l \in Y_2$. Assume *w.l.o.g.* that l is $x_1.\mathrm{id} = y_1.\mathrm{id}$, where $x_2 = h_2(x_1)$, $y_2 = h_2(y_1)$, $y_2 \in [x_2]_{\mathsf{Eq}_{k+1}}$ but $L(h(x_2)) \not\asymp L(h(y_2))$ and $L(h(y_2)) \not\asymp L(h(x_2))$; the proofs for the other cases of conflicts are similar. Then, since G is well defined, $h(x_2)$ and $h(y_2)$ cannot be the same node in G. We can conclude that $h \circ h_2(\bar{x}_2) \not\models (x_1.\mathrm{id} = y_1.\mathrm{id})$ because $h(x_2)$ and $h(y_2)$ are distinct nodes. Hence $h \circ h_2(\bar{x}_2) \not\models Y_2$. □

Complexity. Capitalizing on Theorem 5.2, we next establish the complexity of the satisfiability problem for GEDs and its sub-classes. We refer to GED $Q[\bar{x}](X \Rightarrow Y)$ as a *tree-pattern* GED if Q is a tree. We say that Σ is a set of GEDs *with tree patterns* if each GED in Σ is a tree-pattern GED.

Theorem 5.4. *The satisfiability problem is*

- coNP-*complete for* GEDs, GFDs, GKeys *and* $\mathrm{GED_x}$s*; and*
- *it is in* $O(1)$ *time for* $\mathrm{GFD_x}$s.

The problem remains coNP*-hard for sets of* GEDs, GFDs, GKeys *and* $\mathrm{GED_x}$s *with tree patterns.* □

Theorem 5.4 tells us the following. (1) The intractability of the satisfiability analysis is rather robust: it arises either from constant literals in GFDs, or from id literals in GKeys and $\mathrm{GED_x}$s. As will be seen in the proof, the problem is coNP-hard even when Σ consists of a fixed number of

GEDs, and when all graph patterns that appear in Σ are trees. (2) In the absence of constant and id literals, the problem is trivial: any set of GFD_xs can find a model.

For relational EGDs, a consistency problem was shown coNP-complete [35]; that problem is to decide, given a database D and a set Σ of EGDs, whether the database D can be extended to a new database D_1 such that D_1 satisfies Σ. It is quite different from the satisfiability problem studied in this paper. The satisfiability problem for relational CFDs is NP-complete [25]. A close examination reveals that it is intractable only under a database schema that requires attributes to have a finite domain, *e.g.,* Boolean [25]. It is in PTIME in the absence of finite-domain attributes. As remarked in Section 3, while GEDs can express CFDs when relations are represented as graphs, GEDs cannot enforce an attribute to have a finite domain. The satisfiability problem for GEDs is intractable in the absence of finite-domain attributes. Hence its intractability is not inherited from CFDs, as indicated by the difference between coNP-complete and NP-complete (unless P = NP).

Proof: We first show that any set Σ of GFD_xs is satisfiable. We then show that the satisfiability problem is coNP-complete for GEDs, GFDs, GKeys and GED_xs with tree patterns.

(1) GFD_xs. It suffices to show that chase(G_Σ, Σ) is consistent, by Theorem 5.2. Observe that GFD_xs only deduce literals of the form $x.A = y.B$, with neither constants nor id literals. Hence any equivalence relation Eq generated in a chasing sequence is consistent (see Section 4 for consistency). In other words, the chase leads to no conflicts. Thus chase(G_Σ, Σ) is consistent for Σ.

(2) GEDs, GFDs, GKeys **and** GED_xs. We show that the satisfiability problem is in coNP for GEDs, and is coNP-hard for GFDs and for GKeys without constant literals, all with tree patterns. These suffice since GKeys without constant literals are a special case of GED_xs and GEDs.

Upper bound. We give an NP algorithm to check, given a set Σ of GEDs, whether Σ is not satisfiable:

(1) construct G_Σ;
(2) guess a chasing sequence $\text{Eq}_0 \Rightarrow_{(\varphi_1,h_1)} \text{Eq}_1 \Rightarrow_{(\varphi_2,h_2)} \ldots \Rightarrow_{(\varphi_k,h_k)} \text{Eq}_k$ of G_Σ by Σ such that its number of steps $k \leq 8 \cdot |\Sigma|^2$;
(3) for each i ($0 \leq i \leq k-1$), check whether $\text{Eq}_i \Rightarrow_{(\varphi_{i+1},h_{i+1})} \text{Eq}_{i+1}$ is a chase step; if not, reject the guess; continue otherwise;
(4) for each i ($0 \leq i \leq k-1$), check whether $\text{Eq}_i \Rightarrow_{(\varphi_{i+1},h_{i+1})} \text{Eq}_{i+1}$ is invalid; if any of them is invalid, return true.

We guess $h_1, \ldots, h_k$ without constructing $G_{\text{Eq}_0}, \ldots, G_{\text{Eq}_{k-1}}$, respectively. We first guess mappings $h'_1, \ldots, h'_k$ in G_Σ, and then define $h_i(x) = h'_i(x)_{\text{Eq}_{i-1}}$ (*i.e.,* $[h'_i(x)]_{\text{Eq}_{i-1}}$; see Section 4 for $G_{\text{Eq}_{i-1}}$) for $i \in [1, k]$. That is, x is mapped to the equivalence class of $h'_i(x)$ in Eq_{i-1}. Then h_i is a mapping from the pattern of φ_i to $G_{\text{Eq}_{i-1}}$. Indeed, by the definition of $G_{\text{Eq}_{i-1}}$, nodes in $G_{\text{Eq}_{i-1}}$ are defined as $x_{\text{Eq}_{i-1}}$ for each $x \in V_\Sigma$ (*i.e.,* $[x]_{\text{Eq}_{i-1}}$), where V_Σ is the set of nodes in G_Σ. Here Eq_0 consists of $[x]_{\text{Eq}_0} = \{x\}$ for each $x \in V_\Sigma$. The correctness of the algorithm follows from Theorems 4.2 and 5.2. For its complexity, step (1) is in PTIME by the definition of G_Σ. By Corollary 5.5 below, steps (3) and (4) are also in PTIME. Thus the algorithm is in NP, and the satisfiability problem is in coNP for GEDs.

Corollary 5.5. *(1) Given a graph G and an equivalence relation* Eq *in a chasing sequence of G by a set Σ of* GEDs, *it is in* PTIME *to decide whether* Eq *is consistent in G. (2) For a* GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$, *a graph G, a match h of Q in G, and two equivalence relations* Eq *and* Eq$'$ *in a chasing sequence of G by Σ, it is in* PTIME *to determine whether* $\text{Eq} \Rightarrow_{(\varphi,h)} \text{Eq}'$ *is a valid chase step.* □

Proof: (1) From the proof of Theorem 4.2 we can see that $|\text{Eq}| \leq 4 \cdot |G| \cdot |\Sigma|$. Hence the three conditions for Eq to be consistent can be checked in PTIME (see Section 4 for consistency).

(2) To check whether $\text{Eq} \Rightarrow_{(\varphi,h)} \text{Eq}'$ is valid, it suffices to do the following:

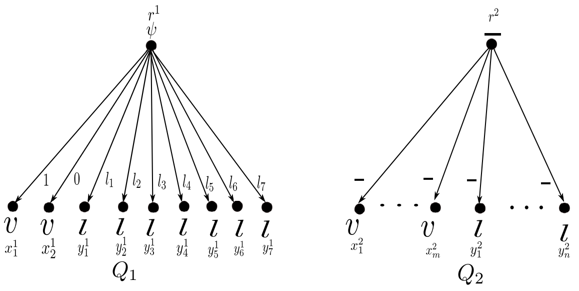

Fig. 5. The graph patterns of φ_1 in the proof of Lemma 5.6

(a) construct G_{Eq};
(b) check whether h is a match of Q in G_{Eq}, where Q is the pattern of φ;
(c) check whether $h(\bar{x}) \models X$;
(d) check whether there is a literal l in Y such that $h(l)$ is not in Eq, and Eq$'$ is Eq $\cup \{h(l)\}$;
(e) check whether Eq$'$ is consistent in G_{Eq}.

Since $|\mathsf{Eq}| \leq 4 \cdot |G| \cdot |\Sigma|$, all these steps can be done in PTIME. Thus it is in PTIME to check whether a chase step Eq $\Rightarrow_{(\varphi,h)}$ Eq$'$ is valid. □

Lower bound. We first settle the lower bound for the satisfiability analysis of GFDs of a restricted form. The proof is quite different from the one given in [30], since we adopt graph homomorphism for the semantics of GFDs in this work, and consider tree-pattern GFDs.

Lemma 5.6. *The satisfiability problem is* coNP-*hard for* GFDs *with tree patterns.* □

Proof: We show the satisfiability problem is coNP-hard by reduction from the complement of the 3SAT problem, which is known to be NP-complete (cf. [32]). The 3SAT problem is to decide, given a 3SAT formula $\psi = C_1 \wedge \ldots \wedge C_n$ with variables in $\{x_1, \ldots, x_m\}$, whether there exists a truth assignment of $\{x_1, \ldots, x_m\}$ such that ψ is true. Here C_i is in the form of $l_1 \vee l_2 \vee l_3$, where l_1, l_2 and l_3 are variables or their negations from $\{x_1, \ldots, x_m\}$.

Given a 3SAT formula ψ, we construct a set Σ of GFDs such that all patterns in Σ are trees. We show that Σ is satisfiable if and only if ψ is not satisfiable. The set Σ consists of two GFDs φ_1 and φ_2. The construction ensures that if Σ is not satisfiable, then we can apply φ_2 to the pattern of φ_1, and further deduce a satisfying truth assignment of ψ, *i.e.,* ψ is satisfiable; conversely, if ψ is satisfiable, then there exists a satisfying truth assignment of ψ, and we can apply φ_2 to the pattern of φ_1 and deduce inconsistency, *i.e.,* Σ is not satisfiable. More specifically, Σ consists of the following GFDs.

(1) The first GFD φ_1 is used to encode Boolean domains of all variables and all satisfying assignments for formulas $l_1 \vee l_2 \vee l_3$. Specifically, $\varphi_1 = Q_1[\bar{x}_1](\emptyset \Rightarrow r^1.A = 1 \wedge Y_1 \wedge Y_2 \wedge Y_1^1 \wedge \ldots \wedge Y_1^7)$, where

(a) as shown in Fig. 5, $Q_1 = (V_1, E_1, L_1)$, and $V_1 = \{r^1, x_1^1, x_2^1, y_1^1, y_2^1, \ldots, y_7^1\}$; it consists of the root r, two nodes x_1^1 and x_2^1 to encode the Boolean domain, and seven nodes for the satisfying truth assignments of clauses C_i; $E_1 = \{(r^1, 1, x_1^1), (r^1, 0, x_2^1), (r^1, l_1, y_1^1), \ldots, (r^1, l_7, y_7^1)\}$; as shown in Fig. 5, these form a tree with r as the root, and all edges have distinct labels; and moreover, $L_1(r^1) = \psi$, $L_1(x_1^1) = L_1(x_2^1) = v$, $L_1(y_1^1) = L_1(y_2^1) = \ldots = L_1(y_7^1) = l$; and

(b) the literals in φ_1 are partitioned into three parts: (i) the first part has only one literal ($r^1.A = 1$); it sets the A attribute of the root to be 1; (ii) the second part consists of four literals to enforce the Boolean domain of variables; that is, $Y_1 = ((x_1^1.A = 1) \wedge (x_1^1.B = 0))$ and $Y_2 = ((x_2^1.A = 0) \wedge (x_2^1.B = 1))$; we use $x_1^1.B$ or $x_2^1.B$ to represent the negation of the variable; (iii) the last part consists of 7 groups of literals, each of which corresponds to one satisfying assignment of clauses C_i ($i \in [1, n]$); that is, $Y_1^1 = ((y_1^1.A = 1) \wedge (y_1^1.B = 1) \wedge (y_1^1.C = 1))$, ...,

$Y_1^7 = ((y_7^1.A = 0) \wedge (y_7^1.B = 0) \wedge (y_7^1.C = 1))$; intuitively, for a conjunct $C_i = l_1 \vee l_2 \vee l_3$, we use the attributes A, B and C to represent the values of l_1, l_2 and l_3, respectively;

(2) The second GFD $\varphi_2 = Q_2[\bar{x}_2]((Y_1 \wedge \ldots \wedge Y_n) \Rightarrow (r^2.A = 2))$ encodes ψ, where

(*a*) $Q_2 = (V_2, E_2, L_2)$ is shown in Fig. 5; here $V_2 = \{r^2, x_1^2, \ldots, x_m^2, y_1^2, y_2^2, \ldots, y_n^2\}$, representing the root, m variables, and $C_1, \ldots, C_n$, respectively; $E_2 = \{(r^2, `\_', x_1^2), \ldots, (r^2, `\_', x_m^2), (r^2, `\_', y_1^2), \ldots, (r^2, `\_', y_n^2)\}$; as shown in Fig. 5, these make a tree with r^2 as the root; and $L_2(r^2) = `\_'$, $L_2(x_j^2) = v$ for $j \in [1, m]$, and $L_2(y_i^2) = l$ for $i \in [1, n]$; and

(b) the literals in $Y_1, \ldots, Y_n$ are used to check whether a given truth assignment μ_X makes ψ true. Note that when φ_2 is applied, there exists a match of nodes $x_1^2, \ldots, x_m^2$, which encodes a truth assignment μ_X of $\{x_1, \ldots, x_m\}$. Each set Y_i ($i \in [1, n]$) of literals encodes the bindings of variables in one conjunct C_i in ψ. For example, let $C_i = x_1 \vee \neg x_2 \vee x_3$. Then the literals for the node y_i^2 are $Y_i = (y_i^2.A = x_1.A) \wedge (y_i^2.B = x_2.B) \wedge (y_i^2.C = x_3.A)$.

Intuitively, to check whether μ_X makes C_i true, we only need to check whether y_i^2 "matches" one of $y_1^1, \ldots, y_7^1$ in Q_1, which encode all possible satisfying truth assignments of a clause. Consider the node y_i^2 above as an example. When $x_1 = 1$, $x_2 = 1$ and $x_3 = 1$ in μ_X, since μ_X satisfies C_i, we have that $y_i^2.A = 1$, $y_i^2.B = 0$, and $y_i^2.C = 1$, *i.e.*, y_i^2 matches y_5^1. Then when all of $y_1^2, \ldots, y_n^2$ match one of $y_1^1, \ldots, y_7^1$, we can conclude that μ_X makes ψ true.

The canonical graph G_Σ of Σ consists of two disjoint components: G_{Q_1} and G_{Q_2} corresponding to Q_1 and Q_2, respectively. Observe the following. (1) Since the labels of the edges in G_{Q_1} are in $\{1, 0, l_1, \ldots, l_7\}$, and the edges in G_{Q_2} are labeled wildcard '_', Q_1 finds no match in G_{Q_2}. (2) Since φ_1 and φ_2 do not carry id literals, no chasing sequence of G_Σ by Σ can identify two distinct nodes in G_Σ. Moreover, (3) since G_{Q_1} and G_{Q_2} are disjoint while each of G_{Q_1} and G_{Q_2} is connected, a chase step is taken only when there exists either a match of Q_1 in G_{Q_1}, or a match of Q_2 in G_{Q_1} or G_{Q_2}.

One can verify that Σ is not satisfiable if and only if ψ is satisfiable (details in the appendix). □

We next show that the satisfiability problem for a special case of GKeys is also intractable.

LEMMA 5.7. *The satisfiability problem is* coNP-*hard for* GKeys *that are defined with tree patterns and do not contain constant literals.* □

Proof: This can be verified by reduction from the complement of 3-colorability problem, which is known to be NP-complete [32]. The 3-colorability problem is to decide, given an undirected graph G, whether there exists a 3-coloring ν of G such that for each edge (u, v) in G, $\nu(u) \neq \nu(v)$. The problem is NP-complete even when G is a connected graph [33].

Consider a given connected undirected graph $G = (V, E)$, where $|V| = m$. We construct a set Σ of GKeys such that the pattern of each GKey ψ in Σ is a tree, and that ψ does not contain constant literals. We show that Σ is not satisfiable if and only if G has a proper 3-coloring.

The reduction is a little involved. We use three groups of GKeys to represent the graph patterns depicted in Fig. 6, which are to represent a proper 3-coloring and the structure of G, respectively. The construction of Q_1^1 (resp. Q_2^1) is straightforward: nodes labeled r, g, and b form a clique, and all these nodes have one edge leading to a node labeled 0 (resp. 1). The pattern Q_3^1 is constructed as follows: Q_3^1 consists of all the nodes in G and an additional node x_0, each undirected edge in G is represented by two directed edges, and all the nodes in G have one edge leading to x_0. In these patterns, all edges are labeled 1, except the ones leading to x_0^1, x_0^2 and x_0, which are labeled 0. Intuitively, when there exists a 3-coloring of G, we can use the GKeys corresponding to Q_3^1 to identify the two nodes labeled 0 and 1 in Q_1^1 and Q_2^1, respectively, and deduce inconsistency.

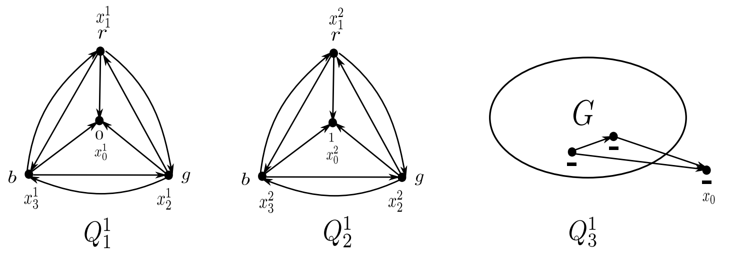

Fig. 6. Graph patterns encoded by the set Σ of GKeys in the proof of Lemma 5.7

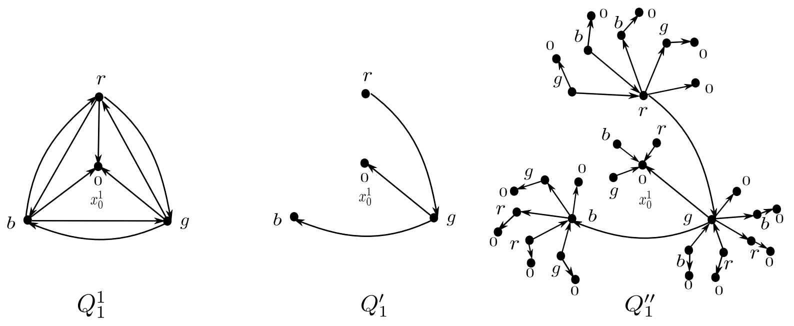

Fig. 7. The graph patterns encoded by φ_1 in the proof of Lemma 5.7

Since we can only use GKeys with tree patterns, the main difficulty of the construction is how to represent these patterns by trees. We solve this problem by using the spanning trees of the graphs, and by using other groups of GKeys or sets of literals to recover the patterns.

Below we define GKeys in Σ group by group, consisting of 15 GKeys in total.

(1) We start with a group of 7 GKeys to encode pattern Q_1^1 depicted in Fig. 6, which is in turn used to encode a proper 3-coloring. More specifically, we use the tree pattern Q_1 of one GKey to encode the main structure of Q_1^1, and another 6 GKeys to "recover" the cyclic pattern Q_1^1.

Pattern Q_1 is constructed in two steps. We first compute a spanning tree of Q_1^1 regardless of the directions of the edges (see Q_1' in Fig. 7 for an example). We then recover all the edges of Q_1^1 (*e.g.,* Q_1'' in Fig. 7). To cope with tree patterns, some edges are encoded with multiple edges in Q_1'', *e.g.,* seven edges in Q_1'' for the edge in Q_1^1 from the node labeled g to the node labeled 0.

More specifically, the first GKey φ_1 is defined as $Q_1[\bar{x}, \bar{y}](\emptyset \Rightarrow x_0^1.\text{id} = (x_0^1)'.\text{id})$, where Q_1 is composed of patterns $Q_1''[x_0^1, \bar{x}_1, \bar{x}_1']$ and $Q_2''[(x_0^1)', \bar{x}_2, \bar{x}_2']$, and $Q_2''[(x_0^1)', \bar{x}_2, \bar{x}_2']$ is a copy of $Q_1''[x_0^1, \bar{x}_1, \bar{x}_1']$. The pattern $Q_1''[\bar{x}] = (V_1, E_1, L_1)$ is defined as follows.

(*a*) The node set $V_1 = \{x_0^1, x_1^1, x_2^1, x_3^1\} \cup V_1^0 \cup V_1^1 \cup V_2^1 \cup V_3^1$, where (i) $\{x_0^1, x_1^1, x_2^1, x_3^1\}$ is the set of nodes in Q_1^1 denoting the 3 colors, and (ii) V_1^0, V_1^1, V_2^1 and V_3^1 contain 15, 5, 5 and 5 copies of x_0^1, x_1^1, x_2^1 and x_3^1, respectively. The node copies are created as follows. Consider the node x_r labeled r. For each (u, ι, v) in Q_1^1, we make two copies of the edge, one attached to u, and the other connected to v. Since x_r has five edges, there are 5 copies of x_r; similarly for the nodes labeled g or b. Moreover, we connect each copy with a node labeled 0, which will be used to recover the pattern Q_1^1. This explains why we need 15 copies of x_0^1.

To simplify the presentation of literals, we use, *e.g.,* $(x_1^1)_{x_0^1}^{(x_1^1, 0, x_0^1)}$ to denote the copy of x_1^1 connected to node x_0^1, which corresponds to the edge $(x_1^1, 0, x_0^1)$.

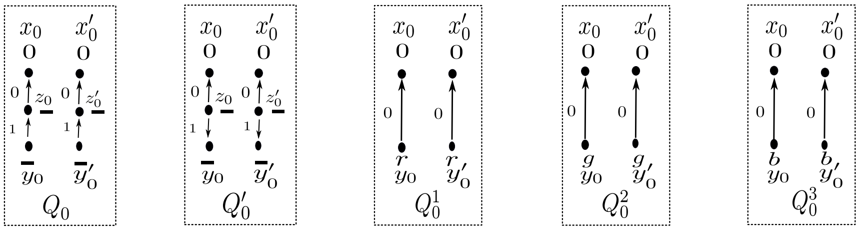

Fig. 8. The graph patterns for recovering φ_1

(b) The edge set $E_1 = \{(x_2^1, 0, x_0^1), (x_1^1, 1, x_2^1), (x_2^1, 1, x_3^1)\} \cup E_0^1 \cup E_1^1 \cup E_2^1 \cup E_3^1$, where (i) $\{(x_2^1, 0, x_0^1), (x_1^1, 1, x_2^1), (x_2^1, 1, x_3^1)\}$ is the set of edges of the spanning tree of Q_1^1 picked, and (ii) E_0^1, E_1^1, E_2^1 and E_3^1 are the sets of copies of the other edges of Q_1^1 corresponding to V_1^0, V_1^1, V_2^1 and V_3^1, respectively, as shown in Fig. 7. We use, *e.g.*, $((x_2^1)_{x_3^1}^{(x_2^1,1,x_3^1)}, 1, x_3^1)$, to denote the copy of the edge $(x_2^1, 1, x_3^1)$ connected to node x_3^1, and $((x_2^1)_{x_3^1}^{(x_2^1,1,x_3^1)}, 0, (x_0^1)_{x_3^1,x_2^1}^{(x_2^1,1,x_3^1)})$ to denote the edge between the copy $(x_2^1)_{x_3^1}^{(x_2^1,1,x_3^1)}$ and the corresponding node labeled 0, denoted by $(x_0^1)_{x_3^1,x_2^1}^{(x_2^1,1,x_3^1)}$.

(c) Labeling L_1 is given as follows: for each node $v \in \{x_0^1\} \cup V_1^0$, $L_1(v) = 0$; for each $v \in \{x_1^1\} \cup V_1^1$, $L_1(v) = r$; for each $v \in \{x_2^1\} \cup V_2^1$, $L_1(v) = g$; and for each $v \in \{x_3^1\} \cup V_3^1$, $L_1(v) = b$.

This completes the construction of the first GKey.

Next, we show how to recover the pattern Q_1^1. Note that although the nodes in Q_1^1 have distinct labels, we cannot simply merge nodes with the same labels, since Q_2^1 also has these labels. The trick is to only allow nodes in Q_1^1 to be labeled 0. Hence we add the following six GKeys:

$$\varphi_1^{0,0} = Q_0[x_0, z_0, y_0, x'_0, z'_0, y'_0](y_0.\text{id} = z'_0.\text{id} \Rightarrow x_0.\text{id} = x'_0.\text{id}), \quad (1)$$

$$\varphi_1^{0,1} = Q'_0[x_0, z_0, y_0, x'_0, z'_0, y'_0](y_0.\text{id} = z'_0.\text{id} \Rightarrow x_0.\text{id} = x'_0.\text{id}), \quad (2)$$

$$\varphi_0^{i} = Q_0^i[x_0, y_0, x'_0, y'_0](x_0.\text{id} = x'_0.\text{id} \Rightarrow y_0.\text{id} = y'_0.\text{id}) \text{ for } i \in [1,3], \quad (3)$$

where Q_0, Q'_0 and Q_0^i are the patterns given in Fig. 8. Note that Q_0 and Q'_0 differ in the directions of edges between z_0 and y_0 and between z'_0 and y'_0. Intuitively, $\varphi_1^{0,0}$ and $\varphi_1^{0,1}$ are used to merge nodes with label 0; and φ_0^1, φ_0^2 and φ_0^3 are to merge nodes with the same labels r, g and b, respectively.

We now illustrate how to recover Q_1^1 by using these GKeys. We show how to merge those nodes labeled 0. It is easy to see that once all nodes labeled 0 are merged (by applying φ_0^1, φ_0^2 and φ_0^3), we can recover pattern Q_1^1. To simplify the presentation, we name the nodes in $G_{Q_1''}$ as shown in Fig. 9. Observe that there are three types of copies of the node labeled 0: (1) the copy connected to the original nodes in Q_1^1, *e.g.*, x_{10}, x_{20}, and x_{30}; (2) the copy connected to the nodes that have edges linked to the original nodes in Q_1^1, *e.g.*, x_3, x_4; and (3) the copy connected to the nodes that have edges linked from the original nodes in Q_1^1, *e.g.*, x_6, x_8. We merge all these nodes with node x_{31}. Here we only show how to merge x_{10} and x_{31}; the other nodes can be merged similarly. To merge x_{10} and x_{31}, we can apply $\varphi_1^{0,0}$ and the following match h_1: $x_0 \mapsto x_{10}$, $y_0 \mapsto x_{11}$, $z_0 \mapsto x_1$, $x'_0 \mapsto x_{31}$, $y'_0 \mapsto x_{21}$, and $z'_0 \mapsto x_{11}$. Since $h_1(z'_0) = h_1(y_0) = x_{11}$, by $\varphi_1^{0,0}$, we have that $h_1(x_0) = h_1(x'_0)$. That is, $x_{10}.\text{id} = x_{31}.\text{id}$. After merging x_{10} and x_{31}, we obtain the pattern Q_1''' given in Fig. 9.

One can verify that if chase(G_Σ, Σ) is consistent, then Q_1^1 is a subgraph of G_{Eq}, where $(\text{Eq}, G_{\text{Eq}})$ is the result of chasing G_Σ by Σ, and Σ contains $\varphi_1^{0,0}$, $\varphi_1^{0,1}$ and φ_0^i for $i \in [1,3]$.

(2) We next define seven keys to encode Q_2^1. The construction is similar to the one for Q_1^1, except that all the nodes labeled 0 in the patterns are replaced by nodes labeled 1. We define

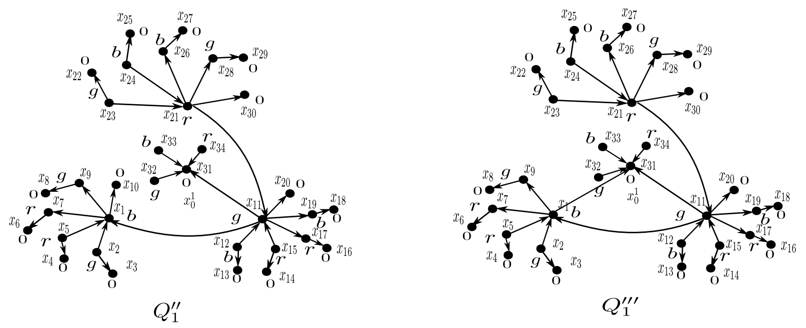

Fig. 9. recovering φ_1

$$\varphi_2^{0,0} \quad = \quad Q_0''[x_0, z_0, y_0, x_0', z_0', y_0'](y_0.\mathrm{id} = z_0'.\mathrm{id} \Rightarrow x_0.\mathrm{id} = x_0'.\mathrm{id}), \tag{4}$$

$$\varphi_2^{0,1} \quad = \quad Q_0'''[x_0, z_0, y_0, x_0', z_0', y_0'](y_0.\mathrm{id} = z_0'.\mathrm{id} \Rightarrow x_0.\mathrm{id} = x_0'.\mathrm{id}), \tag{5}$$

$$(\varphi_0^i)' \quad = \quad (Q_0^i)'[x_0, y_0, x_0', y_0'](x_0.\mathrm{id} = x_0'.\mathrm{id} \Rightarrow y_0.\mathrm{id} = y_0'.\mathrm{id}) \text{ for } i \in [1,3], \tag{6}$$

where Q_0'', Q_0''' and $(Q_0^i)'$ are patterns obtained from Q_0, Q_0' and Q_0^i depicted in Fig. 8 by replacing nodes labeled 0 by nodes labeled 1, respectively.

Similar to (1), one can verify that if chase(G_Σ, Σ) is consistent, then Q_2^1 is a subgraph of G_{Eq}, where $(\mathsf{Eq}, G_{\mathsf{Eq}})$ is the result of chasing G_Σ by Σ, and Σ consists of $\varphi_2^{0,0}$, $\varphi_2^{0,1}$ and $(\varphi_0^i)'$ for $i \in [1,3]$.

(3) The last group consists of a single GKey φ_3, to encode the structure of Q_1^3, which in turn encodes graph G. More specifically, $\varphi_3 = Q[\bar{x}_2](X \Rightarrow x_0.\mathrm{id} = x_0'.\mathrm{id})$, where Q consists of a pattern Q' and its copy Q'', and can be constructed along the same lines as Q_1^1 and Q_2^1. We omit the details here. In the following, we only show the literals in X, which are used to recover the structure of Q_1^3 from Q. Here we use literals to recover Q_1^3, rather than GKeys as we did for Q_1^1 and Q_2^1.

From the constructions in (1) and (2) above, we can see that to recover Q_1^3, we only need to merge the copies of nodes with the original ones. Employing the notations for copies given earlier, we can define the set X of literals as follows. Suppose that E_2 is the set of edges in Q_3^1. Given an edge (u, ι, v) in E_2, there are two copies of this edge, one of which is connected to u, and the other is connected to v. Meanwhile, we connect two copies of node x_0 to these copies of u and v. Then X is

$$\bigwedge_{(u,\iota,v)\in E_2} \big(((u)_v^{(u,\iota,v)}.\mathrm{id} = u.\mathrm{id}) \wedge ((v)_u^{(u,\iota,v)}.\mathrm{id} = v.\mathrm{id}) \wedge ((x_0)_{u,v}^{(u,\iota,v)} = x_0.\mathrm{id}) \wedge ((x_0)_{v,u}^{(u,\iota,v)} = x_0.\mathrm{id})\big).$$

One can verify that $(G_Q)_{\mathsf{Eq}_X}$ is unique, and it contains Q_3^1 as a subgraph. Therefore, if there exists a match of Q in a graph G' such that $h(\bar{x}_2) \models X$, then we can deduce a match of Q_3^1 in G'.

One can show that Σ is not satisfiable if and only if G has a proper 3-coloring (see the appendix). □

This completes the proof of Lemma 5.7 and the proof of Theorem 5.4. □

5.2 The Implication Problem

A set Σ of GEDs *implies* another GED φ, denoted by $\Sigma \models \varphi$, if for all graphs G, if $G \models \Sigma$ then $G \models \varphi$. We consider finite implication, when G is finite.

The *implication* problem for GEDs is as follows:

- Input: A finite set Σ of GEDs and another GED φ.
- Question: Does $\Sigma \models \varphi$?

As remarked earlier, the implication analysis helps us optimize data quality rules and graph pattern queries in practice, among other things.

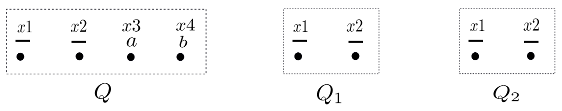

Fig. 10. The implication of GEDs

Characterization. We characterize the implication $\Sigma \models \varphi$ as follows. Assume $\varphi = Q[\bar{x}](X \Rightarrow Y)$, where pattern $Q = (V_Q, E_Q, L_Q)$. We use the following notations.

(a) The *canonical graph* of Q is $G_Q = (V_Q, E_Q, L_Q, F_A)$, where F_A is empty, along the same lines as G_Σ (Section 5.1).

(b) We use Eq_X to denote *the equivalence relation of* X *in* G_Q, such that for any literal l in X, $v \in [u]_{\mathsf{Eq}}$, where l is $u = v$, denoting $x.A = c$, $x.A = y.B$ or $x.\mathrm{id} = y.\mathrm{id}$. Moreover, Eq_X contains $[x]_{\mathsf{Eq}_X} = \{x\}$ for each $x \in V_Q$.

(c) We use $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ to denote the result of the chase of G_Q by Σ starting with Eq_X. Note that it is inconsistent if Eq_X is inconsistent (see Section 4).

(d) We say that a literal l can be *deduced* from an equivalence relation Eq if $v \in [u]_{\mathsf{Eq}}$, where l is $u = v$. That is, the equality specified by l can be deduced from the transitivity of equality literals, and the semantics of id literals in Eq. We say that a set Y of literals can be *deduced* from Eq if each literal of Y can be deduced from Eq.

Theorem 5.8. *For a set* Σ *of* GEDs *and a* GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$, $\Sigma \models \varphi$ *if and only if either (1)* $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ *is inconsistent; or (2)* $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ *is consistent and* Y *can be deduced from* $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$. □

Intuitively, if $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ is inconsistent, then for all graphs $G \models \Sigma$ and for all matches $h(\bar{x})$ of pattern Q in G, $h(\bar{x}) \not\models X$. Condition (1) covers this case. Otherwise, if $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ is consistent, condition (2) ensures that Y is a logical consequence of Σ, Q and X.

Example 5.9. Consider a set $\Sigma_1 = \{\phi_1, \phi_2\}$ and φ:

$$\phi_1 = Q_1[x_1, x_2](x_1.A = x_2.A \Rightarrow x_1.\mathrm{id} = x_2.\mathrm{id}),\ \ \phi_2 = Q_2[x_1, x_2](x_1.B = x_2.B \Rightarrow x_1.A = x_1.B),$$
$$\varphi\ = Q[x_1, x_2, x_3, x_4](X \Rightarrow Y),$$

where Q, Q_1 and Q_2 are shown in Fig. 10, $X = (x_1.A = x_3.A \wedge x_2.B = x_4.B)$, and $Y = (x_1.\mathrm{id} = x_3.\mathrm{id} \wedge x_2.\mathrm{id} = x_4.\mathrm{id})$. Canonical graph G_Q has the same form as pattern Q of Fig. 10. Then $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ yields all literals in Y, and $\Sigma \models \varphi$. Note that x_3 and x_4 have distinct labels, and each is identified with a node labeled '_': $x_3 \in [x_1]_{\mathsf{Eq}}$ and $x_4 \in [x_2]_{\mathsf{Eq}}$, where Eq is the result of the chase. This explains why we use $\asymp$ when comparing labels (see Section 4). □

Theorem 5.8 tells us that to decide whether $\Sigma \models \varphi$, it suffices to chase the canonical graph G_Q of pattern Q. Again we chase a graph pattern G_Q with GEDs.

Proof: To verify the characterization, we prove two lemmas. Consider a terminal chasing sequence $(\mathsf{Eq}_0 = \mathsf{Eq}_X, \mathsf{Eq}_1, \ldots, \mathsf{Eq}_k)$ of G_Q by Σ starting with Eq_X, where G_Q is the canonical graph of Q.

Lemma 5.10. *For graph* G *and match* $h(\bar{x})$ *of* Q *in* G*, if* $G \models \Sigma$ *and* $h(\bar{x}) \models X$*, then* $h(\bar{x}) \models \mathsf{Eq}_k$. □

Lemma 5.11. *When* $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ *is consistent,* Y *can be deduced from* $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ *if and only if for any graph* G *and any match* h *of* Q *in* G*, if* $h(\bar{x}) \models \mathsf{Eq}_k$*, then* $h(\bar{x}) \models Y$. □

The proofs of Lemmas 5.10 and 5.11 are in the electronic appendix.

Using Lemmas 5.10 and 5.11, we prove Theorem 5.8 as follows.

($\Leftarrow$) We consider two cases, when $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ is inconsistent and when $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$

is consistent. First consider inconsistent chase($G_Q, \mathsf{Eq}_X, \Sigma$). If Eq_X is inconsistent, then obviously $\Sigma \models \varphi$, since for any graph G and any match h of Q in G, $h(\bar{x}) \not\models X$. Hence we assume *w.l.o.g.* that Eq_X is consistent. Then for any graph G such that $G \models \Sigma$, if there exists a match h of Q in G such that $h(\bar{x}) \models X$, then $h(\bar{x}) \models \mathsf{Eq}_k$ by Lemma 5.10. Since chase($G_Q, \mathsf{Eq}_X, \Sigma$) is inconsistent, one can verify that $G \not\models \Sigma$ along the same lines as the proof of Theorem 5.2, a contradiction. Hence $\Sigma \models \varphi$.

Now consider the case when chase($G_Q, \mathsf{Eq}_X, \Sigma$) is consistent, from which Y can be deduced. Then for any graph G such that $G \models \Sigma$ and any match h of Q in G such that $h(\bar{x}) \models X$, we have that $h(\bar{x}) \models \mathsf{Eq}_k$ by Lemma 5.10. Since Y can be deduced from chase($G_Q, \mathsf{Eq}_X, \Sigma$), from Lemma 5.11 it follows that $h(\bar{x}) \models Y$. Therefore, $\Sigma \models \varphi$.

($\Rightarrow$) Conversely, assume that $\Sigma \models \varphi$. We show that if chase($G_Q, \mathsf{Eq}_X, \Sigma$) is consistent, then Y can be deduced from chase($G_Q, \mathsf{Eq}_X, \Sigma$). By Lemma 5.11, it suffices to show that for any graph $G = (V, E, L, F_A)$ and any match h of Q in G, if $h(\bar{x}) \models \mathsf{Eq}_k$, then $h(\bar{x}) \models Y$.

Given such G and h, we next show that $h(\bar{x}) \models Y$. Since $h(\bar{x}) \models \mathsf{Eq}_k$ and $\mathsf{Eq}_X \subseteq \mathsf{Eq}_k$, we have that $h(\bar{x}) \models \mathsf{Eq}_X$. That is, $h(\bar{x}) \models X$. We consider two cases. (a) If $G \models \Sigma$, then $h(\bar{x}) \models Y$ since $\Sigma \models \varphi$ and $h(\bar{x}) \models X$. (b) If $G \not\models \Sigma$, we show that there exists a subgraph G_h of G such that $G_h \models \Sigma$, h is also a match of Q in G_h, and moreover, $h(\bar{x}) \models \mathsf{Eq}_k$. From these we can get $h(\bar{x}) \models Y$. Let $Q = (V_Q, E_Q, L_Q)$. We define $G_h = (V', E', L', F'_A)$ as follows:

- $V' = \{h(x) \mid x \in V_Q\}$;
- $E' = \{(h(y_1), h(\iota^{y_1}_{y_2}), h(y_2)) \mid (y_1, \iota, y_2) \in E_Q\}$;
- for each node x in V_Q, $L'(h(x)) = L(h(x))$; and
- for each node $x \in V_Q$, $F'_A(h(x)) = F_A(h(x))$.

Obviously, G_h is a subgraph of G, and h is also a match of Q in G_h. Moreover, $h(\bar{x}) \models \mathsf{Eq}_k$ still holds. We show that $G_h \models \Sigma$ by contradiction. Suppose that $G_h \not\models \Sigma$. Then there exist a GED $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in Σ, and a match h_1 of Q_1 in G_h such that $h_1(\bar{x}_1) \models X_1$, but $h_1(\bar{x}_1) \not\models Y_1$. Then along the same lines as the proof of Theorem 5.2, we can show that the chasing sequence ($\mathsf{Eq}_0 = \mathsf{Eq}_X, \mathsf{Eq}_1, \ldots, \mathsf{Eq}_k$) is either not terminal or invalid, which contradicts the assumption that chase($G_\Sigma, \mathsf{Eq}_X, \Sigma$) is consistent. Therefore, $G_h \models \Sigma$. □

Complexity. Based on the characterization, we settle the complexity of GED implication.

THEOREM 5.12. *The implication problem is* NP*-complete for* GEDs, GFDs, GKeys, $\mathrm{GFD_x}$s *and* $\mathrm{GED_x}$s. *The problem remains* NP*-hard for* GEDs, GFDs, GKeys, $\mathrm{GFD_x}$s, $\mathrm{GED_x}$s *with tree patterns.* □

As opposed to Theorem 5.4, the implication analysis for $\mathrm{GFD_x}$s is NP-hard, in the absence of constant and id literals, although chase($G_Q, \mathsf{Eq}_X, \Sigma$) is always consistent in this case. This is because to check whether Y can be deduced from chase($G_Q, \mathsf{Eq}_X, \Sigma$), we need to examine all possible homomorphic mappings of patterns of Σ in G_Q. The intractability remains intact even when Σ consists of a single GED, and when GEDs of Σ and φ are defined with tree patterns.

Note that the implication for CFDs is coNP-complete [25], for the same reason as for the satisfiability analysis. While the implication problem for EGDs is NP-complete [9], the proofs are quite different, especially for the upper bound for GEDs and lower bound for GKeys, in the presence of id literals. Note that the implication problem for GEDs is not a dual of the satisfiability problem studied in Section 5.1, since that problem additionally requires the existence of pattern matches.

Proof: It suffices to show that the implication problem is in NP for GEDs, and is NP-hard for $\mathrm{GFD_x}$s and GKeys with tree patterns. Indeed, $\mathrm{GFD_x}$s are a special case of GFDs, $\mathrm{GED_x}$s and GEDs.

Upper bound. Given a set Σ of GEDs and a GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$, we develop an NP algorithm to check whether $\Sigma \models \varphi$. The algorithm works as follows:

(1) construct the canonical graph G_Q of Q and the equivalence relation Eq_X of X;
(2) guess a chasing sequence $\mathsf{Eq}_0 = \mathsf{Eq}_X \Rightarrow_{(\varphi_1,h_1)} \mathsf{Eq}_1 \Rightarrow_{(\varphi_2,h_2)} \dots \Rightarrow_{(\varphi_k,h_k)} \mathsf{Eq}_k$ of G_Q by Σ such that its number of steps $k \leq 8 \cdot |\varphi| \cdot |\Sigma|$;
(3) for each i $(0 \leq i \leq k-1)$, check whether $\mathsf{Eq}_i \Rightarrow_{(\varphi_{i+1},h_{i+1})} \mathsf{Eq}_{i+1}$ is a chase step; if not, reject the guess; continue otherwise.
(4) for each i $(0 \leq i \leq k-2)$, check whether each chase step $\mathsf{Eq}_i \Rightarrow_{(\varphi_{i+1},h_{i+1})} \mathsf{Eq}_{i+1}$ is valid; if not, reject the guess; continue otherwise;
(5) check whether Eq_k is inconsistent, if so, return true; otherwise, continue;
(6) check whether Y can be deduced from Eq_k, if so, return true.

We guess $h_1, \dots, h_k$ in the same way as in the proof of Theorem 5.4. The correctness of the algorithm is assured by Theorem 4.2 and the characterization given in Theorem 5.8. In particular, it is easy to verify the bound on the lengths of chasing sequences given in the proof of Theorem 4.2 still holds when we start with Eq_X instead of Eq_0.

For its complexity, note that step (1) is in PTIME, which follows from the definitions of G_Q and Eq_X. By Corollary 5.5, steps (3), (4) and (5) can be done in PTIME. Step (6) is also in PTIME by the following corollary. Therefore, the algorithm is in NP and so is the implication problem for GEDs.

Corollary 5.13. *Given a graph G, a set X of literals, and an equivalence relation* Eq *on G in a chasing sequence of G by Σ, it is in* PTIME *to check whether X can be deduced from* Eq. □

Proof: Since $|\mathsf{Eq}| \leq 4 \cdot |G| \cdot |\Sigma|$, it is in PTIME to check the following conditions: for each literal l in X, (a) when l is $x.A = c$ (resp. $x.A = y.B$), whether (i) $c \in [x.A]_{\mathsf{Eq}}$ (resp. $x.A \in [y.B]_{\mathsf{Eq}}$); or (ii) $c \in [x.A]_{\mathsf{Eq}}$ (resp. $x.A \in [y.B]_{\mathsf{Eq}}$) can be deduced by the closure of Eq under the reflexivity, symmetry and transitivity, as well as the semantics of id literals; and (b) when l is $x.\mathrm{id} = y.\mathrm{id}$, whether $y \in [x]_{\mathsf{Eq}}$. Hence, it is in PTIME to check whether X can be deduced from Eq. □

Lower bound. As remarked earlier, we show that the implication problem is NP-hard for GFD$_\mathsf{x}$s and GKeys with tree patterns. We start with GFD$_\mathsf{x}$s.

Lemma 5.14. *The implication problem is* NP*-hard for* GFD$_\mathsf{x}$s *with tree patterns.* □

Proof: We show that the implication problem is NP-hard by reduction from the 3SAT problem (see the proof of Lemma 5.6 for 3SAT). Given a 3SAT formula ψ, we construct a set Σ of GFD$_\mathsf{x}$s and a GFD$_\mathsf{x}$ φ such that all patterns in $\Sigma \cup \{\varphi\}$ are trees. We show that $\Sigma \models \varphi$ if and only if ψ is satisfiable.

The construction is almost the same as the one for Lemma 5.6, except that we need to remove all constant literals from the GFDs. We introduce two additional nodes to represent constants 0 and 1. The set Σ consists of only one GFD$_\mathsf{x}$ φ_1, which is to encode the structure of ψ (see the appendix). □

We next show that the implication problem is also intractable for GKeys with tree patterns. We show a stronger result: the intractability holds even when GKeys contain no constant literals.

Lemma 5.15. *The implication problem is* NP*-hard for* GKeys *with tree patterns, even in the absence of constant literals.* □

Proof: This is also verified by reduction from the 3-colorability problem (see the proof of Lemma 5.7 for the problem statement). Given a connected undirected graph $G = (V, E)$, we construct a set Σ of GKeys and another GKey φ such that $\Sigma \models \varphi$ if and only if G has a proper 3-coloring. The construction is similar to the proof of Lemma 5.7, except that we do not need the second group of GKeys, which is used to deduce inconsistency in the satisfiability analysis (see the appendix). □

This completes the proof of Theorem 5.12. □

5.3 The Validation Problem

The *validation problem* for GEDs is stated as follows.

- Input: A finite set Σ of GEDs and a graph G.
- Question: Does $G \models \Sigma$?

As remarked earlier, the validation analysis is the basis of inconsistency and spam detection, to find violations of GEDs in a knowledge base or a social graph, among other things.

While the validation problem for relational FDs and CFDs is in PTIME, it is harder for GEDs unless P = NP. Like the implication problem, the validation analysis is intractable even for GFD$_\text{x}$s, which are an extension of relational FDs that carry neither constant literals nor id literals. The intractability remains intact when Σ consists of a single GFD$_\text{x}$ or a single GKey defined with a tree pattern, and even when graph G is a tree.

Theorem 5.16. *The validation problem is* coNP*-complete for* GEDs, GFDs, GKeys, GFD$_\text{x}$s *and* GED$_\text{x}$s. *The problem remains* coNP*-hard for* GEDs, GFDs, GKeys, GFD$_\text{x}$s *and* GED$_\text{x}$s *with tree patterns, even when the given graph G is a tree.* □

The result is a bit surprising since it is in PTIME to decide, given graphs Q and G, whether there exists a homomorphism from Q to G when Q is a tree. As will be seen in the proof below, the presence of attribute dependencies $X \Rightarrow Y$ in GEDs makes the analysis harder. Indeed, we can encode the complement of 3SAT using a set Σ of GFD$_\text{x}$s with tree patterns in Σ and a tree G, and encode the complement of H-coloring problem using GKeys with tree patterns and a tree G [37].

Proof: We show that the validation problem is in coNP for GEDs, and is coNP-hard for GFD$_\text{x}$s and GKeys with tree patterns, when graph G is a tree. These suffice since GFD$_\text{x}$s are a special case of GFDs, GED$_\text{x}$s and GEDs. Hence GFD$_\text{x}$s and GKeys cover all the cases of Theorem 5.16.

Upper bound. We give an NP algorithm to check, given a graph G and a set Σ of GEDs, whether $G \not\models \Sigma$. The algorithm works as follows:

(1) guess a GED $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, and a mapping h from Q to G;
(2) check whether h is a match; if not, reject the current guess; otherwise continue;
(3) check whether $h(\bar{x}) \models X$, but $h(\bar{x}) \not\models Y$; if so, return true.

The correctness of the algorithm follows from the semantics of GEDs. For its complexity, step (2) is obviously in PTIME. Step (3) is also in PTIME since $|X| + |Y| \leq |\Sigma|$. Therefore, the algorithm is in NP, and hence the validation problem is in coNP for GEDs.

Lower bound. We next show that the validation problem is coNP-hard for GFD$_\text{x}$s and GKeys with tree patterns, when G is a tree. We start with the intractability of the validation problem for GFD$_\text{x}$s.

Lemma 5.17. *The validation problem is* coNP*-hard for* GFD$_\text{x}$s *with tree patterns, even when the given graph G is a tree.* □

Proof: This is verified by reduction from the complement of the 3SAT problem. Given a 3SAT formula ψ, we construct a set Σ of GFD$_\text{x}$s and a graph G_1, which are also trees. The construction and the proof are similar to those given for Lemma 5.14 (see the appendix for a proof). □

We next prove the intractability of the validation problem for a special case of GKeys.

LEMMA 5.18. *The validation problem is* coNP*-hard for* GKeys *with tree patterns, even in the absence of constant literals, and even when the given graph G is a tree.* □

Proof: We prove this by reduction from the complement of the H-coloring problem, which is NP-complete [37]. For a fixed graph H, the H-coloring problem is to decide, given a graph G, whether there exists a homomorphism from G to H. It remains NP-complete even when H is a tree.

Given graph G, we construct a graph G_1 and a set Σ of GKeys such that G_1 and the patterns in Σ are all trees. Moreover, we show that $G_1 \not\models \Sigma$ if and only if there exists a homomorphism from G to H. In the construction, we adopt the tree H given in Fig. 6 of [37] (see the appendix for a proof). □

This completes the proof of Theorem 5.16. □

Tractable cases. The main conclusion of this section is that the intractability of the analyses of GEDs is quite robust. As shown above, even for GEDs defined in terms of tree patterns, the satisfiability, implication and validation problems remain intractable. Nonetheless, the static analyses of GEDs are no harder than their counterparts for, *e.g.*, relational CFDs [25].

There are tractable cases that allow us to make practical use of GEDs. For example, one may consider a set Σ of GEDs in which graph patterns have a size at most k, for a predefined bound k. This is practical. Indeed, real-life graph patterns often have a small size [11, 31, 47]: 98% of SPARQL queries have no more than 4 nodes and 5 edges, and single-triple patterns account for 97.25% of patterns in SWDF and 66.41% of DBPedia [31]. One can readily verify that the satisfiability, implication and validation problems for GEDs are in PTIME when patterns have a bounded size k. Indeed, when the sizes of patterns in a set of GEDs are bounded by a constant k, we can enumerate all possible matches of the patterns in PTIME; from this it is easy to develop PTIME algorithms for checking the satisfiability, implication and validation of GEDs. In addition, it has been shown in [26] that under the subgraph isomorphism semantics for pattern matching, for GFDs with k-bounded patterns, both the satisfiability problem and the implication problem are fixed-parameter tractable, and the validation problem is co-W[1]-hard; moreover, all these problems are also in PTIME.

6 FINITE AXIOMATIZABILITY

We next study the finite axiomatizability of GEDs. We naturally want a finite set $\mathcal{A}$ of inference rules to characterize GED implication, along the same lines as Armstrong's axioms for relational FDs [6]. As observed in [1], the finite axiomatizability of a dependency class is a stronger property than the existence of an algorithm for testing its implication. An axiom system reveals insight of logical implication, and can be used to generate symbolic proofs.

For a set Σ of GEDs and a GED φ, *a proof of φ from Σ using inference rules of $\mathcal{A}$* is a sequence

$$\varphi_1, \ \ldots, \ \varphi_n = \varphi,$$

such that each φ_i either is a GED in Σ, or a GED deduced from φ_j's by applying an inference rule (or axiom) in $\mathcal{A}$, for $j < i$ (see [1] for details about proofs).

We say that φ is *provable from Σ using $\mathcal{A}$*, denoted by $\Sigma \vdash_{\mathcal{A}} \varphi$, if there exists a proof of φ from Σ using $\mathcal{A}$. We write it as $\Sigma \vdash \varphi$ when $\mathcal{A}$ is clear from the context.

We say that for GEDs, an inference system $\mathcal{A}$ is

- *sound* if $\Sigma \vdash_{\mathcal{A}} \varphi$ implies $\Sigma \models \varphi$;
- *complete* if $\Sigma \models \varphi$ implies $\Sigma \vdash_{\mathcal{A}} \varphi$;

for all GED sets Σ and GEDs φ; and

- *independent* if for any rule $\mathrm{r} \in \mathcal{A}$, there exist GEDs Σ and φ such that $\Sigma \vdash_{\mathcal{A}} \varphi$ but $\Sigma \nvdash_{\mathcal{A}\setminus\mathrm{r}} \varphi$.

GED_1	$\Sigma \vdash Q[\bar{x}](X \Rightarrow X \wedge X_{\text{id}})$, where X_{id} is $\bigwedge_{i\in[1,n]}(x_i.\text{id} = x_i.\text{id})$, and $\bar{x}$ consists of x_i for all $i \in [1, n]$.
GED_2	If $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ and literal $(u.\text{id} = v.\text{id}) \in Y$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow u.A = v.A)$ for all attributes $u.A$ that appear in Y.
GED_3	If $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ and $(u = v) \in Y$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow v = u)$.
GED_4	If $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$, $(u_1 = v) \in Y$ and $(v = u_2) \in Y$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow u_1 = u_2)$.
GED_5	If $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ and $\text{Eq}_X \cup \text{Eq}_Y$ is inconsistent, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y_1)$ for any set Y_1 of literals of $\bar{x}$.
GED_6	If $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$, $\text{Eq}_X \cup \text{Eq}_Y$ is consistent, $\Sigma \vdash Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$, and if there exists a match h of Q_1 in $(G_Q)_{\text{Eq}_X\cup\text{Eq}_Y}$ such that $h(\bar{x}_1) \models X_1$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y \wedge h(Y_1))$.

Table 3. Axiom system $\mathcal{A}_{\text{GED}}$ for GEDs

Here $\mathcal{A} \setminus \text{r}$ denotes $\mathcal{A}$ excluding r. That is, removing any rule from $\mathcal{A}$ would make it no longer complete. We remark that we focus on finite implication, considering finite graphs.

We refer to $\mathcal{A}$ as a *finite axiom system* or a *finite axiomatization* of GEDs if $\mathcal{A}$ is sound, complete and independent for GEDs. Following [1], we say that GEDs are *finitely axiomatizable* if there exists a finite axiomatization of GEDs.

Inference rules. We give a set $\mathcal{A}_{\text{GED}}$ of rules for GEDs in Table 3, in which we denote by (a) $Q[\bar{x}]$ a pattern; (b) X a set of literals of $\bar{x}$; (c) $h(X)$ the set of literals obtained by substituting $h(x)$ for all variables x in X, for a match h of Q in a graph; (d) G_Q the canonical graph of pattern Q (Section 5.2); (e) Eq_X the equivalence relation of a set X of literals in G_Q; and (f) $(G_Q)_{\text{Eq}}$ the coercion of Eq on G_Q (Section 4). The consistency of an equivalence relation Eq is defined in Section 4. To simplify the presentation, we allow $c = x.A$ as a literal in intermediate results of a proof, for constant c.

Recall that Armstrong's axioms consist of three rules for relational FDs: reflexivity, augmentation and transitivity [6]. Four rules are needed for each of CFDs [25] and EGDs [50]. In contrast, $\mathcal{A}_{\text{GED}}$ has six rules for GEDs over graphs (Table 3), to deal with the semantics of id literals, the consistency of literals, and graph pattern matching for GEDs over graphs. In fact, reflexivity, augmentation and transitivity can be deduced from $\mathcal{A}_{\text{GED}}$, as shown below.

Lemma 6.1. *The axiom system $\mathcal{A}_{\text{GED}}$ has the following properties.*

(*a*) *[Reflexivity] If $\Sigma \vdash \varphi$, $\varphi = Q[\bar{x}](X \Rightarrow Y)$ and $Y_1 \subseteq Y$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y_1)$, where Y_1 is a set $\{u_i = v_i \mid i \in [1, n]\}$ of literals that are also in Y.*
(*b*) *[Augmentation] If $\Sigma \vdash \varphi_1$, then $\Sigma \vdash \varphi$, where $\varphi_1 = Q[\bar{x}](X \Rightarrow Y)$ and $\varphi = Q[\bar{x}](XZ \Rightarrow YZ)$.*
(*c*) *[Transitivity] If $\Sigma \vdash \varphi_1$ and $\Sigma \vdash \varphi_2$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Z)$, where $\varphi_1 = Q[\bar{x}](X \Rightarrow Y)$ and $\varphi_2 = Q[\bar{x}](Y \Rightarrow Z)$.* □

Proof: (a) We first prove the following property: When $X \cup Y$ is inconsistent, by GED_5, $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y_1)$. When $X \cup Y$ is consistent, we have

(1) $Q[\bar{x}](X \Rightarrow Y)$	φ
(2) $Q[\bar{x}](X \Rightarrow (v_1 = u_1))$	(1) and GED_3
(3) $Q[\bar{x}](X \Rightarrow (u_1 = v_1))$	(2) and GED_3
...	
(2n+1) $Q[\bar{x}](X \Rightarrow (u_n = v_n))$	(2n) and GED_3
(2n+2) $Q[\bar{x}](X \Rightarrow (u_1 = v_1)(u_2 = v_2))$	(3), (5) and GED_6
...	
(3n) $Q[\bar{x}](X \Rightarrow Y_1)$	(3n-1), (2n+1) and GED_6

To simplify the presentation, we denote this property as GED_7 and apply it in proofs, although GED_7 is not one of the rules in $\mathcal{A}_{\text{GED}}$.

(b) Recall the augmentation rule of Armstrong's axioms: if $X \Rightarrow Y$ then $XZ \Rightarrow YZ$ for any Z. Analogously, consider $\varphi_1 = Q[\bar{x}](X \Rightarrow Y)$, where $\Sigma \vdash \varphi_1$, and GED $\varphi = Q[\bar{x}](XZ \Rightarrow YZ)$. We show that $\Sigma \vdash \varphi$ using $\mathcal{A}_{\text{GED}}$ as follows. First consider the case when $\text{Eq}_X \cup \text{Eq}_Z$ is consistent:

(1) $Q[\bar{x}](XZ \Rightarrow XZ \wedge X_{\text{id}})$	GED_1
(2) $Q[\bar{x}](XZ \Rightarrow XZ)$	(1) and GED_7
(3) $Q[\bar{x}](X \Rightarrow Y)$	φ_1
(4) $Q[\bar{x}](XZ \Rightarrow XYZ)$	(2), (3) and GED_6
(5) $Q[\bar{x}](XZ \Rightarrow YZ)$	(4) and GED_7

When $\text{Eq}_X \cup \text{Eq}_Z$ is inconsistent, the proof consists of steps (1) and (2) above, followed by:

(3) $Q[\bar{x}](XZ \Rightarrow YZ)$	(2) and GED_5

(c) Let $\Sigma \vdash \varphi_1$ and $\Sigma \vdash \varphi_2$, where $\varphi_1 = Q[\bar{x}](X \Rightarrow Y)$ and $\varphi_2 = Q[\bar{x}](Y \Rightarrow Z)$. We show that $\Sigma \vdash Q[\bar{x}](X \Rightarrow Z)$ using $\mathcal{A}_{\text{GED}}$. When $\text{Eq}_X \cup \text{Eq}_Y$ is consistent, we have the following:

(1) $Q[\bar{x}](X \Rightarrow X \wedge X_{\text{id}})$	GED_1
(2) $Q[\bar{x}](X \Rightarrow X)$	(1) and GED_7
(3) $Q[\bar{x}](X \Rightarrow Y)$	φ_1
(4) $Q[\bar{x}](X \Rightarrow XY)$	(2), (3) and GED_6
(5) $Q[\bar{x}](Y \Rightarrow Z)$	φ_2
(6) $Q[\bar{x}](X \Rightarrow XYZ)$	(4), (5) and GED_6
(7) $Q[\bar{x}](X \Rightarrow Z)$	(6) and GED_7

If Eq_X is inconsistent, the proof has steps (1), (2) and the following:

(3) $Q[\bar{x}](X \Rightarrow Z)$	(2) and GED_5

If $\text{Eq}_X \cup \text{Eq}_Y$ is inconsistent, it has steps (1)–(3) and the following:

(4) $Q[\bar{x}](X \Rightarrow XY)$	(2), (3) and GED_6
(5) $Q[\bar{x}](X \Rightarrow Z)$	(4) and GED_5

These prove the transitivity, and hence the Armstrong's axioms also hold for GEDs. □

Axiomatization. We show that GEDs are finitely axiomatizable.

THEOREM 6.2. *The set $\mathcal{A}_{\text{GED}}$ of rules in Table 3 is sound, complete and independent for* GEDs. □

Proof: We show that $\mathcal{A}_{\text{GED}}$ is sound, complete, and independent one by one.

Soundness. We first show that the axiom system $\mathcal{A}_{\text{GED}}$ is sound, *i.e.*, every GED derived from Σ using $\mathcal{A}_{\text{GED}}$ is also implied by Σ. More specifically, we show that if $\Sigma \vdash \varphi$, then $\Sigma \models \varphi$. Suppose that $\Sigma \vdash \varphi$, where $\varphi = Q[\bar{x}](X \Rightarrow Y)$. We show that $\Sigma \models \varphi$. When chase$(G_Q, \text{Eq}_X, \Sigma)$ is inconsistent, we have that $\Sigma \models \varphi$ by Theorem 5.8. In the following, we only consider the case when chase$(G_Q, \text{Eq}_X, \Sigma)$ is consistent. Below we verify the soundness of $\mathcal{A}_{\text{GED}}$ by induction on the length of the proof $\varphi_0, \ldots, \varphi_n = \varphi$ of φ from Σ using $\mathcal{A}_{\text{GED}}$, by using Theorem 5.8.

Base case: GED_1. This is one of the base cases (the other base case corresponds to citing a GED from Σ and is trivial). Since $x_1, \ldots, x_n$ are all variables in $\bar{x}$, and X_{id} consists of $x.\text{id} = x.\text{id}$ only, it is easy to verify that $\Sigma \models Q[\bar{x}](X \Rightarrow X \wedge (x_1.\text{id} = x_1.\text{id}) \wedge \ldots \wedge (x_n.\text{id} = x_n.\text{id}))$.

Suppose that the soundness holds on proofs of length i or less. We next show that it also holds on proofs of length $i + 1$. Consider step $i + 1$ of the proof with one of the following rules.

GED_2: Assume that $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y_1)$. We show that $\Sigma \models Q[\bar{x}](X \Rightarrow (u.A = v.A))$, where $u.\text{id} = v.\text{id}$ is in Y_1. By the inductive hypothesis, $\Sigma \models Q[\bar{x}](X \Rightarrow Y_1)$. Since chase$(G_Q, \text{Eq}_X, \Sigma)$ is consistent and Y_1 can be deduced from chase$(G_Q, \text{Eq}_X, \Sigma)$, u and v must refer to the same node since the id literal $u.\text{id} = v.\text{id}$ is in Y_1. By Theorem 5.8, we can conclude that $\Sigma \models Q[\bar{x}](X \Rightarrow (u.A = v.A))$.

GED$_3$ *and* GED$_4$: The proofs are similar to the one for GED$_2$. We omit the details here.

GED$_5$: Assume that $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y')$ and $\mathsf{Eq}_X \cup \mathsf{Eq}_{Y'}$ is inconsistent. By the inductive hypothesis, $\Sigma \models Q[\bar{x}](X \Rightarrow Y')$. To verify that $\Sigma \models Q[\bar{x}](X \Rightarrow Y_1)$, it suffices to show that chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ is inconsistent, by Theorem 5.8. We prove this by contradiction. Suppose that chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ is consistent. By $\Sigma \models Q[\bar{x}](X \Rightarrow Y')$ and Theorem 5.8, Y' can be deduced from chase$(G_Q, \mathsf{Eq}_X, \Sigma)$. Thus $\mathsf{Eq}_X \cup \mathsf{Eq}_{Y'}$ can be deduced from chase$(G_Q, \mathsf{Eq}_X, \Sigma)$, and chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ is inconsistent. Thus by condition (1) of Theorem 5.8, $\Sigma \models Q[\bar{x}](X \Rightarrow Y_1)$.

GED$_6$: Assume that $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y')$, $\mathsf{Eq}_X \cup \mathsf{Eq}_{Y'}$ is consistent, and $\Sigma \vdash Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$. By the inductive hypothesis, we have that $\Sigma \models Q[\bar{x}](X \Rightarrow Y')$ and $\Sigma \models Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$. Since chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ is consistent, let chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ be $(\mathsf{Eq}, G_{\mathsf{Eq}})$. By the inductive hypothesis and Theorem 5.8, Y' can be deduced from Eq. We prove the soundness by contradiction. Suppose that $\Sigma \not\models Q[\bar{x}](X \Rightarrow Y' \wedge h(Y_1))$. Along the same line as the proofs of Lemma 4.3 and Theorem 5.2, we can construct a graph G and a match h_1 of Q_1 in G such that $G \models \Sigma$, $h_1(\bar{x}_1) \models X_1$, but $h_1(\bar{x}_1) \not\models Y_1$. That is, $G \models \Sigma$, but $G \not\models Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$. Then $\Sigma \not\models Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$, a contradiction. From this it follows that $\Sigma \models Q[\bar{x}](X \Rightarrow Y' \wedge h(Y_1))$, *i.e.*, the proof with its last step using GED$_6$ is also sound.

Hence the soundness holds for proofs of arbitrary lengths to derive φ from Σ using $\mathcal{A}_{\mathsf{GED}}$.

Completeness. We next show that every GED implied by Σ can also be derived from Σ using $\mathcal{A}_{\mathsf{GED}}$. More specifically, we show that if $\Sigma \models Q[\bar{x}](X \Rightarrow Y)$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$. It suffices to show the following. Suppose that $(\mathsf{Eq}_1 = \mathsf{Eq}_X, \mathsf{Eq}_2, \ldots, \mathsf{Eq}_k)$ is a terminal chasing sequence of G_Q by Σ, denoted by ρ, where for each $i \in [1, k-1]$, $\mathsf{Eq}_i \Rightarrow_{(\varphi_i, h_i)} \mathsf{Eq}_{i+1}$ is a valid chase step, and $\varphi_i = Q_i[\bar{x}_i](X_i \Rightarrow Y_i)$. Then we claim the following.

Claim 1: For each Eq_i $(1 \leq i \leq k)$*,* $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_i)$.

Claim 2: If there exist a GED φ *in* Σ *and a match* h *such that* $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ *and* Eq_{k+1} *is inconsistent in* G_{Eq_k}*, then* $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$.

The detailed proofs of these claims can be found in the electronic appendix.

With these claims, we can verify $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ as follows. Given that $\Sigma \models Q[\bar{x}](X \Rightarrow Y)$, by Theorem 5.8, we consider the following two cases: (a) chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ is inconsistent; or (b) chase$(G_Q, \mathsf{Eq}_X, \Sigma)$ is consistent and Y can be deduced from chase$(G_Q, \mathsf{Eq}_X, \Sigma)$. In case (a), if Eq_X is inconsistent, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ by GED$_1$ and GED$_5$. When Eq_X is consistent, by claim (2) above, there exists an inconsistent Eq_{k+1} such that $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$. Since Eq_{k+1} is inconsistent, $\mathsf{Eq}_X \cup \mathsf{Eq}_{k+1}$ is also inconsistent. Then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ by GED$_5$. In case (b), let $(\mathsf{Eq}, G_{\mathsf{Eq}})$ be the result of a valid terminal chasing sequence. Then by Theorem 4.2, $\mathsf{Eq} = \mathsf{Eq}_k$, and Y can be deduced from Eq by Theorem 5.8. By claim (1), $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq})$. Since Y can be deduced from Eq, we can show that $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ by applying GED$_1$, GED$_2$, GED$_3$, GED$_4$ and GED$_6$. More specifically, $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$ is verified as follows (to save space, below we combine multiple steps into one):

(1) $Q[\bar{x}](X \Rightarrow X \wedge X_{\mathsf{id}})$	GED$_1$
(2) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_1)$	(1), GED$_2$, GED$_3$, GED$_4$, and GED$_6$
(3) $Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$	φ_1 used in chase step $\mathsf{Eq}_1 \Rightarrow_{(\varphi_1, h_1)} \mathsf{Eq}_2$
(4) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_2)$	(2), (3), GED$_6$, and GED$_7$
(5) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_2')$	(4), GED$_2$, GED$_3$, GED$_4$, and GED$_6$
...	
(3k-3) $Q_{k-1}[\bar{x}_{k-1}](X_{k-1} \Rightarrow Y_{k-1})$	φ_{k-1} used in chase step $\mathsf{Eq}_{k-1} \Rightarrow_{(\varphi_{k-1}, h_{k-1})} \mathsf{Eq}_k$
(3k-2) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_k)$	(3k-3), (3k-4), GED$_6$, and GED$_7$
(3k-1) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_k')$	(3k-2), GED$_2$, GED$_3$, GED$_4$, and GED$_6$
(3k) $Q[\bar{x}](X \Rightarrow Y)$	(3k-1), and GED$_7$

Here GED_7 is the property proved in Lemma 6.1, which we use here to simplify the presentation; φ_i is the GED used in the chase step $\text{Eq}_i \Rightarrow_{(\varphi_i, h_i)} \text{Eq}_{i+1}$ $(i \in [1, k-1])$ for computing chase$(G_Q, \text{Eq}_X, \Sigma)$, and Eq'_i $(i \in [2, k])$ denotes the closure of Eq_i under the reflexivity, symmetry and transitivity and the semantics of id literals. In the proof of Claim 1, we will show how to compute Eq_1 from X by using rules in $\mathcal{A}_{\text{GED}}$; similarly for Eq'_i. These verify that $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$. □

Independence. We next show that we cannot remove any rule from $\mathcal{A}_{\text{GED}}$. More specifically, we show that for any rule $\mathrm{r} \in \mathcal{A}_{\text{GED}}$, there exist a set Σ of GEDs and a GED φ such that $\Sigma \vdash_{\mathcal{A}_{\text{GED}}} \varphi$ but $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. That is, Σ and φ witness the necessity of rule r.

We verify this statement for each rule of $\mathcal{A}_{\text{GED}}$ as follows.

$\underline{\text{GED}_1}$. Let $\Sigma = \emptyset$ and $\varphi = Q_1[x](\emptyset \Rightarrow x.\text{id} = x.\text{id})$, where Q_1 consists of a single node x. It is easy to see that $\Sigma \vdash_{\mathcal{A}_{\text{GED}}} \varphi$ by using GED_1. Now consider $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$, where r stands for GED_1. Then $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. Indeed, all the rules in $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$ require as a precondition that there exists another GED φ_1 such that $\Sigma \vdash \varphi_1$. When $\Sigma = \emptyset$, we cannot derive any GED from Σ without using GED_1, *i.e.,* no rule in $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$ can be applied. Thus $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. This shows the independence of GED_1.

$\underline{\text{GED}_2}$. Let $\Sigma = \emptyset$ and $\varphi = Q_2[x, y](X \Rightarrow Y)$, where Q_2 consists of two isolated nodes, X is $(x.\text{id} = y.\text{id}) \wedge (x.A = 2)$, and Y is $y.A = x.A$. Then $\Sigma \vdash_{\mathcal{A}_{\text{GED}}} \varphi$ with GED_1 and GED_2. In contrast, consider $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$, where r stands for GED_2. Since $\Sigma = \emptyset$, and neither $y.A = x.A$ nor $x.A = y.A$ is in X, no rules in $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$ can derive $y.A = x.A$, *i.e.,* $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. Thus GED_2 is independent.

$\underline{\text{GED}_3}$. Consider $\Sigma = \emptyset$ and $\varphi = Q_3[x, y](x.A = y.B \Rightarrow y.B = x.A)$, where Q_3 consists of two isolated nodes, which are labeled with τ_1 and τ_2, respectively. Then $\Sigma \vdash_{\mathcal{A}_{\text{GED}}} \varphi$ by using GED_1 and GED_3. Now consider $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$, where r is GED_3. Then no rule in $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$ allows us syntactically deduce $y.B = x.A$. In particular, GED_6 does not apply since there exists only one match of Q_3 in G_{Q_3}. That is, $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. Hence GED_3 is independent.

$\underline{\text{GED}_4}$. Consider $\Sigma = \emptyset$ and $\varphi = Q_4[x, y, z]((x.A = y.B) \wedge (y.B = z.C) \Rightarrow x.A = z.C)$, where Q_4 consists of three isolated nodes, which are labeled with τ_1, τ_2 and τ_3, respectively. Then $\Sigma \vdash_{\mathcal{A}_{\text{GED}}} \varphi$ by using GED_1 and GED_4. In contrast, consider $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$, where r is GED_4. One can verify that $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. Indeed, since there exist no id literals in Σ and φ, one cannot derive $x.A = z.C$ by applying GED_1 or GED_2. Because the literals in φ are consistent, we cannot apply GED_5. Moreover, since $\Sigma = \emptyset$ and there exists only one match of Q_4 in G_{Q_4}, applying GED_6 cannot deduce new literals. In addition, there exist no other GEDs containing literals like $x.A = z.C$ and $z.C = x.A$ in Σ; hence we cannot derive $x.A = z.C$ using GED_3. Therefore, GED_4 is independent.

$\underline{\text{GED}_5}$. Let $\Sigma = \emptyset$ and $\varphi = Q_5[x]((x.A = 1) \wedge (x.A = 2) \Rightarrow x.A = 3)$, where Q_5 consists of a single node. Then $\Sigma \models_{\mathcal{A}_{\text{GED}}} \varphi$ by using GED_1 and GED_5. Similar to the analysis of GED_4, one can see that no rule in $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$ can derive $x.A = 3$, where r stands for GED_5, since there exists no constant 3 in Σ or $(x.A = 1) \wedge (x.A = 2)$. Therefore, $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. This justifies the independence of GED_5.

$\underline{\text{GED}_6}$. Define $\Sigma = \{\varphi_1\}$, $\varphi_1 = Q'_6[x, y](\emptyset \Rightarrow x.\text{id} = y.\text{id})$ and $\varphi = Q_6[x, y](X \Rightarrow Y)$, where both Q_6 and Q'_6 consist of two isolated nodes of label τ, X is $x.A = 1$ and Y is $y.A = 1$. Then $\Sigma \vdash_{\mathcal{A}_{\text{GED}}} \varphi$ with GED_1, GED_2, GED_4 and GED_6. However, no rule in $\mathcal{A}_{\text{GED}} \setminus \mathrm{r}$ can derive $y.A = 1$, where r is GED_6, since $y.A$ appears in neither X nor Σ. Thus $\Sigma \not\vdash_{\mathcal{A}_{\text{GED}} \setminus \mathrm{r}} \varphi$. Hence GED_6 is independent. □

This completes the proof of Theorem 6.2. □

7 EXTENSIONS OF GEDS

We next extend GEDs by supporting built-in predicates (Section 7.1) or disjunctions (Section 7.2). We show that the extensions complicate the static analyses. We defer the proofs of the results of this section to the electronic appendix due to the space constraint.

7.1 Denial Constraints for Graphs

We extend GEDs with built-in predicates, referred to as *graph denial constraints*, denoted by GDCs.

GDCs. A GDC ϕ is defined as $Q[\bar{x}](X \Rightarrow Y)$, where Q is a pattern, and X and Y are sets of literals of one of the following forms: (a) $x.A \oplus c$, (b) $x.A \oplus y.B$, for constant $c \in U$, and non-id attributes $A, B \in \Upsilon$, and (c) $x.\mathrm{id} = y.\mathrm{id}$; here $\oplus$ is one of built-in predicates $=, \neq, <, >, \leq, \geq$.

Along the same lines as GEDs, we define $G \models \phi$ for a graph G; similarly for other notions. Obviously GEDs are a special case of GDCs when $\oplus$ is equality '=' only. One can verify that GDCs can express denial constraints of [3] when relation tuples are represented as vertices in a graph.

Example 7.1. Denial constraints can be used to enforce domain constraints and catch spam.

(1) We can express "domain constraints" as GDCs, to enforce each node of "type" τ to have an attribute with a finite domain (*e.g.*, Boolean) as follows:

$$\phi_1\colon Q_e[x](\emptyset \Rightarrow x.A = x.A), \qquad \phi_2\colon Q_e[x](x.A \neq 0 \wedge x.A \neq 1 \Rightarrow \mathsf{false}).$$

Here Q_e consists of a single node labeled τ, ϕ_1 is a GED that enforces each τ-node x to have an A-attribute, and ϕ_2 ensures that $x.A$ can only takes values 0 or 1.

(2) GDC $\varphi_7 = Q_7[x, x', z_1, z_2, y_1, \ldots, y_k](X_8 \Rightarrow Y_8)$ specifies the spam-detection rule of Example 1.1. Here Q_7 is the pattern given in Fig. 1, X_8 consists of $x'.\mathrm{is\_fake}=1$, $z_1.\mathrm{keyword}=c$, $z_2.\mathrm{keyword}=c$, $y_i.\mathrm{content} \neq y_j.\mathrm{content}$ for any $i \neq j$ ($i, j \in [1, k]$), where c is a constant; and Y_8 consists of a single literal $x.\mathrm{is\_fake}=1$. The GDC says that for accounts and blogs matching Q_7, if account x' is confirmed fake and if both blogs z_1 and z_2 contain a peculiar keyword c, then x is also a fake account.

Note that this rule cannot be expressed by GEDs under the homomorphism semantics, since we can no longer enforce k blogs to be distinct as opposed to isomorphism semantics in [30]. □

Complexity. The increased expressive power of GDCs comes with a price. Recall that the satisfiability, implication and validation problems for GEDs are coNP-complete, NP-complete and coNP-complete, respectively. In contrast, the static analyses of GDCs have a higher complexity unless P = NP, although their validation problem gets no harder.

Theorem 7.2. *The satisfiability, implication and validation problems for* GDCs *are* Σ_2^p*-complete,* Π_2^p*-complete and* coNP*-complete, respectively.* □

The lower bounds of these problems remain intact when Σ consists of a fixed number of GDCs with variable and constant literals only, without id literals. The proof of Theorem 7.2 is more involved than their counterpart for GEDs (Theorems 5.4, 5.12 and 5.16).

Proof: (1) To prove the upper bound of the satisfiability problem, we establish a small model property, as opposed to the proof of Theorem 5.4 that is based on the chase. We show that if a set Σ of GDCs has a model, then it has a model of size at most $4 \cdot |\Sigma|^3$. The proof requires attribute value normalization and is more involved than its counterpart for Theorem 5.4. Based on the property, we give an Σ_2^p algorithm to check whether a given set of GDCs is satisfiable.

We show the lower bound by reduction from a generalized graph coloring problem (GGCP) [49, 52]. GGCP is to decide, given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, whether there exists a two-coloring of F such that G is not a monochromatic subgraph of F. A monochromatic subgraph of F is a subgraph in which nodes are assigned the same color. The problem is Σ_2^p-complete when G is a complete graph and F contains no self cycles [49].

The reduction is a little complicated. We use a set Σ of four GDCs to encode 2-coloring, monochromatic G and graph F. These GDCs use constant and variable literals with $\neq$ and $\leq$, but employ no id literals. One of them is a forbidding constraint of the form $Q[\bar{x}](X \Rightarrow \text{false})$.

(2) For implication, we also show a small model property: if $\Sigma \not\models \varphi$, then there exists a graph G_h such that $|G_h| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$, $G_h \models \Sigma$ and $G_h \not\models \varphi$. Based on the small model property, we give an Σ_2^p algorithm to check whether $\Sigma \not\models \varphi$. The lower bound is verified by reduction from the complement of GGCP, using a set Σ of three GDCs of the form above.

(3) For validation, the lower bound follows from Theorem 5.16 since GEDs are a special case of GDCs. For the upper bound, we use the algorithm for checking $G \not\models \Sigma$ developed for GEDs in the proof of Theorem 5.16. We show that the algorithm also works for GDCs and remains in NP. □

7.2 Adding Disjunction

We next extend GEDs by adding limited disjunctions.

GED$^\vee$s. A GED ψ *with disjunction*, denoted by GED$^\vee$, has the same syntactic form $Q[\bar{x}](X \Rightarrow Y)$ as GEDs, but Y is interpreted as the disjunction of its literals. That is, for a match $h(\bar{x})$ of Q in a graph G, $h(\bar{x}) \models Y$ if *there exists* a literal $l \in Y$ such that $h(\bar{x}) \models l$. Hence we also write ψ as

$$Q[\bar{x}]\big(\textstyle\bigwedge_{l \in X} l \Rightarrow \bigvee_{l' \in Y} l'\big).$$

The other notions (*e.g.,* satisfiability and implication) remain the same as their GED counterparts.

GED$^\vee$s subsume GEDs. Each GED $Q[\bar{x}](X \Rightarrow Y)$ can be expressed as a set of $Q[\bar{x}](X \Rightarrow l)$ of GED$^\vee$s, one for each $l \in Y$. In contrast, some GED$^\vee$s are not expressible as GEDs.

Example 7.3. Recall GDCs from Example 7.1 that enforce $x.A$ to be Boolean. It is expressible as:

$$\psi: \ Q_e[x](\emptyset \Rightarrow x.A = 0 \vee x.A = 1).$$

This GED$^\vee$ specifies a domain constraint, to ensure that each τ-node x has an A-attribute and moreover, that $x.A$ can only take Boolean values. □

Complexity. Disjunctions also complicate the static analyses but do not make the validation analysis harder. The lower bounds remain intact when Σ consists of a fixed number of GED$^\vee$s with constant and variable literals only, in the absence of id literals.

Theorem 7.4. *The satisfiability, implication and validation problems for* GED$^\vee$s *are* Σ_2^p*-complete,* Π_2^p*-complete and* coNP*-complete, respectively.* □

Proof: The proof is similar to the one for Theorem 7.2. For satisfiability (resp. implication), the upper bound is also verified by means of a small model property, and the lower bound by reduction from (resp. the complement of) GGCP, by using a set Σ consisting of three GED$^\vee$s. □

8 CONCLUSION

We have proposed GEDs, which can uniformly express GFDs and keys for graphs. For GEDs, we have revised the chase with the Church-Rosser property, provided characterizations for their static analyses, settled the complexity of their satisfiability, implication and validation problems in various settings (Table 1), and shown the finite axiomatizability of their finite implication. We have also studied extensions of GEDs with built-in predicates or disjunction.

There is naturally much more to be done. One topic for future work is to identify practical special cases in which the satisfiability, implication and validation problems are tractable. Another topic is to develop parallel scalable algorithms for reasoning about GEDs, to warrant speedup with the increase of processors. A third topic is to develop effective methods to repair graph-structured data after semantic inconsistencies are detected by, *e.g.*, GEDs. It is also interesting to study other practical forms of graph dependencies, *e.g.*, an extension of TGDs to graphs.

ACKNOWLEDGMENTS

Fan and Lu are supported in part by 973 Program 2014CB340302, ERC 652976, NSFC 61421003, EPSRC EP/M025268/1, Foundation for Innovative Research Groups of NSFC, Joint Lab between Edinburgh and Huawei, and Beijing Advanced Innovation Center for Big Data and Brain Computing.

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases.* Addison-Wesley.

[2] Waseem Akhtar, Alvaro Cortés-Calabuig, and Jan Paredaens. 2010. Constraints in RDF. In *SDKB*. 23–39.

[3] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *PODS*.

[4] Marcelo Arenas, Jonny Daenen, Frank Neven, Martin Ugarte, Jan Van Den Bussche, and Stijn Vansummeren. 2014. Discovering XSD keys from XML data. *ACM Transactions on Database Systems (TODS)* 39, 4 (2014), 28.

[5] Marcelo Arenas and Leonid Libkin. 2002. A Normal Form for XML Documents. In *PODS*. 85–96.

[6] William Ward Armstrong. 1974. Dependency Structures of Data Base Relationships. In *IFIP Congress*. 580–583.

[7] Marianne Baudinet, Jan Chomicki, and Pierre Wolper. 1999. Constraint-Generating Dependencies. *JCSS* 59, 1 (1999), 94–115.

[8] Catriel Beeri and Moshe Y. Vardi. 1981. The Implication Problem for Data Dependencies. In *Automata, Languages and Programming*. 73–85.

[9] Catriel Beeri and Moshe Y. Vardi. 1981. *On the Complexity of Testing Implications of Data Dependencies.* Technical Report. The Hebrew University of Jeruslem.

[10] Angela Bonifati, Ioana Ileana, and Michele Linardi. 2016. Functional Dependencies Unleashed for Scalable Data Exchange. In *SSDBM*. 2:1–2:12.

[11] Angela Bonifati, Wim Martens, and Thomas Timm. 2017. An Analytical Study of Large SPARQL Query Logs. *PVLDB* 11, 2 (2017), 149–161.

[12] Peter Buneman, Susan B. Davidson, Wenfei Fan, Carmem S. Hara, and Wang Chiew Tan. 2001. Keys for XML. In *WWW*. 201–210.

[13] Peter Buneman, Susan B. Davidson, Wenfei Fan, Carmem S. Hara, and Wang Chiew Tan. 2003. Reasoning about Keys for XML. *Inf. Syst.* 28, 8 (2003), 1037–1063.

[14] Marco Calautti, Sergio Greco, Cristian Molinaro, and Irina Trubitsyna. 2016. Exploiting Equality Generating Dependencies in Checking Chase Termination. *PVLDB* 9, 5 (2016), 396–407.

[15] Andrea Calì and Andreas Pieris. 2011. On Equality-Generating Dependencies in Ontology Querying - Preliminary Report. In *AMW*.

[16] Diego Calvanese, Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger, and Mantas Simkus. 2014. Capturing Relational Schemas and Functional Dependencies in RDFS. In *AAAI*.

[17] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the Detection of Fake Accounts in Large Scale Social Online Services. In *NSDI*. 197–210.

[18] E. F. Codd. 1972. Relational Completeness of Data Base Sublanguages. *In: R. Rustin (ed.): Database Systems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California* (1972).

[19] Alvaro Cortés-Calabuig and Jan Paredaens. 2012. Semantics of Constraints in RDFS. In *AMW*. 75–90.

[20] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. 2008. The Chase Revisited. In *PODS*. 149–158.

[21] Ronald Fagin, Phokion G Kolaitis, Renée J Miller, and Lucian Popa. 2005. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science* (2005), 89–124.

[22] Ronald Fagin and Moshe Y. Vardi. 1984. The Theory of Data Dependencies - An Overview. In *ICALP*. 1–22.

[23] Wenfei Fan, Zhe Fan, Chao Tian, and Xin Luna Dong. 2015. Keys for graphs. *PVLDB* 8, 12 (2015), 1590–1601.

[24] Wenfei Fan and Floris Geerts. 2012. *Foundations of Databases.* Morgan & Claypool Publishers.

[25] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional Functional Dependencies for Capturing Data Inconsistencies. *TODS* 33, 1 (2008).

[26] Wenfei Fan, Chunming Hu, Xueli Liu, and Ping Lu. 2018. Discovering Graph Functional Dependencies. In *SIGMOD*. 427–439.

[27] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. 2011. Interaction between Record Matching and Data Repairing. In *SIGMOD*. 469–480.

[28] Wenfei Fan and Leonid Libkin. 2002. On XML Integrity Constraints in the Presence of DTDs. *J. ACM* 49, 3 (2002), 368–406.

[29] Wenfei Fan and Ping Lu. 2017. Dependencies for Graphs. In *PODS*. 403–416.

[30] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional Dependencies for Graphs. In *SIGMOD*.

[31] Mario Arias Gallego, Javier D Fernández, Miguel A Martínez-Prieto, and Pablo de la Fuente. 2011. An Empirical Study of Real-World SPARQL Queries. In *USEWOD workshop*.

[32] Michael Garey and David Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

[33] Michael R Garey, David S. Johnson, and Larry Stockmeyer. 1976. Some Simplified NP-Complete Graph Problems. *Theoretical computer science* 1, 3 (1976), 237–267.

[34] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC Data-Cleaning Framework. *PVLDB* 6, 9 (2013), 625–636.

[35] Marc H Graham, Alberto O Mendelzon, and Moshe Y Vardi. 1986. Notions of Dependency Satisfaction. *Journal of the ACM (JACM)* 33, 1 (1986), 105–129.

[36] Ivana Grujic, Sanja Bogdanovic-Dinic, and Leonid Stoimenov. 2014. Collecting and Analyzing Data from E-Government Facebook Pages. In *ICT Innovations*.

[37] Wolfgang Gutjahr, Emo Welzl, and Gerhard Woeginger. 1992. Polynomial Graph-Colorings. *Discrete Applied Mathematics* 35, 1 (1992), 29–45.

[38] Binbin He, Lei Zou, and Dongyan Zhao. 2014. Using Conditional Functional Dependency to Discover Abnormal Data in RDF Graphs. In *SWIM*. 1–7.

[39] Jelle Hellings, Marc Gyssens, Jan Paredaens, and Yuqing Wu. 2014. Implication and Axiomatization of Functional Constraints on Patterns with an Application to the RDF Data Model. In *FoIKS*.

[40] Jelle Hellings, Marc Gyssens, Jan Paredaens, and Yuqing Wu. 2016. Implication and Axiomatization of Functional and Constant Constraints. *Ann. Math. Artif. Intell.* 76, 3-4 (2016), 251–279.

[41] Holger Knublauch and Dimitris Kontokostas. 2017. Shapes Constraint Language (SHACL). W3C Working Draft. (Feb. 2017). *https://www.w3.org/TR/shacl/#dfn-shacl-instance*.

[42] Georg Lausen, Michael Meier, and Michael Schmidt. 2008. SPARQLing Constraints for RDF. In *EDBT*. 499–509.

[43] Bruno Marnette. 2009. Generalized Schema-Mappings: From Termination to Tractability. In *PODS*. 13–22.

[44] Bruno Marnette and Floris Geerts. 2010. Static Analysis of Schema-Mappings Ensuring Oblivious Termination. In *ICDT*. 183–195.

[45] Bruno Marnette, Giansalvatore Mecca, and Paolo Papotti. 2010. Scalable Data Exchange with Functional Dependencies. *PVLDB* 3, 1 (2010), 105–116.

[46] Neo4j Team. 2017. The Neo4j Developer Manual v3.1 (Chapter 3.5.2: Constraints). (2017). *http://neo4j.com/docs/developer-manual/current/*.

[47] François Picalausa and Stijn Vansummeren. 2011. What Are Real SPARQL Queries Like?. In *Proceedings of the International Workshop on Semantic Web Information Management, SWIM 2011, Athens, Greece, June 12, 2011*. 7.

[48] Reinhard Pichler and Sebastian Skritek. 2011. The Complexity of Evaluating Tuple Generating Dependencies. In *ICDT*. 244–255.

[49] Vladislav Rutenburg. 1986. Complexity of Generalized Graph Coloring. In *MFCS*. 573–581.

[50] Fereidoon Sadri. 1980. *Data Dependencies in the Relational Model of Databases, a Generalization*. Ph.D. Dissertation. Princeton Unversity.

[51] Fereidoon Sadri and Jeffrey D. Ullman. 1980. The Interaction between Functional Dependencies and Template Dependencies. In *SIGMOD*. 45–51.

[52] Marcus Schaefer and Christopher Umans. 2002. Completeness in the Polynomial-Time Hierarchy: A Compendium. *SIGACT news* 33, 3 (2002), 32–49.

[53] Schema.org. 2018. MusicAlbum. (2018). *Canonical URL: http://schema.org/MusicAlbum*.

[54] Michael Schmidt, Michael Meier, and Georg Lausen. 2010. Foundations of SPARQL Query Optimization. In *ICDT*. 4–33.

[55] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. 2004. XML Schema Part 1: Structures Second Edition. W3C Recommendation. (Oct. 2004). http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

[56] Yang Yu and Jeff Heflin. 2011. Extending Functional Dependency to Detect Abnormal Data in RDF Graphs. In *ISWC*.

APPENDIX: PROOFS

More details of the proof of Lemma 5.6

It remains to prove the correctness of the construction. That is to show that Σ is not satisfiable if and only if ψ is satisfiable.

($\Rightarrow$) First assume that Σ is not satisfiable. Then by Theorem 5.2, chase(G_Σ, Σ) is inconsistent. Consider a terminal chasing sequence $(\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k)$ of G_Σ by Σ. Since chase(G_Σ, Σ) is inconsistent, there exist a GFD $\varphi = Q(\bar{x})(X \Rightarrow Y)$ in Σ and a match h of Q in G_Σ such that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ and Eq_{k+1} is inconsistent in G_{Eq_k}. Moreover, GFDs do not contain id literals; hence there exists a node x in G_Σ such that both $x.A = 1$ and $x.A = 2$ are deduced from Eq_{k+1} (see Section 4).

By the properties of G_Σ mentioned above, Eq_{k+1} is inconsistent only if there exist a match h of Q_2 in G_{Q_1} such that $h(\bar{x}_2) \models Y_1 \wedge \ldots \wedge Y_n$, where $\bar{x}_2$ are variables in Q_2, and a match of Q_1 in G_{Q_1}; indeed, a match of Q_2 in G_{Q_2} does not lead to conflict, and Q_1 does not have a match in G_{Q_2}. Based on h, we construct a truth assignment ν of ψ as follows: for each node $x \in \{x_1^2, \ldots, x_m^2\}$ in Q_2, $\nu(x)$ is 1 or 0 if $h(x)$ is x_1^1 or x_2^1 in Q_1, respectively. By the definition of G_{Q_1} and $h(\bar{x}) \models Y_1 \wedge \ldots \wedge Y_n$, it is easy to verify that ν is a truth assignment of ψ. Therefore, ν is a truth assignment that satisfies ψ.

($\Leftarrow$) Conversely, assume that there exists a truth assignment ν that satisfies ψ. We show that there exists a chasing sequence ρ: $(\mathsf{Eq}_0, \mathsf{Eq}_1)$ of G_Σ by Σ, and a chase step $\mathsf{Eq}_1 \Rightarrow_{(\varphi_2,h)} \mathsf{Eq}_2$ such that Eq_2 is inconsistent in G_{Eq_1}. Then by Theorems 4.2 and 5.2, Σ is not satisfiable. We give ρ as follows.

(a) The first step of ρ applies φ_1 to G_{Q_1}. It enforces all nodes in G_{Q_1} to have the required attribute values, *e.g.,* $r^1.A = 1$. Since all edges in G_{Q_1} have distinct labels, only one such application exists.

(b) The second step is $\mathsf{Eq}_1 \Rightarrow_{(\varphi_2,h)} \mathsf{Eq}_2$, which applies φ_2 to G_{Q_1}, and leads to inconsistency. Here the match h is defined as follows: (1) $h(r^2) = r^1$, *i.e.,* the root of Q_2 is mapped to the root of G_{Q_1}; (2) for each node $x_i^2 \in \{x_1^2, \ldots, x_m^2\}$ (for $i \in [1, m]$), $h(x_i^2)$ is x_1^1 or x_2^1 in G_{Q_1} when $\nu(x_i)$ is 1 or 0, respectively; and (3) for the other node y_i^2 (for $i \in [1, n]$), suppose that C_i is $l_1 \vee l_2 \vee l_3$, $h(y_i^2)$ is mapped to the node y^1 in Q_1 such that $y^1.A = \nu(l_1)$, $y^1.B = \nu(l_2)$, and $y^1.C = \nu(l_3)$. Here $\nu(l)$ is defined as follows: if $l = x_i$, then $\nu(l) = \nu(x_i)$; otherwise, $\nu(l) = \neg\nu(x_i)$. Since all the nodes in G_{Q_1} labeled l have distinct values of attributes A, B, and C, h is well defined and is unique. Moreover, Eq_2 extends Eq_1 by adding $2 \in [r^1.A]_{\mathsf{Eq}_2}$. We can verify that h is a match of Q_2 in G_Σ, and $\mathsf{Eq}_1 \Rightarrow_{(\varphi_2,h)} \mathsf{Eq}_2$ is a chase step. However, from Eq_1 and Eq_2 we deduce that both $r^1.A = 1$ and $r^1.A = 2$, a conflict. Therefore, Eq_2 is inconsistent. That is, chase(G_Σ, Σ) is inconsistent. □

More details of the proof of Lemma 5.7

It remains to prove that Σ is not satisfiable if and only if G has a proper 3-coloring.

($\Rightarrow$) First assume that Σ is not satisfiable. Then by Theorem 5.2, chase(G_Σ, Σ) is inconsistent. Meanwhile, from the construction, there exists a chasing sequence $(\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k)$ such that $G_{Q_1^1}$ and $G_{Q_2^1}$ are subgraphs of G_{Eq_k}. By Theorem 4.2, there exists a terminal chasing sequence $(\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k, \mathsf{Eq}_{k+1}, \ldots, \mathsf{Eq}_l)$ of G_Σ by Σ. Moreover, one can see that $G_{Q_1^1}$ and $G_{Q_2^1}$ are also subgraphs of G_{Eq_l}. Indeed, since Eq_l gis consistent, and the labels of nodes in $G_{Q_1^1}$ or $G_{Q_2^1}$ are distinct, we know that no node in $G_{Q_1^1}$ or $G_{Q_2^1}$ can be further merged. Since $G_{Q_1^1}$ and $G_{Q_2^1}$ are subgraphs of G_{Eq_k}, and $\mathsf{Eq}_k \subseteq \mathsf{Eq}_l$, we have that $G_{Q_1^1}$ and $G_{Q_2^1}$ are also subgraphs of G_{Eq_l}.

Since chase(G_Σ, Σ) is inconsistent, there exist $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a match h of Q in $(G_\Sigma)_{\mathsf{Eq}_l}$ such that $\mathsf{Eq}_l \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{l+1}$ and Eq_{l+1} is inconsistent in G_{Eq_l}. By the definition of Σ, the conflict can only be introduced by two nodes x and y of G_Σ such that $y \in [x]_{\mathsf{Eq}_{l+1}}$ but the labels of x and y are distinct. A closer examination reveals that Eq_{l+1} can be inconsistent only if there exists either a match h of the two patterns Q' and Q'' of Q in G_{Q_1} and G_{Q_2}, respectively, such that $h(\bar{x}) \models X$, or a match of the two patterns Q' and Q'' in Q in G_{Q_2} and G_{Q_1}, respectively, such that $h(\bar{x}) \models X$. By restricting h to nodes in Q^1_3 and Q^2_3, we can obtain a match of Q^1_3 and Q^2_3 in G_{Q_1} and G_{Q_2}, respectively, or a match of Q^1_3 and Q^2_3 in G_{Q_2} and G_{Q_1}, respectively. Indeed, any other match does not lead to conflict. Moreover, Q^1_3 is connected since G is connected, and hence it can only be mapped to either $G_{Q^1_1}$ or $G_{Q^1_2}$, not to both of them; similarly for Q^2_3. Here we consider *w.l.o.g.* the case when there exists a match h of Q^1_3 and Q^2_3 in $G_{Q^1_1}$ and $G_{Q^1_2}$, respectively.

We can deduce a 3-coloring ν of G as follows: for each node $x \in V$, if $h(x) = x^1_1$, then $\nu(x) = r$; if $h(x) = x^1_2$, then $\nu(x) = g$; and if $h(x) = x^1_3$, then $\nu(x) = b$. Here V is the set of nodes in Q^1_3, which correspond to nodes in G. Note that no node in V can be mapped to x^1_0 by h, since x^1_0 has no outgoing edge. By the definitions of $G_{Q^1_1}$, Q^1_3 and h, one can verify that ν is a 3-coloring of G.

($\Leftarrow$) Conversely, assume that ν is a 3-coloring of G. It suffices to show that there exists a chasing sequence $\rho = (\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k)$ of G_Σ by Σ, and a chase step $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ such that Eq_{k+1} is inconsistent in G_{Eq_k}. For if it holds, then by Theorems 4.2 and 5.2, Σ is not satisfiable.

More specifically, consider ρ as the sequence $(\mathsf{Eq}_0, \ldots, \mathsf{Eq}_k)$ such that $G_{Q^1_1}$ and $G_{Q^1_2}$ are subgraphs of G_{Eq_k}. The existence of such a sequence is ensured by the definition of the first and second groups of GKeys, and can be verified along the same lines as above. The last step is defined as follows: $\mathsf{Eq}_k \Rightarrow_{(\varphi_3,h)} \mathsf{Eq}_{k+1}$, which applies φ_3 to G_{Q_1} and G_{Q_2}, and deduces the inconsistency. The match h is defined as follows: $h(x_0) = h((x_0)^{(u,\iota,v)}_{u,v}) = h((x_0)^{(u,\iota,v)}_{v,u}) = x^1_0$ and $h(x'_0) = h((x'_0)^{(u,\iota,v)}_{u,v}) = h((x'_0)^{(u,\iota,v)}_{v,u}) = x^2_0$ for all edge (u, ι, v) in G; for each other node x in Q^1_3 in Q', if $\nu(x) = r$, then $h(x) = h((x)^{(x,\iota,y)}_y) = x^1_1$; if $\nu(x) = g$, then $h(x) = h((x)^{(x,\iota,y)}_y) = x^1_2$; if $\nu(x) = b$, then $h(x) = h((x)^{(x,\iota,y)}_y) = x^1_3$; similarly for each other node in Q''. One can verify that h is a match of Q in $G_{Q^1_1}$ and $G_{Q^1_2}$ such that $h(\bar{x}_2) \models X$. As mentioned above, from h we can obtain a match of Q^1_3 and its copy in $G_{Q^1_1}$ and $G_{Q^1_2}$, respectively. As a result, Eq_{k+1} deduces $x^1_0.\mathrm{id} = x^2_0.\mathrm{id}$. However, the label of x^1_0 is '0', while the label of x^2_0 is '1', *i.e.*, label conflict. We can conclude that Eq_{k+1} is inconsistent. □

More details of the proof of Theorem 5.8

It remains to show the following two lemmas.

Lemma 5.10. For graph G and match $h(\bar{x})$ of Q in G, if $G \models \Sigma$ and $h(\bar{x}) \models X$, then $h(\bar{x}) \models \mathsf{Eq}_k$.

Lemma 5.11. When $\mathrm{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ is consistent, Y can be deduced from $\mathrm{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ if and only if for any graph G and any match h of Q in G, if $h(\bar{x}) \models \mathsf{Eq}_k$, then $h(\bar{x}) \models Y$.

We next prove Lemmas 5.10 and 5.11.

Proof of Lemma 5.10. Suppose that $G \models \Sigma$ and $h(\bar{x}) \models X$. Then Eq_X is consistent. We show that for all i ($i \in [0, k]$), $h(\bar{x}) \models \mathsf{Eq}_i$ by induction on i. The proof is similar to the one for Theorem 5.2 about the property of terminal chasing sequences. From this it follows that $h(\bar{x}) \models \mathsf{Eq}_k$. □

Proof of Lemma 5.11. We verify the sufficient and necessary condition as follows.

($\Rightarrow$) Suppose that Y can be deduced from $\mathrm{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$. For any graph G and any match h of Q in G such that $h(\bar{x}) \models \mathsf{Eq}_k$, we show that $h(\bar{x}) \models Y$. More specifically, for any literal $l \in Y$, we show that $h(\bar{x}) \models l$ as follows. When l is $x.A = c$, since l can be deduced from Eq_k, we have that $c \in [x.A]_{\mathsf{Eq}_k}$, *i.e.*, $x.A = c$ can be deduced from the transitivity of equality literals, and the semantics of id literals in Eq_k. Denote $x_0.A_0$ as $x.A$. Then there exist equality literals

$x_0.A_0 = x_1.A_1, x_1.A_1 = x_2.A_2, \ldots, x_n.A_n = c$ such that for each $x_i.A_i = x_{i+1}.A_{i+1}$ ($i \in [0, n-1]$), either $x_i.A_i = x_{i+1}.A_{i+1}$ exists in Eq_k, or $x_i.\mathsf{id} = x_{i+1}.\mathsf{id}$ is in Eq_k and $A_i = A_{i+1}$. Moreover, $x_n.A_n = c$ is also in Eq_k. From $h(\bar{x}) \models \mathsf{Eq}_k$ and the semantics of id literals, we know that $h(\bar{x}) \models (x_0.A_0 = x_1.A_1), h(\bar{x}) \models (x_1.A_1 = x_2.A_2), \ldots$, and $h(\bar{x}) \models (x_n.A_n = c)$. By the transitivity of equality literals, we know that $h(\bar{x}) \models (x_0.A_0 = c)$, *i.e.,* $h(\bar{x}) \models (x.A = c)$. Hence $h(\bar{x}) \models l$. The proofs are similar when l is $x.A = y.B$ or $x.\mathsf{id} = y.\mathsf{id}$. Then we can conclude that $h(\bar{x}) \models Y$.

($\Leftarrow$) Conversely, suppose that for any graph G and any match h of Q in G, if $h(\bar{x}) \models \mathsf{Eq}_k$, then $h(\bar{x}) \models Y$. We prove that Y can be deduced from $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ by contradiction. Assume that there exists a literal l in Y such that l cannot be deduced from $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$. Then we construct a graph G and a match h of Q in G such that $h(\bar{x}) \models \mathsf{Eq}_k$, but $h(\bar{x}) \not\models l$, which contradicts the assumption that for any graph G and any match h of Q in G, if $h(\bar{x}) \models \mathsf{Eq}_k$, then $h(\bar{x}) \models Y$.

Graph $G = (V, E, L, F_A)$ is constructed in the same way as its counterpart given in the proof of Theorem 5.2, except here that we use the equivalence relation Eq_k (see the proof of Theorem 5.2 for details). Intuitively, the set of nodes in G contains v_{Eq_k} for every node v in Q (denoting $[v]_{\mathsf{Eq}_k}$; see Section 4); and the attributes are enforced by Eq_k. Since $\mathsf{chase}(G_Q, \mathsf{Eq}_X, \Sigma)$ is consistent, G is well defined. We define the mapping h from Q to G such that for each node $x \in V_Q$, $h(x) = x_{\mathsf{Eq}_k}$ (*i.e.,* $[x]_{\mathsf{Eq}_k}$). By the definition of G, it is easy to verify that h is a match of Q in G.

We show that (a) $h(\bar{x}) \models \mathsf{Eq}_k$, but (b) $h(\bar{x}) \not\models l$. To prove (a), for any literal l_1 in Eq_k, we show that $h(\bar{x}) \models l_1$. When l_1 is $x.A = c$, by the construction of G, there exists an attribute A of node x_{Eq_k} such that $x_{\mathsf{Eq}_k}.A = c$ in G. Since $h(x) = x_{\mathsf{Eq}_k}$, we have that $h(\bar{x}) \models (x.A = c)$. The proofs are similar for the cases when l_1 is $x.A = y.B$ or $x.\mathsf{id} = y.\mathsf{id}$. Hence, $h(\bar{x}) \models \mathsf{Eq}_k$.

To prove (b), consider first the case when l is $x.A = c$, and l cannot be deduced from Eq_k by the transitivity of equality literals and the semantics of id literals in Eq_k. By the construction of G, we have that either node x_{Eq_k} does not have attribute A or $x_{\mathsf{Eq}_k}.A \neq c$ in G. Hence $h(\bar{x}) \not\models Y$. The proofs are similar for the cases when l is $x.A = y.B$ or $x.\mathsf{id} = y.\mathsf{id}$. □

Proof of Lemma 5.14

We show the implication problem is NP-hard by reduction from the 3SAT problem (see the proof of Lemma 5.6 for 3SAT). Given a 3SAT formula ψ, we construct a set Σ of $\mathsf{GFD_x}$s and a $\mathsf{GFD_x}$ φ such that all patterns in Σ are trees. We show that $\Sigma \models \varphi$ if and only if ψ is satisfiable.

The construction is almost the same as the one given in Lemma 5.6, except that we need to remove all constants from the GFDs. To this end, we introduce two additional nodes to represent the constants 0 and 1. The set Σ consists of only one $\mathsf{GFD_x}$ φ_1, which is used to encode the structure of ψ. More specifically, $\varphi_1 = Q_1[\bar{x}_1]((Y_1 \wedge \ldots \wedge Y_n) \Rightarrow (r^1.A = r^1.B))$, where

(*a*) $Q_1 = (V_1, E_1, L_1)$ as depicted in Fig. 11, where $V_1 = \{r^1, x_1^1, \ldots, x_m^1, y_1^1, y_2^1, \ldots, y_n^1\}$, and $E_1 = \{(r^1, \text{'\_'}, x_1^1), \ldots, (r^1, \text{'\_'}, x_m^1), (r^1, \text{'\_'}, y_1^1), \ldots, (r^1, \text{'\_'}, y_n^1)\}$; these form a tree with root r^1; we let $L_1(r^1) = \text{'\_'}$, $L_1(x_1^1) = \ldots = L_1(x_m^1) = v$, $L_1(y_1^1) = L_1(y_2^1) = \ldots = L_1(y_n^1) = l$; and

(b) the literals in $Y_1, \ldots, Y_n$ are to check whether a given truth assignment μ_x satisfies ψ. The construction is the same as the one given in the proof of Lemma 5.6. For example, let $C_i = x_1 \vee \neg x_2 \vee x_3$. Then the literals for the node y_i^1 are defined as follows: $Y_i = (y_i^1.A = x_1.A) \wedge (y_i^1.B = x_2.B) \wedge (y_i^1.C = x_3.A)$. The other conjuncts can be constructed similarly.

We next define the $\mathsf{GFD_x}$ φ, which is used to encode the Boolean domain of each variable and all satisfying assignments of C_i ($i \in [1, n]$). The construction is almost the same as the one given in the proof of Lemma 5.6, except that we replace the two constants 1 and 0 by two attributes $c_1.A$ and $c_0.A$, respectively. More specifically, $\varphi = Q[\bar{x}](Y_1 \wedge Y_2 \wedge Y_1^1 \wedge \ldots \wedge Y_1^7 \Rightarrow (r.A = r.B))$, where

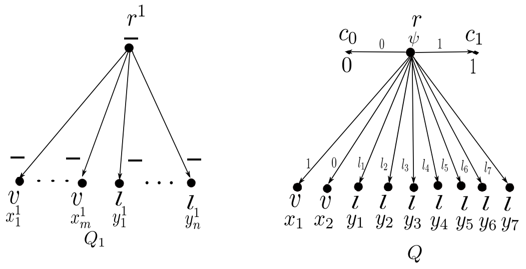

Fig. 11. The tree patterns in the proof of Lemma 5.14

(1) pattern $Q = (V, E, L)$ as shown in Fig. 11, where $V = \{r, c_0, c_1, x_1, x_2, y_1, y_2, \ldots, y_7\}$ and $E = \{(r, 0, c_0), (r, 1, c_1), (r, 1, x_1), (r, 0, x_2), (r, l_1, y_1), \ldots, (r, l_7, y_7)\}$; these make a tree with r as the root; moreover, $L(r) = \psi$, $L(c_0) = 0$, $L(c_1) = 1$, $L(x_1) = L(x_2) = v$, $L(y_1) = L(y_2) = \ldots = L(y_7) = l$; and

(2) the literals in $Y_1 \wedge Y_2 \wedge Y_1^1 \wedge \ldots \wedge Y_1^7$ are almost the same as their counterparts in the proof of Lemma 5.6, except that we do not use constant literals. To cope with Boolean values, we use two attributes $c_0.A$ and $c_1.A$ to represent 0 and 1, respectively.
More specifically, the literals in $Y_1 \wedge Y_2 \wedge Y_1^1 \wedge \ldots \wedge Y_1^7$ are partitioned into two parts: (a) the first part consists of four literals, which are used to enforce the domain of variables. That is, $Y_1 = ((x_1.A = c_1.A) \wedge (x_1.B = c_0.A))$ and $Y_2 = ((x_2.A = c_0.A) \wedge (x_2.B = c_1.A))$. These literals ensure that when a node corresponding to a variable matches x_1, the variable is set 1; otherwise, if it matches x_2, then the variable is set 0. Meanwhile, we also use $x_1.B$ or $x_2.B$ to represent the negation of the variable. (b) The second part consists of 7 groups of literals, each of which encodes a satisfying truth assignment of C_i ($i \in [1, n]$). More specifically, $Y_1^1 = ((y_1.A = c_1.A) \wedge (y_1.B = c_1.A) \wedge (y_1.C = c_1.A))$, …, $Y_1^7 = ((y_7.A = c_0.A) \wedge (y_7.B = c_0.A) \wedge (y_7.C = c_1.A))$. Intuitively, for a conjunct $C_i = l_1 \vee l_2 \vee l_3$, we use the attributes A, B and C to represent l_1, l_2 and l_3, respectively.

One can verify that a chase step can only be taken when there exists a match of Q_1 in G_Q, along the same lines as the proof of Lemma 5.6.

We next show that $\Sigma \models \varphi$ if and only if the 3SAT formula ψ is satisfiable.

($\Rightarrow$) First assume that $\Sigma \models \varphi$. Since both φ and GEDs of Σ are GFD$_x$s, it can be readily verified that chase(G_Q, Eq$_X$, Σ) is always consistent, along the same lines as the proof for the satisfiability of GFD$_x$s (see the proof of Theorem 5.4). Then by Theorem 5.8, we can deduce $r.A = r.B$ from chase(G_Q, Eq$_X$, Σ). Therefore, there exists a match h of Q_1 in G_Q such that $h(\bar{x}_1) \models Y_1 \wedge \ldots \wedge Y_n$. Based on h, we can construct a satisfying truth assignment ν of ψ as follows: for each node $x \in \{x_1^1, \ldots, x_m^1\}$ in Q, $\nu(x)$ is 1 if $h(x)$ is x_1 in Q_1, and $\nu(x)$ is 0 if $h(x)$ is x_2. Similar to the proof for Lemma 5.6, we can verify that ν is indeed a truth assignment that satisfies ψ.

($\Leftarrow$) Conversely, assume that ν is a truth assignment that satisfies ψ. We show that there exists a valid chase step Eq$_0 \Rightarrow_{(\varphi_1, h)}$ Eq$_1$ such that $r.A = r.B$ in G_{Eq_1}. As mentioned above, chase(G_Q, Eq$_X$, Σ) is always consistent. Hence by Theorem 5.8, $\Sigma \models \varphi$.

It remains to define the chase step Eq$_0 \Rightarrow_{(\varphi_1, h)}$ Eq$_1$. Since Σ consists of only one GFD$_x$, we only need to define the match h, which is almost the same as as its counterpart given in the proof of Lemma 5.6. More specifically, we define h as follows: (1) $h(r^1) = r$, *i.e.*, the root of Q_1 is mapped to the root of G_Q; (2) for each node $x \in \{x_1^1, \ldots, x_m^1\}$, $h(x)$ is x_1 or x_2 in G_Q when $\nu(x)$ is 1 or 0, respectively; and (3) for each other node y_i^1 ($i \in [1, n]$), suppose that C_i is $l_1 \vee l_2 \vee l_3$, $h(y_i^1)$ is mapped to the node y such that $y.A = c_{\nu(l_1)}.A$, $y.B = c_{\nu(l_2)}.A$, and $y.C = c_{\nu(l_3)}.A$. Here $\nu(l)$ is defined as follows: if $l = x_i$, then $\nu(l) = \nu(x_i)$; otherwise if $l = \neg x_i$, then $\nu(l) = \neg\nu(x_i)$. Since all the nodes in G_Q have distinct values of attributes A, B, and C with respect to $c_1.A$ and $c_0.A$, h is well defined and is unique. Moreover, Eq_1 extends Eq_0 by adding $r.B \in [r.A]_{\text{Eq}_1}$. Based on these, we can verify that h is a match of Q_1 in G_Q, and $\text{Eq}_0 \Rightarrow_{(\varphi_2,h)} \text{Eq}_1$ is a valid chase step.

Therefore, $r.A = r.B$ can be deduced from chase$(G_Q, \text{Eq}_X, \Sigma)$. By Theorem 5.8, $\Sigma \models \varphi$. □

Proof of Lemma 5.15

This is also verified by reduction from the 3-colorability problem (see the proof of Lemma 5.7 for the problem statement). Given a connected undirected graph $G = (V, E)$, we construct a set Σ of GKeys and another GKey φ. We define $\Sigma = \{\varphi_3\} \cup \{\varphi_1^{0,0}, \varphi_1^{0,1}, \varphi_0^1, \varphi_0^2, \varphi_0^3\}$ and $\varphi = \varphi_1$, where φ_1, φ_3, $\varphi_1^{0,0}, \varphi_1^{0,1}, \varphi_0^1, \varphi_0^2$, and φ_0^3 are GKeys given in the proof of Lemma 5.7:

$$\varphi_1 = Q_1[\bar{x}](\emptyset \Rightarrow x_0^1.\text{id} = (x_0^1)'.\text{id}),$$
$$\varphi_3 = Q[\bar{x}](X \Rightarrow x_0.\text{id} = x_0'.\text{id}),$$
$$\varphi_1^{0,0} = Q_0[x_0, z_0, y_0, x_0', z_0', y_0'](y_0.\text{id} = z_0'.\text{id} \Rightarrow x_0.\text{id} = x_0'.\text{id}),$$
$$\varphi_1^{0,1} = Q_0'[x_0, z_0, y_0, x_0', z_0', y_0'](y_0.\text{id} = z_0'.\text{id} \Rightarrow x_0.\text{id} = x_0'.\text{id}),$$
$$\varphi_0^i = Q_0^i[x_0, y_0, x_0', y_0'](x_0.\text{id} = x_0'.\text{id} \Rightarrow y_0.\text{id} = y_0'.\text{id}) \text{ for each } i \in [1,3].$$

Intuitively, φ_3 encodes the structure of G, while the other GKeys encode a 3-coloring. These GKeys bear tree patterns, and do not contain constant literals. Then along the same lines as the proof of Lemma 5.7, one can verify that $\Sigma \models \varphi$ if and only if G has a proper 3-coloring.

Note that we do not need the second group of GKeys for encoding Q_2^1 in the proof of Lemma 5.7, which is used to deduce inconsistency in the satisfiability analysis. □

Proof of Lemma 5.17

This is verified by reduction from the complement of the 3SAT problem (see the proof of Lemma 5.6 for 3SAT). Given a 3SAT formula ψ, we construct a set Σ of GFD_xs with tree patterns and a graph G_1, which is also a tree. Here we adopt the same set Σ of GFD_xs as defined in the proof of Lemma 5.14, which consists of a single GFD_x $\varphi_1 = Q_1[\bar{x}](X_1 \Rightarrow r^1.A = r^1.B)$. We define graph G_1 as the canonical graph $(G_Q)_{\text{Eq}_X}$ of GFD_x $\varphi = Q[\bar{x}](X \Rightarrow r.A = r.B)$, which is a tree. Intuitively, φ_1 encodes the structure of ψ, and G_1 encodes all possible satisfying assignments. Then along the same lines as the proof of Lemma 5.14, one can verify that $G_1 \not\models \Sigma$ if and only if ψ is satisfiable. □

Proof of Lemma 5.18

We prove this by reduction from the complement of the H-coloring problem, which is NP-complete [37]. For a fixed graph H, the H-coloring problem is to decide, given a graph G, whether there exists a homomorphism from G to H. It remains NP-complete even when H is a tree.

We adopt the tree H given in Fig. 6 of [37], as shown in Fig. 12. Here $T_1, \ldots, T_7$ in H are abbreviations of trees. As mentioned in [37], the tree H has 287 nodes. Each tree T_i ($i \in [1,7]$) is connected to the other edges in H by connecting the nodes labeled x and y.

Given graph G, we construct a graph G_1 and a set Σ of GKeys such that both G_1 and the patterns in Σ are all trees. We show that $G_1 \not\models \Sigma$ if and only if there exists a homomorphism from G to H.

Graph G_1 is constructed from H as follows, as shown in Fig 13: (1) we label all edges in H with 0; (2) we add two new edges (with two new nodes) to each node in H, and label these edges with 1; and (3) we label all the nodes with v. Compared to H, we add two nodes to each vertex in H. To simplify the presentation, for each node v in H, we denote by $v[0]$ and $v[1]$ the two newly added nodes for v. In Fig. 13, we only show how to process T_1; the trees $T_2, \ldots, T_7$ can be handled

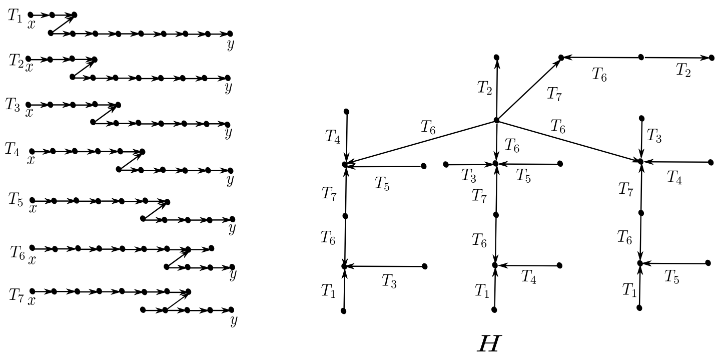
Fig. 12. The patterns used in Lemma 5.18

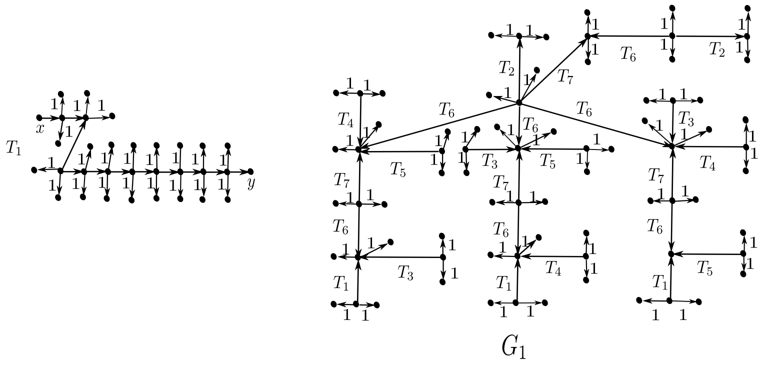
Fig. 13. The graph G_1

similarly. We list all the edges labeled 1, and all the other edges are labeled 0. Obviously, G_1 is also a tree. We will see that the new nodes for each node in H are used to deduce whether $G_1 \models \Sigma$.

We next define Σ to encode G. We first modify the graph G as we did for H, by labeling each edge, adding new nodes for each node in G, and labeling the nodes. Denote the resulting graph as G'. Similar to G_1, for each node v in G, denote by $v[0]$ and $v[1]$ the two newly added nodes for v in G' (those not in G). Then Σ consists of only one GKey $\varphi = Q[\bar{x}][X \Rightarrow x.\text{id} = x'.\text{id}]$, where Q and X are constructed from G' by using the same method given in the proof of Lemma 5.7, x is a newly added node for a node n in G', and x' is the copy of x, *i.e.,* x and x' are the designated nodes in Q (here Q consists of two copies Q'_1 and Q'_2 of a pattern Q', x is a designated node in Q'_1, and x' is in Q'_2 corresponding to x; see the definition of GKeys in Section 3). Such nodes x and x' exist by the construction of G'. More specifically, φ is defined from G' as follows: (1) compute one spanning tree of G'; (2) copy its adjacent edges for each node in G' by adding new nodes; and (3) use literals in X to merge all the newly added nodes in step (2) with the original ones.

Moreover, when there exists a homomorphism h_1 from G to H, we can deduce a violation as follows. Suppose that $v = h_1(n)$ for some node n in G. Then we have that v is a node in H. Recall that $v[0]$ and $v[1]$ are the two newly added nodes for v. Then we can define a match h such that $h(\bar{x}) \models X$, $h(x) = v[0]$ and $h(x') = v[1]$. By φ, we know that $h(x).\text{id} = h(x').\text{id}$. However, $v[0]$ and $v[1]$ are two distinct nodes in G_1; hence a contradiction.

We are ready to show that $G_1 \not\models \Sigma$ if and only if there exists a homomorphism from G to H.

($\Rightarrow$) Suppose that $G_1 \not\models \Sigma$. Then there exists a match h of Q in G_1 such that $h(\bar{x}) \models X$. Since $h(\bar{x}) \models X$, we can get a homomorphism from G to H, by restricting h to the nodes of G in Q. Note that the nodes in G cannot be mapped to the new nodes in G_1 (those not in H), since these new nodes have edges with label 1, while the edges in G are labeled 0.

($\Leftarrow$) Suppose that h_1 is a homomorphism from G to H. Then we can construct a match h of Q in G_1 such that $h(\bar{x}) \models X$, but $h(x)$ and $h(x')$ are distinct nodes. That is, $h(\bar{x}) \not\models \varphi$, and thus $G_1 \not\models \Sigma$.

The match h is constructed as follows. Suppose that Q consists of Q^1 and its copy Q^2. For each node n_v in Q^1, if n_v is a node in G, then $h(n_v) = h_1(n_v)$; if n_v is a copy of node n'_v in G, then $h(n_v) = h_1(n'_v)$; if n_v is $n'_v[0]$ or $n'_v[1]$ (the two newly added nodes for n'_v) for node n'_v in G, then $h(n_v) = h_1(n'_v)[0]$. For each node n_v in Q^2, h is defined similarly, except that when n_v is $n'_v[0]$ or $n'_v[1]$, we let $h(n_v) = h_1(n'_v)[1]$. Note that for each n_v of a copy of node n'_v in G, we set $h(n_v) = h_1(n'_v)$. Hence $h(\bar{x}) \models X$. Since h_1 is a homomorphism from G to H, it is easy to verify that h is a match of Q in G_1 such that $h(\bar{x}) \models X$. However, since x is a newly added node, $h(x) = n_v[0]$ for some node n_v in H, and $h(x') = n_v[1]$. Obviously, $n_v[0]$ and $n_v[1]$ are distinct nodes, and we have that $h(x).\mathrm{id} \neq h(x').\mathrm{id}$. From $h(\bar{x}) \models X$, we have that $h(\bar{x}) \not\models \varphi$. That is, $G_1 \not\models \Sigma$. □

More details of the proof of Theorem 6.2

For the correctness of Theorem 6.2, it remains to show the following two claims.

Claim 1: For each Eq_i $(1 \leq i \leq k)$, $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_i)$.

Claim 2: If there exist a GED φ *in* Σ *and a match* h *such that* $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ *and* Eq_{k+1} *is inconsistent in* G_{Eq_k}, *then* $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$.

We next prove these two claims.

Proof of Claim (1). We show that for each Eq_i $(1 \leq i \leq k)$, $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_i)$. This can be verified by induction on the length of ρ. For the base case $\mathsf{Eq}_1 = \mathsf{Eq}_X$, we show that $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_X)$ by using GED$_1$, GED$_2$, GED$_3$, GED$_4$, GED$_5$ and GED$_6$. When Eq_X is inconsistent, we know that ρ consists of only Eq_X, and no chase step can be carried out since $(G_Q)_{\mathsf{Eq}_X}$ is undefined. Then $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_X)$ can be verified as follows.

(1) $Q[\bar{x}](X \Rightarrow X \wedge X_{\mathrm{id}})$	GED$_1$
(2) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_X)$	(1), and GED$_5$

When Eq_X is consistent, then Eq_X is computed from X as follows, by reflexive, symmetric and transitive relations, as well as the semantics of id literals.

(1) $Q[\bar{x}](X \Rightarrow X \wedge X_{\mathrm{id}})$	GED$_1$
(2) $Q[\bar{x}](X \Rightarrow (x_1.A_1 = y_1.A_1))$	(1), $x_1.A_1 \in X$, $x_1.\mathrm{id} = y_1.\mathrm{id}$, and GED$_2$
$\ldots$	
(n+1) $Q[\bar{x}](X \Rightarrow (x_n.A_n = y_n.A_n))$	(1), $x_n.A_n \in X$, $x_n.\mathrm{id} = y_n.\mathrm{id}$, and GED$_2$
(n+2) $Q[\bar{x}](X \Rightarrow (v_1 = u_1))$	(1), $(u_1 = v_1) \in X$, and GED$_3$
$\ldots$	
(n+m+1) $Q[\bar{x}](X \Rightarrow (v_m = u_m))$	(1), $(u_m = v_m) \in X$, and GED$_3$
(n+m+2) $Q[\bar{x}](X \Rightarrow (v_1^1 = v_3^1))$	(1), $(v_1^1 = v_2^1), (v_2^1 = v_3^1) \in X$, and GED$_4$
$\ldots$	
(n+m+k+1) $Q[\bar{x}](X \Rightarrow (v_1^k = v_3^k))$	(1), $(v_1^k = v_2^k), (v_2^k = v_3^k) \in X$, and GED$_4$
(n+m+k+2) $Q[\bar{x}](X \Rightarrow X \wedge X_{\mathrm{id}} \wedge (x_1.A_1 = x_1.A_1))$	(1), (2) and GED$_6$
$\ldots$	
(2(n+m+k)+1) $Q[\bar{x}](X \Rightarrow X \wedge X_{\mathrm{id}} \wedge (x_1.A_1 = x_1.A_1) \wedge \ldots \wedge (x_n.A_n = x_n.A_n) \wedge (v_1 = u_1) \wedge \ldots \wedge (v_m = u_m) \wedge (v_1^1 = v_3^1) \wedge \ldots \wedge (v_1^k = v_3^k))$	(2(n+m+k)), (n+m+k+1) and GED$_6$

Here $x_1.A_1, \ldots, x_n.A_n$ are all attributes appearing in X; the literals in X are enumerated as $u_i = v_i$ for all $i \in [1, m]$; and $(v_1^i = v_2^i)$ and $(v_2^i = v_3^i)$ for $i \in [1, k]$ are to represent transitivity relationships that appear in X. In the proof, steps (2)-(n+1) are for the computation of reflexive relations and the semantics of id literals, steps (n+2)-(n+m+1) for symmetric relations, and steps (n+m+2)-(n+m+k+1) for transitivity. Steps (n+m+k+2)-(2(n+m+k)+1) combine all the intermediate results together. Steps (n+1)-(n+m+k+1) also handle the literals generated in previous steps (not shown).

Suppose that for any j ($1 \leq j \leq i-1$), $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_j)$. For the induction step, we show that $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_i)$. Since there exist a GED $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in Σ and a match h_1 of Q_1 in $(G_Q)_{\mathsf{Eq}_{i-1}}$ such that $\mathsf{Eq}_{i-1} \Rightarrow_{(\varphi_1,h_1)} \mathsf{Eq}_i$, $h_1(\bar{x}_1) \models X_1$. Since $\mathsf{Eq}_X \subseteq \mathsf{Eq}_{i-1}$, the equivalence relation of $(\mathsf{Eq}_X \cup \mathsf{Eq}_{i-1})$ remains unchanged in Eq_{i-1}. Hence h_1 is also a match of Q_1 in $(G_Q)_{\mathsf{Eq}_X \cup \mathsf{Eq}_{i-1}}$. Since $\varphi_1 \in \Sigma$, by the definition of proofs, $\Sigma \vdash \varphi_1$. Then by GED$_6$, $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{i-1} \wedge h(Y_1))$. Note that Eq_i is the equivalence relation extending $\mathsf{Eq}_{i-1} \cup \{l\}$, where l is a literal in $h(Y_1)$ by the definition of the chase step. Then using GED$_7$, we can prove that $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{i-1} \wedge l)$, where GED$_7$ is the property proved in Lemma 6.1. That is, $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_i)$. □

Proof of Claim (2). We show that if there exist φ and h such that $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ and Eq_{k+1} is inconsistent in G_{Eq_k}, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$. Recall that ρ is a terminal chasing sequence of G_Q by Σ, and ρ is $(\mathsf{Eq}_1 = \mathsf{Eq}_X, \mathsf{Eq}_2, \ldots, \mathsf{Eq}_k)$. Assume *w.l.o.g.* that Eq_k is consistent, since $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$ is a chase step (although invalid; see Section 4). Then by the definition of terminal chasing sequences, Eq_i ($i \in [1,k]$) is also consistent. Suppose that $h(l)$ is the literal added in the chase step $\mathsf{Eq}_k \Rightarrow_{(\varphi,h)} \mathsf{Eq}_{k+1}$, *i.e.*, Eq_{k+1} is the equivalence relation of $\mathsf{Eq}_k \cup \{h(l)\}$, and $\varphi = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ is the GED used in the step. Then $\Sigma \vdash Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$ is verified as follows:

(1) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_k)$	claim (1)
(2) $Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$	φ
(3) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_k \wedge h(Y_1))$	(1), (2), and GED$_6$
(4) $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$	GED$_5$

Observe the following. (a) Since Eq_k is consistent and $\mathsf{Eq}_X \subseteq \mathsf{Eq}_k$, $\mathsf{Eq}_X \cup \mathsf{Eq}_k$ (for step (1) above) is also consistent, and hence we can apply GED$_6$ in step (3). (b) Since Eq_{k+1} is inconsistent, Eq_{k+1} is the equivalence relation of $\mathsf{Eq}_k \cup \{h(l)\}$, and $h(l) \in h(Y_1)$, we have that $\mathsf{Eq}_k \wedge h(Y_1)$ is inconsistent, and thus we can apply GED$_5$ in step (4). These give us $Q[\bar{x}](X \Rightarrow \mathsf{Eq}_{k+1})$.

Putting these together, we can conclude that if $\Sigma \models Q[\bar{x}](X \Rightarrow Y)$, then $\Sigma \vdash Q[\bar{x}](X \Rightarrow Y)$. □

Proof of Theorem 7.2

Proof: We show that the satisfiability, implication and validation problems for GDCs are Σ_2^p-complete, Π_2^p-complete and coNP-complete, respectively.

(1) The satisfiability problem for GDCs. The proof is a little involved. We first show that the satisfiability problem for GDCs has a small model property. Based on this property, we then give an Σ_2^p algorithm to check whether a set Σ of GDCs is satisfiable. Finally, we show that the satisfiability problem is Σ_2^p-hard.

The small model property. We show that if a set Σ of GDCs is satisfiable, then Σ has a model G_h of size $O(4 \cdot |\Sigma|^3)$. That is, there exists a graph G_h such that $|G_h| \leq 4 \cdot |\Sigma|^3$, $G_h \models \Sigma$, and for any GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G_h.

Since Σ is satisfiable, there exists a graph $G = (V, E, L, F_A)$ such that $G \models \Sigma$, and for each GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G. We construct G_h based on G as follows. (a) We first deduce a subgraph G'_h of G by combining matches h_φ for every $\varphi \in \Sigma$, such that G'_h has at most $O(|\Sigma|)$ nodes; and (b) we then revise the attributes of G'_h to deduce G_h such that $|G_h| \leq 4 \cdot |\Sigma|^3$ and $G_h \models \Sigma$, by normalizing attribute values in G'_h.

(a) We deduce G'_h as follows. Recall the canonical graph $G_\Sigma = (V_\Sigma, E_\Sigma, L_\Sigma, F_A^\Sigma)$ of Σ (see Section 5.1). It is the union of patterns in Σ, in which different patterns are disjoint. We define a mapping h from G_Σ to G: for each node $x \in V_\Sigma$, if x is in V_Q for some GDC $\varphi' = Q[\bar{x}](X \Rightarrow Y)$ in Σ, then $h(x) = h_\varphi(x)$, where V_Q is the set of nodes in Q. We define G'_h as the subgraph of G "induced by" h. That is, $G'_h = (V_h, E_h, L_h, F_A^h)$, where

- $V_h = \{h(x) \mid x \in V_\Sigma\}$, where V_Σ is the set of nodes in G_Σ;
- $E_h = \{(h(y_1), h(\iota_{y_2}^{y_1}), h(y_2)) \mid (y_1, \iota, y_2) \in E_\Sigma\}$, where E_Σ is the set of edges in G_Σ;
- L_h is such defined that for each node x in V_h, $L_h(x) = L(x)$; and
- we define F_A^h by taking attributes that only appear in Σ: for each GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, and each match h_φ of Q in G'_h, if $h_\varphi(\bar{x}) \models X$ in G, then for each literal l in X or Y,
  - if l is of the form $x.A \oplus c$, then $F_A^h(v).A = F_A(v).A$, where $v = h_\varphi(x)$; and
  - if l is $x.A \oplus y.B$, then $F_A^h(v).A = F_A(v).A$ and $F_A^h(v').B = F_A(v').B$, where $v = h_\varphi(x)$ and $v' = h_\varphi(y)$.

  This is well-defined since $G \models \Sigma$. Note that we cannot directly copy the attributes in $F_A(x)$ for each node $x \in V_h$, since there is no bound on the number of attributes in $F_A(x)$.

Then $|V_h| \leq |\Sigma|$ and the number of attributes in G'_h does not exceed $|\Sigma|^2$ (see a proof below).

(b) Having defined G'_h, we revise G'_h to get G_h. That is, we "normalize" the values of attributes in G_h, so that G_h does not contain unboundedly large attribute values. The challenge is to ensure that $G_h \models \Sigma$ after the values are changed (normalized). To simplify the discussion, we refer to values on which the built-in predicates $<, >, \leq, \geq$ are defined as *numeric values*, and those on which only $=$ and $\neq$ are defined as *non-numeric values*. We normalize attribute values by distinguishing numeric values from non-numeric values as follows. (i) Suppose that all numeric values in G'_h are $a_1, \ldots, a_n$ with $a_1 < a_2 < \ldots < a_n$. We normalize those values of $a_1, \ldots, a_n$ that do not appear in Σ, such that their values are bounded. Let M and N be the maximum and minimum numbers in Σ, respectively. If there exist no numeric values in Σ, assume *w.l.o.g.* that $M = N = 0$. Let $a_1, \ldots, a_i$ be the values smaller than N, and $a_j, \ldots, a_n$ be the values larger than M. Then we replace a_k ($k \in [1, i]$) with $N - (i + 1) + k$, and replace a_l ($l \in [j, n]$) with $M + l$; the values in the range $[N, M]$ remain unchanged as they may fall into Σ. One can verify that the normalization does not change the relative ordering of the numeric values, since $\oplus$ is among $=, \neq, <, >, \leq, \geq$. (ii) Suppose that $l_1, \ldots, l_m$ are all the non-numerical values that are in G'_h but are not in Σ. We pick distinct new small values $b_1, \ldots, b_m$ that do not appear in G'_h, and replace l_i by b_i ($i \in [1, m]$). Denote the graph obtained by normalizing G'_h as $G_h = (V_h, E_h, L_h, F_A^h)$ with normalized F_A^h. One can verify that after the normalization, no attribute value in G_h is larger than $2 \cdot |\Sigma|$.

Along the same lines, we normalize labels such that no label has size larger than Σ (see below).

We next show that G_h is a model of Σ with a bounded size. First, it is easy to show that for each GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G, since h is a match of G_Σ in G.

We next prove that $G_h \models \Sigma$ by contradiction. Suppose that that $G_h \not\models \Sigma$. Then there exist a GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a match h_φ of Q in G_h such that $h_\varphi(\bar{x}) \models X$ but $h_\varphi(\bar{x}) \not\models Y$. By the definition of G_h, h_φ is also a match of Q in G. Moreover, for each node x in G_h and each attribute A of x, $F_A^h(h(x)).A$ inherits $F_A(h(x)).A$ subject to value normalization that preserves the relative ordering of the values. Therefore, we also have that $h_\varphi(\bar{x}) \models X$ and $h_\varphi(\bar{x}) \not\models Y$ in G, which contradict the assumption that $G \models \Sigma$. Hence $G_h \models \Sigma$.

We now show that $|G_h| \leq 4 \cdot |\Sigma|^3$. Observe that following: (a) $|V_h| + |E_h| \leq |\Sigma|$, since the numbers of nodes and edges in G_h are no larger than their counterparts in G_Σ; (b) L_h has $|V_h| + |E_h|$ labels, and each label has size at most Σ; (c) for each node $x \in V_h$, the number of attributes in $F_A^h(x)$ is at most $|\Sigma|$, since we add $F_A^h(x).A$ only if $x.A$ appears in some GDC φ; (d) the total number of attributes in F_A^h is at most $|\Sigma|^2$ by (a) and (c); and (e) the size $|F_A^h| \leq 2|\Sigma|^3$, since each attribute value is at most $2 \cdot |\Sigma|$. From these we know that the size $|G_h|$ of G_h is at most $2|\Sigma|^3 + |\Sigma|^2 + |\Sigma| \leq 4|\Sigma|^3$. □

Upper bound. Based on the small model property, we give an Σ_2^p algorithm to check whether a given set Σ of GDCs is satisfiable, as follows:

(1) guess a graph $G = (V, E, L, F_A)$ such that $|G| \leq 4 \cdot |\Sigma|^3$, and for each GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, guess a mapping h_Q from V_Q to V, where V_Q is the set of nodes in Q;
(2) check whether h_Q is a match of Q in G for each GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ; if so, continue; otherwise reject the current guess;
(3) check whether $G \models \Sigma$; if so, return true.

The correctness of the algorithm follows from the small model property. For its complexity, step (2) is in PTIME, and step (3) is in coNP (see the proof for the validation problem for GDCs to be given shortly). Therefore, the algorithm is in Σ_2^p, and so is the satisfiability problem for GDCs. □

Lower bound. We show that the satisfiability problem is Σ_2^p-hard for GDCs by reduction from the generalized graph coloring problem, denoted by GGCP, which is Σ_2^p-complete [49, 52]. GGCP is to decide, given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, whether there exists a two-coloring of F that does not contain a monochromatic G as a subgraph. A monochromatic subgraph of F is a subgraph in which nodes are assigned the same color. It is known that GGCP remains Σ_2^p-complete when G is a complete graph and F contains no self cycles [49].

Given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, where G is complete and F does not contain self cycles, we construct a set Σ of GDCs such that Σ is satisfiable if and only if there exists a 2-coloring of F that does not contain a monochromatic G as a subgraph.

We use two groups of GDCs to encode GGCP, one to check 2-coloring, and the other to check the relationship between F and G. We associate an attribute C with each node, denoting its color. (a) In the first group, we use a GDC to ensure that each node has a C attribute in the range of $[1, 2]$, and another GDC to ensure that the C-attribute, if it exists, must carry a value of either 1 or 2. (b) The second group checks the existence of a monochromatic subgraph G and encodes the structure of F. In addition, we address the following issues: (i) the patterns in Σ are directed graphs, while F and G are undirected; and (ii) the matches for GDCs are homomorphic mappings, while GGCP requires subgraph isomorphism. For (i), we can use two directed edges to represent one undirected edge in F and G. For (ii), since F contains no self cycles and G is complete, we have that a homomorphism h from G to F is an isomorphism from G to a subgraph of F. Indeed, if h is not one-to-one, *i.e.,* if two nodes u and v in G are mapped to the same node in F by h, then F must contain a self cycle since G is complete, a contradiction.

Based on the observations, we define the set Σ of GDCs as follows, partitioned into two groups.

(1) The first group consists of two GDCs, to encode the coloring of the vertices. More specifically,
- $\varphi_1 = Q_1[x]\big(\emptyset \Rightarrow (1 \leq x.C) \wedge (x.C \leq 2)\big)$, where $Q_1[x] = (V_1, E_1, L_1)$, $V_1 = \{x\}$, $E_1 = \emptyset$, and $L_1(x) =$'0'; it assures that every node has a C-attribute; and
- $\varphi_2 = Q_2[x]\big((x.C \neq 1) \wedge (x.C \neq 2) \Rightarrow (x.C = 1) \wedge (x.C = 2)\big)$, or equivalently, $\varphi_2 = Q_2[x]\big((x.C \neq 1) \wedge (x.C \neq 2) \Rightarrow \text{false}\big)$ as a forbidding constraint, where $Q_2[x] = (V_2, E_2, L_2)$, $V_2 = \{x\}$, $E_2 = \emptyset$, and $L_2(x) =$'0'. It says that if a node x has a C-attribute $x.C$, then $x.C$ must be either 1 or 2.

These GDCs enforce that each node is colored 1 or 2. Note that φ_2 is a forbidding constraint.

(2) The second group of Σ also has two GDCs. One is to check whether there exists a monochromatic G. It is defined as $\varphi_3 = Q_3[x_0, x_1, \ldots, x_n]\big((x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C) \Rightarrow x_0.C = 1\big)$, where n is the number of nodes in G, and $Q_3[x_0, x_1, \ldots, x_n] = (V_3, E_3, L_3)$ such that

- $V_3 = V_G \cup \{x_0\}$, *i.e.,* the nodes of V_G are encoded by $x_1, \ldots, x_n$, and x_0 encodes an additional node v_0;
- $E_3 = \{(u, 0, v), (v, 0, u) \mid (u, v) \in E_G\} \cup E_0$, where $E_0 = \{(x_1, 1, x_0), \ldots, (x_n, 1, x_0)\}$; *i.e.,* it includes all the edges in G plus an edge from each node of G to x_0; and
- for each vertex $x_i \in V_3$, $L_3(x_i)$ ='0'.

Intuitively, GDC φ_3 encodes the structure of G. It also ensures that if all the nodes in G have the same color, then the color of x_0 must be 1. Note that node x_0 is not a vertex in G. It differs from the other nodes in the following: (a) it has no outgoing edges, and (b) all the edges going to it are labeled with '1', while all the other edges are labeled with '0'.

The other GDC in the group is used to encode the structure of F. We define $\varphi_4 = Q_4[x_0^1, x_0^2, x_1, \ldots, x_m](\emptyset \Rightarrow x_0^1.C = 2)$, where m is the number of nodes in F, and $Q_4[\bar{x}] = (V_4, E_4, L_4)$ such that

- $V_4 = V_F \cup \{x_0^1, x_0^2\}$, including all the nodes in F and two additional nodes x_0^1 and x_0^2;
- $E_4 = \{(u, 0, v), (v, 0, u) \mid (u, v) \in E_F\} \cup E_0 \cup \{(x_0^1, 2, x_0^2)\}$, where $E_0 = \{(x_1, 1, x_0^1), \ldots, (x_n, 1, x_0^1)\}$; *i.e.,* it includes all the edges in F, one edge from each node of F to x_0^1, and an additional edge $(x_0^1, 2, x_0^2)$; and
- for each node $x \in V_4$, $L_4(x)$='0'.

The construction of φ_4 is the same as φ_3 except that we add an extra node x_0^2 and an extra edge $(x_0^1, 2, x_0^2)$. Observe that there exists no match of Q_4 in G_{Q_3}, where G_{Q_3} is the canonical graph of Q_3 in φ_3 (see Section 5.2). Indeed, Q_4 has an edge labeled '2', which does not appear in Q_3. We will use this property to check whether a 2-coloring of F does not contain a monochromatic G.

This completes the construction of Σ. In summary, Σ consists of four GDCs with constant literals and variable literals. None of them contains id literals. One of them is a forbidding constraint.

We next show that Σ is satisfiable if and only if there exists a 2-coloring of F that does not contain a monochromatic G as a subgraph.

($\Rightarrow$) First assume that Σ is satisfiable. Then there exists a graph $G' = (V, E, L, F_A)$ such that $G' \models \Sigma$, and for any GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G'. Based on the match h_{φ_4} of Q_4 in G', we can define a 2-coloring ν of F as follows: for each node $x \in V_F$, let $\nu(x) = h_{\varphi_4}(x).C$. Note that φ_1 and φ_2 can be applied to all nodes $h_{\varphi_4}(x)$, and hence $h_{\varphi_4}(x).C$ is defined for all nodes $h_{\varphi_4}(x)$ in G'. Hence ν is a 2-coloring by φ_1 and φ_2 in Σ. That is, ν is well defined. Moreover, by the definition of φ_4, $h_{\varphi_4}(x_0^1).C$ must be 2, since $h_{\varphi_4}(\bar{x}) \models (x_0^1.C = 2)$.

We next show that the 2-coloring ν of F does not contain a monochromatic G as a subgraph. Suppose by contradiction that G is a monochromatic subgraph in the 2-coloring ν, which is mapped to F via mapping h_I. We show that $h_{\varphi_4}(x_0^1).C$ much be 1, and hence yields a contradiction to $h_{\varphi_4}(x_0^1).C = 2$ determined by h_{φ_4} above. To prove that $h_{\varphi_4}(x_0^1).C = 1$, we apply φ_3 to G'. More specifically, we construct a mapping $h = h_{\varphi_4} \circ h'_I$ from Q_3 to G', where h'_I extends h_I by letting $h'_I(x_0) = x_0^1$, where x_0 is not in G. Here $h_{\varphi_4} \circ h'_I$ is the composition of two homomorphic mappings h_{φ_4} and h'_I, and hence is a homomorphism from Q_3 to G'. We will show below that h is a match of Q_3 in G' and $h(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$. Since $G' \models \varphi_3$, we have that $h(x_0).C = 1$. That is, $h_{\varphi_4}(x_0^1).C = h_{\varphi_4} \circ h'_I(x_0).C = h(x_0).C = 1$.

It remains to show that $h = h_{\varphi_4} \circ h'_I$ is a match of φ_3 in G' and $h_{\varphi_4} \circ h'_I(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$. By the definition of h_I, it is easy to see that h'_I actually makes a match of Q_3 in G_{Q_4}, where G_{Q_4} is the canonical graph of Q_4. Because h_{φ_4} is a match of Q_4 in G', $h_{\varphi_4} \circ h'_I$ is a match of Q_3 in G'. Moreover, since h'_I is the match of Q_3 in G_{Q_4} that makes G a monochromatic subgraph in ν, we have that $h_I(x_1).C = h_I(x_2).C = \ldots = h_I(x_n).C$, where $x_1, \ldots, x_n$ are nodes in V_G. Therefore, $h_{\varphi_4} \circ h'_I(x_1).C = h_{\varphi_4} \circ h'_I(x_2).C = \ldots = h_{\varphi_4} \circ h'_I(x_n).C$. That is, $h_{\varphi_4} \circ h'_I(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$.

Taken together, these verify that there exists a 2-coloring ν of F does not contain a monochromatic G as a subgraph.

($\Leftarrow$) Conversely, assume that ν is a 2-coloring of F that does not contain a monochromatic G as a subgraph. Based on ν, we construct a model G' of Σ, and hence show that Σ is satisfiable. We define $G' = (V, E, L, F_A)$ as follows:

- $V = V_{\varphi_1} \cup V_{\varphi_2} \cup V_{\varphi_3} \cup V_{\varphi_4}$, such that V_{φ_i} and V_{φ_j} are disjoint if $i \neq j$;
- $E = E_{\varphi_1} \cup E_{\varphi_2} \cup E_{\varphi_3} \cup E_{\varphi_4}$; note that there exist no edges between nodes in V_{φ_i} and V_{φ_j} if $i \neq j$;
- L is defined as follows: for each node x in V, $L(x) =$'0'; and
- F_A is given as follows: for each node x in V, we consider the following cases:
  - if $x \in V_{\varphi_1}$ or $x \in V_{\varphi_2}$, then $F_A(x).C = 1$;
  - for all $x \in V_{\varphi_3}$, if $x \neq v_1$ and $x \neq x_0$, then $F_A(x).C = 2$; otherwise, $F_A(x_0).C = F_A(x_1).C = 1$; and
  - for $x \in V_{\varphi_4}$, if $x \neq x_0^1$ and $x \neq x_0^2$, then $F_A(x).C = \nu(x)$; otherwise, $F_A(x_0^1).C = F_A(x_0^2).C = 2$.

We show that G' is a model of Σ. By the definitions of V and E, for any GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G'. We show that $G' \models \Sigma$. Suppose by contradiction that $G' \not\models \Sigma$. Then there exist a GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ and a match h of Q in G' such that $h(\bar{x}) \models X$ and $h(\bar{x}) \not\models Y$. Observe the following. (a) Since for each node x in G', $x.C$ is either 1 or 2, we have that $G' \models \varphi_1$ and $G' \models \varphi_2$. (b) Since there exists only one edge $(x_0^1, 2, x_0^2)$ in G', and $x_0^1.C = x_0^2.C = 2$, for any match h of φ_4 in G', we have that $h(x_0^1).C = h(x_0^2).C = 2$. That is, $G' \models \varphi_4$. (c) Then we must have that $G' \not\models \varphi_3$ since $G' \not\models \Sigma$. Denote by G_{φ_1}, G_{φ_2}, G_{φ_3} and G_{φ_4} the sub-graphs of G' induced by V_{φ_1}, V_{φ_2}, V_{φ_3} and V_{φ_4}, respectively. Observe that G_{Q_3} is connected; hence Q_3 can only be mapped to a connected subgraph in G'. That is, each match of Q_3 in G' is contained in one of G_{φ_1}, G_{φ_2}, G_{φ_3} and G_{φ_4}. It is easy to see that $G_{\varphi_1} \models \varphi_3$ and $G_{\varphi_2} \models \varphi_3$. Moreover, one can verify that $G_{\varphi_3} \models \varphi_3$ since G_{φ_3} does not contain self cycles. Hence if $G' \not\models \varphi_3$, then the only possibility is that $G_{\varphi_4} \not\models \varphi_3$. That is, there exists a match h of Q_3 in G_{φ_4} such that $h(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$, but $h(x_0).C \neq 1$. However, by the construction of Q_3, this implies that G_{φ_4} contains a monochromatic G as a subgraph. Hence the 2-coloring ν of F contains a monochromatic G as a subgraph, a contradiction to the assumption. Thus $G' \models \Sigma$. That is, Σ is satisfiable. □

(2) The implication problem for GDCs. We now study the implication problem. Similar to the satisfiability problem, we first establish a small model property, and then use this property to prove the upper bound. After these, we show that the implication problem is Π_2^p-hard for GDCs.

The small model property. We prove the following property: given a set Σ of GDCs and a GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$, if $\Sigma \not\models \varphi$, then there exists a graph G_h such that $|G_h| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$, $G_h \models \Sigma$, and $G_h \not\models \varphi$. That is, there exists a small witness for $\Sigma \not\models \varphi$.

If $\Sigma \not\models \varphi$, then there exists a graph $G = (V, E, L, F_A)$ such that $G \models \Sigma$ but $G \not\models \varphi$. By $G \not\models \varphi$, there exists a match h of Q in G such that $h(\bar{x}) \models X$, but $h(\bar{x}) \not\models Y$. We construct G_h as follows. (a) We first deduce a subgraph G'_h of G from match h, such that G'_h has at most $O(|\varphi|)$ nodes; and (b) we then normalize the attributes of G'_h to deduce G_h such that $|G_h| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$, *i.e.*, G_h has a bounded size. Moreover, we show that $G_h \models \Sigma$ and $G_h \not\models \varphi$.

The construction of G_h is similar to its counterpart given above for the satisfiability problem. Recall the canonical graph $G_Q = (V_Q, E_Q, L_Q, F_A)$ of Q of φ (Section 5.2). Observe that the match h of Q in G also makes a mapping from G_Q to G. We define G'_h as the subgraph of G "induced by" h. That is, $G'_h = (V_h, E_h, L_h, F^h_A)$, where

- $V_h = \{h(x) \mid x \in V_Q\}$, *i.e.*, it includes matches of nodes of G_Q only;
- $E_h = \{(h(y_1), h(\iota^{y_1}_{y_2}), h(y_2)) \mid (y_1, \iota, y_2) \in E_Q\}$, with the matches of edges in E_Q;
- L_h is defined as follows: for each node x in V_h, $L_h(x) = L(x)$; and
- F^h_A is defined in the same way as its counterpart for satisfiability; for each GDC $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in $\Sigma \cup \{\varphi\}$ and each match h_{φ_1} of Q_1 in G'_h (if it exists), if $h_{\varphi_1}(\bar{x}_1) \models X_1$ in G, then for each literal l in $X_1 \cup Y_1$,
  - if l is $x.A \oplus c$, then $F^h_A(v)$ is defined and it takes the value of $F_A(v).A$, where $v = h_{\varphi_1}(x)$; and
  - if l is $x.A \oplus y.B$, then $F^h_A(v).A$ and $F^h_A(v').B$ are defined and they have the corresponding values of $F_A(v).A$ and $F_A(v').B$, respectively, where $v = h_{\varphi_1}(x)$ and $v' = h_{\varphi_1}(y)$.

  This is well-defined since $F^h_A(\cdot)$ inherits values from $F_A(\cdot)$ of G.

We normalize the attribute values and labels of G'_h in the same way as for its counterpart in the satisfiability proof, such that the sizes of the attribute values in G_h are bounded by $2 \cdot (|\varphi| + |\Sigma|)$, and no label has size larger than $|\varphi| + |\Sigma|$. We define $G_h = (V_h, E_h, L_h, F^h_A)$ to be G'_h with normalized attributes and labels.

We next show the following: (a) $G_h \models \Sigma$, (b) $G_h \not\models \varphi$; and (c) $|G_h| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$.

(a) We first show that $G_h \models \Sigma$. Assume that $G_h \not\models \Sigma$ by contradiction. Then there exist a GDC $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in Σ and a match h_{φ_1} of Q_1 in G_h such that $h_{\varphi_1}(\bar{x}_1) \models X_1$ but $h_{\varphi_1}(\bar{x}_1) \not\models Y_1$. By the definition of G_h, h_{φ_1} is also a match of Q_1 in G. Moreover, for each node $x \in V_h$ and each attribute A of x, $F^h_A(h(x)).A$ inherits $F_A(h(x)).A$ with value normalization that preserves the relative ordering of the values. Thus we also have that $h_{\varphi_1}(\bar{x}_1) \models X$ and $h_{\varphi_1}(\bar{x}_1) \not\models Y$ in G. That is, $G \not\models \varphi_1$, a contradiction to the assumption that $G \models \Sigma$. Hence $G_h \models \Sigma$.

(b) Along the same lines as (a) above, we can verify that $G_h \not\models \varphi$.

(c) We show that $|G_h| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$. Observe that following: (i) $|V_h| + |E_h| \leq |\varphi|$, since G_h uses the same sets of nodes and edges of G_Q; (ii) L_h is bounded by $|V_h| + |E_h|$ and hence is at most $|\varphi|$; its total size is bounded by $|\varphi| \cdot (|\varphi| + |\Sigma|)$; (iii) for each node $x \in V_h$, $F^h_A(x)$ has at most $|\varphi| + |\Sigma|$ attributes by the definition of F^h_A; hence the total number of attributes in F^h_A is at most $|\varphi| \cdot (|\varphi| + |\Sigma|)$; and (iv) the size $|F^h_A| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma|)^2$, since each attribute value has size at most $2 \cdot (|\varphi| + |\Sigma|)$. Therefore, the size $|G_h|$ of G_h is at most $2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma|)^2 + |\varphi| + |\varphi| \cdot (|\varphi| + |\Sigma|)$ $\leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$. □

Upper bound. Based on the small model property, we develop an Σ^p_2 algorithm that, given any set Σ of GDCs and another GDC φ, checks whether $\Sigma \not\models \varphi$, as follows:

(1) guess a graph $G = (V, E, L, F_A)$ such that $|G| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$, and a mapping h_Q from V_Q to V, where V_Q is the set of nodes in Q;

(2) check whether h_Q is a match of Q in G; if so, continue; otherwise reject the current guess;

(3) check whether $h(\bar{x}) \models X$ and $h(\bar{x}) \not\models Y$; if so, continue; otherwise reject the current guess;

(4) check whether $G \models \Sigma$; if so, return true.

The correctness of the algorithm is assured by the small model property. For its complexity, step (2) is in PTIME, by the definition of matches. Step (3) is in PTIME, since $|X| + |Y| \leq |\varphi|$. Step (4) is in coNP (see the proof for the validation problem for GDCs). Therefore, the algorithm is in Σ_2^p, and the implication problem is in Π_2^p. □

<u>Lower bound</u>. We show that the implication problem is Π_2^p-hard for GDCs by reduction from the complement of GGCP (see GGCP in the proof of the satisfiability problem). Given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, where G is complete and F does not contain self cycles, we construct a set Σ of GDCs and another GDC φ such that $\Sigma \not\models \varphi$ if and only if there exists a 2-coloring of F that does not contain a monochromatic G as a subgraph.

We borrow some constructions from the lower bound proof for GDC satisfiability. Recall φ_1, φ_2, φ_3 and φ_4 given there. We define $\Sigma = \{\varphi_1, \varphi_2, \varphi_3\}$, which encodes 2-coloring of nodes and checks the existence of a monochromatic G. We define $\varphi = Q[x_0^1, x_1, \ldots, x_m](\emptyset \Rightarrow x_0^1.C = 1)$ to encode graph F, where $Q[\bar{x}] = (V, E, L)$,

- $V = V_F \cup \{x_0^1\}$; as opposed to φ_4 in the satisfiability proof, we use a single extra node x_0^1;
- $E = \{(u, 0, v), (v, 0, u) \mid (u, v) \in E_F\} \cup E_0$, where $E_0 = \{(x_1, 1, x_0^1), \ldots, (x_n, 1, x_0^1)\}$, simpler than its counterpart in φ_4; we encode an undirected edge with two directed edges, and
- for each node $x \in V$, $L(x)$='0'.

Intuitively, if $\Sigma \not\models \varphi$, then there exist a graph $G' \models \Sigma$ and a match h of Q in G' such that $h(x_0^1).C \neq 1$. Moreover, from h we can deduce a 2-coloring v of F. By $G' \models \varphi_1$ and $G' \models \varphi_2$, $h(x_0^1).C = 1$ or $h(x_0^1).C = 2$. In addition, φ_3 allows us to deduce the existence of a 2-coloring of F that does not contain a monochromatic G as a subgraph. Indeed, consider the 2-coloring v deluded and assume by contradiction that v contains a monochromatic G as a subgraph. Then by combining this 2-coloring (homomorphism) and h, we can get a match h' of Q_3 in G' such that $h'(x_0) = h(x_0^1)$, where Q_3 is the pattern of φ_3 in Σ. Since $G' \models \varphi_3$, we have that $h(x_0^1).C = 1$, which contradicts the assumption that $\Sigma \not\models \varphi$ and hence $h(x_0^1).C \neq 1$. Conversely, if there exists a 2-coloring of F that does not contain a monochromatic G as a subgraph, then we can construct a graph G_1 such that $G_1 \models \Sigma$ and $G_1 \not\models \varphi$. That is, $\Sigma \not\models \varphi$.

We next formalize this intuition and show that $\Sigma \not\models \varphi$ if and only if there exists a 2-coloring of F that does not contain a monochromatic G as a subgraph.

($\Rightarrow$) First assume that $\Sigma \not\models \varphi$. We show that there exists a 2-coloring v of F that there does not contain a monochromatic G as a subgraph. By $\Sigma \not\models \varphi$, there exists a graph $G' = (V, E, L, F_A)$ such that $G' \models \Sigma$ and $G' \not\models \varphi$. Since $G' \not\models \varphi$, there exists a match h of Q in G' such that $h(\bar{x}) \not\models (x_0^1.C = 1)$. Based on h, we define a 2-coloring v of F such that for each node $x \in V_F$, $v(x) = h(x).C$. By the definitions of φ_1 and φ_2, we know that $h(x).C$ is defined and has value 1 or 2. Thus v is well defined. Moreover, $h(x_0^1).C = 2$, since $h(\bar{x}) \not\models (x_0^1.C = 1)$. It remains to show that v does not contain a monochromatic G as a subgraph.

Assume by contradiction that there exists a monochromatic G as a subgraph in the 2-coloring v. Then we show that $h(x_0^1).C$ must be 1, which contradicts that $h(x_0^1).C = 2$ as argued above. To see this, it suffices to apply φ_3. That is, we construct a match h' of Q_3 in G' such that $h'(x_0) = h(x_0^1)$, and $h'(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$. For if it holds, then $h(x_0^1).C = h'(x_0).C = 1$ by $G' \models \varphi_3$.

The match h' can be constructed as follows. Let h_I be the mapping of G in F such that G is a monochromatic subgraph in ν. Let $h' = h \circ h'_I$, where h'_I extends h_I by letting $h'_I(x_0) = x_0^1$, where x_0 is not in G. Here $h \circ h'_I$ is the composition of two homomorphic mappings h from Q (encoding F) to G' and h'_I from G (encoded by Q_3) to F, and is hence a homomorphism from Q_3 to G'. Then we only need to show that $h'(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$. Since h_I is the match of Q_3 in G_Q such that G is a monochromatic subgraph in ν, we have that $h_I(x_1).C = h_I(x_2).C = \ldots = h_I(x_n).C$, where G_Q is the canonical graph of Q of φ. Hence $h \circ h_I(x_1).C = h \circ h_I(x_2).C = \ldots = h \circ h_I(x_n).C$. That is, $h \circ h'_I(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$. This leads to $h(x_0^1).C = 1$ and $h(x_0^1).C = 2$, a contradiction. Thus ν does not contain a monochromatic G as a subgraph.

($\Leftarrow$) Conversely, assume that ν is a 2-coloring of F that does not contain a monochromatic G as a subgraph. Based on ν, we construct a graph G' such that $G' \models \Sigma$ but $G' \not\models \varphi$. That is, $\Sigma \not\models \varphi$.

We next construct G'. Recall the canonical graph $G_Q = (V_Q, E_Q, L_Q, F_A)$ of the pattern Q of φ (Section 5.2). The graph G' is almost the same as G_Q, except that it carries attributes, while G_Q has an empty set of attributes. More specifically, $G' = (V', E', L', F'_A)$ is defined as follows:

- $V' = V_Q$, consisting of all nodes in V_Q;
- $E' = E_Q$, the same set of edges as in G_Q;
- for each node $x_i \in V'$, $L'(x_i)$='0'; and
- F'_A is defined as follows: for each node x in V', if $x \neq x_0^1$, then $F'_A(x).C = \nu(x)$; and $F'_A(x_0^1).C = 2$.

Note that $G' \not\models \varphi$ since Q has a match in G' but $F'_A(x_0^1).C = 2$. It remains to show that $G' \models \Sigma$.

Assume by contradiction that $G' \not\models \Sigma$. Recall that $\Sigma = \{\varphi_1, \varphi_2, \varphi_3\}$. Since for each node x in G', $x.C$ is either 1 or 2, we have that $G' \models \varphi_1$ and $G' \models \varphi_2$. Hence if $G' \not\models \Sigma$, then it must be the case that $G' \not\models \varphi_3$. Therefore, there exists a match h_1 of Q_3 in G' such that $h_1(\bar{x}) \models (x_1.C = x_2.C) \wedge \ldots \wedge (x_n.C = x_1.C)$. One can verify that $h_1(Q_3)$ is a monochromatic subgraph of G', since G is complete and F does not contain self cycles, along the same lines as the argument for its counterpart given in the proof of the satisfiability problem. By the construction of φ_3, Q_3 encodes G. As a result, G' contains a monochromatic G as a subgraph. Hence the 2-coloring ν of F contains a monochromatic G as a subgraph, which contradicts the assumption. Therefore, $G' \models \Sigma$.

Putting these together, we have that $\Sigma \not\models \varphi$. □

(3) The validation problem for GDCs. Finally, we show that the validation problem is coNP-complete for GDCs. The lower bound follows immediately from Theorem 5.16, since GEDs are a special case of GDCs. For the upper bound, for a graph G and a set Σ of GDCs, the algorithm given in the proof of Theorem 5.16 can still check whether $G \not\models \Sigma$. Moreover, the algorithm remains in NP since given a GDC $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a match h of Q in G, it is still in PTIME to check whether $h(\bar{x}) \models X$ but $h(\bar{x}) \not\models Y$. Hence the validation problem for GDCs is in coNP. □

This completes the proof of Theorem 7.2. □

Proof of Theorem 7.4

We show that the satisfiability, implication and validation problems for GED$^\vee$s are Σ_2^p-complete, Π_2^p-complete and coNP-complete, respectively. The proof is similar to the proof of Theorem 7.2 and hence, below we just highlight the difference in the two proofs.

(1) The satisfiability problem for GED$^\vee$s. Similar to GDCs, we first show that the satisfiability problem for GED$^\vee$s has a small model property. Based on this, we give an Σ_2^p algorithm to check whether a set Σ of GED$^\vee$s is satisfiable. Finally, we show that the satisfiability problem is Σ_2^p-hard.

The small model property. We show that if a set Σ of GED$^\vee$s is satisfiable, then Σ has a model G_h of size $O(4 \cdot |\Sigma|^3)$. To this end, we use the same G_h constructed in the proof of Theorem 7.2, which has size $O(4 \cdot |\Sigma|^3)$. Moreover, for each GED$^\vee$ $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ, there exists a match h_φ of Q in G.

We next show that $G_h \models \Sigma$. Suppose by contradiction that $G_h \not\models \Sigma$. Then there exist a GED$^\vee$ $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a match h_φ of Q in G_h such that $h_\varphi(\bar{x}) \models X$ but $h_\varphi(\bar{x}) \not\models Y$. That is, for all literals l in X (resp. in Y), $h_\varphi(\bar{x}) \models l$ (resp. $h_\varphi(\bar{x}) \not\models l$). By the definition of G_h (see details of G_h in the proof of Theorem 7.2), h_φ is also a match of Q in G. Moreover, the normalization process in the construction of G_h preserves the equality of attribute values. Hence for all literals l in X (resp. in Y), we also have that $h_\varphi(\bar{x}) \models l$ (resp. $h_\varphi(\bar{x}) \not\models l$) in G. Putting these together, we have that $h_\varphi(\bar{x}) \models X$ and $h_\varphi(\bar{x}) \not\models Y$ in G, which contradict the assumption that $G \models \Sigma$. Hence $G_h \models \Sigma$. □

Upper bound. Given the small model property, we give an Σ_2^p algorithm to check whether a set Σ of GED$^\vee$s is satisfiable. In fact, the algorithm for checking the satisfiability of GDCs still works on GED$^\vee$s. Moreover, it is in Σ_2^p, since it is in coNP to check whether a graph satisfies a set of GED$^\vee$s (see the proof for the validation problem to be given below); that is, the checking step of the algorithm is in coNP. Hence the satisfiability problem for GED$^\vee$ is in Σ_2^p. □

Lower bound. We show that the satisfiability problem is Σ_2^p-hard for GED$^\vee$s by reduction from the generalized graph coloring problem (GGCP) (see the proof of Theorem 7.2 for GGCP). Given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, where G is complete and F does not contain self cycles, we construct a set Σ of GED$^\vee$s such that Σ is satisfiable if and only if there exists a 2-coloring of F that does not contain a monochromatic G as a subgraph.

To define Σ, we revise the GDCs φ_1, φ_2, φ_3 and φ_4 given in the lower bound proof of Theorem 7.2 for the satisfiability problem of GDCs. Observe that φ_3 and φ_4 are GED$^\vee$s. Moreover, since φ_1 and φ_2 are used to ensure that all nodes are 2-colored, we can use the following single GED$^\vee$ instead:

$$\varphi_1' = Q_1[x]\big(\emptyset \Rightarrow (x.C = 1) \vee (x.C = 2)\big),$$

where $Q_1[x] = (V_1, E_1, L_1)$, $V_1 = \{x\}$, $E_1 = \emptyset$, and $L_1(x) =$'0'. Intuitively, this GED$^\vee$ alone ensures that each vertex has a color attribute C and its value must be either 1 or 2. We define $\Sigma = \{\varphi_1', \varphi_3, \varphi_4\}$. Then similar to the proof of Theorem 7.2, one can verify that Σ is satisfiable if and only if there exists a 2-coloring F that does not contain a monochromatic G as a subgraph.

The reduction uses three GED$^\vee$s defined in terms of variable literals and constant literals, without id literals. No forbidding constraints are needed. □

(2) The implication problem for GED$^\vee$s. We next show that the implication problem is Π_2^p-complete. We start with a small model property of the implication problem for GED$^\vee$s.

The small model property. We show that given a set Σ of GED$^\vee$s and another GED$^\vee$ $\varphi = Q[\bar{x}](X \Rightarrow Y)$, if $\Sigma \not\models \varphi$, then there exists a graph G_h such that $|G_h| \leq 2 \cdot |\varphi| \cdot (|\varphi| + |\Sigma| + 1)^2$, $G_h \models \Sigma$ and $G_h \not\models \varphi$. We use the same G_h constructed in the proof of Theorem 7.2 for the implication analysis of GDCs, which satisfies the bound on the size.

It remains to show that $G_h \models \Sigma$ and $G_h \not\models \varphi$. We first show that $G_h \models \Sigma$. Assume that $G_h \not\models \Sigma$ by contradiction. Then there exist a GED$^\vee$ $\varphi_1 = Q_1[\bar{x}_1](X_1 \Rightarrow Y_1)$ in Σ and a match h_{φ_1} of Q_1 in G_h such that $h_{\varphi_1}(\bar{x}_1) \models X_1$ but $h_{\varphi_1}(\bar{x}_1) \not\models Y_1$. That is, for all literals l in X_1 (resp. all literals l in Y_1), $h_{\varphi_1}(\bar{x}) \models l$ (resp. $h_{\varphi_1}(\bar{x}) \not\models l$). By the definition of G_h, h_{φ_1} is also a match of Q_1 in G. Moreover, the normalization in the construction of G_h preserves the equality of attribute values. As a result, for all literals l in X_1 (resp. all literals l in Y_1), we also have that $h_\varphi(\bar{x}) \models l$ (resp. $h_\varphi(\bar{x}) \not\models l$) in G.

Therefore, $h_{\varphi_1}(\bar{x}_1) \models X_1$ and $h_{\varphi_1}(\bar{x}_1) \not\models Y_1$ in G. That is, $G \not\models \varphi_1$, a contradiction to the assumption that $G \models \Sigma$. Hence $G_h \models \Sigma$. Along the same lines, we can verify that $G_h \not\models \varphi$. □

Upper bound. Leveraging the small model property, we develop an Σ_2^p algorithm to check whether $\Sigma \not\models \varphi$. The algorithm is the same as its counterpart given in the proof of Theorem 7.2 for implication. It remains in Σ_2^p since it is in PTIME to check whether a mapping from a pattern to a graph is a match, and whether a match satisfies a GED$^\vee$. Moreover, it is in coNP to check whether a graph satisfies Σ (see the proof of the validation problem for GED$^\vee$s below). Hence the problem for the implication analysis of GED$^\vee$s is in Π_2^p. □

Lower bound. We show that the implication problem is Π_2^p-hard for GED$^\vee$s by reduction from the complement of the generalized graph coloring problem (GGCP). Given two undirected graphs $F = (V_F, E_F)$ and $G = (V_G, E_G)$, where G is complete and F does not contain self cycles, we construct a set Σ of GED$^\vee$s and another GED$^\vee$ φ such that $\Sigma \models \varphi$ if and only if for any 2-coloring of F, it contains a monochromatic G as a subgraph.

Recall φ_1', φ_3 and φ_4, the GED$^\vee$s defined in the lower bound proof of the satisfiability problem for GED$^\vee$s given above. We define the set $\Sigma = \{\varphi_1', \varphi_3\}$, and GED$^\vee$ $\varphi = Q[x_0, x_1, \ldots, x_m](\emptyset \Rightarrow x_0^1.C = 1)$ to encode graph F, where $Q[\bar{x}] = (V_4, E_4, L_4)$,

- $V_4 = V_F \cup \{x_0^1\}$;
- $E_4 = \{(u, 0, v)(v, 0, u) \mid (u, v) \in E_F\} \cup E_0$, where $E_0 = \{(x_1, 1, x_0^1), \ldots, (x_n, 1, x_0^1)\}$; and
- for each node $x \in V_4$, $L_4(x)$='0'.

Similar to the lower bound proof of Theorem 7.2 for the implication problem for GDCs, one can verify that $\Sigma \models \varphi$ if and only if for each 2-coloring of F, it contains a monochromatic G. □

(3) The validation problem for GED$^\vee$s. We show that the validation problem is coNP-complete for GED$^\vee$s. For the lower bound, as observed in Section 7.2, each GED φ can be expressed as a set Σ_φ of GED$^\vee$s of cardinality bounded by $|\varphi|$, and each GED$^\vee$ in Σ_φ has size at most $|\varphi|$. That is, $\Sigma_\varphi \models \varphi$ and $\varphi \models \phi$ for each $\phi \in \Sigma_\varphi$, and $|\Sigma_\varphi| \leq |\varphi|^2$. Hence as an immediate consequence of Theorem 5.16, the validation problem is coNP-hard for GED$^\vee$s.

For the upper bound, an algorithm is developed in the proof of Theorem 5.16 that, given a graph G and a set Σ_φ of GEDs, checks whether $G \not\models \Sigma$. The algorithm also works on GED$^\vee$s and remains in NP. Indeed, given GED$^\vee$ $\varphi = Q[\bar{x}](X \Rightarrow Y)$ in Σ and a mapping h from Q to G, it is in PTIME to check whether h is a match of Q in G, and whether $h(\bar{x}) \models X$ but $h(\bar{x}) \not\models l$ for each $l \in Y$. Thus the validation problem remains in coNP for GED$^\vee$s.

This completes the proof of Theorem 7.4. □

Received February 2007; revised March 2009; accepted June 2009